



Universidade Estadual de Campinas



Centro Nacional de Processamento de Alto Desempenho - São Paulo

CENAPAD

Apostila de Treinamento:
SAS Programação 2
- SQL, MACRO, Recursos de DATA Step -

Revisão: 2015

CONTEÚDO

1 – Introdução ao SQL	pag.05
1.1 – SQL no SAS	pag.06
1.2 – Comando de Grupo SELECT-FROM	pag.08
1.2.1 – Subcomando WHERE	pag.13
1.2.2 – Subcomando ORDER BY	pag.16
1.2.3 – Subcomando GROUP BY	pag.17
1.2.4 – Subcomando HAVING	pag.20
1.3 – Parâmetro CALCULATED	pag.21
1.4 – Comando de Grupo CREATE TABLE-AS	pag.22
1.5 – “Subquery: IN-LINE VIEW”	pag.25
1º LABORATÓRIO	pag.27
2 – Processamento MACRO	pag.29
2.1 – Tabela Global	pag.32
2.2 – Variável de Macro	pag.32
2.3 – Comando %PUT	pag.32
2.4 – Comando %LET	pag.33
2.5 – Referências de Variável de Macro	pag.34
2.6 – Opção NOSYMBOLGEN/SYMBOLGEN	pag.36
2.7 – Funções Macro	pag.37
2.7.1 – Função %SUBSTR	pag.37
2.7.2 – Função %SCAN	pag.38
2.7.3 – Função %UPCASE	pag.39
2.7.4 – Função %EVAL/%SYSEVALF	pag.40
2.7.5 – Função %SYSFUNC	pag.41
2º LABORATÓRIO – Macro Parte 1	pag.42
2.8 – Rotinas Macro	pag.43
2.8.1 – Opções de Compilação e Execução	pag.43
2.8.2 – Parâmetros de Rotinas Macro	pag.46
2.8.3 – Tabela Local	pag.48
2.8.4 – Comando %IF-%THEN-%ELSE	pag.49
2.8.5 – Comando %DO-%END	pag.50
2.8.6 – Comando %DO-%TO-%END Iterativo	pag.51
2.9 – Criar Variáveis de Macro em DATA Step	pag.52
2.10 – Criar Variáveis de Macro em PROC SQL	pag.54
2.11 – Referências Indiretas de Variáveis de Macro	pag.57
3º LABORATÓRIO – Macro Parte 2	pag.60
3 – Indexação dos Dados	pag.61
3.1 – Indexação com o SAS	pag.62
3.1.1 – Procedimento DATASETS	pag.62
3.1.2 – Procedimento SQL	pag.64
3.1.3 – Criação de Index em DATA Step	pag.65

3.2 – Utilização do Index	pag.67
3.2.1 – Situações em que o SAS poderá utilizar o Index	pag.67
3.2.2 – Situações em que o SAS não utilizará o Index	pag.70
3.2.3 – Utilização de Index Composto	pag.70
3.2.4 – Controlar o uso do Index no Comando WHERE	pag.71
3.3 – Recomendações sobre Indexação	pag.72
3.4 – Manutenção de Index	pag.72
4º LABORATÓRIO	pag.73
4 – Recursos de Programação em SAS	pag.74
4.1 – Performance	pag.74
4.1.1 – Opção BUFSIZE	pag.75
4.1.2 – Opção BUFNO	pag.75
4.1.3 – Comando SASFILE	pag.75
4.1.4 – Opção FULLSTIMER NOFULLSTIMER	pag.75
4.1.5 – Opção COMPRESS	pag.75
4.2 – Combinar Arquivos Utilizando Indexes	pag.78
4.3 – Combinação de Arquivos em Data Step – Opção de Arquivo IN=	pag.80
4.4 – Conjunto de Dados – “ARRAYs”	pag.82
5º LABORATÓRIO – Recursos de Data Step – Parte 1	pag.85
4.5 – Comando SELECT-WHEN-OTHERWISE-END	pag.86
4.6 – Comando MODIFY	pag.87
4.7 – Atualização de Dados com Arquivo de Transação	pag.88
6º LABORATÓRIO – Recursos de Data Step – Parte 2	pag.89
BIBLIOGRAFIA	pag.90

Tipografia utilizada na apostila

Na apresentação de alguns comandos do SAS, foram utilizados símbolos gráficos que identificam, na sintaxe do comando, a característica de ser opcional ou obrigatório:

< característica > É **obrigatório** a informação no comando;

[característica] É **opcional** a informação no comando.

Exemplo: Utilização de procedimentos SAS

```
PROC <tipo> [opção1 opção2 opção3 . . . opção] ;  
    [comando1] ;  
    [comando2] ;  
    [WHERE <expressão lógica>] ;  
    . . .  
    [comandon] ;  
RUN ;
```

1 – Introdução ao SQL

Structured Query Language, ou **Linguagem de Consulta Estruturada** ou **SQL**, é uma linguagem de pesquisa declarativa para banco de dados relacional (Oracle, DB2, Teradata, Informix, SQLServer, Postgres, etc). Muitas das características originais do SQL foram inspiradas na álgebra relacional (Forma de cálculo sobre conjuntos ou relações: Selecionar, Unir, Projetar, Produto Cartesiano entre conjuntos);

O **SQL** foi desenvolvido originalmente no início dos anos 70 nos laboratórios da IBM em San Jose, dentro do projeto de um sistema de Banco de Dados, o **System R**, que tinha por objetivo demonstrar a viabilidade da implementação do modelo relacional proposto por **Edgar Frank Codd** matemático britânico;

A linguagem SQL é um grande padrão de banco de dados. Isto decorre da sua simplicidade e facilidade de uso. Ela se diferencia de outras linguagens de consulta a banco de dados no sentido de que, uma consulta SQL, especifica a forma do resultado e não o caminho para chegar a ele. Ela é uma linguagem declarativa em oposição a outras linguagens procedurais. Isto reduz o ciclo de aprendizado daqueles que se iniciam na linguagem;

Embora o **SQL** tenha sido originalmente criado pela IBM, rapidamente surgiram vários "dialetos" desenvolvidos por outros fabricantes. Essa expansão levou à necessidade de ser criado e adaptado um padrão para a linguagem. Esta tarefa foi realizada pela **American National Standards Institute (ANSI)** em 1986 e **International Organization for Standardization (ISO)** em 1987;

O princípio básico do **SQL** é aplicar uma estrutura de comandos conforme se expressa, ou se fala, da necessidade de se obter os dados de uma base de dados, gerando um relatório.

```
"Eu preciso selecionar o nome, a empresa, o cargo e o salário dos dados da tabela cadastro, aonde o salário seja maior que R$ 10.000,00 e que estejam organizados em ordem alfabética pelo nome"
```

```
SELECT nome, empresa, cargo, salário
FROM cadastro
WHERE salário > 10000
ORDER BY nome
```

A execução de um programa SQL é chamada de "Query", consulta. Uma "query" pode ser executada em qualquer padrão ANSI de SQL.

1.1 - SQL no SAS

O Ambiente SQL no SAS é ativado como um procedimento:

PROC SQL [*opção1 opção2 opção3 ... opção*] ;

<Comando de grupo>
[subcomando1]
[subcomando2]
...
[subcomandon] ;

query 1

<Comando de grupo>
[subcomando1]
[subcomando2]
...
[subcomandon] ;

query 2

...

<Comando de grupo>
[subcomando1]
[subcomando2]
...
[subcomandon] ;

query n

QUIT ;

Opções:	<u>NOPRINT/PRINT</u> <u>NONUMBER/NUMBER</u> <u>NODOUBLE/DOUBLE</u> OUTOBS=MAX INOBS=MAX etc	Gerar relatório; Numerar as linhas do relatório; Espaço entre linhas; Número de linhas apresentadas no relatório; Número de linhas processadas pelo SQL;
Comandos de grupo:	SELECT-FROM CREATE TABLE CREATE INDEX INSERT INTO UPDATE TABLE ALTER TABLE DROP TABLE DELETE etc	Selecionar dados; Criar uma tabela; Criar um índice em uma tabela; Inserir dados em uma tabela que já exista; Atualizar dados de uma tabela; Alterar a estrutura de uma tabela; Apagar uma tabela; Apagar linhas de uma tabela;
Subcomandos:	WHERE SET GROUP BY ORDER BY HAVING INTO etc	

Observações:

- 1 – Toda “query” que inicia com o comando SELECT, normalmente, gera relatório;
- 2 – Os comandos em uma “query” possuem uma seqüência lógica correta;
- 3 – O símbolo ; só deve ser colocado no último comando de cada “query”;
- 4 – A PROC SQL finaliza com o comando QUIT;
- 5 – Terminologia em SQL:

SAS/BASE	SQL
Arquivo, uma base de dados	tabela (table)
Variáveis, campos de dados	coluna (column)
Observações, registros	linha (row)

Exemplo:

```
proc sql;
  select nome, empresa, funcao, salario
  from arq.cadastro
  where funcao='ANALISTA' and salario gt 5000
  order by empresa ;
quit;
```

The SAS System

nome	empresa	funcao	salario
MOUA, TANIA	ATLAS S.A.	ANALISTA	5418.33
MOUA, CARLA	ATLAS S.A.	ANALISTA	7221.58
MOUA, LIGIA	ATLAS S.A.	ANALISTA	5047.05
MOUA, LAURA	ATLAS S.A.	ANALISTA	7402.21
MOUA, ELIANE	MALTA LTDA	ANALISTA	13694.19
MOUA, MONICA	MALTA LTDA	ANALISTA	12568.82
MOUA, MARCO	MALTA LTDA	ANALISTA	9988.53
MOUA, MADALENA	MALTA LTDA	ANALISTA	14821.37
MOUA, PAULO	PARIS INSTITUTO	ANALISTA	9548.36

1.2 – Comando SELECT-FROM

- Comando que **seleciona colunas** de dados de um arquivo SAS;
- O comando SELECT sempre funciona em parceria com o comando FROM, ou seja, pode selecionar colunas a partir de uma tabela;
- Com o comando SELECT é possível criar novas colunas de dados, alterar os dados das colunas existentes, montar expressões aritméticas e melhorar a aparência dos dados das colunas selecionadas;

```
SELECT coluna1 [opções], coluna2 [opções], ..., colunan [opções]
    [*]
    [AS]
    [DISTINCT]
    [INTO]
FROM tabela1, tabela2, ..., tabelan
    [AS]
;
```

coluna1...n [opções]	Seleciona, cria uma nova coluna ou altera uma coluna existente; Modifica a aparência da coluna: AS , LABEL= , FORMAT=
*	Seleciona todas as colunas das tabelas especificadas no comando FROM;
AS	Especifica um novo nome para a coluna, no SELECT, ou, um “apelido” para a tabela, no FROM;
DISTINCT	Elimina as ocorrências duplicadas da combinação de colunas especificadas;
INTO	Cria variáveis macro a partir dos dados de uma coluna;

Detalhes:

- As colunas especificadas em um comando **SELECT** devem vir **separadas por vírgulas**;
- **ATENÇÃO!** Quando se especifica mais de uma tabela no comando FROM, o SQL efetua uma combinação de todas as linhas de dados (registros) de uma tabela com as linhas de dados das outras tabelas, ou seja, é o **produto cartesiano** entre as tabelas. Isso causa uma degradação da performance da “query”, dependendo do número de linhas existentes em cada tabela.

```
SELECT *
FROM tabelaA, tabelaB, tabelaC ;
```

tabelaA=1000 registros tabelaB=1000 registros tabelaC=1000 registros	} Relatório = 1000 x 1000 x 1000 = 1.000.000.000 linhas
--	---

- O uso otimizado do produto cartesiano na combinação de tabelas, com o comando FROM, é feito utilizando-se o comando WHERE, que seleciona os dados a partir de uma condição lógica;

Ex.1 – Seleção de dados

```

options ls=100 nodate pageno=1;
proc sql outobs=10 number;

    select nome, empresa, funcao, salario
           from arq.cadastro;

select *
       from arq.media;

quit;

```

The SAS System

1

Row	nome	empresa	funcao	salario
1	MALÁ, ROSANE	PARIS INSTITUTO	PROGRAMADOR	1662.28
2	SILVA, CECILIA	PARIS INSTITUTO	PROGRAMADOR	3612.03
3	PINTOTO, TANIA	PARIS INSTITUTO	PROGRAMADOR	3133.36
4	MARQUES, LIGIA	PARIS INSTITUTO	PROGRAMADOR	1584.08
5	SERPA, RENATO	PARIS INSTITUTO	PROGRAMADOR	3317.27
6	PISCO, PAULO	PARIS INSTITUTO	PROGRAMADOR	3900.55
7	MARKO, MARCO	PARIS INSTITUTO	PROGRAMADOR	2139.16
8	GUEDES, CARLA	PARIS INSTITUTO	PROGRAMADOR	2076.92
9	YATAKA, CARLA	PARIS INSTITUTO	PROGRAMADOR	1534.09
10	MILIA, FLAVIA	PARIS INSTITUTO	PROGRAMADOR	1795.6

The SAS System

2

Row	empresa	funcao	Número de Funcionários	Gastos com Salário	Média Salarial
1	ATLAS S.A.	GERENTE	3	45.503,53	15.167,84
2	MALTA LTDA	DIRETOR	1	25.377,28	25.377,28
3	MALTA LTDA	GERENTE	4	74.481,05	18.620,26
4	PARIS INSTITUTO	GERENTE	1	19.303,66	19.303,66

Ex.2 – Seleção de dados de mais de um arquivo

```
proc sql inobs=4 number;
  select *
    from arq.valores, arq.media;
quit;
```

SAS Log

```
132 proc sql inobs=4 number;
133   select *
134     from arq.valores, arq.media;
NOTE: The execution of this query involves performing one or more Cartesian product joins
      that can not be optimized.
WARNING: Only 4 records were read from ARQ.VALORES due to INOBS= option.
WARNING: Only 4 records were read from ARQ.MEDIA due to INOBS= option.
135 quit;
NOTE: PROCEDURE SQL used (Total process time):
      real time           0.01 seconds
      cpu time            0.00 seconds
```

The SAS System

Row	funcionario	sal	nome	sexo	idade	peso	altura	salario
1	MALA, ROSANE	1662.28	MALA, ROSANE	F	20	74.4	1.78	1662.28
2	MALA, ROSANE	1662.28	SILVA, CECILIA	F	30	74.3	1.86	3612.03
3	MALA, ROSANE	1662.28	PINTOTO, TANIA	F	29	59.9	1.73	3133.36
4	MALA, ROSANE	1662.28	MARQUES, LIGIA	F	23	76.8	1.88	1584.08
5	SILVA, CECILIA	3612.03	MALA, ROSANE	F	20	74.4	1.78	1662.28
6	SILVA, CECILIA	3612.03	SILVA, CECILIA	F	30	74.3	1.86	3612.03
7	SILVA, CECILIA	3612.03	PINTOTO, TANIA	F	29	59.9	1.73	3133.36
8	SILVA, CECILIA	3612.03	MARQUES, LIGIA	F	23	76.8	1.88	1584.08
9	PINTOTO, TANIA	3133.36	MALA, ROSANE	F	20	74.4	1.78	1662.28
10	PINTOTO, TANIA	3133.36	SILVA, CECILIA	F	30	74.3	1.86	3612.03
11	PINTOTO, TANIA	3133.36	PINTOTO, TANIA	F	29	59.9	1.73	3133.36
12	PINTOTO, TANIA	3133.36	MARQUES, LIGIA	F	23	76.8	1.88	1584.08
13	MARQUES, LIGIA	1584.08	MALA, ROSANE	F	20	74.4	1.78	1662.28
14	MARQUES, LIGIA	1584.08	SILVA, CECILIA	F	30	74.3	1.86	3612.03
15	MARQUES, LIGIA	1584.08	PINTOTO, TANIA	F	29	59.9	1.73	3133.36
16	MARQUES, LIGIA	1584.08	MARQUES, LIGIA	F	23	76.8	1.88	1584.08

Ex.3 – Seleção de colunas que possuem o mesmo nome em mais de um arquivo

```
proc sql inobs=4 number;
  select nome, empresa, cpf, salario
    from arq.cadastro, arq.media;
quit;
```

SAS Log

```
152 proc sql inobs=4 number;
153   select nome, empresa, cpf, salario
154     from arq.cadastro, arq.media;
ERROR: Ambiguous reference, column nome is in more than one table.
ERROR: Ambiguous reference, column salario is in more than one table.
155 quit;
NOTE: The SAS System stopped processing this step because of errors.
NOTE: PROCEDURE SQL used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds
```

```
proc sql inobs=3 number;
  select cadastro.nome, empresa, cpf, media.salario
    from arq.cadastro, arq.media;
quit;
```

The SAS System

Row	nome	empresa	cpf	salario
1	MALA, ROSANE	PARIS INSTITUTO	12862709772	1662.28
2	SILVA, CECILIA	PARIS INSTITUTO	15777878374	1662.28
3	PINTOTO, TANIA	PARIS INSTITUTO	84879489045	1662.28
4	MALA, ROSANE	PARIS INSTITUTO	12862709772	3612.03
5	SILVA, CECILIA	PARIS INSTITUTO	15777878374	3612.03
6	PINTOTO, TANIA	PARIS INSTITUTO	84879489045	3612.03
7	MALA, ROSANE	PARIS INSTITUTO	12862709772	3133.36
8	SILVA, CECILIA	PARIS INSTITUTO	15777878374	3133.36
9	PINTOTO, TANIA	PARIS INSTITUTO	84879489045	3133.36

Ex.4 – Criando novas colunas no relatório

```
proc sql outobs=10 number;
  select nome,
         empresa,
         admissao,
         "Tempo de Empresa:",
         int((today()-admissao)/365.25),
         "anos",
         salario*0.1 as bonus
  from arq.cadastro;
quit;
```

} Colunas existentes no arquivo cadastro;
 } Novas colunas **sem nome**;
 } Nova coluna com nome **"bonus"**

The SAS System

Row	nome	empresa	admissao			bonus
1	MALA, ROSANE	PARIS INSTITUTO	15772	Tempo de Empresa:	7 anos	166.228
2	SILVA, CECILIA	PARIS INSTITUTO	16064	Tempo de Empresa:	6 anos	361.203
3	PINTOTO, TANIA	PARIS INSTITUTO	16021	Tempo de Empresa:	6 anos	313.336
4	MARQUES, LIGIA	PARIS INSTITUTO	15970	Tempo de Empresa:	6 anos	158.408
5	SERPA, RENATO	PARIS INSTITUTO	16027	Tempo de Empresa:	6 anos	331.727
6	PISCO, PAULO	PARIS INSTITUTO	15818	Tempo de Empresa:	7 anos	390.055
7	MARKO, MARCO	PARIS INSTITUTO	15914	Tempo de Empresa:	7 anos	213.916
8	GUEDES, CARLA	PARIS INSTITUTO	16002	Tempo de Empresa:	6 anos	207.692
9	YATAKA, CARLA	PARIS INSTITUTO	15885	Tempo de Empresa:	7 anos	153.409
10	MILIA, FLAVIA	PARIS INSTITUTO	15877	Tempo de Empresa:	7 anos	179.560

Ex.5 – Melhorando a aparência das colunas

```
title "Relatório de Tempo de Empresa";
proc sql outobs=10;
  select nome label="Nome do Funcionário",
         empresa label="Empresa",
         admissao label="Data de Admissão" format=ddmmyy10.,
         "Tempo de Empresa:",
         int((today()-admissao)/365.25) format=2.,
         "anos",
         salario*0.1 as bonus label="Bonus" format=comma8.2
  from arq.cadastro;
quit;
```

Relatório de Tempo de Empresa

Nome do Funcionário	Empresa	Data de Admissão			Bonus
MALA, ROSANE	PARIS INSTITUTO	08/03/2003	Tempo de Empresa:	7 anos	166,23
SILVA, CECILIA	PARIS INSTITUTO	25/12/2003	Tempo de Empresa:	6 anos	361,20
PINTOTO, TANIA	PARIS INSTITUTO	12/11/2003	Tempo de Empresa:	6 anos	313,34
MARQUES, LIGIA	PARIS INSTITUTO	22/09/2003	Tempo de Empresa:	6 anos	158,41
SERPA, RENATO	PARIS INSTITUTO	18/11/2003	Tempo de Empresa:	6 anos	331,73
PISCO, PAULO	PARIS INSTITUTO	23/04/2003	Tempo de Empresa:	7 anos	390,06
MARKO, MARCO	PARIS INSTITUTO	28/07/2003	Tempo de Empresa:	7 anos	213,92
GUEDES, CARLA	PARIS INSTITUTO	24/10/2003	Tempo de Empresa:	6 anos	207,69
YATAKA, CARLA	PARIS INSTITUTO	29/06/2003	Tempo de Empresa:	7 anos	153,41
MILIA, FLAVIA	PARIS INSTITUTO	21/06/2003	Tempo de Empresa:	7 anos	179,56

Ex.6 – Seleccionando dados distintos de columnas

```
options nodate number pageno=1;
proc sql outobs=20;
  select distinct empresa, funcao, sexo
     from arq.cadastro;

  select distinct empresa, funcao, e_civil
     from arq.cadastro;
quit;
```

The SAS System

1

empresa	funcao	sexo
	DESEMPREGADO	F
	DESEMPREGADO	M
ATLAS S.A.	ANALISTA	F
ATLAS S.A.	GERENTE	F
ATLAS S.A.	GERENTE	M
ATLAS S.A.	PROGRAMADOR	F
ATLAS S.A.	PROGRAMADOR	M
MALTA LTDA	ANALISTA	F
MALTA LTDA	ANALISTA	M
MALTA LTDA	DIRETOR	M
MALTA LTDA	GERENTE	F
MALTA LTDA	GERENTE	M
MALTA LTDA	PROGRAMADOR	F
MALTA LTDA	PROGRAMADOR	M
PARIS INSTITUTO	ANALISTA	M
PARIS INSTITUTO	GERENTE	F
PARIS INSTITUTO	PROGRAMADOR	F
PARIS INSTITUTO	PROGRAMADOR	M

The SAS System

2

empresa	funcao	e_civil
	DESEMPREGADO	2
ATLAS S.A.	ANALISTA	1
ATLAS S.A.	ANALISTA	3
ATLAS S.A.	GERENTE	1
ATLAS S.A.	PROGRAMADOR	1
ATLAS S.A.	PROGRAMADOR	3
MALTA LTDA	ANALISTA	1
MALTA LTDA	ANALISTA	2
MALTA LTDA	DIRETOR	1
MALTA LTDA	GERENTE	1
MALTA LTDA	PROGRAMADOR	1
MALTA LTDA	PROGRAMADOR	2
PARIS INSTITUTO	ANALISTA	1
PARIS INSTITUTO	GERENTE	1
PARIS INSTITUTO	PROGRAMADOR	1

1.2.1 – Subcomando WHERE

- Comando para **selecionar linhas** de dados de acordo com uma expressão lógica, dentro do comando de grupo SELECT-FROM;
- O comando WHERE deve ser posicionado logo após o comando FROM;
- O comando WHERE ativa a otimização do produto cartesiano, quando mais de um arquivo é especificado no comando FROM.

WHERE <expressão> ;

expressão Combinação de variáveis com operadores de comparação e/ou operadores lógicos que determinam uma condição.

