

# Bases de Dados

**Módulo 18a: Subconsultas em Instruções SELECT**

**Prof. André Bruno de Oliveira**

14/05/24 09:32

# Explorando banco de dados da POF

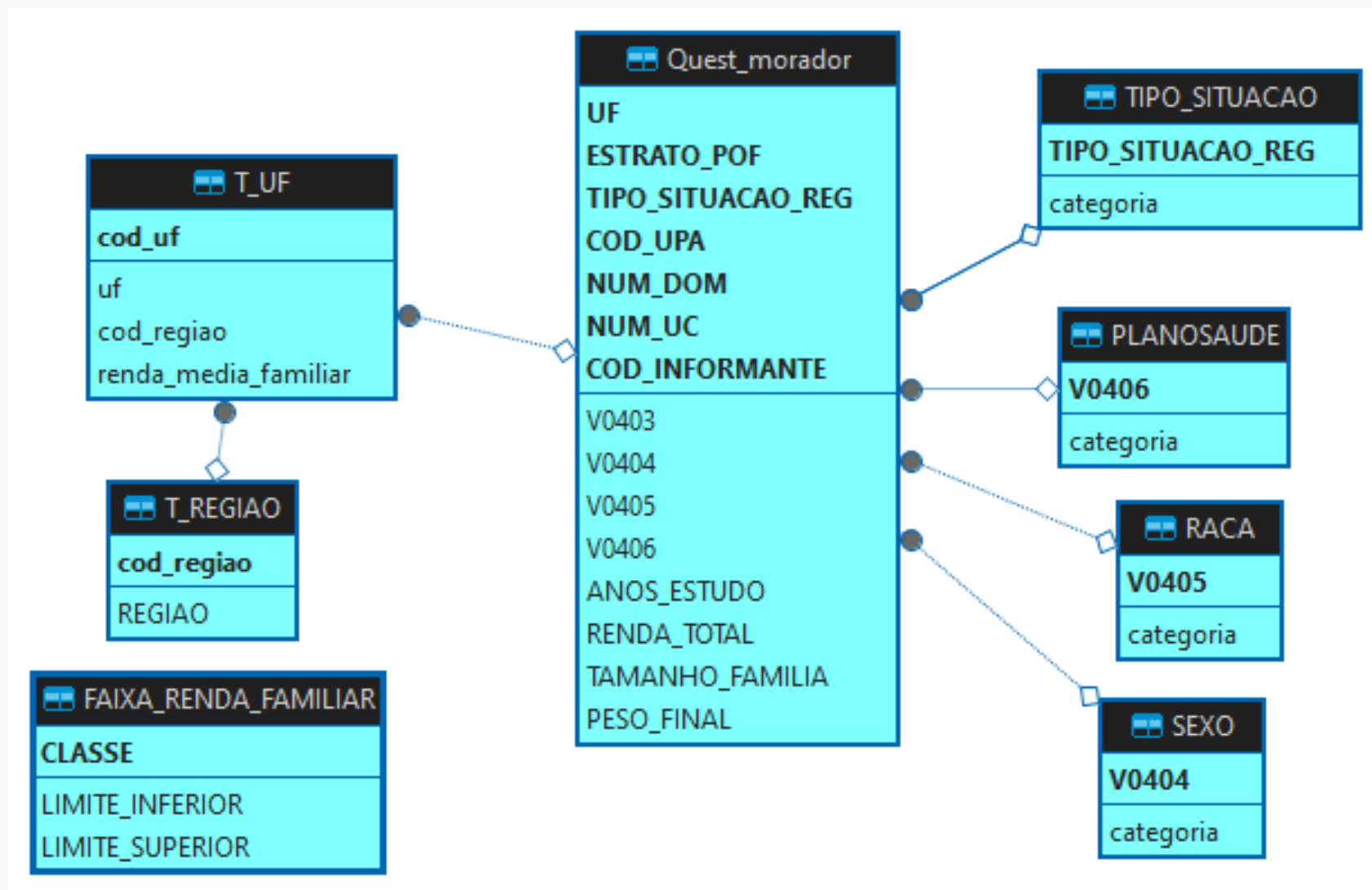
- Este módulo usa microdados da POF 2017-2018 para estudar subconsultas SQL.
- A Pesquisa de Orçamentos Familiares – POF avalia as estruturas de consumo, de gastos, de rendimentos e parte da variação patrimonial das **famílias**, oferecendo um perfil das condições de vida da população a partir da análise dos orçamentos domésticos.
- O desenho atual da amostra da POF propicia a publicação de resultados nos seguintes níveis: Brasil, Grandes Regiões, por situações urbana e rural, por Unidades da Federação, Regiões Metropolitanas e nos Municípios das Capitais.
- Os dados utilizados neste módulo são do questionário de Morador do domicílio.

# Explorando banco de dados da POF

- Este BD SQLite (POF.db) inclui uma versão simplificada da POF com 15 variáveis na tabela *Quest\_Morador*. A versão original do questionário do morador está em *Questionario* com 56 variáveis e 178.431 ocorrências.
- O banco de dados está disponível na pasta do professor.
- Os dados foram importados para o SQLite com uso do SQLite Studio (houve uma etapa anterior de importação para SAS e depois exportação para o formato CSV que o SQLite studio reconhece).
- A fonte de dados original juntamente com a documentação está disponível em <https://www.ibge.gov.br/estatisticas/sociais/populacao/24786-pesquisa-de-orcamentos-familiares-2.html?edicao=37681&t=downloads>.

# Explorando banco de dados da POF

- O BD POF.bd inclui 178.431 linhas com 8 tabelas apresentadas no diagrama da figura 1. A tabela central Quest\_Questionario possui 4 FK com 1 FK para cada tabela auxiliar:
  - T\_regiao (cod\_região, regiao),
  - T\_uf(cod\_uf, UF);
  - Tipo\_situação(tipo\_situação\_reg, categoria);
  - PlanoSaude (V0406, categoria);
  - Raca (V0405, categoria);
  - Sexo (V0404, categoria)
  - *Faixa\_renda\_familiar* (classe, limite\_inferior, limite\_superior)
- As tabelas auxiliares foram criadas e populadas primeiro, pois cada tabela inclui um atributo que faz parte da regra de integridade referencial (FK) da tabela principal Quest\_Questionario, com exceção da *Faixa\_renda\_familiar*. A tabela Faixa\_renda\_familiar foi criada a partir das informações do SIDRA.

**Figura 1 – Diagrama do BD da POF 2017-2018**

## Curiosidade

### O que SAS ?

“O SAS, no mercado desde 1976, é um dos mais reputados sistemas de análises de dados em microcomputadores, utilizado por cerca de 5.000 empresas no mundo inteiro . Trata-se de um sistema integrado de aplicações para o processamento e análise estatística de dados, consistindo em módulos de Acesso e Recuperação de Dados, Gerenciamento de Arquivos, rotinas de Geração de Gráficos e Geração de Relatórios.”