Operadores de Comparação

GT	>	maior que
LT	<	menor que
EQ	=	igual a
LE	<=	menor ou igual a
GE	>=	maior ou igual a
NE	~=	não é igual (diferente)
CONTAINS	?	contém caracteres
NL		não é menor
NG		não é maior
IN (... ..)		está no conjunto
BETWEEN-AND		está entre
<u>LIKE</u>		se parece com
		% substitui qualquer conjunto de caracteres _ substitui apenas um caractere

Operadores Lógicos

AND	&	e, ambos
OR		ou, um ou outro
NOT	~	não, negação

Ex.7 – Seleccionando linhas de dados

```
options nodate number pageno=1;
proc sql number;
  select empresa, nome, salario
  from arq.cadastro
  where funcao="GERENTE";
quit;
```

The SAS System 1

Row	empresa	nome	salario
1	PARIS INSTITUTO	MOUA, MARIA	19303.66
2	MALTA LTDA	MOUA, RENATO	20457.36
3	MALTA LTDA	MOUA, MIRIAM	13075.91
4	MALTA LTDA	MOUA, ROSANE	20336.22
5	MALTA LTDA	MOUA, JOAO	20611.56
6	ATLAS S.A.	MOUA, MARCIO	10828.7
7	ATLAS S.A.	MOUA, LUIS	17414.14
8	ATLAS S.A.	MOUA, LICIA	17260.69

```
options nodate number pageno=1;
proc sql number;
  select empresa, nome, salario
  from arq.cadastro
  where salario between 5000 and 8000 and sexo='F';
quit;
```

The SAS System 1

Row	empresa	nome	salario
1	MALTA LTDA	LUILA, CARLA	5178.65
2	MALTA LTDA	ANJOA, MONICA	5066.22
3	ATLAS S.A.	MOUA, CARLA	7221.58
4	ATLAS S.A.	MOUA, TANIA	5418.33
5	ATLAS S.A.	MOUA, LIGIA	5047.05
6	ATLAS S.A.	MOUA, LAURA	7402.21

```
options nodate number pageno=1;
proc sql number;
  select empresa, nome, salario
  from arq.cadastro
  where nome like "%IA,LI%";

  select empresa, nome, salario
  from arq.cadastro
  where nome like "%IA__LI%";
quit;
```

The SAS System 1

Row	empresa	nome	salario
1	MALTA LTDA	MILIA, LICIA	3197.88
2	ATLAS S.A.	MILIA, LIGIA	3328.5

The SAS System 2

Row	empresa	nome	salario
1	MALTA LTDA	MILIA, ELIANE	3828.07

Ex.8 – Combinação de arquivos com o comando WHERE

```
proc sql;
  select * from arq.aumento;
quit;
```

The SAS System

empresa	funcao	aumento
PARIS INSTITUTO	GERENTE	1000
PARIS INSTITUTO	DIRETOR	2000
PARIS INSTITUTO	ANALISTA	500
PARIS INSTITUTO	PROGRAMADOR	800
MALTA LTDA	GERENTE	1500
MALTA LTDA	PROGRAMADOR	100
ATLAS S.A.	DIRETOR	2500

```
options ls=100 nodate number pageno=1;
proc sql number outobs=20;
  select c.empresa, nome, c.funcao, salario, aumento,
         salario+aumento as novo_sal
         from arq.cadastro as c, arq.aumento as a
         where c.empresa=a.empresa ;
quit;
```

The SAS System

1

Row	empresa	nome	funcao	salario	aumento	novo_sal
1	PARIS INSTITUTO	MALA, ROSANE	PROGRAMADOR	1662.28	1000	2662.28
2	PARIS INSTITUTO	MALA, ROSANE	PROGRAMADOR	1662.28	800	2462.28
3	PARIS INSTITUTO	MALA, ROSANE	PROGRAMADOR	1662.28	500	2162.28
4	PARIS INSTITUTO	MALA, ROSANE	PROGRAMADOR	1662.28	2000	3662.28
5	PARIS INSTITUTO	SILVA, CECILIA	PROGRAMADOR	3612.03	1000	4612.03
6	PARIS INSTITUTO	SILVA, CECILIA	PROGRAMADOR	3612.03	800	4412.03
7	PARIS INSTITUTO	SILVA, CECILIA	PROGRAMADOR	3612.03	500	4112.03
8	PARIS INSTITUTO	SILVA, CECILIA	PROGRAMADOR	3612.03	2000	5612.03
9	PARIS INSTITUTO	PINTOTO, TANIA	PROGRAMADOR	3133.36	1000	4133.36
10	PARIS INSTITUTO	PINTOTO, TANIA	PROGRAMADOR	3133.36	800	3933.36
11	PARIS INSTITUTO	PINTOTO, TANIA	PROGRAMADOR	3133.36	500	3633.36
12	PARIS INSTITUTO	PINTOTO, TANIA	PROGRAMADOR	3133.36	2000	5133.36
13	PARIS INSTITUTO	MARQUES, LIGIA	PROGRAMADOR	1584.08	1000	2584.08
14	PARIS INSTITUTO	MARQUES, LIGIA	PROGRAMADOR	1584.08	800	2384.08
15	PARIS INSTITUTO	MARQUES, LIGIA	PROGRAMADOR	1584.08	500	2084.08
16	PARIS INSTITUTO	MARQUES, LIGIA	PROGRAMADOR	1584.08	2000	3584.08
17	PARIS INSTITUTO	SERPA, RENATO	PROGRAMADOR	3317.27	1000	4317.27
18	PARIS INSTITUTO	SERPA, RENATO	PROGRAMADOR	3317.27	800	4117.27
19	PARIS INSTITUTO	SERPA, RENATO	PROGRAMADOR	3317.27	500	3817.27
20	PARIS INSTITUTO	SERPA, RENATO	PROGRAMADOR	3317.27	2000	5317.27

```
options ls=100 nodate number pageno=1;
proc sql number outobs=20;
  select c.empresa, nome, c.funcao, salario, aumento,
         salario+aumento as novo_sal
         from arq.cadastro as c, arq.aumento as a
         where c.empresa=a.empresa and c.funcao=a.funcao;
quit;
```

The SAS System

1

Row	empresa	nome	funcao	salario	aumento	novo_sal
1	PARIS INSTITUTO	MALA, ROSANE	PROGRAMADOR	1662.28	800	2462.28
2	PARIS INSTITUTO	SILVA, CECILIA	PROGRAMADOR	3612.03	800	4412.03
3	PARIS INSTITUTO	PINTOTO, TANIA	PROGRAMADOR	3133.36	800	3933.36
4	PARIS INSTITUTO	MARQUES, LIGIA	PROGRAMADOR	1584.08	800	2384.08
5	PARIS INSTITUTO	SERPA, RENATO	PROGRAMADOR	3317.27	800	4117.27
6	PARIS INSTITUTO	PISCO, PAULO	PROGRAMADOR	3900.55	800	4700.55
7	PARIS INSTITUTO	MARKO, MARCO	PROGRAMADOR	2139.16	800	2939.16
8	PARIS INSTITUTO	GUEDES, CARLA	PROGRAMADOR	2076.92	800	2876.92
9	PARIS INSTITUTO	YATAKA, CARLA	PROGRAMADOR	1534.09	800	2334.09
10	PARIS INSTITUTO	MILIA, FLAVIA	PROGRAMADOR	1795.6	800	2595.6
11	PARIS INSTITUTO	MARUEL, CARLA	PROGRAMADOR	1950.75	800	2750.75
12	PARIS INSTITUTO	MENDES, FLAVIA	PROGRAMADOR	3888.4	800	4688.4
13	PARIS INSTITUTO	CERTO, FRANCISCA	PROGRAMADOR	1713.6	800	2513.6
14	PARIS INSTITUTO	SUNAY, FRANCISCA	PROGRAMADOR	2070.6	800	2870.6
15	PARIS INSTITUTO	APARECIDO, EDUARDO	PROGRAMADOR	1770.37	800	2570.37
16	PARIS INSTITUTO	SONTAS, ELIANE	PROGRAMADOR	4152.76	800	4952.76
17	PARIS INSTITUTO	APARECIDO, ELIANE	PROGRAMADOR	3685.27	800	4485.27
18	PARIS INSTITUTO	MARKO, MARIA	PROGRAMADOR	3781.36	800	4581.36
19	PARIS INSTITUTO	BENTES, MARIA	PROGRAMADOR	1679.65	800	2479.65
20	PARIS INSTITUTO	ANJOA, ROSANE	PROGRAMADOR	3626.59	800	4426.59

1.2.2 – Subcomando ORDER BY

- Comando que **organiza** os dados de uma ou mais colunas em ordem ascendente ou descendente, dentro do comando de grupo SELECT-FROM;
- **Não é necessário** executar um PROC SORT antes;
- O padrão do ORDER BY é sempre organizar os dados em ordem ascendente;
- O ORDER BY é sempre o **último comando** em uma “query”;

ORDER BY <coluna1> [opção] , <coluna2> [opção] , . . . , <colunan> [opção] ;

coluna1...n Variáveis dos arquivos especificados no comando FROM ou a posição numérica da coluna no comando SELECT;

opção Especificar a ordenação descendente da coluna: **DESC**

Ex.9 – Organizar dados

```
proc sql number outobs=10;
select empresa, nome, funcao, salario
from arq.cadastro
where lowercase(empresa) contains "atlas"
order by salario desc;
quit;
```

The SAS System

1

Row	empresa	nome	funcao	salario
1	ATLAS S.A.	MOUA, LUIS	GERENTE	17414.14
2	ATLAS S.A.	MOUA, LICIA	GERENTE	17260.69
3	ATLAS S.A.	MOUA, MARCIO	GERENTE	10828.7
4	ATLAS S.A.	MOUA, LAURA	ANALISTA	7402.21
5	ATLAS S.A.	MOUA, CARLA	ANALISTA	7221.58
6	ATLAS S.A.	MOUA, TANIA	ANALISTA	5418.33
7	ATLAS S.A.	MOUA, LIGIA	ANALISTA	5047.05
8	ATLAS S.A.	LUILA, JOAO	PROGRAMADOR	3492.88
9	ATLAS S.A.	LONAS, EDUARDO	PROGRAMADOR	3483.55
10	ATLAS S.A.	SONTAS, MADALENA	PROGRAMADOR	3471.46

```
proc sql number outobs=10;
select empresa, nome, funcao, salario, salario*1.15
from arq.cadastro
where lowercase(empresa) contains "atlas"
order by funcao,4 desc;
quit;
```

The SAS System

1

Row	empresa	nome	funcao	salario
1	ATLAS S.A.	MOUA, LAURA	ANALISTA	7402.21 8512.542
2	ATLAS S.A.	MOUA, CARLA	ANALISTA	7221.58 8304.817
3	ATLAS S.A.	MOUA, TANIA	ANALISTA	5418.33 6231.08
4	ATLAS S.A.	MOUA, LIGIA	ANALISTA	5047.05 5804.108
5	ATLAS S.A.	MOUA, LUIS	GERENTE	17414.14 20026.26
6	ATLAS S.A.	MOUA, LICIA	GERENTE	17260.69 19849.79
7	ATLAS S.A.	MOUA, MARCIO	GERENTE	10828.7 12453.01
8	ATLAS S.A.	LUILA, JOAO	PROGRAMADOR	3492.88 4016.812
9	ATLAS S.A.	LONAS, EDUARDO	PROGRAMADOR	3483.55 4006.083
10	ATLAS S.A.	SONTAS, MADALENA	PROGRAMADOR	3471.46 3992.179

1.2.3 – Subcomando GROUP BY

- Comando que agrupa valores de uma ou mais colunas de acordo com uma **função de sumarização**, dentro do comando de grupo SELECT-FROM;
- O comando GROUP BY **só funciona** se for utilizada uma **função de sumarização** em alguma coluna especificada no comando SELECT;
- O comando deve ser posicionado após o comando FROM ou após o comando WHERE;
- As funções de sumarização podem ser utilizadas independente do uso do comando GROUP BY;
- Algumas funções de sumarização:

AVG, MEAN	Calcula a média de valores de uma coluna de dados numérica;
COUNT, N, FREQ	Conta valores, excluindo os valores “missing”;
MAX	Maior valor;
MIN	Menor valor;
NMISS	Conta valores “missing”;
STD	Calcula do desvio padrão;
SUM	Calcula o somatório de valores de uma coluna de dados numérica;
VAR	Calcula a variância de valores de uma coluna de dados numérica;

- As funções de sumarização funcionam de duas formas:

1 – Sumarizam valores **entre colunas**, atuando para cada linha de dados: Ação Horizontal;

2 – Sumarizam valores em uma **única coluna**, atuando para todas as linhas de dados: Ação Vertical;

GROUP BY <coluna1> , <coluna2> , . . . , <colunan> ;

coluna1...n Variáveis dos arquivos especificados no comando FROM ou a posição numérica da coluna no comando SELECT;

Observações:

1. O uso de uma função de sumarização na ação vertical, **na única coluna de dados** especificada no comando SELECT, determina um único valor como resultado da “query”;
2. O uso de uma função de sumarização na ação vertical, **em uma das colunas de dados** especificadas no comando SELECT, determina **duas execuções da “query”** sobre o arquivo especificado: Uma para **efetuar a sumarização** e a segunda para **combinar o valor resultante da sumarização** com os dados das outras colunas do comando SELECT;
3. O comando GROUP BY identifica a coluna de dados que possibilita criar grupos, permitindo valores sumarizados para cada grupo.

Ex.10 – Sumarização de dados

```

/* Ação Horizontal */
proc sql;
  select nome label="Nome" format=$15.,
         empresa label="Empresa",
         funcao label="Função",
         salario label="Salário" format=comma10.2,
         salfam label="Salário Família" format=comma10.2,
         salanos label="Salário por Tempo" format=comma10.2,
         salcom label="Salário Comissionado" format=comma10.2,
         sum(salario,salfam,salanos,salcom) as sal_total label="Salário Total" format=comma10.2
  from arq.valores
  where funcao in ("DIRETOR","GERENTE");
quit;

```

The SAS System

Nome	Empresa	Função	Salário	Salário Família	Salário por Tempo	Salário Comissionado	Salário Total
MOUA, MARIA	PARIS INSTITUTO	GERENTE	19.303,66	772,15	22.199,21	1.930,37	44.205,38
MOUA, RENATO	MALTA LTDA	GERENTE	20.457,36	409,15	23.525,96	2.045,74	46.438,21
MOUA, MIRIAM	MALTA LTDA	GERENTE	13.075,91	392,28	15.037,30	1.307,59	29.813,07
MOUA, ROSANE	MALTA LTDA	GERENTE	20.336,22	406,72	24.403,46	2.033,62	47.180,03
MOUA, MARCELO	MALTA LTDA	DIRETOR	25.377,28	507,55	55.830,02	5.075,46	86.790,30
MOUA, JOAO	MALTA LTDA	GERENTE	20.611,56	412,23	23.703,29	2.061,16	46.788,24
MOUA, MARCIO	ATLAS S.A.	GERENTE	10.828,70	108,29	12.453,01	1.082,87	24.472,86
MOUA, LUIS	ATLAS S.A.	GERENTE	17.414,14	174,14	20.896,97	1.741,41	40.226,66
MOUA, LICIA	ATLAS S.A.	GERENTE	17.260,69	172,61	20.712,83	1.726,07	39.872,19

```

/* Ação Vertical */
proc sql;
  select sum(salario) as sal_total label="Salário Total" format=comma10.2
  from arq.valores
  where funcao in ("DIRETOR","GERENTE");
quit;

```

The SAS System

```

          Salário
          Total
          -----
          164.665,52

```

```

/* Ação Vertical com duas execuções da "query"*/
proc sql;
  select empresa, sum(salario) as sal_total label="Salário Total" format=comma10.2
  from arq.valores
  where funcao in ("DIRETOR","GERENTE");
quit;

```

SAS Log

```

173 proc sql;
174   select empresa, sum(salario) as sal_total label="Salário Total" format=comma10.2
175   from arq.valores
176   where funcao in ("DIRETOR","GERENTE");
NOTE: The query requires remerging summary statistics back with the original data.
177 quit;
NOTE: PROCEDURE SQL used (Total process time):
      real time          0.01 seconds
      cpu time            0.00 seconds

```

The SAS System

empresa	Salário Total
PARIS INSTITUTO	164.665,52
MALTA LTDA	164.665,52
MALTA LTDA	164.665,52
MALTA LTDA	164.665,52
MALTA LTDA	164.665,52
MALTA LTDA	164.665,52
ATLAS S.A.	164.665,52
ATLAS S.A.	164.665,52
ATLAS S.A.	164.665,52

Ex.11 – Agrupando dados

```
proc sql;
  select empresa,
         count(*) as Num label="Número de Funcionários",
         sum(salario) as sal_total label="Gastos com Salário" format=comma10.2,
         avg(salario) as med_total label="Média Salarial" format=comma10.2
  from arq.valores
  where funcao ne "DESEMPREGADO"
  group by empresa;
quit;
```

The SAS System

empresa	Número de Funcionários	Gastos com Salário	Média Salarial
ATLAS S.A.	97	297.237,99	3.064,31
MALTA LTDA	174	671.338,26	3.858,27
PARIS INSTITUTO	163	477.604,28	2.930,09

```
proc sql;
  select empresa, funcao,
         count(*) as Num label="Número de Funcionários",
         sum(salario) as sal_total label="Gastos com Salário" format=comma10.2,
         avg(salario) as med_total label="Média Salarial" format=comma10.2
  from arq.valores
  where funcao ne "DESEMPREGADO"
  group by empresa, funcao;
quit;
```

The SAS System

empresa	funcao	Número de Funcionários	Gastos com Salário	Média Salarial
ATLAS S.A.	ANALISTA	4	25.089,17	6.272,29
ATLAS S.A.	GERENTE	3	45.503,53	15.167,84
ATLAS S.A.	PROGRAMADOR	90	226.645,29	2.518,28
MALTA LTDA	ANALISTA	4	51.072,91	12.768,23
MALTA LTDA	DIRETOR	1	25.377,28	25.377,28
MALTA LTDA	GERENTE	4	74.481,05	18.620,26
MALTA LTDA	PROGRAMADOR	165	520.407,02	3.153,98
PARIS INSTITUTO	ANALISTA	1	9.548,36	9.548,36
PARIS INSTITUTO	GERENTE	1	19.303,66	19.303,66
PARIS INSTITUTO	PROGRAMADOR	161	448.752,26	2.787,28

```
proc sql;
  select empresa,
         int((today()-admissao)/365.25) label="Tempo de Empresa",
         count(*) as Num label="Número de Funcionários",
         avg(salario) as med_total label="Média Salarial" format=comma10.2
  from arq.valores
  where funcao ne "DESEMPREGADO"
  group by empresa,2;
quit;
```

The SAS System

empresa	Tempo de Empresa	Número de Funcionários	Média Salarial
ATLAS S.A.	6	27	2.623,51
ATLAS S.A.	7	63	2.473,18
ATLAS S.A.	13	2	7.311,90
ATLAS S.A.	14	2	5.232,69
ATLAS S.A.	23	1	10.828,70
ATLAS S.A.	24	2	17.337,42
MALTA LTDA	6	60	3.124,20
MALTA LTDA	7	105	3.171,00
MALTA LTDA	13	1	14.821,37
MALTA LTDA	14	3	12.083,85
MALTA LTDA	23	3	18.048,28
MALTA LTDA	24	1	20.336,22
MALTA LTDA	44	1	25.377,28
PARIS INSTITUTO	6	57	2.776,30
PARIS INSTITUTO	7	104	2.793,30
PARIS INSTITUTO	14	1	9.548,36
PARIS INSTITUTO	23	1	19.303,66

1.2.5 – Subcomando HAVING

- Comando que seleciona **grupos de dados** de acordo com uma expressão lógica que envolva uma sumarização de dados, dentro do comando de grupo SELECT-FROM;
- O comando HAVING **só funciona** se for utilizada uma **função de sumarização** em alguma coluna especificada no comando SELECT ou na própria expressão lógica do comando;
- O comando deve ser posicionado após o comando FROM ou GROUP BY;
- **ATENÇÃO!** o comando **WHERE** seleciona **linhas de dados** do arquivo; o comando **HAVING** seleciona **grupos de dados** da sumarização de uma coluna de dados;

HAVING <expressão>

expressão Expressão lógica que envolva uma coluna de dados sumarizada do comando SELECT, ou uma expressão lógica que utiliza uma função de sumarização;

Ex.12 – Selecionando grupo de dados

```
proc sql number;
  select empresa, funcao,
         count(*) as Num label="Número de Funcionários",
         sum(salario) as sal_total label="Gastos com Salário" format=commmax10.2,
         avg(salario) as med_total label="Média Salarial" format=commmax10.2
  from arq.valores
  where funcao ne "DESEMPREGADO"
  group by empresa, funcao
  having med_total > 15000;
quit;
```

The SAS System

Row	empresa	funcao	Número de Funcionários	Gastos com Salário	Média Salarial
1	ATLAS S.A.	GERENTE	3	45.503,53	15.167,84
2	MALTA LTDA	DIRETOR	1	25.377,28	25.377,28
3	MALTA LTDA	GERENTE	4	74.481,05	18.620,26
4	PARIS INSTITUTO	GERENTE	1	19.303,66	19.303,66

```
title "Empresas com Idade Média acima de 50 anos";
proc sql number;
  select empresa, funcao, sexo
  from arq.cadastro
  group by empresa, funcao, sexo
  having mean(idade) gt 50;
quit;
```

Empresas com Idade Média acima de 50 anos

Row	empresa	funcao	sexo
1	ATLAS S.A.	GERENTE	F
2	ATLAS S.A.	GERENTE	M
3	MALTA LTDA	DIRETOR	M
4	MALTA LTDA	GERENTE	F
5	MALTA LTDA	GERENTE	M
6	PARIS INSTITUTO	ANALISTA	M
7	PARIS INSTITUTO	GERENTE	F

1.3 – Parâmetro CALCULATED

- Toda coluna nova criada no comando SELECT, à princípio, NÃO pode ser utilizada novamente, no mesmo comando SELECT, no comando WHERE ou no comando GROUP BY;
- Utiliza-se o parâmetro CALCULATED, antes da nova coluna, para indicar que a coluna já foi calculada.

Ex.13 – Utilização de novas colunas

```
proc sql number;
  select nome,
         empresa,
         funcao,
         salario,
         salario*1.15 as novo_sal,
         novo_sal*0.1 as taxa
  from arq.cadastro
  where taxa > 1000;
quit;
```

SAS Log

```
633 options nonumber;
634 title "The SAS System";
635 proc sql number;
636     select nome,
637            empresa,
638            funcao,
639            salario,
640            salario*1.15 as novo_sal,
641            novo_sal*0.1 as taxa
642     from arq.cadastro
643     where taxa > 1000;
ERROR: The following columns were not found in the contributing tables: novo_sal, taxa.
644 quit;
NOTE: The SAS System stopped processing this step because of errors.
NOTE: PROCEDURE SQL used (Total process time):
      real time           0.00 seconds
      cpu time             0.00 seconds
```

```
proc sql number;
  select nome,
         empresa,
         funcao,
         salario,
         salario*1.15 as novo_sal,
         calculated novo_sal*0.1 as taxa
  from arq.cadastro
  where calculated taxa > 1000;
quit;
```

The SAS System

Row	nome	empresa	funcao	salario	novo_sal	taxa
1	MOUA, PAULO	PARIS INSTITUTO	ANALISTA	9548.36	10980.61	1098.061
2	MOUA, MARIA	PARIS INSTITUTO	GERENTE	19303.66	22199.21	2219.921
3	MOUA, RENATO	MALTA LTDA	GERENTE	20457.36	23525.96	2352.596
4	MOUA, MARCO	MALTA LTDA	ANALISTA	9988.53	11486.81	1148.681
5	MOUA, MONICA	MALTA LTDA	ANALISTA	12568.82	14454.14	1445.414
6	MOUA, MADALENA	MALTA LTDA	ANALISTA	14821.37	17044.58	1704.458
7	MOUA, ELIANE	MALTA LTDA	ANALISTA	13694.19	15748.32	1574.832
8	MOUA, MIRIAM	MALTA LTDA	GERENTE	13075.91	15037.3	1503.73
9	MOUA, ROSANE	MALTA LTDA	GERENTE	20336.22	23386.65	2338.665
10	MOUA, MARCELO	MALTA LTDA	DIRETOR	25377.28	29183.87	2918.387
11	MOUA, JOAO	MALTA LTDA	GERENTE	20611.56	23703.29	2370.329
12	MOUA, MARCIO	ATLAS S.A.	GERENTE	10828.7	12453.01	1245.301
13	MOUA, LUIS	ATLAS S.A.	GERENTE	17414.14	20026.26	2002.626
14	MOUA, LICIA	ATLAS S.A.	GERENTE	17260.69	19849.79	1984.979

1.4 – Comando de Grupo CREATE TABLE – AS

- Comando de grupo que cria uma tabela com estrutura SAS a partir de um SELECT-FROM, ou seja, a partir de uma seleção de dados ou “query”;
- É sempre posicionado no início da “query”;
- Uma “query” com o comando CREATE TABLE – AS, **NÃO** gera relatório;
- As colunas sem nome, no comando SELECT, são automaticamente nomeadas quando se usa o comando CREATE TABLE;

```
CREATE TABLE <tabela> AS  
  <"query">
```

tabela Nome de uma tabela seguindo as regras de nomenclatura do SAS e, se possível, indicando a biblioteca de armazenamento;

“query” Uma estrutura de comandos SELECT-FROM;

Ex.14 – Salvando dados de uma “query”

```
options pageno=1 nodate;  
proc sql number;  
  create table arq.atlas as  
  select empresa, nome, funcao, salario  
  from arq.cadastro  
  where lowercase(empresa) contains "atlas";  
quit;
```

SAS Log

```
15 options pageno=1 nodate;  
16 proc sql number;  
17   create table arq.atlas as  
18     select empresa, nome, funcao, salario  
19     from arq.cadastro  
20     where lowercase(empresa) contains "atlas";  
NOTE: Table ARQ.ATLAS created, with 97 rows and 4 columns.  
21 quit;  
NOTE: PROCEDURE SQL used (Total process time):  
  real time          0.03 seconds  
  cpu time           0.00 seconds
```

```
options pageno=1 nodate;  
proc sql number outobs=10;  
  create table arq.atlas as  
  select empresa, substr(empresa,index(empresa," ")),nome,funcao,salario,salario*1.1  
  from arq.cadastro  
  where lowercase(empresa) contains "atlas";  
  
  select *  
  from arq.atlas;  
quit;
```

The SAS System

1

Row	empresa	TEMA001	nome	funcao	salario	TEMA002
1	ATLAS S.A.	S.A.	PISCO,ROSANE	PROGRAMADOR	3045.33	3349.863
2	ATLAS S.A.	S.A.	ANJOA, PAULO	PROGRAMADOR	3007.47	3308.217
3	ATLAS S.A.	S.A.	SILVA, MARCIO	PROGRAMADOR	3069.12	3376.032
4	ATLAS S.A.	S.A.	SUNAY, MONICA	PROGRAMADOR	3445.2	3789.72
5	ATLAS S.A.	S.A.	YATAKA, MADALENA	PROGRAMADOR	3057.14	3362.854
6	ATLAS S.A.	S.A.	MENDES, PAULO	PROGRAMADOR	3213.9	3535.29
7	ATLAS S.A.	S.A.	MOUA, MARCIO	GERENTE	10828.7	11911.57
8	ATLAS S.A.	S.A.	LUILA, LUIS	PROGRAMADOR	3098.58	3408.438
9	ATLAS S.A.	S.A.	MARQUES, LAURA	PROGRAMADOR	3179.68	3497.648
10	ATLAS S.A.	S.A.	SANTOS, ROSANE	PROGRAMADOR	3108.51	3419.361

Ex.15 – Salvando os dados e gerando relatório

```
proc sql number;
  create table arq.atlas as
  select empresa, nome, funcao, salario
  from arq.cadastro
  where lowercase(empresa) contains "atlas";

  select *
  from arq.atlas (obs=10);
quit;
```

SAS Log

```
32 options pageno=1 nodate;
33 proc sql number;
34   create table arq.atlas as
35   select empresa, nome, funcao, salario
36   from arq.cadastro
37   where lowercase(empresa) contains "atlas";
NOTE: Table ARQ.ATLAS created, with 97 rows and 4 columns.
38
39   select *
40   from arq.atlas (obs=10);
41 quit;
NOTE: PROCEDURE SQL used (Total process time):
      real time          0.03 seconds
      cpu time           0.01 seconds
```

The SAS System

1

Row	empresa	nome	funcao	salario
1	ATLAS S.A.	PISCO,ROSANE	PROGRAMADOR	3045.33
2	ATLAS S.A.	ANJOA, PAULO	PROGRAMADOR	3007.47
3	ATLAS S.A.	SILVA,MARCIO	PROGRAMADOR	3069.12
4	ATLAS S.A.	SUNAY,MONICA	PROGRAMADOR	3445.2
5	ATLAS S.A.	YATAKA,MADALENA	PROGRAMADOR	3057.14
6	ATLAS S.A.	MENDES, PAULO	PROGRAMADOR	3213.9
7	ATLAS S.A.	MOUA,MARCIO	GERENTE	10828.7
8	ATLAS S.A.	LUILA, LUIS	PROGRAMADOR	3098.58
9	ATLAS S.A.	MARQUES, LAURA	PROGRAMADOR	3179.68
10	ATLAS S.A.	SANTOS,ROSANE	PROGRAMADOR	3108.51

Ex.16 – Exemplo completo

```
options ls=120 pageno=1;
title "Relatório de Salários por Empresa e Função";
proc sql;
  create table arq.salarios as
  select nome label="Nome" format=$15.,
         empresa label="Empresa",
         funcao label="Função",
         salario label="Salário" format=commmax10.2,
         salfam label="Salário Família" format=commmax10.2,
         salanos label="Salário por Tempo" format=commmax10.2,
         salcom label="Salário Comissionado" format=commmax10.2,
         sum(salario,salfam,salanos,salcom) as sal_total label="Salário Total" format=commmax10.2,
         avg(calculated sal_total) as med label="Salário Médio" format=commmax10.2
  from arq.valores
  where funcao in ("DIRETOR","GERENTE")
  group by empresa,funcao
  having med > 35000
  order by empresa,funcao;

  select *
  from arq.salarios;
quit;
```

Relatório de Salários por Empresa e Função 1

Nome	Empresa	Função	Salário	Salário Família	Salário por Tempo	Salário Comissionado	Salário Total	Salário Médio
MOUA, MARCELO	MALTA LTDA	DIRETOR	25.377,28	507,55	55.830,02	5.075,46	86.790,30	86.790,30
MOUA, MIRIAM	MALTA LTDA	GERENTE	13.075,91	392,28	15.037,30	1.307,59	29.813,07	42.554,89
MOUA, JOAO	MALTA LTDA	GERENTE	20.611,56	412,23	23.703,29	2.061,16	46.788,24	42.554,89
MOUA, RENATO	MALTA LTDA	GERENTE	20.457,36	409,15	23.525,96	2.045,74	46.438,21	42.554,89
MOUA, ROSANE	MALTA LTDA	GERENTE	20.336,22	406,72	24.403,46	2.033,62	47.180,03	42.554,89
MOUA, MARIA	PARIS INSTITUTO	GERENTE	19.303,66	772,15	22.199,21	1.930,37	44.205,38	44.205,38

```
proc sql;
  create table arq.salarios as
  select nome label="Nome" format=$15.,
         empresa label="Empresa",
         funcao label="Função",
         salario label="Salário" format=commmax10.2,
         salfam label="Salário Família" format=commmax10.2,
         salanos label="Salário por Tempo" format=commmax10.2,
         salcom label="Salário Comissionado" format=commmax10.2,
         sum(salario,salfam,salanos,salcom) as sal_total label="Salário Total" format=commmax10.2
  from arq.valores
  where funcao in ("DIRETOR","GERENTE");

  select empresa,
         funcao,
         avg(sal_total) as med label="Salário Médio" format=commmax10.2
  from arq.salarios
  group by empresa,funcao
  having med > 40000
  order by empresa,funcao;
quit;
```

Relatório de Salários por Empresa e Função 1

Empresa	Função	Salário Médio
MALTA LTDA	DIRETOR	86.790,30
MALTA LTDA	GERENTE	42.554,89
PARIS INSTITUTO	GERENTE	44.205,38

1.5 – “Subquery: IN-LINE VIEW”

- “Subquery” é uma “query” dentro de outra, ou seja, o resultado de uma “query” será utilizada para processar outra “query”;
- Uma “subquery” montada no comando **FROM**, recebe o nome especial de “**IN-LINE VIEW**”, e deve vir entre parêntesis;
- É uma tabela temporária em memória.
- Não é possível utilizar ORDER BY em “IN-LINE VIEW”;

```
PROC SQL;  
    SELECT ...  
        FROM ( SELECT ...  
                FROM ... ) ;  
QUIT;
```

Ex.17 – Subquery 1 (“IN-LINE VIEW”)

```
proc sql;
  select empresa, funcao, subtotal/sum(subtotal) format=percent8.2
  from ( select empresa, funcao, sum(salario) as subtotal format=comma12.2
        from arq.cadastro
        group by empresa, funcao )
  where funcao ne "DESEMPREGADO"
  group by empresa
  order by empresa;
quit;
```

Resultado da "Subquery"

```
(select empresa, funcao, sum(salario) as subtotal format=comma12.2
 from arq.cadastro
 group by empresa, funcao )
```

The SAS System

empresa	funcao	subtotal
	DESEMPREGADO	.
ATLAS S.A.	ANALISTA	25.089,17
ATLAS S.A.	GERENTE	45.503,53
ATLAS S.A.	PROGRAMADOR	226.645,29
MALTA LTDA	ANALISTA	51.072,91
MALTA LTDA	DIRETOR	25.377,28
MALTA LTDA	GERENTE	74.481,05
MALTA LTDA	PROGRAMADOR	520.407,02
PARIS INSTITUTO	ANALISTA	9.548,36
PARIS INSTITUTO	GERENTE	19.303,66
PARIS INSTITUTO	PROGRAMADOR	448.752,26

Resultado Final da "Query"

The SAS System

empresa	funcao	
ATLAS S.A.	ANALISTA	8.44%
ATLAS S.A.	GERENTE	15.31%
ATLAS S.A.	PROGRAMADOR	76.25%
MALTA LTDA	ANALISTA	7.61%
MALTA LTDA	DIRETOR	3.78%
MALTA LTDA	GERENTE	11.09%
MALTA LTDA	PROGRAMADOR	77.52%
PARIS INSTITUTO	ANALISTA	2.00%
PARIS INSTITUTO	GERENTE	4.04%
PARIS INSTITUTO	PROGRAMADOR	93.96%

1º Laboratório – SQL

Este treinamento irá trabalhar com uma base de dados de informações sobre o processamento da execução de programas em filas, do ambiente do CENAPAD-SP. Essas informações são importantes para identificar problemas no nosso ambiente e gerar relatórios com estatísticas de uso do ambiente. O nome da tabela é “**analise**” e está localizada na pasta **c:\curso\sas2** e possui a seguinte estrutura:

N	variavel	Tipo	Tamanho	Formato	Label
1	id	Caracter	20		Identificação do Job
2	usuario	Caracter	12		Nome do Usuário
3	inst	Caracter	20		Instituição
4	estado	Caracter	2		Estado
5	jobname	Caracter	20		Nome do Job
6	queue	Caracter	12		Nome da Fila
7	data_submissao	Numérico	8	DDMMYY10.	Data de Submissão
8	data_inicio	Numérico	8	DDMMYY10.	Data de Início da Execução
9	data_fim	Numérico	8	DDMMYY10.	Data de Finalização da Execução
10	ncpus	Numérico	8		Quantidade de CPUs utilizadas
11	tesp	Numérico	8	TIME20.	Tempo de Espera na Fila

1. Criar uma relatório que selecione todos os usuários do ambiente CENAPAD que finalizaram programas, na fila “paralela”, no período de 01/Janeiro/2006 a 31/Março/2006;

Colunas : usuario, data_fim, queue
Comandos : SELECT-FROM, DISTINCT, WHERE
Dica : Montar a expressão do WHERE com constantes data do SAS – “ddmmyy”d Ex. “23apr2012”d

2. Verifique se existem nomes de “jobs” com a seguinte e única formação de caracteres:

job “*quaisquer caracteres*”•”2 caracteres quaisquer”•”quaisquer caracteres”

Colunas : jobname
Comandos : SELECT-FROM, DISTINCT, WHERE
Dica : Utilize o operador LIKE na construção do comando WHERE

3. Quais foram os usuários que utilizaram mais de 30 dias de processamento;

Colunas : usuario, data_inicio, data_fim
Comandos : SELECT-FROM, CALCULATED, WHERE
Dica : Será necessário montar um expressão aritmética com a diferença entre as datas

4. Melhorando a questão 3; quantos jobs, por usuário, gastaram mais de 30 dias de processamento?

Comando : GROUP BY
Dica : Será necessário utilizar uma função de sumarização para possibilitar o agrupamento, no caso, a função de frequência, para contar o número jobs por usuário

5. Melhorando a questão 4; gere o relatório por ordem descendente do número de vezes;

Colunas : usuario, data_inicio, data_fim
Comandos : SELECT-FROM, WHERE, GROUP BY, ORDER BY
Dica : Será necessário utilizar uma função de frequência para contar o número jobs por usuário, e montar um expressão aritmética com a diferença entre as datas

6. Criar um relatório **agrupado** por estado e por instituição, com o número **distintos** de usuários dentro desses grupos, e em ordem descendente do número de usuários.

Colunas : usuario, estado, inst
Comandos : SELECT-FROM, DISTINCT, GROUP BY, ORDER BY
Dica : Será necessário utilizar uma função de sumarização para calcular a frequência do número de usuários

7. Em um único SQL, responda as perguntas abaixo. Inclua o texto das perguntas no relatório.

Quantos usuários já utilizaram o CENAPAD-SP ?
Quantas instituições já utilizaram o CENAPAD-SP ?
Quantos estados já utilizaram o CENAPAD-SP ?
Qual é a instituição com maior número de Usuários no CENAPAD-SP ?

8. Selecione as instituições que iniciaram mais de 5000 “jobs” no ambiente CENAPAD.

Colunas : inst
Comandos : SELECT-FROM, GROUP BY, HAVING
Dica : Será necessário utilizar uma função de sumarização para calcular a frequência do número de “jobs”

9. **Primeira parte:** Crie uma tabela SAS que armazene o estado, a instituição, a fila de processamento e o número de processadores (CPUs) utilizados, para todos os “jobs” que tenham sido finalizados em 2009 e que tenham utilizado mais de 1 processador (CPU);

Colunas : estado, inst, queue, ncpus, data_fim
Comandos : CREATE TABLE, SELECT-FROM, WHERE
Dica : Será necessário utilizar uma função de data para extrair o ano da data_fim

Segunda parte: Crie um relatório com a tabela criada na primeira parte, com o número de jobs executados, em ordem descendente, e agrupados por estado, instituição e fila;

Colunas : estado, inst, queue, ncpus, data_fim
Comandos : SELECT-FROM, GROUP BY, ORDER BY
Dica : Será necessário utilizar uma função de sumarização para calcular a frequência do número de “jobs”

10. Modifique o exercício 9; crie uma subquery “in-line view”, aonde a tabela criada na primeira parte passe a ser uma subquery da segunda parte.

11. **Primeira parte:** Crie uma tabela nova a partir da tabela “**analise**” com os dados de fila, ano do início da execução do “job” e a média do tempo de espera, agrupados por fila e ano de início e organizados por fila e ano de início. Gere um relatório com os dados desse novo arquivo.

Colunas : queue, data_inicio, tesp
Comandos : CREATE TABLE, SELECT-FROM, GROUP BY, ORDER BY
Dica : Será necessário utilizar uma função de sumarização para calcular a média do tempo de espera

Segunda parte: Crie um relatório com o resultado da combinação dessa nova tabela, com a tabela “**analise**”, pela fila e ano de início, e conte quantos “jobs”, agrupados por fila e ano, ficaram acima da média do tempo de espera.

Colunas : queue, data_inicio, média do tempo de espera calculada na primeira parte
Comandos : SELECT-FROM, WHERE, GROUP BY
Dica : Será necessário utilizar uma função de sumarização para calcular a frequência do número de “jobs”.

2 – Processamento MACRO

Macro é um recurso de processamento de texto, e quando identificado dentro de um programa, é substituído por algum valor texto que tenha sido previamente definido. Em SAS, dois caracteres especiais identificam e inicializam o processamento macro, são chamados de “**macro triggers**”:

- &** Identifica uma variável de macro de referência ou variável simbólica;
- %** Identifica uma rotina macro, um comando macro ou uma função macro;

Todo o processamento macro ocorre antes da compilação de um programa SAS, ou seja, quando um programa SAS é executado, tudo que é referência de macro, já foi resolvido e substituído por algum valor texto.

O processamento com macros facilita a codificação de programas SAS nas seguintes situações:

- **Substituição de valor**

```
proc print data=sas2.cadastro;  
  where admin>"01jan1990"d and empresa="ATLAS S.A.";  
  title "Funcionários da Empresa ATLAS S.A a partir de 1990";  
  footnote "Fonte: arquivo Cadastro";  
run;
```

- **Execução condicional de steps**

Execução Diária

```
proc append base=sas2.cadastro data=dia_de_hoje;  
run;
```

Execução só no Domingo ???

```
proc sort data=sas2.cadastro out=sas2.cadastro_ordenado;  
  by cpf;  
run;
```

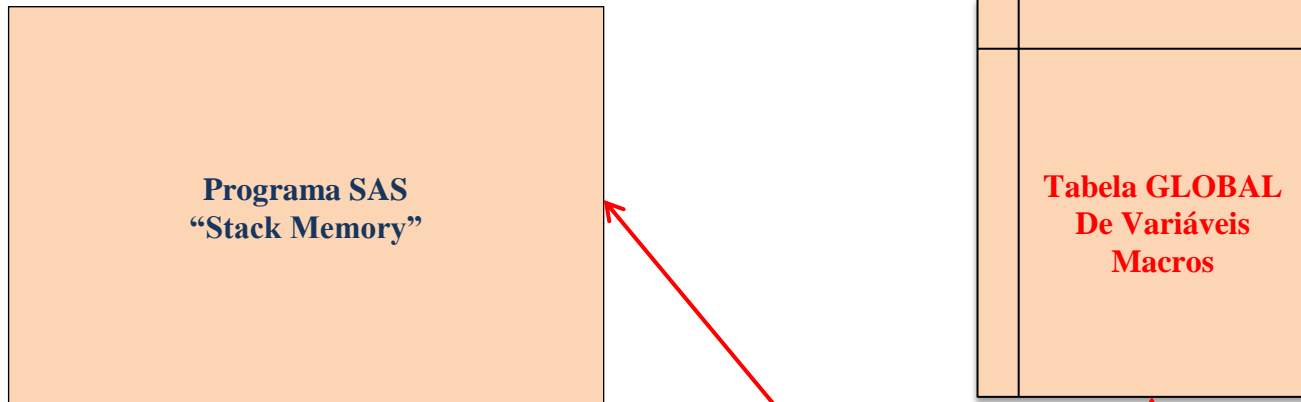
- **Execução repetitiva**

```
proc print data=sas2.cadastro;  
  where year(admissao)=1990;  
run;  
proc print data=sas2.cadastro;  
  where year(admissao)=1991;  
run;  
proc print data=sas2.cadastro;  
  where year(admissao)=1992;  
run;  
proc print data=sas2.cadastro;  
  where year(admissao)=1993;  
run;  
...  
...  
...  
proc print data=sas2.cadastro;  
  where year(admissao)=2010;  
run;
```

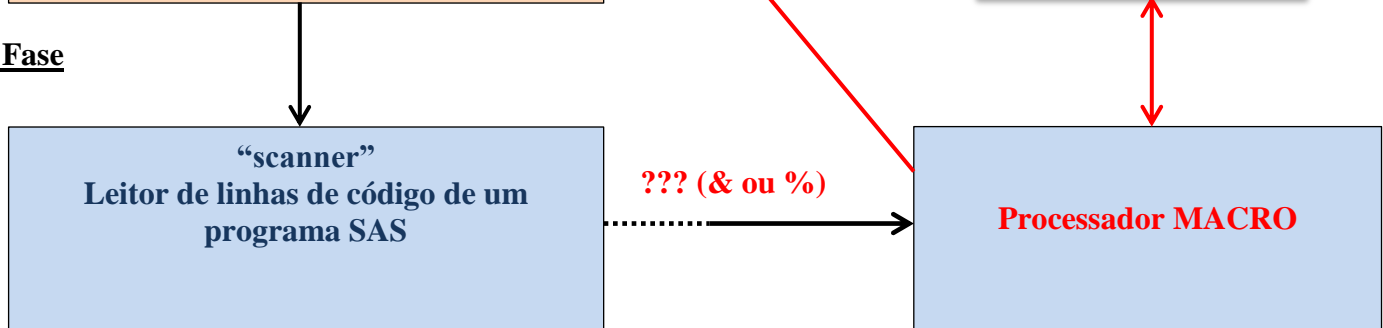
Memória do computador

“Em uma sessão SAS Windows ou Enterprise Guide”

1ª Fase



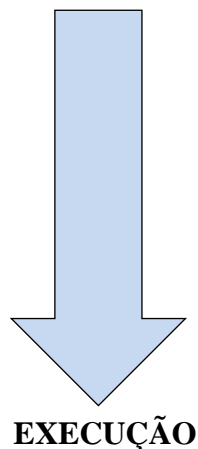
2ª Fase



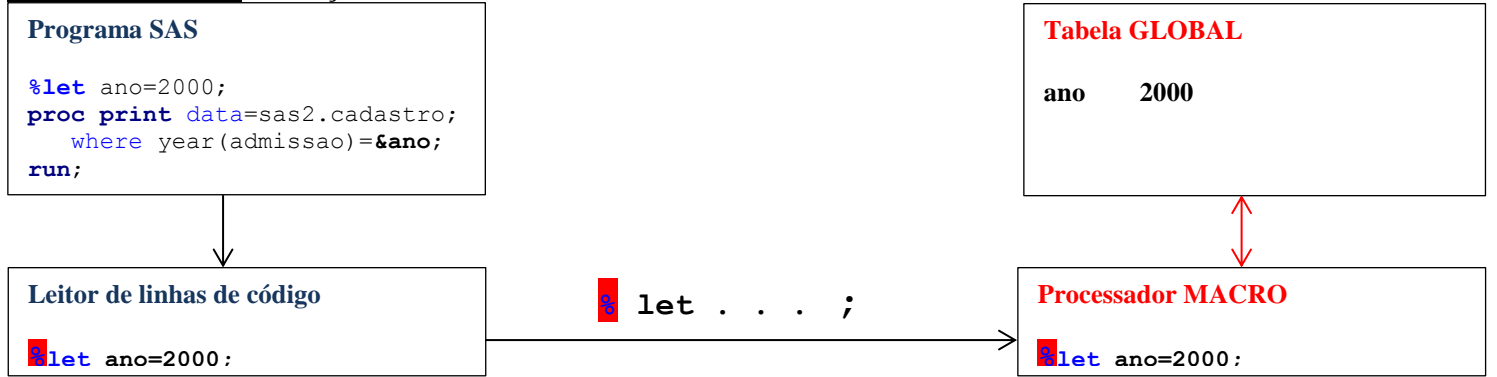
3ª Fase



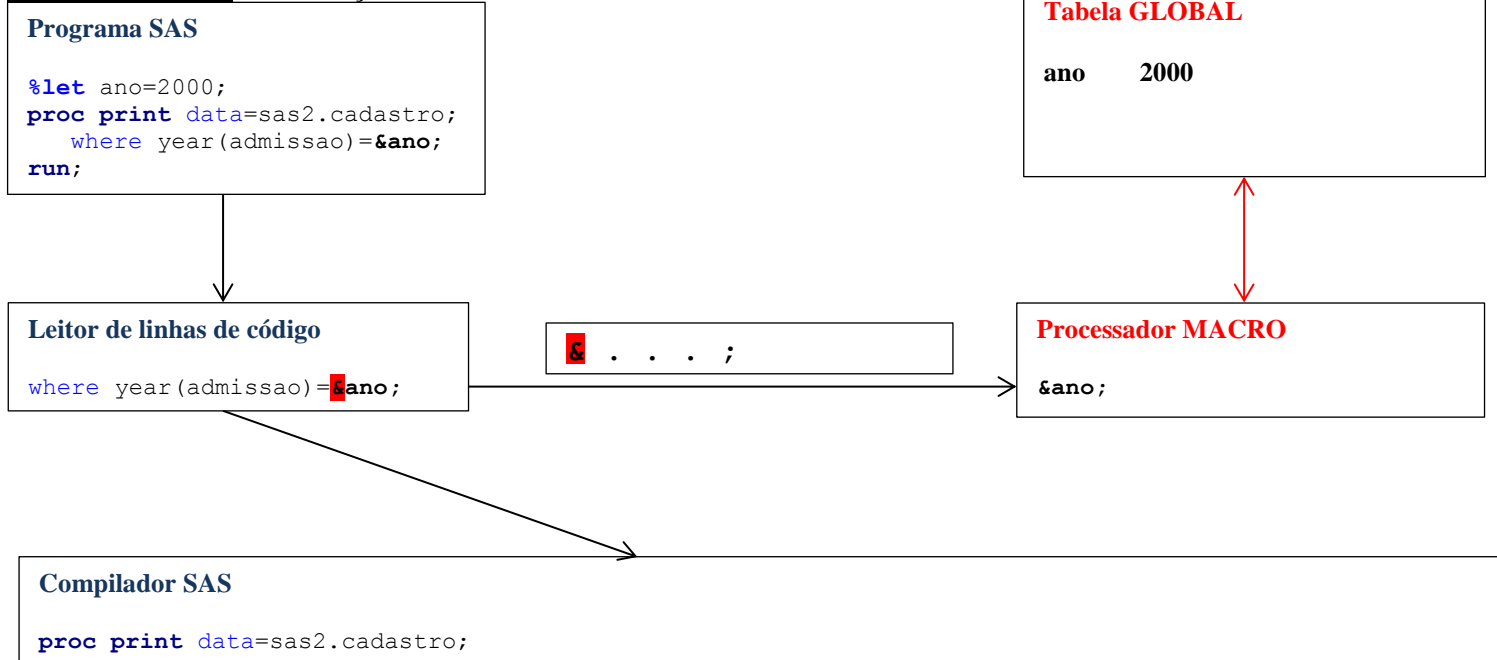
4ª Fase



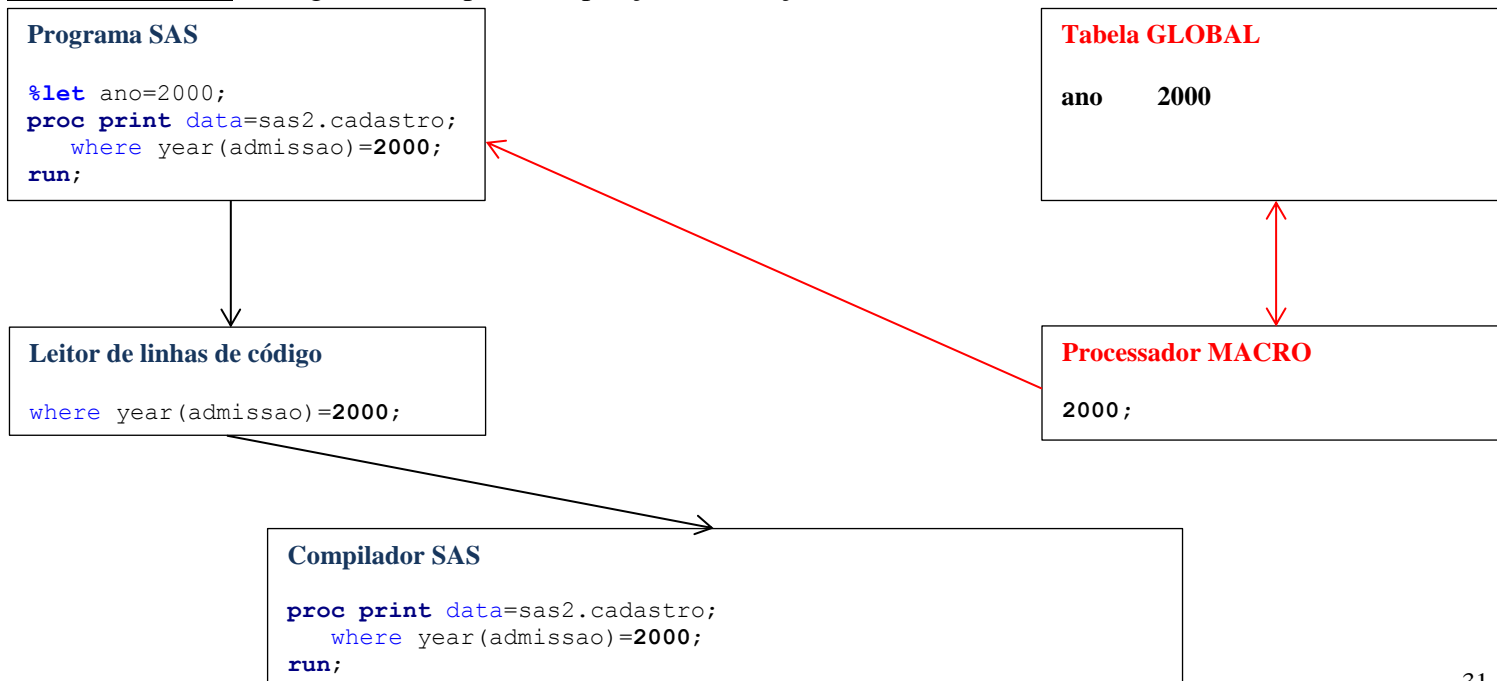
Exemplo parte 1 – Criação da variável macro *ano*



Exemplo parte 2 – Utilização da variável macro *ano*



Exemplo parte 3 – Programa final para compilação e execução



2.1 – Tabela Global

Toda vez que se abre uma sessão SAS ou executa um programa SAS, uma tabela global de variáveis de macro é criada, e algumas [variáveis de macro são definidas automaticamente pelo SAS](#). As variáveis de macro definidas na tabela global estão sempre disponíveis para qualquer execução de um programa SAS.

Variáveis de Macro Automáticas

Nome	Descrição
SYSDATE	Data do sistema na abertura da sessão SAS. Padrão DATE7.
SYSDATE9	Data do sistema na abertura da sessão SAS. Padrão DATE9.
SYSDAY	Dia da semana na abertura da sessão SAS.
SYSTIME	Hora do sistema na abertura da sessão SAS. Padrão TIME5.
SYSHOSTNAME	Nome da máquina de execução do SAS.
SYSUSERID	Nome do usuário que executa o SAS.
SYSLAST	Nome do último arquivo criado em uma execução SAS.

2.2 – Variável de Macro

- Toda variável de macro inicia com o caractere especial “&” seguido por um nome padrão SAS;

¯o ;

- As variáveis de macro, também são chamadas de referências simbólicas;
- Podem aparecer em qualquer lugar de um programa SAS;
- Normalmente, são armazenadas na tabela global de referência;
- São enviadas e resolvidas pelo processador Macro;
- Os valores de uma variável de macro são sempre do tipo caractere; são considerados como texto;
- Variáveis de macro entre **aspas duplas (“”)** são resolvidas pelo processador macro;
- Variáveis de macro entre **aspas simples ou apostrofes (‘)** são consideradas como texto e não são resolvidas pelo processador Macro;
- Se o processador Macro não conseguir resolver uma variável de macro, é enviada uma mensagem de WARNING para o log do SAS.

O comando: TITLE "A execução do programa SAS ocorreu as &systime";

Compilado como: TITLE "A execução do programa SAS ocorreu as 11:10";

O comando: TITLE 'A execução do programa SAS ocorreu as &systime';

Compilado como: TITLE 'A execução do programa SAS ocorreu as &systime';

2.3 – Comando %PUT

Comando de macro que pode ser colocado em qualquer lugar de um programa SAS. Serve imprimir no log do SAS valores de variáveis de macro.

%PUT [¯1 ¯2 ... ¯n] [_AUTOMATIC_] [_USER_] [_ALL_] ;

¯1...n Variáveis de macro;

AUTOMATIC Variáveis de macro definidas pelo SAS na tabela global;

USER Variáveis de macro definidas pelo usuário na tabela global;

ALL Todas as variáveis de macro definidas na tabela global;

2.4 – Comando %LET

Comando que define variável macro e seu respectivo valor.