Fonte: <https://www2.ufjf.br/estatistica/eventos-e-projetos/projeto-sas/o-que-e-o-sas/>

## Curiosidade

### O que é SIDRA ?

Sistema IBGE de Recuperação Automática – SIDRA permite a consulta livres aos dados armazenados no Banco de Tabelas Estatísticas.

O objetivo do Banco de Tabelas Estatísticas é armazenar tabelas contendo os dados agregados das pesquisas que o IBGE realiza. Um dado agregado pode ser obtido, por exemplo, através do somatório dos valores de quesitos contidos em um questionário respondido pelos informantes da pesquisa.

Fonte: <https://sidra.ibge.gov.br/ajuda>

# Explorando banco de dados da POF

- A partir da documentação da POF 2017-2018 criou-se as tabelas auxiliares da figura 1 para ajudar na avaliação da base de dados. Cada tabela auxiliar possui um atributo PK e um atributo categoria que descreve a variável PK. A figura 2 abaixo exemplifica como é o dicionário de variáveis da pesquisa: código da variável (ou nome do atributo); descrição; código (valor numérico inteiro categórico); categoria no formato alfanumérico.

Figura 2 - Exemplo do dicionário da POF

DICIONÁRIO DAS VARIÁVEIS - POF 2017-2018 REGISTRO – MORADOR					
Posição Inicial	Tamanho	Decimais	Código da variável	Descrição	Categorias
3	4		ESTRATO_POF	da capital, resto da região metropolitana, resto da UF e área rural. A estratificação estatística foi realizada a partir das definições implementadas na Amostra Mestra, que utiliza informações da variável renda total do domicílio, obtida a partir dos dados do Censo 2010.	Ver arquivo em anexo "Estratos POF 2017-2018"
7	1		TIPO_SITUACAO_REG	Situação do Domicílio	1 – Urbano 2 – Rural
8	9		COD_UPA	Código da Unidade Primária de Amostragem: UF (2) + Número Sequencial (6) + DV (1). As 2 primeiras posições representam o código da Unidade da Federação	
17	2		NUM_DOM	Número do Domicílio. Corresponde a um código atribuído sequencialmente a cada domicílio selecionado em cada UPA.	



# Descrição das variáveis

Variável	CHAVE	Descrição das variável de Quest_morador
UF	PK; FK	Código da UF
ESTARTO_POF	PK	Identifica os estratos do plano amostral da pesquisa
TIPO_SITUACAO_RE G	PK; FK	Situação do Domicílio URBANO RURAL
COD_UPA	PK	Código da Unidade Primária de Amostragem: UF (2) + Número Sequencial (6) + DV (1). As 2 primeiras posições representam o código da Unidade da Federação
NUM_DOM	PK	Número do Domicílio. Corresponde a um código atribuído sequencialmente a cada domicílio selecionado em cada UPA.
NUM_UC	PK	Número da unidade de consumo (UC) pertencente ao domicílio pesquisado. Este número varia de 1 até o número total de unidades de consumo do domicílio. <b>(Família)</b>
COD_INFORMANTE	PK	Número de ordem atribuído ao informante, que é utilizado para identificá-lo em todos questionários. <b>(morador)</b>
V0403		Idade em anos
V0404	FK	Sexo
V0405	FK	Cor ou raça (1 – Branca; 2 – Preta; 3 – Amarela; 4 – Parda; 5 – Indígena; 9 – Sem declaração)
V0406	FK	tem plano ou seguro-saúde?
ANOS_ESTUDO		Anos de estudo da pessoa. Variável derivada, construída a partir dos quesitos referentes a educação.
RENDA_TOTAL		Valor em reais (R\$), considerando os centavos, do rendimento bruto total mensal da Unidade de Consumo <b>(renda da Família)</b>
TAMANHO_FAMÍLIA		Variável derivada criada para usar na aula de base de dados.

# Subconsultas em Instruções SELECT

## Subconsultas

Uma subconsulta (subquery) consiste em um comando SELECT que existe dentro de um outro comando SQL. O comando externo é chamado de comando “pai” da subconsulta. Conforme já mencionado em aulas anteriores.

Esta aula apresenta exemplos práticos de utilização de diferentes tipos de subconsulta com a utilização da base de dados da POF 2017-2018.

# Subconsultas em Instruções SELECT

## Subconsultas

O uso mais comum para as subconsultas é dentro de outra instrução SELECT, com o objetivo de retornar resultados que, de alguma forma, serão utilizados pela consulta pai. No entanto, as subqueries também são aceitas em comandos INSERT, UPDATE, DELETE e no CREATE TABLE. Esta aula aborda apenas o uso de subconsultas no SELECT, o chamado “SELECT dentro de SELECT”.

# Subconsultas em Instruções SELECT

## Subconsultas

As subconsultas com SELECT podem ser de diferentes tipos:

- Subconsulta de única linha (*single-row*): retornará sempre uma única linha em seu resultado (independente do número de colunas).

```
SELECT UF, AVG(RENDA_TOTAL) FROM Quest_morador
```

```
Where (uf, tipo_situacao_reg) IN
```

```
(SELECT MIN(uf),      MAXT(tipo_situacao_reg)
```

```
FROM Quest_morador WHERE renda_total > 0);
```

	UF	AVG(RENDA_TOTAL)
1	11	3709.863576233172

# Subconsultas em Instruções SELECT

## Subconsultas

- Subconsulta de múltiplas linhas (*multiple-row*): pode retornar zero, uma ou mais linhas em seu resultado (independente do número de colunas). Não é garantido que muitas linhas serão retornadas, mas como isto pode acontecer, a consulta pai deverá estar estruturada para receber muitas linhas.

```
SELECT UF, AVG(RENDA_TOTAL) RENDA_MEDIA  
FROM Quest_morador Where UF IN (SELECT COD_UF FROM T_UF)  
GROUP BY UF  
HAVING RENDA_TOTAL > RENDA_MEDIA;
```

	UF	RENDA_MEDIA
1	17	3368.815431154388
2	24	4228.366577123091
3	32	4939.553044699301
4	35	7039.474957737322
5	43	6549.507053582458
6	52	5109.4952391203815

# Subconsultas em Instruções SELECT

## Subconsultas

- Subconsulta de única coluna (*single-column*): retornará sempre uma única coluna no resultado (independente do número de linhas).

```
SELECT UF, AVG(RENDA_TOTAL) FROM Quest_morador
```

```
Where uf = (SELECT MIN(uf)
```

```
FROM Quest_morador WHERE renda_total > 0);
```