%LET <variável>=[valor] ;

- Se a variável existir o valor será sobreposto;
- O valor é sempre texto. Números serão tratados como texto.
- Se a variável ou o valor possuírem macro triggers, estes serão resolvidos primeiros antes da definição ser feita;
- O valor pode ter um tamanho máximo de 65.534 caracteres e um tamanho mínimo de 0, valor nulo;
- Expressões matemáticas **não** são resolvidas;
- Brancos a esquerda e a direita do valor, são retirados;

```
%let valor= PapaGaio ;           → valor=PapaGaio
%let valor2="Luis Felipe";       → valor2="Luis Felipe"
%let valor3=;                    → valor3=
%let exp1=(10*1.1)/5;           → exp1=(10*1.1)/5
%let exp2=100;                  → exp2=100
%let exp3=exp1+exp2;            → exp3=exp1+exp2
%let exp4=&exp1+&exp2;           → exp4=(10*1.1)/5+100
%let &valor=e meu                → papagaio=e meu
```

Ex.18 – Definição de variável macro

```
%let ano=1986;
proc print data=sas2.cadastro;
  format admissao ddmmyy10.;
  var nome empresa funcao admissao;
  title "Relatório de pessoas admitidas em &ano";
  where year(admissao)=&ano;
run;
```

SAS Log

```
57 %let ano=1986;
58 proc print data=sas2.cadastro;
59   format admissao ddmmyy10.;
60   var nome empresa funcao admissao;
61   title "Relatório de pessoas admitidas em &ano";
62   where year(admissao)=&ano;
63 run;
```

NOTE: There were 8 observations read from the data set SAS2.CADASTRO.

WHERE YEAR(admissao)=1986;

NOTE: PROCEDURE PRINT used (Total process time):

real time 0.01 seconds
cpu time 0.01 seconds

SAS Output

Relatório de pessoas admitidas em 1986

Obs	nome	empresa	funcao	admissao
20	MOUA, JOAO	MALTA LTDA	GERENTE	10/09/1986
48	MOUA, LUIS	ATLAS S.A.	GERENTE	03/06/1986
116	MOUA, LICIA	ATLAS S.A.	GERENTE	19/01/1986
167	MOUA, MIRIAM	MALTA LTDA	GERENTE	12/09/1986
258	MOUA, MARIA	PARIS INSTITUTO	GERENTE	25/09/1986
363	MOUA, RENATO	MALTA LTDA	GERENTE	10/09/1986
396	MOUA, MARCIO	ATLAS S.A.	GERENTE	04/12/1986
454	MOUA, ROSANE	MALTA LTDA	GERENTE	21/06/1986

2.5 – Referências de Variáveis de Macro

É possível referenciar uma variável de macro em qualquer lugar do programa, mas é necessário ficar atento á situações especiais que possam ocorrer devido a posição do nome da variável macro, podendo haver confusão na solução da variável macro pelo processador Macro.

texto¯o

¯otexto } Como que o processador Macro irá identificar o fim
texto**¯o**texto } do nome da variável macro e início do texto???

¯o¯o

Todos os caracteres especiais podem funcionar como delimitadores dos nomes de variáveis de macro, mas o caractere “.”, é especial, pois indica o fim do nome de uma variável de macro e é eliminado pelo processador de Macro, não sendo repassado para o compilador SAS.

Ex.19 – Delimitador de Variável Macro

```
%let mes=01;
proc print data=sas2.cad&mes1986;
  format admissao ddmmyy10.;
  var nome empresa funcao admissao;
  title "Relatório Cadastral de Admissão de &mes de 1986";
run;
```

SAS Log

```
95 %let mes=01;
96 proc print data=sas2.cad&mes1986;
WARNING: Apparent symbolic reference MES1986 not resolved.
96 proc print data=sas2.cad&mes1986;
-
22
ERROR 22-322: Syntax error, expecting one of the following: ;, (, BLANKLINE, DATA, DOUBLE,...
96 proc print data=sas2.cad&mes1986;
-
200
ERROR 200-322: The symbol is not recognized and will be ignored.

WARNING: Apparent symbolic reference MES1986 not resolved.
ERROR: File SAS2.CAD.DATA does not exist.
97 format admissao ddmmyy10.;
98 var nome empresa funcao admissao;
99 title "Relatório Cadastral de Admissão de &mes de 1986";
100 run;
```

NOTE: The SAS System stopped processing this step because of errors.
NOTE: PROCEDURE PRINT used (Total process time):
real time 0.00 seconds
cpu time 0.00 seconds

```
%let mes=01;
proc print data=sas2.cad&mes.1986;
  format admissao ddmmyy10.;
  var nome empresa funcao admissao;
  title "Relatório Cadastral de Admissão de &mes de 1986";
run;
```

SAS Log (Parcial)

NOTE: There were 1 observations read from the data set SAS2.CAD011986.
NOTE: PROCEDURE PRINT used (Total process time):
real time 0.01 seconds
cpu time 0.00 seconds

OBS: ATENÇÃO! Existem situações nas quais a posição da variável macro pode anular um “.” que é importante para a sintaxe do comando, neste caso, repita o “.” para que o processamento ocorra sem problemas.

Ex.20 – Situação Especial com Delimitador

```
%let mes=01;
%let bib=sas2;

proc print data=&bib.cad&mes.1986;
  var nome empresa funcao admissao;
run;
```

SAS Log

```
114 %let mes=01;
115 %let bib=sas2;
116
117 proc print data=&bib.cad&mes.1986;
ERROR: File WORK.SAS2CAD011986.DATA does not exist.
118   format admissao ddmmyy10.;
119   var nome empresa funcao admissao;
120   title "Relatório Cadastral de Admissão de &mes de 1986";
121 run;

NOTE: The SAS System stopped processing this step because of errors.
NOTE: PROCEDURE PRINT used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds
```

```
proc print data=&bib..cad&mes.1986;
```

Delimitador ←

→ Texto

```
%let mes=01;
%let bib=sas2;

proc print data=&bib..cad&mes.1986;
  var nome empresa funcao admissao;
run;
```

SAS Log

```
122 %let mes=01;
123 %let bib=sas2;
124
125 proc print data=&bib..cad&mes.1986;
126   var nome empresa funcao admissao;
127 run;

NOTE: There were 1 observations read from the data set SAS2.CAD011986.
NOTE: PROCEDURE PRINT used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds
```

2.6 – Opção NOSYMBOLGEN/SYMBOLGEN

Opção que ativa no SAS log a impressão da solução dada pelo processador Macro, de uma variável de macro. Esta opção, normalmente está desligada.

```
options symbolgen ;
```

Ex.21 – Opção SYMBOLGEN

```
options symbolgen;

%let ano=1986;
%let bib=sas2;

proc print data=&bib..cadastro;
  format admissao ddmmyy10.;
  var nome empresa funcao admissao;
  title "Relatório de pessoas admitidas em &ano";
  where year(admissao)=&ano;
run;
```

SAS Log

```
147 options symbolgen;
148
149 %let ano=1986;
150 %let bib=sas2;
151
152 proc print data=&bib..cadastro;
SYMBOLGEN: Macro variable BIB resolves to sas2
153   format admissao ddmmyy10.;
154   var nome empresa funcao admissao;
SYMBOLGEN: Macro variable ANO resolves to 1986
155   title "Relatório de pessoas admitidas em &ano";
156   where year(admissao)=&ano;
SYMBOLGEN: Macro variable ANO resolves to 1986
157 run;

NOTE: There were 8 observations read from the data set SAS2.CADASTRO.
      WHERE YEAR(admissao)=1986;
NOTE: PROCEDURE PRINT used (Total process time):
      real time          0.00 seconds
      cpu time           0.00 seconds

%put _user_;

158 %put _user_;
GLOBAL MES 01
GLOBAL ANO 1986
GLOBAL BIB sas2
```

2.7 – Funções Macro

- As funções macro possuem sintaxe similar as funções de Data step;
- Possuem resultados similares;
- Manipulam variáveis de macro e expressões macro;
- São macro triggers, sendo por sua vez, executadas pelo processador Macro.

2.7.1 – Função %SUBSTR

Função que quebra e separa o conteúdo de uma variável ou expressão macro;

%SUBSTR(<argumento1>,<argumento2>,[argumento3]) ;

- O valor do *argumento1* é separado a partir da posição indicada pelo *argumento2* até a posição indicada pelo *argumento3* ou até a última posição;
- Os valores dos argumentos podem ser:
 1. Constante texto;
 2. Variáveis de macro;
 3. Funções macro;
 4. Rotinas macro.
- Não é necessário, e não se deve colocar os argumentos entre aspas, pois as funções macro tratam todos os seus argumentos como caracteres.

Ex.22 – Função %SUBSTR

```
%let data1=01/01/1985;
%let data2=31/12/2000;

options symbolgen;

proc print data=sas2.cadastro noobs;
  var nome empresa admissao funcao salario;
  where admissao between "01jan%substr(&data1,7)"d
    and "31dec%substr(&data2,7)"d;
  format admissao ddmmyy10.;
run;
```

SAS Log

```
262 %let data1=01/01/1985;
263 %let data2=31/12/2000;
264
265 options symbolgen;
266
267 proc print data=sas2.cadastro noobs;
268     var nome empresa admissao funcao salario;
269     where admissao between "01jan%substr(&data1,7)"d
SYMBOLGEN: Macro variable DATA1 resolves to 01/01/1985
270         and "31dec%substr(&data2,7)"d;
SYMBOLGEN: Macro variable DATA2 resolves to 31/12/2000
271     format admissao ddmmyy10.;
272 run;

NOTE: There were 17 observations read from the data set SAS2.CADASTRO.
WHERE (admissao>='01JAN1985'D and admissao<='31DEC2000'D);
NOTE: PROCEDURE PRINT used (Total process time):
  real time          0.01 seconds
  cpu time           0.01 seconds
```

2.7.2 – Função %SCAN

Função que busca palavras a partir da posição relativa em relação a delimitadores;

%SCAN(<argumento>,<posição> [,delimitadores]) ;

- Busca uma palavra do *argumento* com a *posição* definida pelo *delimitador*.
- O valor do argumento, da posição e dos delimitadores pode ser:
 1. Constante texto;
 2. Variáveis de macro;
 3. Funções macro;
 4. Rotinas macro.
- Não é necessário, e não se deve colocar o argumento e os delimitadores entre aspas, pois as funções macro tratam todos os seus argumentos como caracteres;
- O campo de delimitadores é opcional. Se não for informado, todos os seguintes caracteres especiais serão delimitadores: **branco . (& ! \$ *) ; - / , %**

Ex.23 – Função %SCAN

```
%let bib=%scan(&syslast,1);  
%let arq=%scan(&syslast,2,.);
```

```
proc print data=&bib.&arq;  
run;
```

SAS Log

```
274 %let bib=%scan(&syslast,1);  
SYMBOLGEN: Macro variable SYSLAST resolves to WORK.TESTE2  
275 %let arq=%scan(&syslast,2,.);  
SYMBOLGEN: Macro variable SYSLAST resolves to WORK.TESTE2  
276  
277 proc print data=&bib.&arq;  
SYMBOLGEN: Macro variable BIB resolves to WORK  
SYMBOLGEN: Macro variable ARQ resolves to TESTE2  
278 run;  
  
NOTE: There were 370 observations read from the data set WORK.TESTE2.  
NOTE: PROCEDURE PRINT used (Total process time):  
real time 0.00 seconds  
cpu time 0.00 seconds
```

2.7.3 – Função %UPCASE

Função que coloca o argumento em maiúsculo;

%UPCASE(argumento1) ;

- O valor do *argumento* pode ser:
 1. Constante texto;
 2. Variáveis de macro;
 3. Funções macro;
 4. Rotinas macro.

Ex.24 – Função %UPCASE

```
%let sexo=feminino;
proc print data=sas2.cadastro(obs=10);
  var nome sexo empresa;
  where sexo="%substr(%upcase(&sexo),1,1)";
  title "Relatório das Pessoas do sexo %upcase(&sexo)";
run;
```

SAS Log

```
284 %let sexo=feminino;
285 proc print data=sas2.cadastro(obs=10);
286     var nome sexo empresa;
287     where sexo="%substr(%upcase(&sexo),1,1)";
SYMBOLGEN: Macro variable SEXO resolves to feminino
SYMBOLGEN: Macro variable SEXO resolves to feminino
288     title "Relatório das Pessoas do sexo %upcase(&sexo)";
289 run;

NOTE: There were 10 observations read from the data set SAS2.CADASTRO.
      WHERE sexo='F';
NOTE: PROCEDURE PRINT used (Total process time):
      real time          0.00 seconds
      cpu time           0.00 seconds
```

Relatório das Pessoas do sexo FEMININO

Obs	nome	sexo	empresa
7	CERTO,CARLA	F	PARIS INSTITUTO
8	PISCO,ROSANE	F	ATLAS S.A.
9	MALA,ELIANE	F	
10	APARECIDO,CARLA	F	PARIS INSTITUTO
11	YATAKA,ROSANE	F	PARIS INSTITUTO
12	MARUEL,ELIANE	F	ATLAS S.A.
13	MARQUES,CARLA	F	PARIS INSTITUTO
14	MILIA,CARLA	F	PARIS INSTITUTO
15	SERPA,ROSANE	F	MALTA LTDA
23	PISCO,MONICA	F	

2.7.4 – Função %EVAL / %SYSEVALF

Funções que permitem a interpretação e execução de uma operação aritmética e lógica.

%EVAL(*expressão*) Aritmética somente com números inteiros;
%SYSEVALF(*expressão*) Aritmética com números reais (com decimais);

- Para a função **%EVAL**, trunca resultados não inteiros;
- Para a função **%EVAL**, retorna um valor nulo, se houverem valores não inteiros na expressão, e imprime uma mensagem de erro no log;
- Retornam o resultado como caracteres;
- Retorna 1 (verdadeiro) ou 0 (falso) para operações lógicas;

Ex.25 – Função %EVAL e %SYSEVALF

```
%let val1=10+25;%put &val1;

%let val2=%eval(10+25);%put &val2;

%let val3=%eval(25/10);%put &val3;

%let val4=%sysevalf(25/10);%put &val4;

%let val5=%eval(&val4*2);%put &val5;

%let val6=%sysevalf(&val4*2);%put &val6;

%let val7=%eval(&val3>&val4);%put &val7;

%let val8=%sysevalf((&val2*2)/(&val6+3.85));%put &val8;
```

SAS Log

```
419 %let val1=10+25;%put &val1;
SYMBOLGEN: Macro variable VAL1 resolves to 10+25
10+25
420
421 %let val2=%eval(10+25);%put &val2;
SYMBOLGEN: Macro variable VAL2 resolves to 35
35
422
423 %let val3=%eval(25/10);%put &val3;
SYMBOLGEN: Macro variable VAL3 resolves to 2
2
424
425 %let val4=%sysevalf(25/10);%put &val4;
SYMBOLGEN: Macro variable VAL4 resolves to 2.5
2.5
426
427 %let val5=%eval(&val4*2);%put &val5;
SYMBOLGEN: Macro variable VAL4 resolves to 2.5
ERROR: A character operand was found in the %EVAL function or %IF condition where a
numeric operand is required. The condition was: 2.5*2
SYMBOLGEN: Macro variable VAL5 resolves to
428
429 %let val6=%sysevalf(&val4*2);%put &val6;
SYMBOLGEN: Macro variable VAL4 resolves to 2.5
SYMBOLGEN: Macro variable VAL6 resolves to 5
5
430
431 %let val7=%eval(&val3>&val4);%put &val7;
SYMBOLGEN: Macro variable VAL3 resolves to 2
SYMBOLGEN: Macro variable VAL4 resolves to 2.5
SYMBOLGEN: Macro variable VAL7 resolves to 0
0
432
433 %let val8=%sysevalf((&val2*2)/(&val6+3.85));%put &val8;
SYMBOLGEN: Macro variable VAL2 resolves to 35
SYMBOLGEN: Macro variable VAL6 resolves to 5
SYMBOLGEN: Macro variable VAL8 resolves to 7.90960451977401
7.90960451977401
```


2.7.5 – Função %SYSFUNC

Função macro genérica que permite executar a maioria das funções de Data step como funções macro e opcionalmente formatar o resultado;

%SYSFUNC(Função SAS [,formato]) ;

- Utiliza-se as funções de Data step com seus respectivos argumentos, que podem ser:
 1. Constante texto;
 2. Variáveis de macro;
 3. Funções macro;
 4. Rotinas macro.
- Não é necessário colocar esses argumentos entre aspas;

Ex.26 – Função %SYSFUNC

```
title1 "Este Relatório foi criado em %sysfunc(today()), ddmmyy10.";
title2 "às %sysfunc(time()), time5.";
```

```
proc print data=sas2.cadastro(obs=5);
  var nome cpf empresa;
run;
```

```

                Este Relatório foi criado em 07/12/2010
                às 14:47
Obs          nome          cpf          empresa
1   MARKO, PAULO    01017503989
2   MOUA, MARCO    01211304778   MALTA LTDA
3   SANTOS, PAULO  01518593290
4   GUEDES, PAULO  01614523771   PARIS INSTITUTO
5   SONTAS, MARCO  01712354666   MALTA LTDA
```

```
%let sexo=feminino;
options ls=90;
proc print data=sas2.cadastro(obs=10);
  var nome sexo empresa;
  where sexo="%sysfunc(substr(%sysfunc(upcase(&sexo)),1,1))";
  title "Relatório das Pessoas do sexo %sysfunc(upcase(&sexo))";
run;
```

OBS: Para cada função de Data Step utilizada como função macro deve-se utilizar a função %SYSFUNC

```
499 %let sexo=feminino;
500 options ls=90;
501 proc print data=sas2.cadastro(obs=10);
502   var nome sexo empresa;
503   where sexo="%sysfunc(substr(%sysfunc(upcase(&sexo)),1,1))";
SYMBOLGEN: Macro variable SEXO resolves to feminino
SYMBOLGEN: Macro variable SEXO resolves to feminino
504   title "Relatório das Pessoas do sexo %sysfunc(upcase(&sexo))";
505 run;
```

NOTE: There were 10 observations read from the data set SAS2.CADASTRO.
WHERE sexo='F';

NOTE: PROCEDURE PRINT used (Total process time):
real time 0.01 seconds
cpu time 0.00 seconds

2º Laboratório – Processamento Macro - Parte1

1. Utilizando a tabela SAS, “**lavouras**”, na pasta **c:\curso\sas2**, monte um programa SAS com a PROC SQL, e crie um relatório com as seguintes características:

Estrutura da tabela lavouras						
#	Variável	Tipo	Tamanho	Formato	Label	
1	uf	Char	2		Estado	
2	produto	Char	120		Produto	
3	a_colheita	Num	8		Área de Colheita (ha)	
4	a_colhida	Num	8		Área Colhida (ha)	
5	quant	Num	8		Quantidade Colhida (t)	
6	rend	Num	8		Rendimento (kg/ha)	
7	valor	Num	8	COMMAX12.2	Valor da Produção (x 1000 R\$)	

- Selecione os estados que produzem “Banana”;
- Coloque um título apropriado incluindo o nome do produto;
- Crie uma variável macro que possua o nome de um produto;
- Substitua essa variável macro nos locais aonde é necessário no programa;
- Execute novamente o programa modificando o nome do produto na variável macro. Ex: Coco, Pimenta, Cafe,...

Colunas : uf, produto, valor

Comandos : SELECT-FROM, WHERE

Dica : Será necessário utilizar o comando macro que cria variáveis macro: %LET

- 2 – Abra o arquivo **c:\curso\sas2\lab2_tabulate.sas**. Crie as seguintes variáveis de macro:

- bib Com o nome da biblioteca;
- arq Com o nome do arquivo SAS;
- uf Com a sigla do estado;
- prod Com o nome do produto. Ex: Banana, Cafe, Abacate, Goiaba, ...;
- padrão Com o padrão ODS de formatação do relatório. Ex.: html, pdf, rtf ou ps
- local Com o caminho para o arquivo a ser salvo pelo padrão ODS;
- narq Com o nome do arquivo que será gerado pelo padrão ODS;
- estilo Com o estilo do padrão HTML;
- est Com a estatística do proc tabulate;
- estl Com o label associado a estatística;
- class1 Com o nome da primeira variável classificatória;
- class2 Com o nome da segunda variável classificatória;
- var1 Com o nome da primeira variável de análise;
- var2 Com o nome da segunda variável de análise;
- condição Com o comando where e sua expressão;

- Modifique o programa substituindo o que for necessário pelas variáveis de macro e execute-o.
- Mude alguns valores das variáveis de macro, gere um relatório em pdf.

2.8 – Rotinas Macro

Uma rotina macro é todo texto inserido entre o comando %MACRO e o comando %MEND.

%MACRO <nome da rotina> ;

<texto>

<texto>

...

<texto>

%MEND [nome da rotina] ;

nome da rotina Segue o padrão de nomes do SAS. É obrigatório no comando %MACRO e opcional no comando %MEND.

texto Pode ser:

- 1 - Qualquer texto;
- 2 - Data step e/ou Proc step;
- 3 - Variáveis de macro, funções macro, comandos de macro ou rotinas de macro;
- 4 - Qualquer combinação dos itens acima;

Uma macro é **compilada** quando se submete a rotina e, basicamente, ocorrem três ações importantes:

- 1) Tudo que tiver macro triggers, será analisada a sintaxe e compilado pelo processador macro;
- 2) Comandos de Data Step e Proc Step **não** são analisados e **não** são processados. Esses comandos só serão analisados e compilados quando a rotina macro for executada;
- 3) A macro é armazenada na biblioteca temporária como **work.sasmacr**.

Uma macro é **executada** quando se submete apenas o nome da rotina com o macro trigger %:

%<nome da rotina>

- 1) Uma rotina de macro pode ser colocada em qualquer lugar de um programa SAS;
- 2) Representa um macro trigger, ou seja, será processada pelo processador Macro;
- 3) **Não** é um comando SAS, não necessita do caractere “;”, no final do comando;

2.8.1 – Opções de Compilação e Execução

MCOMPILENOTE=ALL / **NONE**

Opção que indica no log a compilação com sucesso da rotina de macro;

NOMPRINT / **MPRINT**

Opção que escreve no log as linhas de código executadas de uma rotina de macro;

NOMLOGIC / **MLOGIC**

Opção que escreve no log a lógica de execução dos comandos de macro dentro de uma rotina de macro;

Ex.27 – Rotina de Macro

```
options symbolgen mcompilenote=all mprint mlogic;

%let ano=1986;

%macro imp;
  proc print data sas2.cadastro;
    format admissao ddmmyy10.;
    var nome empresa funcao admissao;
    title "Relatório de pessoas admitidas em &ano";
    where year(admissao)=&ano;
  run;
%mend imp;
```

SAS Log

```
367 options symbolgen mcompilenote=all mprint mlogic;
368
369 %let ano=1986;
370
371 %macro imp;
372   proc print data sas2.cadastro;
373     format admissao ddmmyy10.;
374     var nome empresa funcao admissao;
375     title "Relatório de pessoas admitidas em &ano";
376     where year(admissao)=&ano;
377   run;
378 %mend imp;
```

NOTE: The macro IMP completed compilation without errors.
3 instructions 224 bytes.

Cadê o erro ???

OBS: O processador Macro não analisa comandos de Data step e Proc Step.

```
%imp
```

SAS Log

```
379 %imp
MLOGIC(IMP): Beginning execution.
NOTE: Line generated by the invoked macro "IMP".
1   proc print data sas2.cadastro;           format admissao ddmmyy10.;           var nome empresa
      -----
      73
1   ! funcao admissao;           title "Relatório de pessoas admitidas em &ano";           where
MPRINT(IMP):   proc print data sas2.cadastro;
MPRINT(IMP):   format admissao ddmmyy10.;
MPRINT(IMP):   var nome empresa funcao admissao;
SYMBOLGEN:    Macro variable ANO resolves to 1986
MPRINT(IMP):   title "Relatório de pessoas admitidas em 1986";
SYMBOLGEN:    Macro variable ANO resolves to 1986
MPRINT(IMP):   where year(admissao)=1986;
MPRINT(IMP):   run;
```

ERROR 73-322: Expecting an =.

NOTE: The SAS System stopped processing this step because of errors.

NOTE: PROCEDURE PRINT used (Total process time):

```
real time      0.00 seconds
cpu time       0.00 seconds
```

MLOGIC(IMP): Ending execution.

OBS: Os erros só aparecem durante a execução da rotina de Macro. O processador Macro resolve a rotina e todos os comandos, funções e variáveis macro, devolve para a memória todo o texto contido na rotina, para então ser compilada pelo compilador SAS.

```

options symbolgen mcompilenote=all nomprint nomlogic;

%let ano=1986;

%macro imp;
  proc print data=sas2.cadastro;
    format admissao ddmmyy10.;
    var nome empresa funcao admissao;
    title "Relatório de pessoas admitidas em &ano";
    where year(admissao)=&ano;
  run;
%mend imp;

%imp

```

SAS Log

```

380 options symbolgen mcompilenote=all nomprint nomlogic;
381
382 %let ano=1986;
383
384 %macro imp;
385   proc print data=sas2.cadastro;
386     format admissao ddmmyy10.;
387     var nome empresa funcao admissao;
388     title "Relatório de pessoas admitidas em &ano";
389     where year(admissao)=&ano;
390   run;
391 %mend imp;
NOTE: The macro IMP completed compilation without errors.
      3 instructions 224 bytes.
392
393 %imp
SYMBOLGEN: Macro variable ANO resolves to 1986
SYMBOLGEN: Macro variable ANO resolves to 1986

NOTE: There were 8 observations read from the data set SAS2.CADASTRO.
      WHERE YEAR(admissao)=1986;
NOTE: PROCEDURE PRINT used (Total process time):
      real time          0.01 seconds
      cpu time           0.01 seconds

```

Relatório de pessoas admitidas em 1986

Obs	nome	empresa	funcao	admissao
20	MOUA, JOAO	MALTA LTDA	GERENTE	10/09/1986
48	MOUA, LUIS	ATLAS S.A.	GERENTE	03/06/1986
116	MOUA, LÍCIA	ATLAS S.A.	GERENTE	19/01/1986
167	MOUA, MIRIAM	MALTA LTDA	GERENTE	12/09/1986
258	MOUA, MARIA	PARIS INSTITUTO	GERENTE	25/09/1986
363	MOUA, RENATO	MALTA LTDA	GERENTE	10/09/1986
396	MOUA, MARCIO	ATLAS S.A.	GERENTE	04/12/1986
454	MOUA, ROSANE	MALTA LTDA	GERENTE	21/06/1986

2.8.2 – Parâmetros de Rotinas Macro

As rotinas macro podem ser definidas com uma lista de parâmetros de entrada, permitindo agilizar a execução da rotina. A lista de parâmetros deve ser especificada entre parêntesis e separados por vírgulas.

Na definição:

Parâmetros posicionais

```
%MACRO <nome da rotina> (par1, ..., parn) ;  
    <texto>  
%MEND ;
```

par1, ..., parn Variáveis de macro que aparecem na rotina;

Parâmetros com palavra chave e valor inicial

```
%MACRO <nome da rotina> (par1=val1, ..., parn=valn) ;  
    <texto>  
%MEND ;
```

par1=val1, ..., parn=valn Variáveis de macro que aparecem na rotina;

Na execução:

Parâmetros posicionais

```
%<nome da rotina> (val1, ..., valn)
```

val1, ..., valn Valores que serão repassados para as variáveis de macro. Esses valores podem ser: nulo, variável de macro ou rotina de macro;

Parâmetros com palavra chave e valor inicial

```
%<nome da rotina> (par1=val1, ...,parn=valn)
```

par1=val1, ...,parn=valn Valores iniciais serão repassados para as variáveis de macro, caso não seja especificado o parâmetro e o seu valor. Esses valores podem ser: nulo, variável de macro ou rotina de macro;

OBS: Com relação aos parâmetros com palavra chave, na execução da rotina, não é obrigatório especificar os parâmetros e seus valores, pois valores iniciais já foram especificados na definição da rotina;

Ex.28 – Parâmetros Posicionais

```
options symbolgen mcompilenote=all nomprint nomlogic;  
%macro imp(ano, formato) ;  
    proc print data=sas2.cadastro;  
        format admissao &formato;  
        var nome empresa funcao admissao;  
        title "Relatório de pessoas admitidas em &ano";  
        where year(admissao)=&ano;  
    run;  
%mend imp;  
  
%imp(1986, ddmmyy10.)  
  
SAS Log  
  
408 options symbolgen mcompilenote=all nomprint nomlogic;  
409 %macro imp(ano, formato);  
410     proc print data=sas2.cadastro;  
411         format admissao &formato;  
412         var nome empresa funcao admissao;  
413         title "Relatório de pessoas admitidas em &ano";  
414         where year(admissao)=&ano;  
415     run;  
416 %mend imp;  
NOTE: The macro IMP completed compilation without errors.  
      5 instructions 320 bytes.  
417  
418 %imp(1986, ddmmyy10.)  
SYMBOLGEN: Macro variable FORMATO resolves to ddmmyy10.  
SYMBOLGEN: Macro variable ANO resolves to 1986  
SYMBOLGEN: Macro variable ANO resolves to 1986  
NOTE: There were 8 observations read from the data set SAS2.CADASTRO.  
      WHERE YEAR(admissao)=1986;  
NOTE: PROCEDURE PRINT used (Total process time):  
      real time          0.01 seconds  
      cpu time           0.01 seconds
```

Ex.29 – Parâmetros com Palavra Chave

```
options mcompile=all nomprint nomlogic;
%macro imp(ano=1986,formato=date9.);
  proc print data=sas2.cadastro;
    format admissao &formato;
    var nome empresa funcao admissao;
    title "Relatório de pessoas admitidas em &ano";
    where year(admissao)=&ano;
  run;
%mend imp;
```

```
%imp(ano=1996,formato=ddmmyy10.)
```

Relatório de pessoas admitidas em 1996

Obs	nome	empresa	funcao	admissao
2	MOUA, MARCO	MALTA LTDA	ANALISTA	26/04/1996
102	MOUA, LIGIA	ATLAS S.A.	ANALISTA	11/01/1996
214	MOUA, TANIA	ATLAS S.A.	ANALISTA	20/02/1996
223	MOUA, MADALENA	MALTA LTDA	ANALISTA	23/12/1996
249	MOUA, CARLA	ATLAS S.A.	ANALISTA	03/10/1996
262	MOUA, MONICA	MALTA LTDA	ANALISTA	11/03/1996
397	MOUA, ELIANE	MALTA LTDA	ANALISTA	03/01/1996
489	MOUA, LAURA	ATLAS S.A.	ANALISTA	28/12/1996
504	MOUA, PAULO	PARIS INSTITUTO	ANALISTA	28/03/1996

```
%imp(formato=julian7.)
```

Relatório de pessoas admitidas em 1986

Obs	nome	empresa	funcao	admissao
20	MOUA, JOAO	MALTA LTDA	GERENTE	1986253
48	MOUA, LUIS	ATLAS S.A.	GERENTE	1986154
116	MOUA, LICIA	ATLAS S.A.	GERENTE	1986019
167	MOUA, MIRIAM	MALTA LTDA	GERENTE	1986255
258	MOUA, MARIA	PARIS INSTITUTO	GERENTE	1986268
363	MOUA, RENATO	MALTA LTDA	GERENTE	1986253
396	MOUA, MARCIO	ATLAS S.A.	GERENTE	1986338
454	MOUA, ROSANE	MALTA LTDA	GERENTE	1986172

```
%imp()
```

Relatório de pessoas admitidas em 1986

Obs	nome	empresa	funcao	admissao
20	MOUA, JOAO	MALTA LTDA	GERENTE	10SEP1986
48	MOUA, LUIS	ATLAS S.A.	GERENTE	03JUN1986
116	MOUA, LICIA	ATLAS S.A.	GERENTE	19JAN1986
167	MOUA, MIRIAM	MALTA LTDA	GERENTE	12SEP1986
258	MOUA, MARIA	PARIS INSTITUTO	GERENTE	25SEP1986
363	MOUA, RENATO	MALTA LTDA	GERENTE	10SEP1986
396	MOUA, MARCIO	ATLAS S.A.	GERENTE	04DEC1986
454	MOUA, ROSANE	MALTA LTDA	GERENTE	21JUN1986

2.8.3 – Tabela Local

Quando uma rotina de macro, **com parâmetros**, é executada, uma tabela local de variáveis de macro é criada para a rotina, e essas variáveis de macro só estarão disponíveis para a rotina. Quando a rotina de macro finalizar a execução, a tabela local é destruída.

```
%let nome=Ricardo;
%let arq=sas2.cadastro;
%imp(ano=1996,formato=ddmmyy10.)
```

Tabela Global

```
SYSDAY Friday
SYSUSERID      kusel
...
nome           Ricardo
arq            sas2.cadastro
```

Tabela Local

```
ano           1996
formato      ddmmyy10.
```

Ex.30 – Tabela Local

```
%let nome=Ricardo;
%let arq=sas2.cadastro;

%put _global_;

%macro imp(ano=1986,formato=date9.);
  %put _global_;
  %put _local_;
  proc print data=sas2.cadastro;
    format admissao &formato;
    var nome empresa funcao admissao;
    title "Relatório de pessoas admitidas em &ano";
    where year(admissao)=&ano;
  run;
%mend imp;

%imp()
%put _global_;
```

SAS Log

```
92  %let nome=Ricardo;
93  %let arq=sas2.cadastro;
94
95  %put _global_;
GLOBAL ARQ sas2.cadastro
GLOBAL NOME Ricardo
96
97  %macro imp(ano=1986,formato=date9.);
98    %put _global_;
99    %put _local_;
100   proc print data=sas2.cadastro;
101     format admissao &formato;
102     var nome empresa funcao admissao;
103     title "Relatório de pessoas admitidas em &ano";
104     where year(admissao)=&ano;
105   run;
106 %mend imp;
NOTE: The macro IMP completed compilation without errors.
      7 instructions 380 bytes.

107
108 %imp()
GLOBAL ARQ sas2.cadastro
GLOBAL NOME Ricardo
IMP FORMATO date9.
IMP ANO 1986

NOTE: There were 8 observations read from the data set SAS2.CADASTRO.
      WHERE YEAR(admissao)=1986;
NOTE: PROCEDURE PRINT used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds

109 %put _global_;
GLOBAL ARQ sas2.cadastro
GLOBAL NOME Ricardo
```


2.8.4 – Comando %IF-%THEN-%ELSE

Comando de macro que permite condicionar a substituição de texto em um programa SAS. **Este comando só pode ser utilizado dentro de uma definição de rotina macro.**

```
%IF <expressão> %THEN <texto> ;  
      [%ELSE <texto>];
```

expressão Qualquer expressão macro válida;

texto Pode ser: comando macro, texto simples, expressão macro, variável macro, rotina macro.

OBS: 1 – As expressões macro são similares as expressões de Data step, mas a expressão de intervalo, a seguir, **não é permitida: 1996 <= &ano <= 2000** ;

2 – Os operadores lógicos **OR** e **AND** não são precedidos pelo caractere %

Ex.31 – Comando %IF

```
options symbolgen mprint mlogic;  
%macro imp;  
  proc print data=sas2.cadastro;  
    format admissao ddmmyy10.;  
    var nome empresa funcao admissao;  
    title "Relatório de pessoas admitidas em &ano";  
    where year(admissao)=&ano;  
  run;  
%mend imp;  
  
%macro fre;  
  title "Distribuição dos funcionários admitidos a partir de &ano";  
  proc freq data=sas2.cadastro;  
    where year(admissao)>=&ano;  
    table empresa*sexo;  
  run;  
%mend fre;  
  
%macro cond(ano=1986);  
  %imp  
  %if &ano>=2000 %then %fre;  
%mend cond;  
%cond(ano=2003)
```

SAS Log (Parcial)

```
210 %cond(ano=2003)  
MLOGIC(COND): Beginning execution.  
MLOGIC(COND): Parameter ANO has value 2003  
MLOGIC(IMP): Beginning execution.  
MPRINT(IMP): proc print data=sas2.cadastro;  
MPRINT(IMP): format admissao ddmmyy10.;  
MPRINT(IMP): var nome empresa funcao admissao;  
SYMBOLGEN: Macro variable ANO resolves to 2003  
MPRINT(IMP): title "Relatório de pessoas admitidas em 2003";  
SYMBOLGEN: Macro variable ANO resolves to 2003  
MPRINT(IMP): where year(admissao)=2003;  
MPRINT(IMP): run;  
NOTE: There were 416 observations read from the data set SAS2.CADASTRO.  
WHERE YEAR(admissao)=2003;  
NOTE: PROCEDURE PRINT used (Total process time):  
real time 0.00 seconds  
cpu time 0.00 seconds  
MLOGIC(IMP): Ending execution.  
SYMBOLGEN: Macro variable ANO resolves to 2003  
MLOGIC(COND): %IF condition &ano>=2000 is TRUE  
MLOGIC(FRE): Beginning execution.  
SYMBOLGEN: Macro variable ANO resolves to 2003  
MPRINT(FRE): title "Distribuição dos funcionários admitidos a partir de 2003";  
MPRINT(FRE): proc freq data=sas2.cadastro;  
SYMBOLGEN: Macro variable ANO resolves to 2003  
MPRINT(FRE): where year(admissao)>=2003;  
MPRINT(FRE): table empresa*sexo;  
MPRINT(FRE): run;  
NOTE: There were 416 observations read from the data set SAS2.CADASTRO.  
WHERE YEAR(admissao)>=2003;  
NOTE: PROCEDURE FREQ used (Total process time):  
real time 0.40 seconds  
cpu time 0.00 seconds  
  
MLOGIC(FRE): Ending execution.  
MLOGIC(COND): Ending execution.
```

2.8.5 – Comando %DO-%END

Comando de macro que permite executar blocos de comandos, como uma alternativa de expandir e melhorar o funcionamento lógico de um %IF. **Este comando só pode ser utilizado dentro de uma definição de rotina macro.**

```
%IF <expressão> %THEN %DO ;
    <texto> ;<texto>;...
%END ;
[%ELSE %DO ;
    <texto> ;<texto>;...
%END];
```

Ex.32 – Comando de bloco %DO

```
options symbolgen mprint mlogic;

%macro cond(ano=1986);
  proc print data=sas2.cadastro;
    format admissao ddmmyy10.;
    var nome empresa funcao admissao;
    title "Relatório de pessoas admitidas em &ano";
    where year(admissao)=&ano;
  run;
  %if &ano>=2000
    %then %do;
      title "Distribuição dos funcionários admitidos a partir de &ano";
      proc freq data=sas2.cadastro;
        where year(admissao)>=&ano;
        table empresa*sexo;
      run;
    %end;
%mend cond;

%cond()

295 %cond()
MLOGIC(COND): Beginning execution.
MLOGIC(COND): Parameter ANO has value 1986
MPRINT(COND): proc print data=sas2.cadastro;
MPRINT(COND): format admissao ddmmyy10.;
MPRINT(COND): var nome empresa funcao admissao;
SYMBOLGEN: Macro variable ANO resolves to 1986
MPRINT(COND): title "Relatório de pessoas admitidas em 1986";
SYMBOLGEN: Macro variable ANO resolves to 1986
MPRINT(COND): where year(admissao)=1986;
MPRINT(COND): run;

NOTE: There were 8 observations read from the data set SAS2.CADASTRO.
WHERE YEAR(admissao)=1986;
NOTE: PROCEDURE PRINT used (Total process time):
real time 0.00 seconds
cpu time 0.00 seconds

SYMBOLGEN: Macro variable ANO resolves to 1986
MLOGIC(COND): %IF condition &ano>=2000 is FALSE
MLOGIC(COND): Ending execution.
```

2.8.6 – Comando %DO-%TO-%END Iterativo

O comando macro %DO iterativo pode executar repetidas vezes: comandos macros, Data step e Proc Step, totinas, ou seja, qualquer texto. **Este comando só pode ser utilizado dentro de uma definição de rotina macro.**

```
%DO <índice>=<início> %TO <final> [%BY incremento] ;  
    <texto>  
%END ;
```

índice	É automaticamente criada como uma variável macro e armazenada na tabela local;
início,final,incremento	Podem ser qualquer expressão macro que resolva para valores numéricos;
texto	Qualquer texto;

Ex. 33 – Comando %DO Iterativo

```
options symbolgen nomprint mlogic;  
%macro cond(ini=1952,fim=2010);  
    %do ano=&ini %to &fim;  
        proc print data=sas2.cadastro;  
            format aniversario ddmmyy10.;  
            var nome empresa funcao aniversario;  
            title "Relatório de pessoas com Aniversário em &ano";  
            where year(aniversario)=&ano;  
        run;  
    %end;  
%mend cond;  
  
%cond()  
  
SAS Log (Parcial)  
  
435 %cond()  
MLOGIC(COND): Beginning execution.  
MLOGIC(COND): Parameter INI has value 1952  
MLOGIC(COND): Parameter FIM has value 2010  
SYMBOLGEN: Macro variable INI resolves to 1952  
SYMBOLGEN: Macro variable FIM resolves to 2010  
MLOGIC(COND): %DO loop beginning; index variable ANO; start value is 1952; stop value is 2010;  
by value is 1.  
SYMBOLGEN: Macro variable ANO resolves to 1952  
SYMBOLGEN: Macro variable ANO resolves to 1952  
  
NOTE: There were 1 observations read from the data set SAS2.CADASTRO.  
WHERE YEAR(aniversario)=1952;  
NOTE: PROCEDURE PRINT used (Total process time):  
real time 0.00 seconds  
cpu time 0.00 seconds  
  
MLOGIC(COND): %DO loop index variable ANO is now 1953; loop will iterate again.  
SYMBOLGEN: Macro variable ANO resolves to 1953  
SYMBOLGEN: Macro variable ANO resolves to 1953  
  
NOTE: No observations were selected from data set SAS2.CADASTRO.  
NOTE: There were 0 observations read from the data set SAS2.CADASTRO.  
WHERE YEAR(aniversario)=1953;  
NOTE: PROCEDURE PRINT used (Total process time):  
real time 0.00 seconds  
cpu time 0.00 seconds  
...  
...  
...  
MLOGIC(COND): %DO loop index variable ANO is now 2010; loop will iterate again.  
SYMBOLGEN: Macro variable ANO resolves to 2010  
SYMBOLGEN: Macro variable ANO resolves to 2010  
NOTE: No observations were selected from data set SAS2.CADASTRO.  
NOTE: There were 0 observations read from the data set SAS2.CADASTRO.  
WHERE YEAR(aniversario)=2010;  
NOTE: PROCEDURE PRINT used (Total process time):  
real time 0.01 seconds  
cpu time 0.01 seconds  
  
MLOGIC(COND): %DO loop index variable ANO is now 2011; loop will not iterate again.  
MLOGIC(COND): Ending execution.
```

2.9 – Criar Variáveis de Macro em DATA Step

Devido ao processamento macro ocorrer antes da compilação, portanto, antes da execução de um programa SAS, não é possível utilizar o comando %LET para definir variáveis de macro durante a execução lógica do programa SAS.

Ex.34 – Variáveis de Macro em DATA Step

```
data _null_;
  set sas2.cadastro (keep=e_civil) end=fi;
  retain sol 0 cas 0 sep 0;
  if e_civil="1" then sol=sol+1;
  else if e_civil="2" then cas=cas+1;
  else sep=sep+1;
  if fi=1
  then do;
    %let solteiros=sol;
    %let casados=cas;
    %let separados=sep;
  end;
run;

%put &solteiros;
%put &casados;
%put &separados;

SAS Log (Parcial)

222 %put &solteiros;
sol
223 %put &casados;
cas
224 %put &separados;
sep
```

A rotina SYMPUT permite ao SAS criar variáveis de macro durante a execução de um Data step, salvando a variável de macro na tabela global. Esta rotina mantém todos os caracteres brancos a esquerda e a direita do conteúdo da variável de macro. Possui sintaxe variada:

CALL SYMPUT("macvar", "texto");

macvar	Nome da variável de macro;
texto	Qualquer texto, que será o conteúdo da variável de macro;

CALL SYMPUT("macvar", variável);

macvar	Nome da variável de macro;
variável	Variável de data step, cujo conteúdo será da variável de macro;

CALL SYMPUT("macvar", expressão);

macvar	Nome da variável de macro;
expressão	Qualquer expressão, cujo resultado será o conteúdo da variável de macro;

CALL SYMPUT(expressão1, expressão2);

expressão1	Qualquer expressão, cujo resultado será o nome da variável de macro;
expressão2	Qualquer expressão, cujo resultado será o conteúdo da variável de macro;

CALL SYMPUTX(expressão1, expressão2);

expressão1	Qualquer expressão, cujo resultado será o nome da variável de macro;
expressão2	Qualquer expressão, cujo resultado será o conteúdo da variável de macro;

OBS: Esta rotina em especial, **remove os brancos a esquerda e a direita** de ambos os argumentos;

Ex.35 – Call SYMPUT

```
data _null_;
  set sas2.cadastro (keep=e_civil) end=fi;
  retain sol 0 cas 0 sep 0;
  if e_civil="1" then sol=sol+1;
  else if e_civil="2" then cas=cas+1;
  else sep=sep+1;
  if fi=1
    then do;
      call symput('solteiros',sol);
      call symput('casados',cas);
      call symput('separados',sep);
    end;

run;

%put &solteiros;
%put &casados;
%put &separados;

SAS Log (Parcial)

240 %put &solteiros;
337
241 %put &casados;
169
242 %put &separados;
44

options nosymbolgen;
data _null_;
  set sas2.cadastro end=fi;
  retain n 0;
  if funcao in ("GERENTE","DIRETOR")
    then do;
      n=n+1;
      call symputx('nome' ||left(n),scan(nome,2,' '));
      call symput('empresa' ||left(n),empresa);
      call symput('funcao' ||left(n),funcao);
      call symputx('          salario' ||left(n),int(salario)+1000);
    end;

run;

%put _user_;
```

SAS Log (Parcial)

```
15 %put _user_ ;
GLOBAL FUNCAO5 GERENTE
GLOBAL NOME7 MARCIO
GLOBAL EMPRESA8 MALTA LTDA
GLOBAL FUNCAO2 GERENTE
GLOBAL NOME4 MIRIAM
GLOBAL EMPRESA9 MALTA LTDA
GLOBAL FUNCAO3 GERENTE
GLOBAL NOME5 MARIA
GLOBAL NOME2 LUIS
GLOBAL FUNCAO1 GERENTE
GLOBAL NOME3 LICIA
GLOBAL SALARIO1 21611
GLOBAL NOME1 JOAO
GLOBAL EMPRESA2 ATLAS S.A.
GLOBAL SALARIO3 18260
GLOBAL SALARIO2 18414
GLOBAL EMPRESA3 ATLAS S.A.
GLOBAL SALARIO5 20303
GLOBAL EMPRESA1 MALTA LTDA
GLOBAL SALARIO4 14075
GLOBAL EMPRESA6 MALTA LTDA
GLOBAL SALARIO7 11828
GLOBAL SALARIO6 21457
GLOBAL EMPRESA7 ATLAS S.A.
GLOBAL FUNCAO8 GERENTE
GLOBAL EMPRESA4 MALTA LTDA
GLOBAL FUNCAO9 DIRETOR
GLOBAL SALARIO9 26377
GLOBAL EMPRESA5 PARIS INSTITUTO
GLOBAL FUNCAO6 GERENTE
GLOBAL NOME8 ROSANE
GLOBAL SALARIO8 21336
```

2.10 – Criar Variáveis de Macro em PROC SQL

O comando SELECT com o sub-comando INTO criam ou atualizam variáveis de macro das seguintes formas:

Variáveis de macro individuais

```
PROC SQL;  
  SELECT coll, col2, ... INTO :mvar1, :mvar2, ...  
  FROM <tabela>  
  WHERE <expressão>  
  ... ;  
QUIT;
```

coll,col2,... Colunas cujo conteúdo será armazenado em variáveis de macro;
:mvar1:mvar2,... Nome das variáveis macro. Na sintaxe do SQL, é obrigatório colocar **:** no início do nome da variável de macro.

- O valor da *coll* será armazenado na *mvar1* e assim respectivamente;
- O sub-comando INTO não remove brancos a esquerda e a direita do conteúdo;
- Apenas a primeira linha de valores ou a primeira linha selecionada pelo comando WHERE, será armazenada nas variáveis de macro;

Intervalos de variáveis de macro

```
PROC SQL;  
  SELECT coll,col2, ... INTO :m1var1-:m1varn, :m2var1-:m2varn,...  
  FROM <tabela>  
  WHERE <expressão>  
  ... ;  
QUIT;
```

:m1var1-:m1varn Intervalo de variáveis de macro.

- As variáveis de macro irão armazenar *n* valores de *n* linhas das colunas selecionadas;

Todos os valores em uma única variável de macro

```
PROC SQL;  
  SELECT coll,col2, ... INTO :mvar1 SEPARATED BY 'delimitador' ,  
  :mvar2 SEPARATED BY 'delimitador' , ...  
  FROM <tabela>  
  WHERE <expressão>  
  ... ;  
QUIT;
```

:mvar Variável de macro que irá armazenar todos os valores da coluna especificada e das linhas selecionada, separadas por um caractere determinado;

delimitador Caractere que separa, na variável de macro, os valores da coluna especificada;

Ex.36 – SELECT-INTO

```
proc sql;
    select nome,empresa into :no,:em
        from sas2.cadastro;
quit;

%put &no;
%put &em;
```

SAS Log (parcial)

```
70
71 %put &no;
GUEDES, PAULO
72 %put &em;
PARIS INSTITUTO
```

The SAS System

nome	empresa
-----	-----
GUEDES, PAULO	PARIS INSTITUTO
CERTO, CARLA	PARIS INSTITUTO
APARECIDO, CARLA	PARIS INSTITUTO
YATAKA, ROSANE	PARIS INSTITUTO
MARQUES, CARLA	PARIS INSTITUTO
MILIA, CARLA	PARIS INSTITUTO
SUNAY, MARCELO	PARIS INSTITUTO
...	
...	

```
proc sql;
    select nome,empresa,funcao,salario into :no1-:no9,:em1-:em9,:fu1-:fu9,:sa1-:sa9
        from sas2.cadastro
        where funcao in ("GERENTE","DIRETOR");
quit;

%put _global_;
```

SAS Log (parcial)

```
8 %put _global_ ;
GLOBAL NO1 MOUA,MARIA
GLOBAL FU3 GERENTE
GLOBAL NO2 MOUA, JOAO
GLOBAL FU4 GERENTE
GLOBAL NO3 MOUA, MIRIAM
GLOBAL FU5 GERENTE
GLOBAL NO4 MOUA, RENATO
GLOBAL FU6 DIRETOR
GLOBAL NO5 MOUA, ROSANE
GLOBAL FU7 GERENTE
GLOBAL NO6 MOUA, MARCELO
GLOBAL FU8 GERENTE
GLOBAL EM1 PARIS INSTITUTO
GLOBAL NO7 MOUA, LUIS
GLOBAL FU9 GERENTE
GLOBAL EM2 MALTA LTDA
GLOBAL NO8 MOUA, LICIA
GLOBAL SA1 19303.66
GLOBAL EM3 MALTA LTDA
GLOBAL NO9 MOUA, MARCIO
GLOBAL SA2 20611.56
GLOBAL EM4 MALTA LTDA
GLOBAL SA3 13075.91
GLOBAL EM5 MALTA LTDA
GLOBAL SA4 20457.36
GLOBAL EM6 MALTA LTDA
GLOBAL SA5 20336.22
GLOBAL EM7 ATLAS S.A.
GLOBAL SA6 25377.28
GLOBAL EM8 ATLAS S.A.
```

The SAS System

nome	empresa	funcao	salario
-----	-----	-----	-----
MOUA, MARIA	PARIS INSTITUTO	GERENTE	19303.66
MOUA, JOAO	MALTA LTDA	GERENTE	20611.56
MOUA, MIRIAM	MALTA LTDA	GERENTE	13075.91
MOUA, RENATO	MALTA LTDA	GERENTE	20457.36
MOUA, ROSANE	MALTA LTDA	GERENTE	20336.22
MOUA, MARCELO	MALTA LTDA	DIRETOR	25377.28
MOUA, LUIS	ATLAS S.A.	GERENTE	17414.14
MOUA, LICIA	ATLAS S.A.	GERENTE	17260.69
MOUA, MARCIO	ATLAS S.A.	GERENTE	10828.7

```

proc sql noprint;
  select count(*) into :x
    from sas2.cadastro
    where funcao in ("GERENTE","DIRETOR");

  %let x=&x; ← Artificio para eliminar os brancos a esquerda e a direita.

  select nome,empresa,funcao,salario into :no1-:no&x,:em1-:em&x,:fu1-:fu&x,:sa1-:sa&x
    from sas2.cadastro
    where funcao in ("GERENTE","DIRETOR");
quit;

%put _global_;

```

SAS Log (parcial)

```

9  options nodate nonumber;
10 proc sql noprint;
11   select count(*) into :x
12     from sas2.cadastro
13     where funcao in ("GERENTE","DIRETOR");
14
15   %let x=&x;
SYMBOLGEN: Macro variable X resolves to 9
16
17   select nome,empresa,funcao,salario into :no1-:no&x,:em1-:em&x,:fu1-:fu&x,:sa1-:sa&x
SYMBOLGEN: Macro variable X resolves to 9
SYMBOLGEN: Macro variable X resolves to 9
SYMBOLGEN: Macro variable X resolves to 9
SYMBOLGEN: Macro variable X resolves to 9
18     from sas2.cadastro
19     where funcao in ("GERENTE","DIRETOR");
20 quit;
NOTE: PROCEDURE SQL used (Total process time):
      real time          0.01 seconds
      cpu time           0.01 seconds

```

```

proc sql noprint;
  select nome,empresa,funcao,salario into :no separated by "*",
                                           :em separated by "*",
                                           :fu separated by "*",
                                           :sa separated by "*"

    from sas2.cadastro
    where funcao in ("GERENTE","DIRETOR");
quit;

```

```

%put &no;
%put &em;
%put &fu;
%put &sa;

```

SAS Log (parcial)

```

74
75 %put &no;
MOUA, MARIA*MOUA, JOAO*MOUA, MIRIAM*MOUA, RENATO*MOUA, ROSANE*MOUA, MARCELO*MOUA, LUIS*MOUA, LICIA*MOUA, MARCIO

76 %put &em;
PARIS INSTITUTO*MALTA LTDA*MALTA LTDA*MALTA LTDA*MALTA LTDA*MALTA LTDA*ATLAS S.A.*ATLAS S.A.*ATLAS
S.A.

77 %put &fu;
GERENTE*GERENTE*GERENTE*GERENTE*GERENTE*DIRETOR*GERENTE*GERENTE*GERENTE

78 %put &sa;
19303.66*20611.56*13075.91*20457.36*20336.22*25377.28*17414.14*17260.69*10828.7

```


2.11 – Referências Indiretas de Variáveis de Macro

De uma maneira simples, referências indiretas de variáveis de macro, significa uma variável de macro ter como valor outra variável de macro, e assim por diante, até chegar a um valor texto. A idéia é fazer com que o processador macro fique em “loop” tentando resolver as sequências de resultados, também, macros.