	UF	AVG(RENDA_TOTAL)
1	11	4206.323168867279

# Subconsultas em Instruções SELECT

## Subconsultas

- Subconsulta de múltiplas colunas (*multiple-column*): pode retornar uma ou mais colunas em seu resultado (independente do número de linhas).

```
SELECT count(UF) as total FROM Quest_morador
```

```
WHERE (UF,renda_total) IN
```

```
(SELECT uf,max(renda_total) FROM Quest_morador group by uf);
```

	total
1	81

# Subconsultas em Instruções SELECT – tipos diferentes

## Subconsultas

- Subconsulta escalar: retornará sempre um resultado contendo uma linha e uma única coluna (ou seja, um único valor). Desta forma, uma subconsulta escalar corresponde a uma subconsulta **single-row, single-column**.

```
SELECT avg(renda_media_familiar) FROM T_UF  
WHERE T_UF.cod_regiao =  
(SELECT R.cod_regiao FROM T_REGIAO R  
WHERE Regiao='Sul');
```

	avg(renda_media_familiar)
1	5739.676415328969



# Subconsultas em Instruções SELECT – tipos diferentes

## Subconsultas

- Subconsulta correlacionada (correlated subquery): este tipo de subconsulta contém uma referência para uma ou mais colunas da tabela pai dentro de sua própria definição. Normalmente, uma condição especificada no WHERE da subconsulta envolverá a comparação de uma coluna da consulta pai com uma coluna da subconsulta, permitindo assim, a efetivação de associações entre uma linha da tabela pai com uma linha gerada pela subconsulta.

```
SELECT UF, AVG(RENDA_TOTAL)
```

```
FROM Quest_morador QM
```

```
Where EXISTS
```

```
(SELECT NULL FROM T_UF WHERE T_UF.cod_uf=QM.uf AND cod_regiao=1)
```

```
GROUP BY UF
```


	UF	AVG(RENDA_TOTAL)
1	11	4206.323168867271
2	12	4590.190494989356
3	13	3621.8991138338833
4	14	4489.387786589763
5	15	2872.6019484765498
6	16	4671.565116511324
7	17	3368.815431154388

# Subconsultas em Instruções SELECT - COM WHERE

- **EXEMPLO (1) – 2 SELECT**

- Suponha que você deseja obter uma relação todas as UF cuja renda média esta superior à média de todas as UF. Neste caso você pode pensar a princípio que seja necessário **dois SELECT** para resolver este problema, conforme mostra o exemplo a seguir:

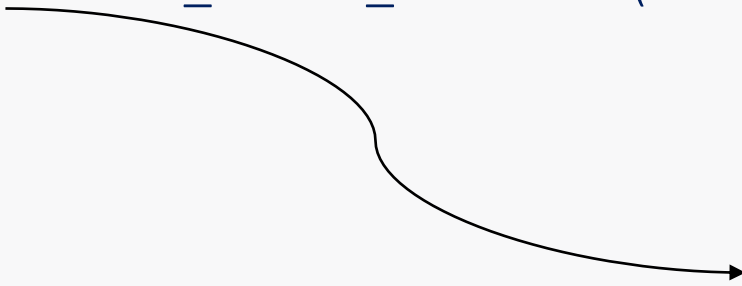
```
SELECT AVG(RENDA_MEDIA_FAMILIA) FROM T_UF;
```



AVG(renda_media_familiar)	
1	4560.5521197975995

```
SELECT UF,RENDA_MEDIA_FAMILIA FROM T_UF
```

```
WHERE RENDA_MEDIA_FAMILIA > (4560.552);
```



	UF	renda_media_familiar
1	Minas Gerais	4890.362424977962
2	Espírito Santo	4643.3496664443
3	Rio de Janeiro	5350.7800125352505
4	São Paulo	6762.966904536868
5	Paraná	5206.860550964191
6	Santa Catarina	5785.853017515916
7	Rio Grande do Sul	6226.315677506797
8	Mato Grosso do Sul	5464.784925883692
9	Mato Grosso	5238.526224707133
10	Goiás	4741.3998074339515
11	Distrito Federal	10342.211659159171

# Subconsultas em Instruções SELECT - COM WHERE

- **EXEMPLO (1) – 2 SELECT**

- Vejamos como resolver:

```
SELECT AVG(renda_media_familiar) FROM T_UF;
```

	AVG(renda_media_familiar)
1	4560.5521197975995

Este primeiro SELECT serve para descobrir a o valor médio da renda familiar nacional 4.560, um valor escalar. No passo seguinte usa-se este valor escalar na cláusula WHERE para selecionar as UF com renda média acima deste valor, mais especificamente `renda_media_familiar > 4560.552`.

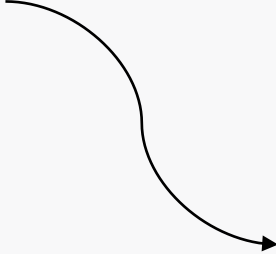
	UF	renda_media_familiar
1	Minas Gerais	4890.362424977962
2	Espirito Santo	4643.3496664443
3	Rio de Janeiro	5350.7800125352505
4	São Paulo	6762.966904536868
5	Paraná	5206.860550964191
6	Santa Catarina	5785.853017515916
7	Rio Grande do Sul	6226.315677506797
8	Mato Grosso do Sul	5464.784925883692
9	Mato Grosso	5238.526224707133
10	Goiás	4741.3998074339515
11	Distrito Federal	10342.211659159171

# Subconsultas em Instruções SELECT - COM WHERE

- **EXEMPLO (1) – 1 SELECT**
- A SQL nos permite combinar esses dois SELECTs em um único comando, bastando para isso transformar o primeiro SELECT em uma subconsulta que retornará um valor escalar para o primeiro SELECT:

```
SELECT UF,RENDA_MEDIA_FAMILIA FROM T_UF
```

```
WHERE RENDA_MEDIA_FAMILIA > (SELECT AVG(RENDA_MEDIA_FAMILIA)  
FROM T_UF);
```



	UF	renda_media_familiar
1	Minas Gerais	4890.362424977962
2	Espirito Santo	4643.3496664443
3	Rio de Janeiro	5350.7800125352505
4	São Paulo	6762.966904536868
5	Paraná	5206.860550964191
6	Santa Catarina	5785.853017515916
7	Rio Grande do Sul	6226.315677506797
8	Mato Grosso do Sul	5464.784925883692
9	Mato Grosso	5238.526224707133
10	Goiás	4741.3998074339515
11	Distrito Federal	10342.211659159171

# Subconsultas em Instruções SELECT - COM WHERE

- **EXEMPLO (1) – 1 SELECT (Explicação)**
- A consulta interna (subconsulta) é executada primeiro pelo SGBD (por isso mesmo, ela é está especificada entre parênteses). O resultado de sua execução corresponde a uma relação contendo uma linha e uma coluna (valor escalar), que armazena o valor médio. Tendo sido determinado, este valor médio é então utilizado no WHERE da consulta externa.

```
SELECT UF,RENDA_MEDIA_FAMILIA FROM T_UF
```

```
WHERE RENDA_MEDIA_FAMILIA > (SELECT AVG(RENDA_MEDIA_FAMILIA)  
FROM T_UF);
```