¯v1 → ¯v2 → ¯v3 → ... → texto

- Cada vez que uma variável de macro aparece em um texto, o processador macro tenta resolver e retornar um valor puramente texto;
- Colocando vários caracteres & antes de um texto qualquer, indica referência indireta a várias variáveis de macro, que serão resolvidas, uma a uma, até o processador macro não encontrar mais nenhum caractere &;
- Dois caracteres && juntos resolvem para um caractere &;

Ex.37 – Referência Indireta

```
%let aviao=g1132;
%let g1132=Brasilia;

%put &&&aviao;

%let n=033;
%let banco033=Santander;

%put &&banco&n;
```

SAS Log

```
1  %let aviao=g1132;
2  %let g1132=Brasilia;
3
4  %put &&&aviao;
SYMBOLGEN:  && resolves to &.
SYMBOLGEN:  Macro variable AVIAO resolves to g1132
SYMBOLGEN:  Macro variable G1132 resolves to Brasília
Brasília
5
6  %let n=033;
7  %let banco033=Santander;
8
9  %put &&banco&n;
SYMBOLGEN:  && resolves to &.
SYMBOLGEN:  Macro variable N resolves to 033
SYMBOLGEN:  Macro variable BANCO033 resolves to Santander
Santander
```

Ex.38 – Referência Indireta

```
proc print data=sas2.tabfunc(obs=5) noobs;
  var codir cfun descricao;
run;
```

The SAS System

codir	cfun	descricao
211	F211	Matemático, estatístico, atuário e afins
212	F212	Analista de sistemas, desenvolvedor de software, admin. de redes e bancos de dados e outros especial
213	F213	Físico, químico, meteorologista, geólogo, oceanógrafo e afins
214	F214	Engenheiro, arquiteto e afins
215	F215	Piloto de aeronaves, comandante de embarcações e oficiais de máquinas

/* Alternativa 1 - Usando a variável codir */

OBS: Nome de variável de macro não pode começar com numeros, será necessário criar um prefixo.

```
data _null_;
  set sas2.tabfunc;
  call symputx('func' || left(codir), descricao);
run;
```

SAS Log (parcial)

```
8 %put _user_;
GLOBAL FUNC211 Matemático, estatístico, atuário e afins
GLOBAL FUNC222 Agrônomo e afins
GLOBAL FUNC221 Biólogo, biomédico e afins
GLOBAL FUNC258 Assistente social e economista doméstico
GLOBAL FUNC259 Filósofo
GLOBAL FUNC256 Geógrafo
GLOBAL FUNC257 Historiador
```

```
%let cod=211;
proc print data=sas2.cadfunc;
  title " Fucionários com a Formação de &&func&cod ";
  where prof=&cod;
run;
```

SAS Log (parcial)

```
106 %let cod=211;
107 proc print data=sas2.cadfunc;
SYMBOLGEN: && resolves to &.
SYMBOLGEN: Macro variable COD resolves to 211
SYMBOLGEN: Macro variable FUNC211 resolves to Matemático, estatístico, atuário e afins
108 title " Fucionários com a Formação de &&func&cod ";
109 where prof=&cod;
SYMBOLGEN: Macro variable COD resolves to 211
110 run;
```

NOTE: There were 59 observations read from the data set SAS2.CADFUNC.

WHERE prof=211;

NOTE: PROCEDURE PRINT used (Total process time):

real time 0.01 seconds
cpu time 0.01 seconds

Fucionários com a Formação de Matemático, estatístico, atuário e afins

Obs	nome	sexo	idade	e_civil	cpf	prof
1	GUEDES, PAULO	M	29	1	01614523771	211
3	APARECIDO, CARLA	F	21	1	02243537624	211
7	SUNAY, MARCELO	M	21	1	03035241713	211
14	SUNAY, FRANCISCA	F	20	1	05181928990	211
22	MEREDITE, FLAVIA	F	27	1	07602864766	211
	...					
	...					
	...					

```
/* Alternativa 2 - Usando a variável cfun*/
```

OBS: Neste caso, o conteúdo da variável cfun já possui um prefixo.

```
data _null_;  
  set sas2.tabfunc;  
  call symputx(cfun,descricao);  
run;
```

```
%put _user_;
```

SAS Log (parcial)

```
7  
8 %put _user_;  
GLOBAL F211 Matemático, estatístico, atuário e afins  
GLOBAL F212 Analista de sistemas, desenvolvedor de software, admin. de redes e bancos de dados e  
outros especial  
GLOBAL F221 Biólogo, biomédico e afins  
GLOBAL F213 Físico, químico, meteorologista, geólogo, oceanógrafo e afins  
GLOBAL F222 Agrônomo e afins  
GLOBAL F214 Engenheiro, arquiteto e afins
```

```
%let cod=f226;  
proc print data=sas2.cadfunc;  
  title " Fucionários com a Formação de &&cod ";  
  where prof=%substr(&cod,2);  
run;
```

SAS Log (parcial)

```
9 %let cod=f226;  
10 proc print data=sas2.cadfunc;  
SYMBOLGEN: && resolves to &  
SYMBOLGEN: Macro variable COD resolves to f226  
SYMBOLGEN: Macro variable F226 resolves to Médico, odontólogo e afins  
11 title " Fucionários com a Formação de &&cod ";  
12 where prof=%substr(&cod,2);  
SYMBOLGEN: Macro variable COD resolves to f226  
13 run;
```

NOTE: There were 48 observations read from the data set SAS2.CADFUNC.

WHERE prof=226;

NOTE: PROCEDURE PRINT used (Total process time):

real time 0.03 seconds

cpu time 0.03 seconds

Fucionários com a Formação de Médico, odontólogo e afins

Obs	nome	sexo	idade	e_civil	cpf	prof
9	ANJOA, JOAO	M	24	1	03361854537	226
17	PINTOTO, MIRIAM	F	29	1	05689948012	226
21	MENDES, LAURA	F	29	1	06879575222	226
28	MALA, ROSANE	F	20	1	12862709772	226
49	SUNAY, LIGIA	F	24	1	25856274809	226
54	PINTOTO, MARCO	M	25	1	31417344223	226
61	MENDES, MADALENA	F	28	1	35588026111	226
65	MARKO, MARCO	M	22	1	41213304015	226
79	MILIA, FLAVIA	F	21	1	47107884229	226

3º LABORATÓRIO – Macro – Parte 2

1 – A tabela SAS **ufcod** na pasta **c:\curso\sas2** possui duas variáveis: código da Unidade da Federação (**cod**) e a descrição desse código (**uf**). Monte um programa que crie, automaticamente, variáveis macros que possuam a descrição de cada código da unidade da federação; seria algo similar ao comando manual `%LET uf5=ce` .

Dicas: - Crie as variáveis macros com o prefixo “uf”;
- Utilize a rotina `call symputx`;

2 – O programa **c:\curso\sas2\importa_uf.sas**, importa os dados de uma específica Unidade da Federação, do excel para o SAS, utilizando a PROC IMPORT, e adapta os dados, corrigindo alguns detalhes com um Data step. Coloque esse programa em uma rotina de macro, de maneira que, a rotina consiga ler automaticamente todas as variáveis de macro criadas no exercício anterior, identificando, um a um, os arquivos em Excel das Unidades da Federação, e convertendo-os para SAS.

3 – Acrescente a rotina macro do exercício anterior um condição lógica de macro: **Se** , a Unidade da Federação, que está sendo importado os dados, for São Paulo “**sp**”, **então** execute um PROC MEANS para gerar um relatório somente com o somatório e a média da variável valor.

3 – Indexação de Dados

Indexar, organizar, mapear os dados de uma arquivo significa criar apontadores para as linhas de dados, para permitir acesso mais rápido e eficiente; A indexação é feita com variáveis de arquivo, consideradas como chaves de acesso aos dados e que são utilizadas com muita frequência;

Um arquivo indexado possui um arquivo auxiliar com apenas os valores da chave indexada e com a localização dos valores dessas chaves no arquivo principal;

As técnicas de indexação e de busca pelas chaves indexadas são privativas de cada software, e valorizam o software no quesito eficiência e performance;

ARQUIVO PRINCIPAL			ARQUIVO INDEX
OBS	RG	NOME	CHAVE (REGISTROS)
1	1234	Marcelo	0001 (584 , 586)
2	5678	Flávia	0011 (3 , 126)
3	0011	Paulo	0045 (125)
4	1234	Marcelo	0123 (127 , 585)
.	.	.	1234 (1 , 4 , 583)
.	.	.	5678 (2 , 128)
.	.	.	
125	0045	Felipe	
126	0011	Paulo	
127	0123	Catarina	
128	5678	Flávia	
.	.	.	
.	.	.	
.	.	.	
583	1234	Marcelo	
584	0001	Solange	
585	0123	Catarina	
586	0001	Solange	

O index pode ser construído como:

Simples Formado pelo valor de uma única variável do arquivo, caractere ou numérica. O nome do index será o mesmo do nome da variável;

Composto Formado pela combinação dos valores de mais de uma variável do arquivo, caracteres, ou numéricas, ou misturadas. O nome do index segue o padrão do SAS para nomes e **não pode** ter o nome de qualquer uma das variáveis que irão compor o index;

É possível construir diversos indexes: simples, compostos, ou ambos, para um mesmo arquivo, mas cuidado, a busca pelos dados no arquivo indexado pode se tornar lenta e prejudicar a performance no processamento do arquivo.

3.1 – Indexação com o SAS

O SAS possui três alternativas básicas para a criação de indexes:

- 1 – PROC DATASETS
- 2 – PROC SQL
- 3 – Opção INDEX= do comando DATA

As duas primeiras alternativas criam indexes a partir de arquivos SAS que já existam, e a terceira alternativa cria indexes quando o arquivo SAS está sendo criado.

3.1.1 – Procedimento DATASETS

```
PROC DATASETS lib=<biblioteca> NOLIST ;  
    MODIFY <arquivo> ;  
        INDEX CREATE <index> [ / unique nomiss];  
        INDEX DELETE <index> ;  
QUIT ;
```

Opções:

lib= Opção obrigatória. Serve para identificar a biblioteca aonde se localiza o arquivo SAS;

NOLIST Opção que desliga a criação de um relatório com informações de todos os arquivos SAS da biblioteca informada;

MODIFY Comando que indica qual o arquivo da biblioteca será modificado, neste caso, a criação de indexes;

INDEX CREATE Sub-comando para criar indexes;

INDEX DELETE Sub-comando para apagar indexes;

OBS: Os comandos INDEX CREATE e INDEX DELETE, são sub-comandos do comando principal MODIFY

index Nome do index que será criado

simple INDEX CREATE <variável>

composto INDEX CREATE <nome sas>=(variável1 variável2...variáveln)

/ unique Opção que indica que a chave de indexação **tem que ser única**, ou seja, **não pode ter valores repetidos** nas variáveis que compõem o index.

/ nomiss Opção que indica que a chave de indexação **não pode conter valores “missing”**, em nenhuma das variáveis que compõem o index.

Ex.39 – Criação de index com PROC DATASETS

```
proc datasets lib=arq nolist;
  modify cadastro;
    index create cpf / unique nomiss;
    index create empfunc=(empresa funcao);
quit;
```

SAS Log

```
21  proc datasets lib=arq nolist;
22      modify cadastro;
23          index create cpf / unique nomiss;
NOTE: Simple index cpf has been defined.
24          index create empfunc=(empresa funcao);
NOTE: Composite index empfunc has been defined.
25  quit;

NOTE: MODIFY was successful for ARQ.CADASTRO.DATA.
NOTE: PROCEDURE DATASETS used (Total process time):
      real time      0.03 seconds
      cpu time       0.01 seconds
```

```
proc datasets lib=arq nolist;
  modify cadastro;
    index delete cpf;
    index delete empfunc;
quit;
```

SAS Log

```
37  proc datasets lib=arq nolist;
38      modify cadastro;
39          index delete cpf;
NOTE: Index cpf deleted.
40          index delete empfunc;
NOTE: All indexes defined on ARQ.CADASTRO.DATA have been deleted.
41  quit;

NOTE: MODIFY was successful for ARQ.CADASTRO.DATA.
NOTE: PROCEDURE DATASETS used (Total process time):
      real time      0.00 seconds
      cpu time       0.00 seconds
```

```
proc contents data=arq.cadastro;
run;
```

The SAS System

The CONTENTS Procedure

Data Set Name	ARQ.CADASTRO	Observations	550
Member Type	DATA	Variables	15
Engine	V9	Indexes	2
Created	Seg, 09 de Ago de 2010 17h37min54s	Observation Length	152
Last Modified	Qua, 22 de Set de 2010 17h11min47s	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	YES
Label			
Data Representation	WINDOWS_32		
Encoding	wlatin1 Western (Windows)		

...
...
...
...

Alphabetic List of Indexes and Attributes

#	Index	Unique Option	NoMiss Option	# of Unique Values	Variables
1	cpf	YES	YES	550	
2	empfunc			11	empresa funcao

3.1.2 – Procedimento SQL

```
PROC SQL ;  
    CREATE [unique] INDEX <index>  
        ON <arquivo(variável1, variável2, ... , variáveln)> ;
```

```
    DROP INDEX <index>  
    FROM <arquivo> ;
```

```
QUIT ;
```

opções No SQL só existe a opção **unique**;

index Nome do index que será criado ou apagado;

arquivo Nome do arquivo que será indexado;

variável Variável ou variáveis que irão compor o index;

OBS: Não é possível criar um index com o mesmo nome, se o arquivo já estiver indexado. O index tem que ser apagado para ser criado novamente.

Ex.40 – Criação de index com PROC SQL

```
proc sql;  
    drop index cpf from arq.cadastro;  
    drop index empfunc from arq.cadastro;  
  
    create unique index cpf  
        on arq.cadastro(cpf);  
  
    create index empfunc  
        on arq.cadastro(empresa,funcao);  
quit;
```

SAS Log

```
59  proc sql;  
60      drop index cpf from arq.cadastro;  
NOTE: Index cpf has been dropped.  
61      drop index empfunc from arq.cadastro;  
NOTE: Index empfunc has been dropped.  
62  
63      create unique index cpf  
64          on arq.cadastro(cpf);  
NOTE: Simple index cpf has been defined.  
65  
66      create index empfunc  
67          on arq.cadastro(empresa,funcao);  
NOTE: Composite index empfunc has been defined.  
68  quit;  
NOTE: PROCEDURE SQL used (Total process time):  
      real time          0.07 seconds  
      cpu time           0.01 seconds
```


3.1.3 – Criação de index em DATA Step

É possível criar indexes quando se cria o arquivo SAS. Seria essa a melhor opção, pois a indexação necessita ler todo o arquivo, e neste caso, a gravação dos dados e a criação dos indexes seriam feitos juntos numa única vez, melhorando a performance e economizando recursos do ambiente.

```
DATA <arquivo> ( INDEX= ( <index1>[/opção1 /opção2]
                        <index2>[/opção1 /opção2]
                        ...
                        <indexn>[/opção1 /opção2] ) );
```

arquivo Nome do arquivo SAS que será criado;

index Nome do index que será criado:

simple <variável>[/opção1/opção2]

composto <nome sas>=(variável1 variável2 ... variáveln)/[opção1 /opção2]

opções **unique nomiss**

OBS: A princípio, a criação de indexes em DATA step, durante a criação do arquivo SAS, não gera nenhuma informação na janela de LOG. Em DATA step, é necessário ativar uma opção que ativa as informações sobre criação e utilização de indexes.

```
OPTIONS MSGLEVEL=I
```

Ex.41 – Criação de index em DATA Step

```
proc sql;
  drop index cpf from arq.cadastro;
  drop index empfunc from arq.cadastro;
quit;

options msglevel=i;

data arq.cadastro (INDEX=(cpf/unique /nomiss
                        empfunc=(empresa funcao)));

  infile "cadastro.dat" lrecl=130 missover;
  input nome $30. sexo $1. idade 2. peso 6.2 altura 4.2 aniversario ddmmyy10.
        e_civil $1. filhos 2. rg $15. cpf $11. empresa $15. funcao $12.
        admissao date9. salario commax12.2 @89 t_emp $9.;
run;
```

SAS Log

```
129 proc sql;
130   drop index cpf from arq.cadastro;
NOTE: Index cpf has been dropped.
131   drop index empfunc from arq.cadastro;
NOTE: Index empfunc has been dropped.
132 quit;
NOTE: PROCEDURE SQL used (Total process time):
      real time          0.03 seconds
      cpu time           0.01 seconds

133
134 options msglevel=i;
135
136 data arq.cadastro (INDEX=(cpf /unique /nomiss
137                       empfunc=(empresa funcao)));
138
139   infile "cadastro.dat" lrecl=130 missover;
140   input nome $30. sexo $1. idade 2. peso 6.2 altura 4.2 aniversario ddmmyy10.
141         e_civil $1. filhos 2. rg $15. cpf $11. empresa $15. funcao $12.
142         admissao date9. salario commax12.2 @89 t_emp $9.;
143 run;
```

NOTE: The infile "cadastro.dat" is:
Filename=D:\kusel\Cursos\sas\Curso_SAS_Revisao_2010\cadastro.dat,
RECFM=V,LRECL=130,File Size (bytes)=72600,
Last Modified=14 de Setembro de 2006 16h04mi,
Create Time=13 de Agosto de 2010 11h19min4

NOTE: 550 records were read from the infile "cadastro.dat".
The minimum record length was 130.
The maximum record length was 130.

NOTE: The data set ARQ.CADASTRO has 550 observations and 15 variables.

NOTE: Composite index empfunc has been defined.

NOTE: Simple index cpf has been defined.

NOTE: DATA statement used (Total process time):
real time 0.04 seconds
cpu time 0.01 seconds

3.2 – Utilização do INDEX

A utilização do index em processamento possibilita uma acesso mais rápido aos dados, e de certa forma, é considerado como acesso “**direto**” ao dado, ao invés do acesso seqüencial, ou seja, leitura de registro após registro, até encontrar os dados.

No SAS, quando se define um index, é criado e armazenado na estrutura do arquivo, informações e estatísticas da distribuição dos dados indexados no arquivo principal, essas informações permitirão ao SAS decidir, automaticamente, **o uso ou não do index** no processamento dos dados. O SAS considera os recursos que serão utilizados, como memória, cpu e I/O, para decidir se a leitura dos dados serão no modo de acesso direto, utilizando o index, ou, no modo de acesso seqüencial.

O index, também permite o processamento dos dados indexados ordenados em ordem crescente, sendo uma alternativa ao uso do PROC SORT;

3.2.1 - Situações em que o SAS poderá utilizar o index

- Volume de dados que serão selecionados

O SAS possui faixas percentuais de volume de dados do arquivo que definem o uso provável do index:

0% a 3% do arquivo	O SAS utilizará o index;
3% a 33% do arquivo	O SAS provavelmente utilizará o index;
33% a 100% do arquivo	O SAS talvez utilize o index;

- Em expressões do comando **WHERE**

Com operadores relacionais: = , < , <= , > , >= , ~= , IN , NOT , CONTAINS , BETWEEN-AND , LIKE

Com operador lógico: **AND**

- Comando **BY**

Como o index ordena os dados na ordem crescente, é possível utiliza-lo em todos os processamentos que necessitem do comando BY, ou seja, de agrupamento dos dados; não sendo obrigado a ordenar o arquivo com o PROC SORT.

OBS: O SAS só utiliza **um index para processamento no comando WHERE**, escolhendo o que, provavelmente, terá melhor performance.

Ex.42 – INDEX utilizado no comando WHERE

```
options msglevel=i nodate nonumber;
```

```
proc print data=arq.cadastro;  
  var nome cpf empresa funcao salario;  
  where cpf in ("04979489998","02437587453","13064972095");  
run;
```

SAS Log

```
1  options msglevel=i nodate nonumber;  
2  
3  proc print data=arq.cadastro;  
4      where cpf in ("04979489998","02437587453","13064972095");  
INFO: Index cpf selected for WHERE clause optimization.  
5  run;  
NOTE: There were 3 observations read from the data set ARQ.CADASTRO.  
WHERE cpf in ('02437587453', '04979489998', '13064972095');  
NOTE: PROCEDURE PRINT used (Total process time):  
real time          0.15 seconds  
cpu time           0.03 seconds
```

The SAS System

Obs	nome	cpf	empresa	funcao	salario
12	MARUEL,ELIANE	02437587453	ATLAS S.A.	PROGRAMADOR	3024.76
32	MALA,TANIA	04979489998	PARIS INSTITUTO	PROGRAMADOR	3063.13
77	SANTOS,RENATO	13064972095	PARIS INSTITUTO	PROGRAMADOR	1816.50

OBS: Os três CPFs informados representam apenas 0,54% dos dados do arquivo cadastro (total: 550) e a expressão do WHERE utiliza um operador adequado ao uso com index. O SAS, neste caso, irá utilizar os dados indexados pela variável CPF.

Ex.43 – INDEX utilizado no comando BY

```
options msglevel=i;
```

```
proc print data=arq.cadastro(obs=5 bufno=7);  
  var nome empresa funcao salario;  
  where empresa ne "";  
  by empresa funcao;  
  sum salario;  
run;
```

```
171 options msglevel=i;  
172  
173 proc print data=arq.cadastro(obs=5 bufno=7);  
174     var nome empresa funcao salario;  
175     where empresa ne "";  
INFO: Index empfunc selected for WHERE clause optimization.  
176     by empresa funcao;
```

```
INFO: Index empfunc selected for BY clause processing.  
NOTE: An index was selected to execute the BY statement.  
The observations will be returned in index order rather than in physical order. The selected index is for the variable(s):
```

```
  empresa  
  funcao  
177     sum salario;  
178 run;
```

```
NOTE: There were 5 observations read from the data set ARQ.CADASTRO.  
WHERE empresa not = ' ';  
NOTE: PROCEDURE PRINT used (Total process time):  
real time          0.00 seconds  
cpu time           0.00 seconds
```

Ex.44 – INDEX não utilizado

```
proc print data=arq.cadastro;  
  var nome cpf empresa funcao salario;  
  where empresa='MALTA LTDA' ;  
run;
```

SAS Log

```
60  proc print data=arq.cadastro;  
61      var nome cpf empresa funcao salario;  
62      where empresa='MALTA LTDA' ;  
INFO: Index empfunc not used. Sorting into index order may help.  
INFO: Index empfunc not used. Increasing bufno to 6 may help.  
63  run;
```

NOTE: There were **174 observations** read from the data set ARQ.CADASTRO.

WHERE empresa='MALTA LTDA';

NOTE: PROCEDURE PRINT used (Total process time):

real time 0.00 seconds

cpu time 0.00 seconds

OBS1: São selecionados 174 registros o que representa 31,63% dos dados do arquivo cadastro (Total: 550); o SAS provavelmente utilizaria o index, no entanto, não utiliza, pois a partir de uma análise da distribuição dos dados, o consumo com acessos a disco seria menor com o processamento seqüencial;

OBS2: O log da execução ainda indica o que pode ser feito para, num processamento futuro, o index seja utilizado: ordenar os dados pela chave indexada ou aumentar o número de páginas de dados na memória durante o processamento, a opção BUFNO; **Atenção!** Ordenar os dados pela chave indexada destrói o index o que inviabiliza essa opção. A ordenação deve ser feita antes da indexação.

```
proc print data=arq.cadastro(bufno=6);  
  var nome cpf empresa funcao salario;  
  where empresa='MALTA LTDA' ;  
run;
```

SAS Log

```
71  proc print data=arq.cadastro(bufno=6);  
72      var nome cpf empresa funcao salario;  
73      where empresa='MALTA LTDA' ;  
INFO: Index empfunc selected for WHERE clause optimization.  
74  run;
```

NOTE: There were 174 observations read from the data set ARQ.CADASTRO.

WHERE empresa='MALTA LTDA';

NOTE: PROCEDURE PRINT used (Total process time):

real time 0.00 seconds

cpu time 0.00 seconds

OBS3: A primeira variável utilizada na definição de um index composto é chamada de chave primária do index e, por sua vez, pode ser utilizada sozinha, como uma chave indexada simples. **Somente a primeira variável ou primária!**

Index composto: empfunc → (empresa funcao)
 primária secundária

3.2.2 - Situações em que o SAS, com certeza, não utilizará o index

- Comando IF/THEN/ELSE

O index não funciona em expressões do comando IF em Data Step. Os dados serão selecionados de forma seqüencial.

- Com algumas condições do comando WHERE

Expressões com o operador lógico **OR** no comando WHERE, o index não é selecionado para o processamento. Os dados serão selecionados de forma seqüencial.

3.2.3 - Utilização de Index Composto

Para a utilização de um index composto, a expressão do comando WHERE deve ser construída utilizando-se as variáveis que compõem o index, pelo menos uma das expressões com o operador relacional = ou **IN**, e as expressões ligadas pelo operador **AND**. Se apenas a variável primária for utilizada na expressão, é possível que o SAS, também, utilize o index composto.