# Subconsultas em Instruções SELECT - COM WHERE

## EXEMPLO (1) – 1 SELECT (Explicação)

- Pode-se ver que o valor retornado pela subconsulta foi utilizado em um teste do tipo “maior que” (operador “ > ”). Este tipo de teste pode ser utilizado apenas para avaliar um valor escalar. Caso a subquery não fosse do tipo escalar (retornasse mais de uma linha ou mais de uma coluna), o teste com “>” ficaria sem sentido e ocorreria um erro de execução do comando SQL!.
- Apenas as subqueries do tipo escalar podem ser utilizadas em testes que envolvam os operadores =, <>, <, >, >= e <=.

```
SELECT UF,RENDA_MEDIA_FAMILIA FROM T_UF
```

```
WHERE RENDA_MEDIA_FAMILIA > (SELECT AVG(RENDA_MEDIA_FAMILIA)  
FROM T_UF);
```

# Subconsultas em Instruções **SELECT** Single-row

# Subconsultas em Instruções SELECT - COM WHERE

## EXEMPLO (2) – Retorno de uma linha

- A query a seguir apresenta uma subconsulta que identifica a maior renda familiar, sua UF de residência e calcula a razão entre sua renda e a renda média da UF que reside, segundo dados da POF 2017-2018. Como o teste da consulta externa é feito com “=”, a subconsulta precisa ser escalar).

```
SELECT DISTINCT A.RENDA_TOTAL as 'Renda Familiar'  
,T_UF.UF  
,A.RENDA_TOTAL /T_UF.renda_media_familiar as 'Razão de renda'  
FROM Quest_morador A  
INNER JOIN T_UF ON T_UF.cod_uf=A.uf  
WHERE RENDA_TOTAL = (SELECT MAX (RENDA_TOTAL) FROM Quest_morador)
```

	Renda Familiar	UF	Razão de renda
1	675212.2	Minas Gerais	138.06997136884854



# Subconsultas em Instruções SELECT - COM WHERE

## EXEMPLO (3) – Retorno de uma linha

- As subconsultas podem ser utilizadas normalmente em qualquer tipo de SELECT, incluindo os que possuem **GROUP BY**, **DISTINCT**, etc., como a query a seguir.
- A query a seguir lista um tipo de família que se caracteriza com menor renda familiar acima de zero, com UF de residência no Pará e sua renda está 338 vezes abaixo da média do Pará.

```
SELECT A.RENDA_TOTAL as 'Renda Familiar'  
,T_UF.UF  
,T_UF.renda_media_familiar/A.RENDA_TOTAL as 'Razão de renda'  
FROM Quest_morador A  
INNER JOIN T_UF ON T_UF.cod_uf=A.uf  
WHERE RENDA_TOTAL = (SELECT MIN(RENDA_TOTAL) FROM Quest_morador WHERE  
RENDATOTAL>0)  
GROUP BY 'Renda Familiar',T_UF.UF,'Razão de renda'
```

	Renda Familiar	UF	Razão de renda
1	8.38	Para	338.7402203253732

# Subconsultas em Instruções SELECT - COM WHERE

## EXEMPLO (4) – Retorno de uma linha

- A consulta a seguir inclui duas condições como subconsulta no WHERE: i) a primeira seleciona a família que tem maior número de membros e que haja ao menos um membro com plano de saúde e seja do sexo feminino; ii) a segunda condição verifica se a renda da família está menor do que a menor média de renda nacional entre as UF.
- Veja que a query principal encontra três tipos de famílias nesta condição.

```
SELECT DISTINCT A.RENDA_TOTAL as 'Renda Familiar'
,T_UF.UF
,T_UF.renda_media_familiar/ A.RENDA_TOTAL as 'Razão de renda'
,tamanho_familia
FROM Quest_morador A
INNER JOIN T_UF ON T_UF.cod_uf=A.uf
WHERE A.tamanho_familia = (SELECT MAX(tamanho_familia) FROM Quest_morador WHERE V0406=1
AND V0404=1)
and A.RENDA_TOTAL < (SELECT AVG(RENDA_MEDIA_FAMILIAR) FROM T_UF)
GROUP BY 'Renda Familiar',T_UF.UF,'Razão de renda',tamanho_familia;
```

	Renda Familiar	UF	Razão de renda	tamanho_familia
1	4090.56	Acre	1.08948506906576	13
2	3184.56	Ceará	1.03406586705893	13
3	692.64	Para	4.09829499642906	13

# Subconsultas em Instruções **SELECT** mult-row

# Subconsultas em Instruções SELECT - COM WHERE

## EXEMPLO (5) – Retorno de mais de uma linha

- Uma subconsulta multi-row é aquela que retorna mais de uma linha. O operador IN é um dos poucos operadores que podem ser utilizados com este tipo de subconsulta na cláusula WHERE, pois ele é aplicado sobre listas de valores.

# Subconsultas em Instruções SELECT - COM WHERE

## EXEMPLO (5) – Retorno de mais de uma linha

- No exemplo a seguir, produzimos um relatório com classe de renda das regiões que tem ao menos uma UF com renda média abaixo da nacional. Neste caso, **utilizamos uma subconsulta com 16 linhas**. Vamos analisar esta *query* por etapas.

```
SELECT
RG.Regiao,F.LIMITE_INFERIOR,F.LIMITE_SUPERIOR,F.CLASSE
,SUM(PESO_FINAL/TAMANHO_FAMILIA) as TOTAL_FAMILIAS
FROM Quest_morador A
INNER JOIN Faixa_renda_familiar F
    ON (RENDA_TOTAL BETWEEN F.LIMITE_INFERIOR
        AND F.LIMITE_SUPERIOR)
    OR (RENDA_TOTAL >= F.LIMITE_INFERIOR AND CLASSE='CLASSE 6')
INNER JOIN T_UF ON T_UF.COD_UF=A.UF
INNER JOIN T_REGIAO RG ON RG.cod_regiao=T_UF.cod_regiao
WHERE A.UF IN
    (SELECT COD_UF FROM T_UF WHERE RENDA_MEDIA_FAMILIAR <
        (SELECT AVG(RENDA_MEDIA_FAMILIAR) FROM T_UF ) )
GROUP BY RG.Regiao,F.LIMITE_INFERIOR,F.LIMITE_SUPERIOR,F.CLASSE
ORDER BY RG.Regiao,F.LIMITE_INFERIOR,F.LIMITE_SUPERIOR,F.CLASSE;
```

# Subconsultas em Instruções SELECT - COM WHERE

## EXEMPLO (5) – Retorno de mais de uma linha

Resultado referente a *query* anterior exhibe as regiões Nordeste e Norte e suas respectivas totalizações por classes de renda. A “Classe 1 – E” tem os maiores totais em ambas as regiões brasileiras.

	Regiao	LIMITE_INFERIOR	LIMITE_SUPERIOR	CLASSE	TOTAL_FAMILIAS
1	Nordeste	0	1908	CALASE 1 - E	7180512
2	Nordeste	1908	2862	CALSSE 2 - D	4012648
3	Nordeste	2862	5724	CLASSE 3 - C	4280488
4	Nordeste	5724	9540	CLASSE 4 - B	1330126
5	Nordeste	14310	23850	CLASSE 5 - A	347198
6	Nordeste	23850	NULL	CLASSE 6	177941
7	Norte	0	1908	CALASE 1 - E	1981821
8	Norte	1908	2862	CALSSE 2 - D	1037374
9	Norte	2862	5724	CLASSE 3 - C	1262716
10	Norte	5724	9540	CLASSE 4 - B	433521
11	Norte	14310	23850	CLASSE 5 - A	98609
12	Norte	23850	NULL	CLASSE 6	47230

(veja resultado semelhante no \*SIDRA, endereço:  
<https://sidra.ibge.gov.br/tabela/6977>).

# Subconsultas em Instruções SELECT - COM WHERE

## EXEMPLO (5) – Retorno de mais de uma linha

- **(Etapa 1)** A *query* retorna as regiões que têm ao menos uma UF com renda média familiar abaixo da renda nacional. A subconsulta que retorna 16 linhas faz uso de uma query mais interna para comparar os casos em que o atributo **renda\_media\_familiar** está **menor** do que o valor escalar retornado pela subconsulta mais interna. A subconsulta mais interna retorna a renda média nacional das famílias. Vejamos a seguir mais detalhes:

```
SELECT
RG.Regiao,F.LIMITE_INFERIOR,F.LIMITE_SUPERIOR,F.CLASSE
,SUM(PESO_FINAL/TAMANHO_FAMILIA) as TOTAL_FAMILIAS
FROM Quest_morador A
INNER JOIN Faixa_renda_familiar F
    ON (RENDA_TOTAL BETWEEN F.LIMITE_INFERIOR
        AND F.LIMITE_SUPERIOR)
    OR (RENDA_TOTAL >= F.LIMITE_INFERIOR AND CLASSE='CLASSE 6')
INNER JOIN T_UF ON T_UF.COD_UF=A.UF
INNER JOIN T_REGIAO RG ON RG.cod_regiao=T_UF.cod_regiao
WHERE A.UF IN
    (SELECT COD_UF FROM T_UF WHERE RENDA_MEDIA_FAMILIAR <
        (SELECT AVG(RENDA_MEDIA_FAMILIAR) FROM T_UF ) )
GROUP BY RG.Regiao,F.LIMITE_INFERIOR,F.LIMITE_SUPERIOR,F.CLASSE
ORDER BY RG.Regiao,F.LIMITE_INFERIOR,F.LIMITE_SUPERIOR,F.CLASSE;
```

# Subconsultas em Instruções SELECT - COM WHERE

## EXEMPLO (5) – Retorno de mais de uma linha

- **(Etapa 2)** São feitos dois INNER JOIN para recuperar a identificação do nome da Região:
  - i) Primeiro é feito um INNER JOIN entre Quest\_morador e T\_uf; e
  - ii) depois outro entre T\_uf e T\_regiao.

```
SELECT
RG.Regiao,F.LIMITE_INFERIOR,F.LIMITE_SUPERIOR,F.CLASSE
,SUM(PESO_FINAL/TAMANHO_FAMILIA) as TOTAL_FAMILIAS
FROM Quest_morador A
INNER JOIN Faixa_renda_familiar F
    ON (RENDA_TOTAL BETWEEN F.LIMITE_INFERIOR
        AND F.LIMITE_SUPERIOR)
    OR (RENDA_TOTAL>=F.LIMITE_INFERIOR AND CLASSE='CLASSE 6')
INNER JOIN T_UF ON T_UF.COD_UF=A.UF
INNER JOIN T_REGIAO RG ON RG.cod_regiao=T_UF.cod_regiao
WHERE A.UF IN
    (SELECT COD_UF FROM T_UF WHERE RENDA_MEDIA_FAMILIAR <
        (SELECT AVG(RENDA_MEDIA_FAMILIAR) FROM T_UF ))
GROUP BY RG.Regiao,F.LIMITE_INFERIOR,F.LIMITE_SUPERIOR,F.CLASSE
ORDER BY RG.Regiao,F.LIMITE_INFERIOR,F.LIMITE_SUPERIOR,F.CLASSE;
```



# Subconsultas em Instruções SELECT - COM WHERE

## EXEMPLO (5) – Retorno de mais de uma linha

- **(Etapa 3)** Para encontrar a classe de renda fez-se um INNER JOIN entre a renda\_total (renda total familiar) da tabela Quest\_morador e da tabela Faixa\_renda\_familiar.
  - São usados dois testes neste JOIN:
    - i) o primeiro verifica em qual faixa (de E a A) a renda\_total pertence com o uso da cláusula *Between*; e
    - ii) o segundo, como não há limite superior, é feito um teste que verifica se renda\_total é maior ou igual ao limite inferior e caso a seja, a classe mais alta de renda (classe 6 acima de A) inclui a família.

```

SELECT
  RG.Regiao,F.LIMITE_INFERIOR,F.LIMITE_SUPERIOR,F.CLASSE
,SUM(PESO_FINAL/TAMANHO_FAMILIA) as TOTAL_FAMILIAS
FROM Quest_morador A
INNER JOIN Faixa_renda_familiar F
  ON (RENDA_TOTAL BETWEEN F.LIMITE_INFERIOR
      AND F.LIMITE_SUPERIOR)
  OR (RENDA_TOTAL >= F.LIMITE_INFERIOR AND CLASSE='CLASSE 6')
INNER JOIN T_UF ON T_UF.COD_UF=A.UF
INNER JOIN T_REGIAO RG ON RG.cod_regiao=T_UF.cod_regiao
WHERE A.UF IN
  (SELECT COD_UF FROM T_UF WHERE RENDA_MEDIA_FAMILIAR <
    (SELECT AVG(RENDA_MEDIA_FAMILIAR) FROM T_UF) )
GROUP BY RG.Regiao,F.LIMITE_INFERIOR,F.LIMITE_SUPERIOR,F.CLASSE
ORDER BY RG.Regiao,F.LIMITE_INFERIOR,F.LIMITE_SUPERIOR,F.CLASSE;

```

	CLASSE	LIMITE_INFERIOR	LIMITE_SUPERIOR
1	CLASSE 1 - E	0	1908
2	CLASSE 2 - D	1908	2862
3	CLASSE 3 - C	2862	5724
4	CLASSE 4 - B	5724	9540
5	CLASSE 5 - A	14310	23850
6	CLASSE 6	23850	NULL