Ex.45 – INDEX composto

```
proc print data=arq.cadastro;
  var nome cpf empresa funcao salario;
  where empresa='MALTA LTDA' and funcao='ANALISTA' ;
run;
```

SAS Log

```
86  proc print data=arq.cadastro;
87      var nome cpf empresa funcao salario;
88      where empresa='MALTA LTDA' and funcao='ANALISTA' ;
INFO: Index empfunc selected for WHERE clause optimization.
89  run;
```

NOTE: There were 4 observations read from the data set ARQ.CADASTRO.

WHERE (empresa='MALTA LTDA') and (funcao='ANALISTA');

NOTE: PROCEDURE PRINT used (Total process time):

```
real time          0.00 seconds
cpu time           0.00 seconds
```

The SAS System

Obs	nome	cpf	empresa	funcao	salario
2	MOUA, MARCO	01211304778	MALTA LTDA	ANALISTA	9988.53
223	MOUA, MADALENA	35685046178	MALTA LTDA	ANALISTA	14821.37
262	MOUA, MONICA	44358657634	MALTA LTDA	ANALISTA	12568.82
397	MOUA, ELIANE	72233587148	MALTA LTDA	ANALISTA	13694.19

3.2.4 - Controlar o uso do Index no Comando WHERE

Existem duas opções de arquivo que determinam como será utilizado o index no processamento dos dados com o comando WHERE, caso exista um index na expressão:

IDXWHERE=YES ou NO Força o uso do processamento utilizando o index ou não, da expressão do comando WHERE, desconsiderando a análise do SAS;

IDXNAME=<nome do index> Especifica qual index utilizar, caso apareça mais de um, na expressão do comando WHERE;

Ex.46 – Determinar o uso do INDEX

```
proc print data=arq.cadastro(idxwhere=no);
  var nome cpf empresa funcao salario;
  where cpf in ("04979489998","02437587453","13064972095");
run;
SAS Log
189 proc print data=arq.cadastro(idxwhere=no);
190   var nome cpf empresa funcao salario;
191   where cpf in ("04979489998","02437587453","13064972095");
INFO: Data set option (IDXWHERE=NO) forced a sequential pass of the data rather than use of an index for where-clause processing.
192 run;
NOTE: There were 3 observations read from the data set ARQ.CADASTRO.
      WHERE cpf in ('02437587453', '04979489998', '13064972095');
NOTE: PROCEDURE PRINT used (Total process time):
      real time          0.00 seconds
      cpu time           0.00 seconds

proc print data=arq.cadastro;
  var nome cpf empresa funcao salario;
  where cpf in ("04979489998","02437587453","13064972095") and
    empresa='ATLAS S.A.' and funcao='PROGRAMADOR';
run;
SAS Log
204 proc print data=arq.cadastro;
205   var nome cpf empresa funcao salario;
206   where cpf in ("04979489998","02437587453","13064972095") and
207         empresa='ATLAS S.A.' and funcao='PROGRAMADOR';
INFO: Index empfunc selected for WHERE clause optimization.
208 run;
NOTE: There were 1 observations read from the data set ARQ.CADASTRO.
      WHERE cpf in ('02437587453', '04979489998', '13064972095') and (empresa='ATLAS S.A.') and
      (funcao='PROGRAMADOR');
NOTE: PROCEDURE PRINT used (Total process time):
      real time          0.00 seconds
      cpu time           0.00 seconds

proc print data=arq.cadastro(idxname=cpf);
  var nome cpf empresa funcao salario;
  where cpf in ("04979489998","02437587453","13064972095") and
    empresa='ATLAS S.A.' and funcao='PROGRAMADOR';
run;
SAS Log
210 proc print data=arq.cadastro(idxname=cpf);
211   var nome cpf empresa funcao salario;
212   where cpf in ("04979489998","02437587453","13064972095") and
213         empresa='ATLAS S.A.' and funcao='PROGRAMADOR';
INFO: Index cpf selected for WHERE clause optimization.
214 run;
NOTE: There were 1 observations read from the data set ARQ.CADASTRO.
      WHERE cpf in ('02437587453', '04979489998', '13064972095') and (empresa='ATLAS S.A.') and
      (funcao='PROGRAMADOR');
NOTE: PROCEDURE PRINT used (Total process time):
      real time          0.01 seconds
      cpu time           0.01 seconds
```

3.3 – Recomendações sobre Indexação

- Não crie index em arquivos pequenos, que ocupam menos de três páginas de dados; no ambiente windows, á princípio, uma página de dados possui 16Kbytes de informação; Em arquivos pequenos, o processamento seqüencial é mais, eficiente.
- Considere o custo de manutenção de um index em um arquivo de dados que é freqüentemente alterado;
- Crie um index quando perceber a necessidade de recuperar uma pequena quantidade de registros em relação ao total do arquivo (recuperar ~25% do total de dados de um arquivo);
- Crie o mínimo possível de indexes, para reduzir o espaço de armazenamento em disco do arquivo index e reduzir os custos de manutenção;
- Crie index com variáveis discriminantes, ou seja, variáveis que possuem vários grupos de dados:

SEXO Não é uma variável discriminante, possui apenas 2 grupos;
ESTADO Variável discriminante, representa os estados do Brasil, possui 27 grupos;
- Para melhorar a performance, se possível, ordene os dados pela variável ou variáveis que serão utilizadas para indexação, antes de indexar;
- Não crie index para satisfazer a todas as expressões do comando WHERE. O custo seria muito alto!
- Na criação de um index composto, utilize como variável primária, a variável mais discriminante;

3.4 – Manutenção de Index

- Cópia do arquivo de dados, através do SAS → O index é criado no novo arquivo;
- Re-nomear o arquivo de dados → O arquivo index é renomeado;
- Re-nomear a variável indexada → O index é renomeado;
- Adicionar dados no arquivo de dados → É adicionado ao index o novo valor;
- Eliminar dados do arquivo → É eliminado do index o valor;
- Atualizar dados do arquivo → O index é atualizado;
- O Arquivo de dados é eliminado → O index é eliminado;
- O arquivo é reconstruído em Data Step → O index é eliminado;
- O arquivo é ordenado pela PROC SORT → O index é eliminado;

4º Laboratório – Indexação dos Dados

1 – Utilizando a PROC DATASETS, crie três indexes no arquivo ANALISE:

- Index simples variável: **usuario** OBS: Não pode possuir valores missing;
- Index simples variável: **queue** OBS: Não pode possuir valores missing;
- Index composto variáveis: **estado,inst**

Verifique a estrutura do arquivo com a PROC CONTENTS.

2 – Repita o exercício anterior, mas utilizando a PROC SQL. Lembre-se, não é possível criar o mesmo index novamente, portanto, será necessário destruir os indexes criados no exercício anterior. É possível forçar o index a não possuir valores missing?

3 – Crie um novo arquivo SAS a partir do arquivo ANALISE, com a mesma definição de index utilizado nos exercícios anteriores, através do comando DATA de Data step, mas somente com os dados sobre *processamento paralelo*, ou seja, para valores da variável **queue**: [paralela](#), [gaussian](#), [exp32](#) e [par22](#)

4 – Crie uma tabela de frequência que informa quantos processos foram executados na fila [paralela](#), por instituições, somente do estado de São Paulo. Verifique no log, se algum index foi utilizado. Execute novamente o programa, mas dessa vez, force o uso do index simples **queue**.

5 – Crie uma tabela com o PROC TABULATE, com o percentual e numero de processos que foram finalizados (data maior que zero), apenas dos usuários da [UNICAMP](#)

Variáveis: **usuario, data_fim, inst**

Função que gera percentual, frequência e totalizador: **pctn, n, all**

Tabela: Nas linhas: usuários e totalizador;
 Nas colunas: percentual, frequência e totalizador;

OBS: Tente montar o comando WHERE, de maneira que ele utilize o index composto do arquivo;

6 – Crie um relatório, utilizando a PROC PRINT:

- Somente do usuário [kusel](#), (**usuario**) ;
- Agrupados por fila de execução (**queue**) ;
- Somente processos que finalizaram com sucesso (**data_fim maior que zero**);
- Inclua no relatório as variáveis: identificação do processo (**id**), data do fim da execução (**data_fim**) e tempo de espera para ser executado (**tesp**) ;
- Somatório do tempo de espera;

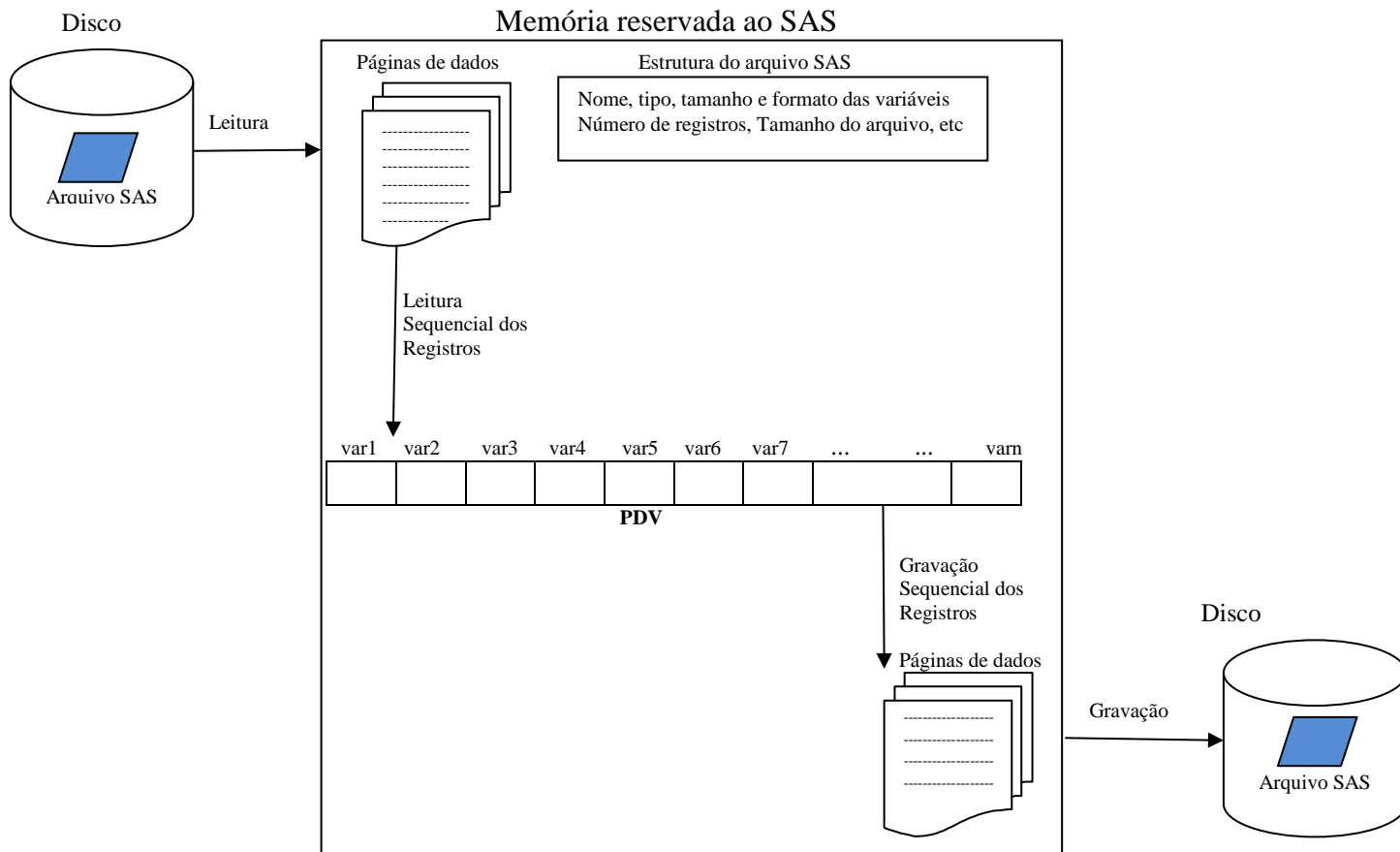
4 – Recursos de Programação em SAS

4.1 – Performance

Três recursos básicos de um computador, influenciam na performance de um programa: **cpu**, **memória** e **leitura e gravação (I/O)** em disco. Esses três recursos, normalmente, são inversamente proporcionais, a melhora no uso de um recurso implica na piora do outro.

Melhorar o uso da cpu ↔ Aumentar o uso de mais memória

Usar menos memória ↔ Aumentar o uso de disco



Existem opções no SAS que permitem controlar o número de acessos de leitura e gravação em disco, normalmente aumentando ou diminuindo o uso da memória e melhorando o tempo de processamento. O SAS automaticamente reconhece o número de CPUs existentes no computador e utiliza essas CPUs para processamento paralelo em alguns procedimentos.

4.1.1 - Opção BUFSIZE

- Controla tamanho em Kbytes das páginas de dados. Pode ser uma opção de arquivo ou uma opção geral; O valor especificado por essa opção, só é utilizado quando o arquivo está sendo criado. Esse tamanho fica armazenado na estrutura do arquivo. Toda vez que se processa o arquivo, o tamanho armazenado é utilizado para definir o tamanho das páginas de dados na memória; esse valor não pode ser sobreposto por um novo valor da opção; O SAS define automaticamente um tamanho de página de dados, que pode variar de 4Kbytes até 16Kbytes.

options BUFSIZE=<nKbytes|nMbytes|nGbytes> ;

<arquivo SAS> (BUFSIZE=<nKbytes|nMbytes|nGbytes>) ;

4.1.2 - Opção BUFNO

- Controla o número de páginas de dados na memória. Esse valor não é armazenado na estrutura do arquivo, portanto, pode variar à cada execução do programa. O padrão no ambiente Windows é de 1 página.

options BUFNO=<número de páginas> ;

<arquivo SAS> (BUFNO=<número de páginas>) ;

4.1.3 – Comando SASFILE

- Comando que automaticamente define um número de páginas de dados suficientes para armazenar todo um arquivo de dados em memória; se não houver memória suficiente para armazenar o arquivo, o comando não é executado e uma mensagem de aviso aparece no log informando o quanto de memória faltou para a alocação dos dados; o arquivo fica carregado na memória enquanto a sessão SAS estiver ativa ou, até finalizar o uso com o comando SASFILE novamente; a princípio um arquivo em memória só será utilizado para leitura, não sendo possível atualizar variáveis, inserir dados ou mesmo recriar o arquivo, mas, existem exceções (PROC APPEND e PROC SQL);

SASFILE <nome do arquivo> OPEN | LOAD | CLOSE ;

OPEN	Reserva a memória, mas só carrega o arquivo quando for utilizado pela 1ª vez;
LOAD	Reserva a memória e carrega imediatamente o arquivo;
CLOSE	Fecha o arquivo e libera a memória.

4.1.4 -Opção FULLSTIMER|NOFULLSTIMER

- Ativa, no log, algumas notificações sobre o uso de memória e do tempo de processamento. Importantes para analisar a performance de execução dos programas. Após a análise, desligue essa opção.

4.1.5 -Opção COMPRESS

- Ativa a compactação dos dados pelo SAS reduzindo o uso de espaço em disco.

options COMPRESS=NO|YES ou CHAR|BINARY ;

<nome do arquivo> (COMPRESS=NO|YES ou CHAR|BINARY) ;

YES ou CHAR	Deve ser utilizado quando a maioria das variáveis do arquivo forem caracteres;
BINARY	Deve ser utilizado quando a maioria das variáveis do arquivo forem numéricas;

Ex.47 – Verificação de Performance

```
data arq.cadastro1;
  infile "cadastro.dat" lrecl=130 missover;
  input nome $30. sexo $1. idade 2. @83 empresa $15. funcao $12. @119 salario commax12.2;
  output;output;output;output;output;output;output;output;output;
run;

options fullstimer;

proc means data=arq.cadastro1;
  class empresa funcao sexo;
  var idade salario;
run;
```

SAS Log (parcial: somente da PROC MEANS)

```
171 options fullstimer;
172
173 proc means data=arq.cadastro1;
174   class empresa funcao sexo;
175   var idade salario;
176 run;
```

NOTE: Multiple concurrent threads will be used to summarize data.
NOTE: There were 11968000 observations read from the data set ARQ.CADASTRO1.
NOTE: PROCEDURE MEANS used (Total process time):

real time	14.23 seconds
user cpu time	5.26 seconds
system cpu time	0.81 seconds
Memory	7088k
OS Memory	15000k
Timestamp	14/10/2010 11:13:31

```
178 options nofullstimer;
```

Informações do FULLSTIMER:

real time	Tempo total do processamento do Step. (CPU, acessos a memória, disco e rede);
user cpu time	Tempo somente de uso da CPU (Cálculo e lógica);
system cpu time	Tempo de CPU necessário para o sistema operacional executar o Step;
Memory	Memória necessária para executar o código, o Step;
OS Memory	Memória total que o Step requisitou ao sistema operacional, para executar;
Timestamp	Data e hora da execução do Step.

```
data arq.cadastro2(bufsize=200K);
  infile "cadastro.dat" lrecl=130 missover;
  input nome $30. sexo $1. idade 2. @83 empresa $15. funcao $12. @119 salario commax12.2;
  output;output;output;output;output;output;output;output;output;
run;

options fullstimer bufno=10;

proc means data=arq.cadastro2;
  class empresa funcao sexo;
  var idade salario;
run;
```

SAS Log (parcial: somente da PROC MEANS)

```
227 options fullstimer bufno=10;
228
229 proc means data=arq.cadastro2;
230   class empresa funcao sexo;
231   var idade salario;
232 run;
```

NOTE: Multiple concurrent threads will be used to summarize data.
NOTE: There were 11968000 observations read from the data set ARQ.CADASTRO2.
NOTE: PROCEDURE MEANS used (Total process time):

real time	4.37 seconds
user cpu time	5.54 seconds
system cpu time	1.78 seconds
Memory	12564k
OS Memory	20612k
Timestamp	14/10/2010 11:24:05

Ex.48 – Performance utilizando o comando SASFILE

```
options msglevel=i fullstimer;

sasfile arq.cadastr01 load;

proc means data=arq.cadastr01;
  class empresa funcao sexo;
  var idade salario;
run;

sasfile arq.cadastr01 close;

options nofullstimer;
```

247 options msglevel=i fullstimer;
248
249 sasfile arq.cadastr01 load;
NOTE: The file ARQ.CADASTRO1.DATA has been loaded into memory by the SASFILE statement.
250
251 proc means data=arq.cadastr01;
252 class empresa funcao sexo;
253 var idade salario;
254 run;

NOTE: Multiple concurrent threads will be used to summarize data.
NOTE: There were 11968000 observations read from the data set ARQ.CADASTRO1.
NOTE: PROCEDURE MEANS used (Total process time):

real time	2.90 seconds	
user cpu time	5.46 seconds	
system cpu time	0.06 seconds	
Memory		7034k
OS Memory		1113496k
Timestamp	14/10/2010	11:44:21

255
256 sasfile arq.cadastr01 close;
NOTE: The file ARQ.CADASTRO1.DATA has been closed by the SASFILE statement.
257
258 options nofullstimer;

4.2 – Combinar Arquivos Utilizando Indexes

Dois arquivos SAS podem ser combinados para selecionar dados e criar um novo arquivo SAS. Um dos arquivos teria a chave dos dados que seriam selecionados no segundo arquivo, que por sua vez, possuiria a chave indexada. A idéia é selecionar uma pequena amostra de dados de um arquivo muito grande.

SET <nome do arquivo> KEY=<chave indexada> ;

- O comando SET com a opção KEY permite abrir um arquivo SAS com acesso direto aos dados, via indexes. É necessário um segundo comando SET que informará os dados da chave para a seleção.

Arquivo Indexado

Variável indexada: matricula			
matricula	nome	sexo	idade
12321	Paulo	M	25
34500	Marcia	F	32
45680	Sandra	F	43
...
...
89000	Julio	M	38

Arquivo Seleção

matricula	curso
34500	Engenharia
89000	Ed.Física

Arquivo Resultado

matricula	nome	sexo	idade	curso
34500	Marcia	F	32	Engenharia
89000	Julio	M	38	Ed.Física

- Todo processamento no SAS que envolve o uso de indexes, ativa, automaticamente uma variável de controle das operações de leitura e gravação, a *_IORC_* (*Input Output Return Code*). Essa variável pode ser analisada, e indica se operação de leitura e busca com indexes foi com sucesso ou não.

IORC=0

IORC ne 0

Sucesso. Foi encontrado o valor da chave de busca.

Erro. Não foi encontrado o valor da chave de busca e ativa uma mensagem de erro no log (*_ERROR_*).

Ex.49 – Combinação utilizando Index

```
options msglevel=i nofullstimer;
data arq.ncpf;
  set arq.amostra;
  set arq.cadastro (keep=nome cpf rg empresa funcao salario)
    key=cpf;
  novosalario=salario*1.18;
run;

proc print data=arq.ncpf noobs;run;
```

SAS Log

```
133 options msglevel=i nofullstimer;
134 data arq.ncpf;
135   set arq.amostra;
136   set arq.cadastro (keep=nome cpf rg empresa funcao salario)
137     key=cpf;
138   novosalario=salario*1.18;
139 run;
```

```
nome=GUEDES,JOANA  cpf=01614523770  rg=  empresa=  funcao=  salario=.  novosalario=.  _ERROR_=1  _IORC_=1230015  _N_=1
nome=AUGUSTO,PEDRO  cpf=76355083900  rg=2946644SSPPR  empresa=PARIS INSTITUTO  funcao=PROGRAMADOR  salario=3031.38
novosalario=3577.0284  _ERROR_=1  _IORC_=1230015  _N_=10
NOTE: There were 10 observations read from the data set ARQ.AMOSTRA.
NOTE: The data set ARQ.NCPF has 10 observations and 7 variables.
NOTE: DATA statement used (Total process time):
      real time           0.01 seconds
      cpu time            0.01 seconds
```

The SAS System

nome	cpf	rg	empresa	funcao	salario	novosalario
GUEDES, JOANA	01614523770				.	.
SUNAY, MARCELO	03035241713	841697878SSPPR	PARIS INSTITUTO	PROGRAMADOR	1530.04	1805.45
SANTOS, CECILIA	05872838477	9283965897SSPPR	PARIS INSTITUTO	PROGRAMADOR	4226.21	4986.93
GUEDES, ROSANE	12365709767	3002200SSPRJ		DESEMPREGADO	.	.
YATAKA, LIGIA	15253254937	93951898SSPSP	ATLAS S.A.	PROGRAMADOR	3131.49	3695.16
SONTAS, MARCELO	33332201134	448607878SSPSP	ATLAS S.A.	PROGRAMADOR	3448.63	4069.38
CERTO, ROSANE	42862709770	9002400SSPPR	PARIS INSTITUTO	PROGRAMADOR	3355.42	3959.40
ANJOA, MIRIAM	55285998980	86440308SSPPE	ATLAS S.A.	PROGRAMADOR	3198.42	3774.14
SUNAY, LAURA	66572575644	2946644SSPPR	PARIS INSTITUTO	PROGRAMADOR	3031.38	3577.03
AUGUSTO, PEDRO	76355083900	2946644SSPPR	PARIS INSTITUTO	PROGRAMADOR	3031.38	3577.03

```
options msglevel=i nofullstimer;
data arq.ncpf;
  set arq.amostra;
  set arq.cadastro (keep=nome cpf rg empresa funcao salario)
    key=cpf;
  if _iorc_=0 then do;
    novosalario=salario*1.18;
    output;
  end;
  else _error_=0;
run;

proc print data=arq.ncpf noobs;run;
```

SAS Log

```
164 options msglevel=i nofullstimer ls=110;
165 data arq.ncpf;
166   set arq.amostra;
167   set arq.cadastro (keep=nome cpf rg empresa funcao salario)
168     key=cpf;
169   if _iorc_=0 then do;
170     novosalario=salario*1.18;
171     output;
172   end;
173   else _error_=0;
174 run;
```

```
NOTE: There were 10 observations read from the data set ARQ.AMOSTRA.
NOTE: The data set ARQ.NCPF has 8 observations and 7 variables.
NOTE: DATA statement used (Total process time):
      real time           0.01 seconds
      cpu time            0.00 seconds
```

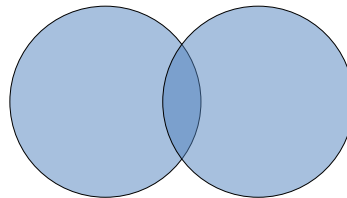
The SAS System

nome	cpf	rg	empresa	funcao	salario	novosalario
SUNAY, MARCELO	03035241713	841697878SSPPR	PARIS INSTITUTO	PROGRAMADOR	1530.04	1805.45
SANTOS, CECILIA	05872838477	9283965897SSPPR	PARIS INSTITUTO	PROGRAMADOR	4226.21	4986.93
GUEDES, ROSANE	12365709767	3002200SSPRJ		DESEMPREGADO	.	.
YATAKA, LIGIA	15253254937	93951898SSPSP	ATLAS S.A.	PROGRAMADOR	3131.49	3695.16
SONTAS, MARCELO	33332201134	448607878SSPSP	ATLAS S.A.	PROGRAMADOR	3448.63	4069.38
CERTO, ROSANE	42862709770	9002400SSPPR	PARIS INSTITUTO	PROGRAMADOR	3355.42	3959.40
ANJOA, MIRIAM	55285998980	86440308SSPPE	ATLAS S.A.	PROGRAMADOR	3198.42	3774.14
SUNAY, LAURA	66572575644	2946644SSPPR	PARIS INSTITUTO	PROGRAMADOR	3031.38	3577.03

4.3 – Combinação de Arquivos em DATA Step - Opção de Arquivo IN=

- O comando MERGE permite combinar arquivos, combinando os registros que possuem uma ou mais valores de chave, em comum. O resultado dessa combinação é um arquivo com todos os dados de todos os arquivos combinados (FULL JOIN).

```
DATA <arquivo> ;  
  MERGE <arquivo1> <arquivo2> ... <arquivon> ;  
  BY <chave1> ... <chaven> ;  
RUN ;
```

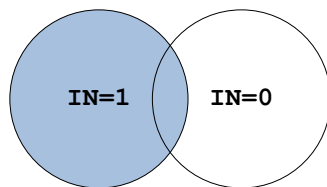


FULL JOIN

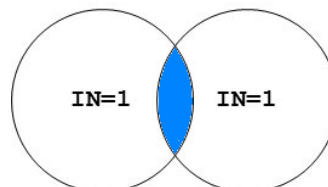
- É possível combinar os arquivos em Data Step e controlar os dados que serão salvos no novo arquivo: Somente os dados que combinam; Somente os dados que não combinam; Todos os dados de um determinado arquivo mais os dados que combinam. Esse controle pode ser feito com a opção IN= de arquivo, que cria uma variável lógica, cujo valor indica se o dado de um arquivo, combinou ou não, com o dado de outro arquivo pela chave comum.

```
DATA <arquivo> ;  
  MERGE <arquivo1> (IN=<variável1>)  
        <arquivo2> (IN=<variável2>)  
        ...  
        <arquivon> (IN=<variáveln>);  
  BY <chave1> ... <chaven> ;  
RUN ;
```