# Subconsultas em Instruções SELECT - COM WHERE

## EXEMPLO (5) – Retorno de mais de uma linha

- **(Etapa 4)** Para encontrar o total de famílias usa-se o atributo peso\_final (que representa o fator de expansão da unidade de consumo [família]). Por exemplo se houver 1 família com 3 pessoas esta família representa um número de família que corresponde ao peso\_final. Como no SQL soma-se o peso\_final para cada pessoa da família, então divide-se esta soma pelo número de membros para encontrar uma família  $((\text{PESO\_FINAL} + \text{PESO\_FINAL} + \dots + \text{PESO\_FINAL})/3)$ . O somatório final exibe o total de famílias. Vejamos um exemplo na próximo *slide*.

```
SELECT
RG.Regiao,F.LIMITE_INFERIOR,F.LIMITE_SUPERIOR,F.CLASSE
,SUM(PESO_FINAL/TAMANHO_FAMILIA) as TOTAL_FAMILIAS
FROM Quest_morador A
INNER JOIN Faixa_renda_familiar F
  ON (RENDA_TOTAL BETWEEN F.LIMITE_INFERIOR
      AND F.LIMITE_SUPERIOR)
  OR (RENDA_TOTAL >= F.LIMITE_INFERIOR AND CLASSE='CLASSE 6')
INNER JOIN T_UF ON T_UF.COD_UF=A.UF
INNER JOIN T_REGIAO RG ON RG.cod_regiao=T_UF.cod_regiao
WHERE A.UF IN
  (SELECT COD_UF FROM T_UF WHERE RENDA_MEDIA_FAMILIAR <
    (SELECT AVG(RENDA_MEDIA_FAMILIAR) FROM T_UF ) )
GROUP BY RG.Regiao,F.LIMITE_INFERIOR,F.LIMITE_SUPERIOR,F.CLASSE
ORDER BY RG.Regiao,F.LIMITE_INFERIOR,F.LIMITE_SUPERIOR,F.CLASSE;
```

\*SIDRA - Sistema IBGE de Recuperação Automática - SIDRA

# Subconsultas em Instruções SELECT - COM WHERE

## EXEMPLO (5) – Retorno de mais de uma linha

- (Etapa 4)
- $SUM(PESO\_FINAL/TAMANHO\_FAMILIA)$  as TOTAL\_FAMILIAS

	UF	ESTRATO_POF	TIPO_SITUACAO_REG	COD_UPA	NUM_DOM	NUM_UC	COD_INFORMANTE	PESO_FINAL	TAMANHO_FAMILIA
1	11	1101	1	110000016	2	1	1	690.88373818	3
2	11	1101	1	110000016	2	1	2	690.88373818	3
3	11	1101	1	110000016	2	1	3	690.88373818	3

As variáveis UF,

ESTRATO\_POF,

TIPO\_SITUACAO\_REG,

COD\_UPA,

NUM\_DOM,

NUM\_UC,

COD\_INFORMANTE

definem um membro da família ou unidade de consumo.

$$(690.88373818 + 690.88373818 + 690.88373818) / 3 = 690.88373818$$

# Subconsultas em Instruções SELECT - COM WHERE

## EXEMPLO (5) – Retorno de mais de uma linha

- **(Etapa 5)** O GROUP BY é usado para agrupar o resultado por região e classes de renda, incluindo as mesmas variáveis selecionadas para exibição.

SELECT

RG.Regiao,F.LIMITE\_INFERIOR,F.LIMITE\_SUPERIOR,F.CLASSE  
,SUM(PESO\_FINAL/TAMANHO\_FAMILIA) as TOTAL\_FAMILIAS

FROM Quest\_morador A

INNER JOIN Faixa\_renda\_familiar F

ON (RENDA\_TOTAL BETWEEN F.LIMITE\_INFERIOR  
AND F.LIMITE\_SUPERIOR)

OR (RENDA\_TOTAL >= F.LIMITE\_INFERIOR AND CLASSE='CLASSE 6')

INNER JOIN T\_UF ON T\_UF.COD\_UF=A.UF

INNER JOIN T\_REGIAO RG ON RG.cod\_regiao=T\_UF.cod\_regiao

WHERE A.UF IN

(SELECT COD\_UF FROM T\_UF WHERE RENDA\_MEDIA\_FAMILIAR <  
(SELECT AVG(RENDA\_MEDIA\_FAMILIAR) FROM T\_UF ))

GROUP BY RG.Regiao,F.LIMITE\_INFERIOR,F.LIMITE\_SUPERIOR,F.CLASSE

ORDER BY RG.Regiao,F.LIMITE\_INFERIOR,F.LIMITE\_SUPERIOR,F.CLASSE;

# Resultado exemplo 5

	Regiao	LIMITE_INFERIOR	LIMITE_SUPERIOR	CLASSE	TOTAL_FAMILIAS
1	Nordeste	0	1908	CALASE 1 - E	7180512
2	Nordeste	1908	2862	CALSSE 2 - D	4012648
3	Nordeste	2862	5724	CLASSE 3 - C	4280488
4	Nordeste	5724	9540	CLASSE 4 - B	1330126
5	Nordeste	14310	23850	CLASSE 5 - A	347198
6	Nordeste	23850	NULL	CLASSE 6	177941
7	Norte	0	1908	CALASE 1 - E	1981821
8	Norte	1908	2862	CALSSE 2 - D	1037374
9	Norte	2862	5724	CLASSE 3 - C	1262716
10	Norte	5724	9540	CLASSE 4 - B	433521
11	Norte	14310	23850	CLASSE 5 - A	98609
12	Norte	23850	NULL	CLASSE 6	47230

# Subconsultas em Instruções SELECT - COM WHERE

## EXEMPLO (5) – Retorno de mais de uma linha

- **(Etapa 6)** O ORDER BY é usado para organizar o resultado por região e classes de renda.

```
SELECT
RG.Regiao,F.LIMITE_INFERIOR,F.LIMITE_SUPERIOR,F.CLASSE
,SUM(PESO_FINAL/TAMANHO_FAMILIA) as TOTAL_FAMILIAS
FROM Quest_morador A
INNER JOIN Faixa_renda_familiar F
    ON (RENDA_TOTAL BETWEEN F.LIMITE_INFERIOR
        AND F.LIMITE_SUPERIOR)
    OR (RENDA_TOTAL >= F.LIMITE_INFERIOR AND CLASSE='CLASSE 6')
INNER JOIN T_UF ON T_UF.COD_UF=A.UF
INNER JOIN T_REGIAO RG ON RG.cod_regiao=T_UF.cod_regiao
WHERE A.UF IN
    (SELECT COD_UF FROM T_UF WHERE RENDA_MEDIA_FAMILIAR <
        (SELECT AVG(RENDA_MEDIA_FAMILIAR) FROM T_UF ))
GROUP BY RG.Regiao,F.LIMITE_INFERIOR,F.LIMITE_SUPERIOR,F.CLASSE
ORDER BY RG.Regiao,F.LIMITE_INFERIOR,F.LIMITE_SUPERIOR,F.CLASSE;
```

## Subconsultas em Instruções SELECT - COM WHERE

- Além da cláusula IN, é também possível utilizar os operadores ANY, SOME e ALL para lidar com subconsultas multi-row no WHERE. Porém esses operadores não são tão utilizados na prática e não são suportados pelo SQLite (maiores informações em: [https://www.w3schools.com/sql/sql\\_any\\_all.asp](https://www.w3schools.com/sql/sql_any_all.asp)).