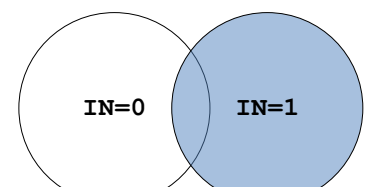
- A variável lógica definida na opção IN, pode assumir somente dois valores: **1 (Verdadeiro)**, o dado no arquivo existe para o valor da chave; **0 (Falso)**, o dado no arquivo não existe para o valor da chave. Analisando essa variável em um comando IF, podemos selecionar os dados que serão salvos no arquivo. As variáveis criadas com a opção IN, **não são salvas** no arquivo final.
- No caso em que apenas dois arquivos estão sendo combinados, podemos ter as seguintes situações:



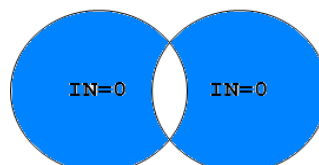
LEFT JOIN



INTERSECTION



RIGHT JOIN



EXCEPTION

Ex.50 – Combinação com opção IN

1 - Todos os dados

```
data arq3;
  merge arq1(in=a)
        arq2(in=b);
  by cod;
  esquerda=a;direita=b;
run;
proc print data=arq3 noobs;run;
```

The SAS System				
cod	nome	resp	esquerda	direita
1	Paulo	Sim	1	1
2	Julio	Não	1	1
3	Carlos		1	0
4	Mauricio	Sim	1	1
5	Sandra		1	0
6		Sim	0	1
7		Sim	0	1

2 - a=1 and b=1 **Somente os dados que combinam de arq1 e arq2;**

```
data arq3;
  merge arq1(in=a)
        arq2(in=b);
  by cod;
  esquerda=a;direita=b;
  if a=1 and b=1 then output;
run;
proc print noobs data=arq3;run;
```

The SAS System				
cod	nome	resp	esquerda	direita
1	Paulo	Sim	1	1
2	Julio	Não	1	1
4	Mauricio	Sim	1	1

3 - a=0 or b=0 **Somente os dados que não combinam de arq1 e arq2;**

```
data arq3;
  merge arq1(in=a)
        arq2(in=b);
  by cod;
  esquerda=a;direita=b;
  if a=0 or b=0 then output;
run;
proc print noobs data=arq3;run;
```

The SAS System				
cod	nome	resp	esquerda	direita
3	Carlos		1	0
5	Sandra		1	0
6		Sim	0	1
7		Sim	0	1

3 - a=1 **Todos os dados de arq1 mais os dados que combinam de arq2;**

```
data arq3;
  merge arq1(in=a)
        arq2(in=b);
  by cod;
  esquerda=a;direita=b;
  if a=1 then output;
run;
proc print noobs data=arq3;run;
```

The SAS System				
cod	nome	resp	esquerda	direita
1	Paulo	Sim	1	1
2	Julio	Não	1	1
3	Carlos		1	0
4	Mauricio	Sim	1	1
5	Sandra		1	0

4 - b=1 **Todos os dados de arq2 mais os dados que combinam de arq1;**

```
data arq3;
  merge arq1(in=a)
        arq2(in=b);
  by cod;
  esquerda=a;direita=b;
  if b=1 then output;
run;
proc print noobs data=arq3;run;
```

The SAS System				
cod	nome	resp	esquerda	direita
1	Paulo	Sim	1	1
2	Julio	Não	1	1
4	Mauricio	Sim	1	1
6		Sim	0	1
7		Sim	0	1

5 - a=0 **Todos os dados de arq1 que não combinam com arq2;**

```
data arq3;
  merge arq1(in=a)
        arq2(in=b);
  by cod;
  esquerda=a;direita=b;
  if a=0 then output;
run;
proc print noobs data=arq3;run;
```

The SAS System				
cod	nome	resp	esquerda	direita
6		Sim	0	1
7		Sim	0	1

6 - b=0 **Todos os dados de arq2 que não combinam com arq1;**

```
data arq3;
  merge arq1(in=a)
        arq2(in=b);
  by cod;
  esquerda=a;direita=b;
  if b=0 then output;
run;
proc print noobs data=arq3;run;
```

The SAS System				
cod	nome	resp	esquerda	direita
3	Carlos		1	0
5	Sandra		1	0


4.4 – Conjuntos de Dados - “ARRAYs”

Em programação, as vezes é necessário repetir o mesmo cálculo para diversas variáveis, o que, a princípio, seria a repetição da formula para cada uma dessas variáveis. Em toda linguagem de programação existe o conceito de variável escalar que armazena apenas um dado e de variável vetorial que armazena uma estrutura de dados, e esses dados são recuperados através da sua posição dentro da estrutura.


Variável Escalar
Variável Vetorial

A=1
B=1, 2, 3, 4, 5, 6, 7, 8, 9, ...

```
data;
  a=10;
  aux=a*1.2;
run;
  aux=10*1.2
  aux=12
```



```
data;
  b=1, 2, 3, 4, ...; ???
  aux=b*1.2;
run;
  aux=1*1.2, 2*1.2, 3*1.2, ... ???
  aux=1.2, 2.4, 3.6
```



O ambiente de programação SAS não possui a definição de variável vetorial, mas permite a criação de conjuntos de variáveis ou conjunto de dados, o que facilita a programação repetitiva.

ARRAY <nome> <(tamanho do conjunto)> [\$] [tamanho] [elementos] [(valores iniciais)];

nome	Nome do conjunto de dados;
tamanho do conjunto	Quantidade de elementos que possui o conjunto. O * define o tamanho do conjunto, de acordo com o número de elementos;
\$	Indica que os elementos serão caracteres;
tamanho	Indica o tamanho dos elementos caracteres;
elementos	Variáveis que irão compor o conjunto de dados: lista de variáveis, _CHARACTER_, _NUMERIC_, _TEMPORARY_
valores iniciais	Valor inicial de cada variável;

```
array dias(7) d1-d7;
array mes(*) jan feb jul oct nov;
array x(*) _NUMERIC_;
array qbx(10);
array meal(3);
array test(4) t1 t2 t3 t4 (90 80 70 70);
array test(4) _TEMPORARY_ (90 80 70 70);
array test2(*) $ a1 a2 a3 ('a','b','c');
array x(5,3) score1-score15;
```

Ex.51 – Conjunto de Dados – Cálculo Repetitivo

A partir de um arquivo com o nome do poupador e um valor inicial, gerar um novo arquivo com a simulação do crescimento do valor inicial a uma taxa pré-fixada, em 12 meses e armazenando o valor creditado de cada mês.

Solução SEM utilização de ARRAY

```
data simular;
  set poupanca;
  tx=1.015;
  mes1=inicial;
  mes2=mes1*tx;
  mes3=mes2*tx;
  mes4=mes3*tx;
  mes5=mes4*tx;
  mes6=mes5*tx;
  mes7=mes6*tx;
  mes8=mes7*tx;
  mes9=mes8*tx;
  mes10=mes9*tx;
  mes11=mes10*tx;
  mes12=mes11*tx;
run;

proc print noobs data=simular; run;
```

```

                                     The SAS System

nome  inicial  tx  mes1  mes2    mes3    mes4    mes5    mes6    mes7    mes8    mes9    mes10    mes11    mes12
Carlos   10   1.015  10  10.150 10.3022 10.4568 10.6136 10.7728 10.9344 11.0984 11.2649 11.4339 11.6054 11.7795
Flavia   12   1.015  12  12.180 12.3627 12.5481 12.7364 12.9274 13.1213 13.3181 13.5179 13.7207 13.9265 14.1354
Luis     45   1.015  45  45.675 46.3601 47.0555 47.7614 48.4778 49.2049 49.9430 50.6922 51.4525 52.2243 53.0077
```

Solução COM utilização de ARRAY

```
data simular (drop=i);
  set poupanca;
  tx=1.015;
  array mes(12) mes1-mes12;
  mes(1)=inicial;
  do i=2 to 12;
    mes(i)=mes(i-1)*tx;
  end;
run;

proc print noobs data=simular; run;
```

```

                                     The SAS System

nome  inicial  tx  mes1  mes2    mes3    mes4    mes5    mes6    mes7    mes8    mes9    mes10    mes11    mes12
Carlos   10   1.015  10  10.150 10.3022 10.4568 10.6136 10.7728 10.9344 11.0984 11.2649 11.4339 11.6054 11.7795
Flavia   12   1.015  12  12.180 12.3627 12.5481 12.7364 12.9274 13.1213 13.3181 13.5179 13.7207 13.9265 14.1354
Luis     45   1.015  45  45.675 46.3601 47.0555 47.7614 48.4778 49.2049 49.9430 50.6922 51.4525 52.2243 53.0077
```

Ex.52 – Conjunto de Dados – Tabela Temporária e Criação de Variáveis

Conhecendo-se o valor do dólar em relação ao real, nos 6 primeiros meses de 2009, verifique a variação do valor em dólar de uma determinada aplicação inicial.

```
data variacao (drop=i);
  aplicacao=15000;
  array dolar2009(6) _temporary_ (2.3803,2.2680,2.3012,2.1992,2.0762,1.9420);
  array mes(6);
  array va(6);
  do i=1 to 6;
    mes(i)=aplicacao*dolar2009(i);
    if i>1 then va(i)=mes(i)-mes(i-1);
  end;
end;
```

```
run;
```

```
proc print data=variacao noobs;
  var mes1 va2 mes2 va3 mes3 va4 mes4 va5 mes5 va6 mes6;
  format mes1-mes6 va1-va6 commax9.2;
```

```
run;
```

The SAS System

mes1	va2	mes2	va3	mes3	va4	mes4	va5	mes5	va6	mes6
35.704,50	-1.684,50	34.020,00	498,00	34.518,00	-1.530,00	32.988,00	-1.845,00	31.143,00	-2.013,00	29.130,00

```
data;
```

```
array diasem(7) $ 15 _temporary_ ("Domingo","Segunda-feira","Terça-feira",
  "Quarta-feira","Quinta-feira","Sexta-feira",
  "Sábado");
```

```
array meses(12) $ 15 _temporary_ ("Janeiro","Fevereiro","Março",
  "Abril","Maio","Junho",
  "Julho","Agosto","Setembro",
  "Outubro","Novembro","Dezembro");
```

```
data=today();
diasemana=weekday(data);
dia=day(data);
mes=month(data);
ano=year(data);
```

```
/* Alternativa 1 */
hoje1=catx(' ',diasem(diasemana),', ',dia,'de',meses(mes),',de',ano);
```

```
/* Alternativa 2 */
hoje2=catx(' ',diasem(weekday(today()))', ',day(today()),',de',meses(month(today()))',',de',year(today()));
```

```
/* Alternativa 3 */
hoje3=data;
format hoje3 ptgdfwkx38.;
```

```
run;
```

```
options ls=64;
```

```
proc print noobs width=min;run;
```

The SAS System

data	diasemana	dia	mes	ano
18577	5	11	11	2010

hoje1

Quinta-feira , 11 de Novembro de 2010

hoje2

Quinta-feira , 11 de Novembro de 2010

hoje3

Quinta-feira, 11 de novembro de 2010

5º Laboratório – Recursos de Data Step – Combinar Arquivos e “Arrays”

1 – Com SET/INDEX.

- O arquivo **lavouras**, localizado na pasta **c:\curso\sas2**, é um arquivo cujos dados representam a produção de lavouras permanentes, por estado, em 2009. Possui uma **chave indexada composta**, por **estado** e **produto**, de nome **“estprod”**.
- O arquivo **lavprev**, também localizado na pasta **c:\curso\sas2**, é um arquivo cujos dados representam apenas dez lavouras com previsão de crescimento para o próximo ano, no valor da “quantidade produzida”.
- Combine esses dois arquivos pela chave indexada, utilizando o método de dois comandos **SET**. Adicione a operação aritmética que calcula uma previsão para a próxima safra da quantidade do produto:

```
quant_prevista=quant*previsao;
```

OBS: Verifique no LOG, se ocorrem incoerências nos resultados. Salve no novo arquivo, apenas os dados que combinam, de acordo com a variável interna do SAS, **_IORC_**.

2 – Com MERGE/INTERSECTION

- Repita o exercício anterior, mas utilizando o comando **MERGE** para combinar os arquivos. Lembre-se, salve **somente** os dados que combinam.

3 – Com MERGE/LEFT ou RIGHT JOIN

- Altere o exercício 2, de maneira que, gere um relatório com todos os produtos do arquivo **lavprev**, os que combinam e os que não combinam.

4 – Com MERGE/LEFT ou RIGHT JOIN

- Altere novamente o exercício 2, de maneira que, gere um relatório com todos os produtos do arquivo **lavouras**, incluindo os que não combinam.

5 – Com MERGE/EXCEPTION

- Altere novamente o exercício 2, de maneira que, gere um relatório com todos os dados que **não** combinam.

6 – Leia os dados do arquivo sas, **lavouras** ; Altere todas as variáveis numéricas dos produtos que possuam na sua descrição, as palavras: **"Banana"** ou **"Laranja"** ou **"Cacau"** ou **"Cafe"** ou **"Goiaba"**, multiplicando por 5 o valor original dessas variáveis.

- a) Dica1: Crie um conjunto de dados temporário com os produtos: **Banana , Laranja , Cacau , Cafe , Goiaba;**
- b) Dica2: Crie um conjunto de dados com todas as variáveis numéricas;
- c) Dica3: Nem sempre é possível determinar quantas variáveis numéricas existem em um arquivo; utilize uma palavra reservada do SAS que identifica a lista de variáveis numéricas: **numeric_**.
- d) Dica4: Como não é possível determinar o número de variáveis numéricas, represente com ***** a dimensão do conjunto de dados;
- e) Dica5: Existe uma função de tratamento de conjuntos que retorna o valor da dimensão de um conjunto: **DIM(conjunto)**;

7 – O Arquivo SAS, **temperaturas**, possui os dados de temperatura (média, média máxima, média mínima, máxima absoluta, mínima absoluta) da região de Campinas de 1988 a 2008, em graus Centígrados. São doze meses de temperaturas como variáveis e apenas 5 registros indicando o tipo de temperatura. Converta todos esses valores de temperatura para graus Fahrenheit, utilizando a fórmula:

```
tempF=(tempC*9/5)+32 ;
```

4.5 - Comando SELECT-WHEN-OTHERWISE-END

Construção alternativa para execução condicional de comandos, servindo como substituto do comando IF e melhorando visualmente a lógica de execução dos comandos;

```
SELECT [ (expressão) ] ;
    WHEN <(expressão1, expressão2, ..., expressão)> <comando> ;
    WHEN <(expressão1, expressão2, ..., expressão)> <comando> ;
    WHEN <(expressão1, expressão2, ..., expressão)> <comando> ;
    ...
    [OTHERWISE [comando] ] ;
END ;
```

- 1) A expressão lógica do comando SELECT não é obrigatória;
- 2) Pelo menos um WHEN deve existir em uma estrutura SELECT-END;
- 3) Pelo menos, uma expressão lógica no WHEN é obrigatória;
- 4) Cada WHEN pode conter várias expressões lógicas separadas por vírgulas, que representam o operador OR;
- 5) Cada WHEN só permite a execução de um único comando, se pelo menos, uma das expressões for verdadeira;
- 6) Se todas as expressões, de todos os WHEN forem falsas, então será executado o comando que vier após o OTHERWISE;
- 7) O comando OTHERWISE não é obrigatório;

Ex.53 – Select-When

```
data analistas programadores chefes desempregados;
  set sas2.cadastro;
  select(funcao);
    when ("ANALISTA") output analistas;
    when ("GERENTE","DIRETOR") output chefes;
    when ("PROGRAMADOR") do;
      sal=salario*1.15;
      output programadores;
    end;
  otherwise output desempregados;
end;
run;
```



```
data analistas (drop=sal) programadores chefes (drop=sal) desempregados (drop=sal);
  set sas2.cadastro;
  select;
    when (funcao="ANALISTA") output analistas;
    when (funcao="GERENTE",funcao="DIRETOR") output chefes;
    when (funcao="PROGRAMADOR") do;sal=salario*1.15;output programadores;end;
    when (funcao="DESEMPREGADO") output desempregados;
  end;
run;
```



```
data analistas programadores chefes desempregados;
  set sas2.cadastro;
  select;
    when (funcao="ANALISTA") output analistas;
    when (funcao in ("GERENTE", "DIRETOR")) output chefes;
    when (funcao="PROGRAMADOR") output programadores;
  end;
run;
```



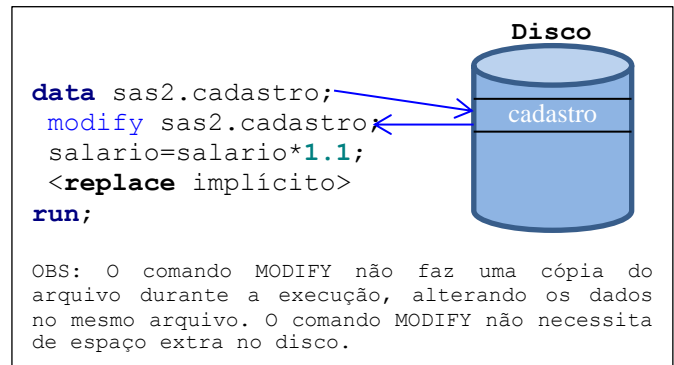
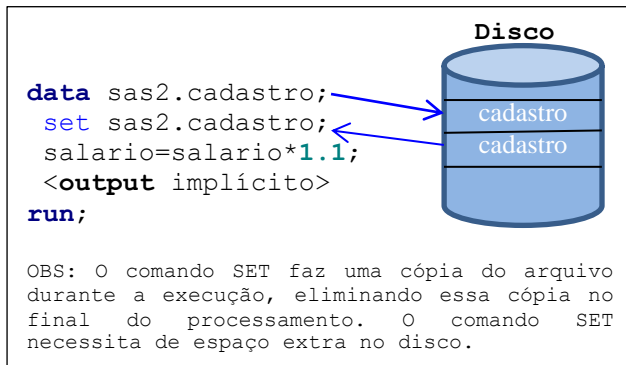
SAS Log (parcial)

```
ERROR: Unsatisfied WHEN clause and no OTHERWISE clause at line 141 column 5.
nome=MARKO,PAULO sexo=M idade=26 peso=69.3 altura=1.71 aniversario=7596 e_civil=2 filhos=5
rg=04156809SSPMG cpf=01017503989 empresa= funcao=DESEMPREGADO admissao=. salario=. t_emp=
_ERROR_=1 _N_=1
NOTE: The SAS System stopped processing this step because of errors.
NOTE: There were 1 observations read from the data set SAS2.CADASTRO.
```

4.6 – Comando MODIFY

Comando que possui como principal função, somente modificar, substituir os dados que existem em um arquivo SAS, sendo a sua ação similar ao do comando SET, mas com execuções diferentes.

```
DATA <arquivo mestre>;  
    MODIFY <arquivo mestre>;  
    ...  
RUN;
```



- 1) O comando MODIFY substitui os dados que existem em um arquivo. Não é possível alterar a estrutura de um arquivo, não é possível salvar uma variável nova no arquivo que está sendo criada no Data step.
- 2) Não é possível criar um novo arquivo a partir da leitura de um arquivo utilizando o comando MODIFY, ou seja, o nome do arquivo no comando DATA deve ser o mesmo utilizado no comando MODIFY.
- 3) Normalmente o comando MODIFY executa um comando REPLACE implícito no final do Data step, mas é possível utilizar o comando OUTPUT para salvar um novo registro e o comando REMOVE para eliminar o registro.

Ex.54

```
data sas2.cadastro2 (drop=salario);  
    modify sas2.cadastro2;  
    sal=salario*1.1;  
    put sal=;  
run;  
proc print;  
    var nome empresa salario sal;  
run;
```

OBS: O comando **PUT**, nesse contexto, é utilizado para imprimir dados no SAS Log, bastando informar o nome da variável com ou sem o sinal de igualdade.

SAS Log (parcial)

NOTE: Data set options for master data set should be specified in MODIFY statement.

```
sal=.  
sal=10987.383  
sal=.  
sal=4211.647  
sal=5608.306  
sal=3411.452  
...  
...  
...
```

NOTE: Missing values were generated as a result of performing an operation on missing values. Each place is given by: (Number of times) at (Line):(Column).

116 at 783:16

NOTE: There were 550 observations read from the data set SAS2.CADASTRO2.

NOTE: The data set SAS2.CADASTRO2 has been updated. There were 550 observations rewritten, 0 observations added and 0 observations deleted.

NOTE: DATA statement used (Total process time):

```
real time      0.34 seconds  
cpu time       0.34 seconds
```

786

```
787 proc print;
```

```
788     var nome empresa salario sal;
```

```
ERROR: Variable SAL not found.
```

```
789 run;
```

NOTE: The SAS System stopped processing this step because of errors.

NOTE: PROCEDURE PRINT used (Total process time):

```
real time      0.00 seconds  
cpu time       0.00 seconds
```

4.7 – Atualização de Dados com Arquivo de Transação

É possível utilizar o comando MODIFY para realizar modificações em um arquivo principal a partir de um arquivo secundário, desde que possuam uma chave/variável em comum;

```
DATA <mestre> ;  
  MODIFY <mestre> <transação> ;  
  BY <variável comum>  
  
  ...  
RUN;
```

- 1) A lógica de execução do comando MODIFY é de atualizar os dados no arquivo mestre com os dados do arquivo de transação. Não é combinação de registros como o comando MERGE;
- 2) Se houverem registros com chave duplicada no arquivo mestre, somente o primeiro registro do grupo, será atualizado com os dados do arquivo de transação;
- 3) Se houverem registros com chave duplicada no arquivo de transação, todas as transações serão aplicadas na ordem que aparecem;
- 4) O comando MODIFY **localiza os dados** no arquivo mestre, utilizando o processamento dinâmico do comando **WHERE (implícito)** com a chave do arquivo de transação;
- 5) **NÃO é necessário ordenar os dados em nenhum arquivo.**

Ex.55

Arquivo: sas2.altera

Obs	nome	sexo	idade	peso	altura	aniversario	e_civil	filhos	rg	cpf	empresa	funcao	admissao	salario	t_emp
1	MARKO,PAULO		01017503989	UNICAMP	ENGENHEIRO	16064	3000	
2	MALA,ELIANE		02131547749	USP	PROFESSOR	16100	3500	

```
data sas2.cadastro2;  
  modify sas2.cadastro2 sas2.altera;  
  by cpf;  
run;
```

where cpf="01017503989"

SAS Log (Parcial)

```
NOTE: There were 1 observations read from the data set SAS2.CADASTRO2.  
NOTE: The data set SAS2.CADASTRO2 has been updated. There were 2 observations rewritten, 0  
      observations added and 0 observations deleted.  
NOTE: There were 2 observations read from the data set SAS2.ALTERA.  
NOTE: DATA statement used (Total process time):  
      real time          0.03 seconds  
      cpu time           0.01 seconds
```

nome	sexo	idade	peso	altura	aniversario	e_civil	filhos	rg	cpf	empresa	funcao	admissao	salario	t_emp
MARKO,PAULO	M	26	69.3	1.71	7596	2	5	04156809SSPMG	01017503989	UNICAMP	ENGENHEIRO	16064	3000.00	
MOUA,MARCO	M	37	111.5	2.06	3471	2	4	245121098SSPMG	01211304778	MALTA LTDA	ANALISTA	13265	9988.53	LTDA
SANTOS,PAULO	M	20	76.1	1.78	9801	2	1	54352809SSPRJ	01518593290		DESEMPREGADO	.	.	
CERTO,CARLA	F	27	71.6	1.62	7255	1	4	25463800SSPPR	02043527085	PARIS INSTITUTO	PROGRAMADOR	15843	3972.30	INSTITUTO
MALA,ELIANE	F	27	80.4	1.89	6972	2	1	4864289032SSPRJ	02131547749	USP	PROFESSOR	16100	3500.00	

6º Laboratório – Recursos de Data Step – Parte 2

1 – Monte um programa SAS que leia o arquivo **analise** e que distribua em vários arquivos os jobs processados de acordo com a fila de execução. Utilize a estrutura SELECT-WHEN-OTHERWISE-END para efetuar a seleção.

- a) Criar os arquivos: **pequena, media, grande e paralelas**
- b) Filas: **pequena, media, grande**
- c) Filas paralelas: **paralela, exp32, par22, gaussian**

2 – Atualize os dados do arquivo **instituicao2** com os dados do arquivo **instalt**.

- a) Utilize o comando MODIFY para atualizar os dados, por usuário;
- b) Faça uma cópia do arquivo instituicao2, pois se ocorrer algum problema, o arquivo instituicao pode ser danificado;
- c) Verifique o log para possíveis mensagens de WARNING. Se houver, como prevenir esse problema?
- d) Execute uma comparação entre os dois arquivos:

```
proc compare base=sas2.instituicao compare=sas2.instituicao2; run;
```

BIBLIOGRAFIA

SAS® 9.2 SQL Procedure User's Guide

Copyright © 2009, SAS Institute Inc., Cary, NC, USA.
ISBN 978-1-599944-853-9

<http://support.sas.com/documentation/cdl/en/sqlproc/62086/HTML/default/titlepage.htm>

SAS® 9.2 Language Reference: Dictionary, Third Edition

Copyright © 2010, SAS Institute Inc., Cary, NC, USA
ISBN 978-1-60764-492-7

<http://support.sas.com/documentation/cdl/en/lrdict/63026/HTML/default/viewer.htm#titlepage.htm>

SAS® 9.2 Language Reference: Concepts, Second Edition

Copyright © 2010, SAS Institute Inc., Cary, NC, USA
ISBN 978-1-60764-448-4

<http://support.sas.com/documentation/cdl/en/lrcon/62955/HTML/default/viewer.htm#titlepage.htm>

SAS® 9.2 Macro Language: Reference

Copyright © 2009, SAS Institute Inc., Cary, NC, USA
ISBN 978-1-59994-708-2

<http://support.sas.com/documentation/cdl/en/mcrolref/61885/HTML/default/viewer.htm#titlepage.htm>

Step-by-Step Programming with Base SAS

Copyright © 2001 by SAS Institute Inc., Cary, NC, USA.
ISBN 978-1-58025-791-6

<http://support.sas.com/documentation/cdl/en/basess/58133/PDF/default/basess.pdf>