## Subconsultas *multi-column* na Cláusula WHERE



# Subconsultas em Instruções SELECT - COM WHERE

## EXEMPLO (6) – Retorno de mais de uma linha

- As subconsultas apresentadas até agora, retornavam apenas uma coluna. Mas a SQL permite com que você monte um teste no WHERE comparando várias colunas de uma vez.
- Um exemplo envolvendo uma subquery do tipo multi-row, multi-column é apresentado a seguir. Observe que a consulta externa utiliza um par de colunas entre parênteses (A.UF, V0403, V0405, 1, RENDA\_TOTAL) que são utilizados na comparação com cada par de resultados retornados pela subconsulta.

```
SELECT RG.Regiao,V0403,SUM(1) AS TOTAL
FROM Quest_morador A
INNER JOIN T_UF ON T_UF.COD_UF=A.UF
INNER JOIN T_REGIAO RG ON RG.cod_regiao=T_UF.cod_regiao
WHERE (A.UF, V0403, V0405, 1, RENDA_TOTAL)
      IN (SELECT UF, V0403, V0405, V0406, RENDA_TOTAL
          FROM Quest_morador WHERE RENDA_TOTAL<100)
GROUP BY RG.cod_regiao,V0403;
```

# Subconsultas em Instruções SELECT - COM WHERE

## EXEMPLO (6) – Retorno de mais de uma linha

- A consulta retorna a região, idade do morador e total de moradores que possuem plano de saúde para uma renda inferior a 1000 reais. A query demonstra o uso de 4 colunas e a coluna com valor 1 corresponde ao atributo V0406 (1= Sim) possui plano de saúde.
- Esta consulta não precisa ser resolvida com uso de subconsulta, está elaborada assim apenas para efeito de demonstração.

```
SELECT RG.Regiao,V0403,SUM(1) AS TOTAL  
FROM Quest_morador A  
INNER JOIN T_UF ON T_UF.COD_UF=A.UF  
INNER JOIN T_REGIAO RG ON RG.cod_regiao=T_UF.cod_regiao  
WHERE (A.UF, V0403, V0405, 1, RENDA_TOTAL)  
      IN (SELECT UF, V0403, V0405, V0406, RENDA_TOTAL  
          FROM Quest_morador WHERE RENDA_TOTAL<100)  
GROUP BY RG.cod_regiao,V0403;
```

## **Subconsultas na Cláusula FROM**

## Subconsultas na Cláusula FROM

- Subconsultas podem ser utilizadas na cláusula FROM, com o intuito de gerar resultados que poderão ser utilizados como se fossem uma tabela real (por este motivo uma subconsulta é normalmente chamada de “tabela virtual”). Esta forma de utilização de subconsultas é muito comum.

# Subconsultas na Cláusula FROM

- **Exemplo (7) – Subconsultas na Cláusula FROM**
- No exemplo a seguir, uma subconsulta é utilizada no FROM para permitir o uso da região como atributo de condição no where, uma vez que quest\_morador não tem essa informação.

```
SELECT SX.categoria SEXO
,F.CLASSE, F.LIMITE_INFERIOR, F.LIMITE_SUPERIOR
, ROUND(SUM(PESO_FINAL/TAMANHO_FAMILIA)) Famalias
FROM Quest_morador A
INNER JOIN SEXO SX ON SX.V0404=A.V0404
INNER JOIN Faixa_renda_familiar F
    ON (A.RENDA_TOTAL BETWEEN F.LIMITE_INFERIOR
        AND F.LIMITE_SUPERIOR)
    OR (A.RENDA_TOTAL>=F.LIMITE_INFERIOR AND CLASSE='CLASSE 6')
INNER JOIN (SELECT REGIAO,cod_uf
    FROM T_UF
    INNER JOIN T_REGIAO R ON T_UF.cod_regiao=R.cod_regiao)
    V ON A.UF=V.COD_UF
WHERE V0403 = 60 AND REGIAO='Norte'
GROUP BY V0403,SEXO,F.CLASSE, F.LIMITE_INFERIOR, F.LIMITE_SUPERIOR
```


# Subconsultas na cláusula FROM

```

SELECT SX.categoria SEXO
,F.CLASSE, F.LIMITE_INFERIOR, F.LIMITE_SUPERIOR
, ROUND(SUM(PESO_FINAL/TAMANHO_FAMILIA)) Famalias
FROM Quest_morador A
INNER JOIN SEXO SX ON SX.V0404=A.V0404
INNER JOIN Faixa_renda_familiar F
  ON (A.RENDA_TOTAL BETWEEN F.LIMITE_INFERIOR
      AND F.LIMITE_SUPERIOR)
  OR (A.RENDA_TOTAL >= F.LIMITE_INFERIOR AND CLASSE='CLASSE 6')
INNER JOIN (SELECT REGIAO,cod_uf
  FROM T_UF
  INNER JOIN T_REGIAO R ON T_UF.cod_regiao=R.cod_regiao)
  V ON A.UF=V.COD_UF
WHERE V0403 = 60 AND REGIAO='Norte'
GROUP BY V0403,SEXO,F.CLASSE, F.LIMITE_INFERIOR, F.LIMITE_SUPERIOR

```

Retorna classes de renda das famílias com pessoas acima de 60 anos residentes na região Norte.



	SEXO	CLASSE	LIMITE_INFERIOR	LIMITE_SUPERIOR	Famalias
1	Homem	CLASSE 1 - E	0	1908	9555
2	Homem	CLASSE 2 - D	1908	2862	7015
3	Homem	CLASSE 3 - C	2862	5724	2470
4	Homem	CLASSE 4 - B	5724	9540	398
5	Homem	CLASSE 5 - A	14310	23850	682
6	Homem	CLASSE 6	23850	NULL	428
7	Mulher	CLASSE 1 - E	0	1908	5789
8	Mulher	CLASSE 2 - D	1908	2862	4563
9	Mulher	CLASSE 3 - C	2862	5724	9324
10	Mulher	CLASSE 4 - B	5724	9540	2666
11	Mulher	CLASSE 5 - A	14310	23850	516

# Subconsultas na cláusula FROM

- **Exemplo (7)**
- Como essa consulta é resolvida? Como acontece com todo SQL, o processamento começa no FROM, onde existem 3 INNER JOIN:
  - i) O primeiro JOIN é entre a tabela Quest\_morador (apelidada de “A”) com uma subconsulta (apelidada de “V”). Esta subconsulta retorna dois campos: regioao e cod\_uf. Se você pegar esta subconsulta e executá-la isoladamente, verá que ela produz o resultado abaixo:

A subconsulta é utilizada normalmente como se fosse uma view ou tabela virtual.

```
SELECT SX.categoria SEXO
, F.CLASSE, F.LIMITE_INFERIOR, F.LIMITE_SUPERIOR
, ROUND(SUM(PESO_FINAL/TAMANHO_FAMILIA)) Familias
FROM Quest_morador A
INNER JOIN SEXO SX ON SX.V0404=A.V0404
INNER JOIN Faixa_renda_familiar F
  ON (A.RENDA_TOTAL BETWEEN F.LIMITE_INFERIOR
      AND F.LIMITE_SUPERIOR)
  OR (A.RENDA_TOTAL >= F.LIMITE_INFERIOR AND CLASSE='CLASSE 6')
INNER JOIN (SELECT REGIAO, cod_uf
  FROM T_UF
  INNER JOIN T_REGIAO R ON T_UF.cod_regiao=R.cod_regiao)
  V ON A.UF=V.COD_UF
WHERE V0403 = 60 AND REGIAO='Norte'
GROUP BY V0403, SEXO, F.CLASSE, F.LIMITE_INFERIOR, F.LIMITE_SUPERIOR
```

	REGIAO	cod_uf
1	Norte	11
2	Norte	12
3	Norte	13
4	Norte	14
5	Norte	15
6	Norte	16
7	Norte	17
8	Nordeste	21
9	Nordeste	22
10	Nordeste	23
11	Nordeste	24
12	Nordeste	25
13	Nordeste	26
14	Nordeste	27
15	Nordeste	28
16	Nordeste	29
17	Sudeste	31
18	Sudeste	32
19	Sudeste	33
20	Sudeste	35
21	Sul	41
22	Sul	42
23	Sul	43
24	Centro-Oeste	50
25	Centro-Oeste	51
26	Centro-Oeste	52
27	Centro-Oeste	53

# Subconsulta Correlacionada



## Subconsultas na cláusula WHERE

- Todos os exemplos apresentados até agora envolveram o uso de subconsultas autônomas, isto é subconsultas que executariam de forma bem-sucedida caso fosse separadas da consulta pai.

## Subconsultas na cláusula WHERE

- De maneira oposta, as subconsultas correlacionadas incluem referências para elementos da consulta pai e, por isso, não podem ser executadas de forma independente.
- Suponha, por exemplo, que você deseja listar, por **situação Rural ou Urbana** o tamanho médio das famílias, mínimo e máximo quando a renda da família for 50 vezes superior a renda média familiar da sua UF de residência.
- Existem algumas formas diferentes para resolver esse problema e uma é exatamente com o uso de uma subconsulta correlacionada.

## Subconsultas na cláusula FROM

- **Exemplo (8)**
- Veja o SQL desta consulta resolvido abaixo. **Como ela funciona?**

```
SELECT T.categoria, AVG(TAMANHO_FAMILIA)
Tamanho_Medio_Familia, min(TAMANHO_FAMILIA)
Mínimo,max(TAMANHO_FAMILIA) Maximo
FROM Quest_morador A
INNER JOIN Tipo_situacao T ON
T.tipo_situacao_reg=A.tipo_situacao_reg
WHERE A.RENDA_TOTAL > (SELECT 50*renda_media_familiar FROM
T_UF WHERE T_UF.COD_UF=A.UF)
GROUP BY T.categoria;
```

# Subconsultas na cláusula FROM

- Exemplo (8)
- Primeiro veja com a tenção a *subquery*:

```
(SELECT 50*renda_media_familiar FROM T_UF WHERE  
T_UF.COD_UF=A.UF)
```

•Veja o WHERE... Você percebeu o uso do apelido “a.” no lado direito? Isto corresponde a uma referência para uma coluna da consulta pai (e não da *subquery*). Se você observar a consulta pai, verá que ela realmente foi apelidada de “a”. Isto é o que chamamos de “correlação” da subconsulta correlacionada! Esta subconsulta não pode ser executada isoladamente, uma única vez, retornando um conjunto de resultados para a consulta externa.

# Subconsultas na cláusula FROM

- Exemplo (8)
- Primeiro veja com a tenção a *subquery*:

```
(SELECT 50*renda_media_familiar FROM T_UF WHERE  
T_UF.COD_UF=A.UF)
```

•Veja o WHERE... Você percebeu o uso do apelido “a.” no lado direito? Isto corresponde a uma referência para uma coluna da consulta pai (e não da *subquery*). Se você observar a consulta pai, verá que ela realmente foi apelidada de “a”. Isto é o que chamamos de “correlação” da subconsulta correlacionada! Esta subconsulta não pode ser executada isoladamente, uma única vez, retornando um conjunto de resultados para a consulta externa. Ao invés disso, a subconsulta correlacionada é sempre reexecutada para cada linha gerada pela consulta pai (melhor explicando: se a consulta pai retornar n linhas, a subconsulta será executada n vezes, uma para cada linha).

## Subconsultas na cláusula FROM

- **Exemplo (8)**
- Em nosso exemplo, a cada iteração (cada linha da consulta pai), a subconsulta receberá o valor da “UF” da consulta pai e utilizará esse valor no teste do seu WHERE (WHERE T\_UF.COD\_UF=A.UF). Com isto, será computada o valor máximo da renda média familiar referente à UF da linha que está sendo correntemente processada na consulta pai.
- Uma vez computado, este resultado poderá ser normalmente utilizado no teste da consulta externa (WHERE A.RENDA\_TOTAL > ...) permitindo com que seja possível identificar se a renda total da família é maior do que a média da UF correspondente. Veja que a consulta externa e a subconsulta correlacionada “conversam” e trabalham de forma colaborativa para obter os resultados.

....

```
WHERE A.RENDA_TOTAL > (SELECT 50*renda_media_familiar  
                        FROM T_UF WHERE  
                        T_UF.COD_UF=A.UF)
```

## Subconsultas na cláusula FROM

- **Exemplo (9)** com Operadores EXISTS e NOT EXISTS
- O operador EXISTS testa a existência de uma linha em uma subconsulta correlacionada. Se nenhuma linha é encontrada, o resultado é FALSE. Caso contrário, TRUE. O NOT EXISTS reverte os resultados (caso seja FALSO retorna VERDADE).
- Utilizando o operador EXISTS para identificar de há famílias com renda 50 vezes superior a média de sua respectiva UF.

```
SELECT T.categoria, P.categoria TemPlanodeSaude
, AVG(TAMANHO_FAMILIA) Tamanho_Medio_Familia
, min(TAMANHO_FAMILIA) Mínimo
, max(TAMANHO_FAMILIA) Maximo
FROM Quest_morador A
INNER JOIN Tipo_situacao T ON T.tipo_situacao_reg=A.tipo_situacao_reg
INNER JOIN PlanoSaude P ON P.V0406 = A.V0406
WHERE EXISTS (SELECT NULL FROM T_UF
              WHERE T_UF.COD_UF=A.UF AND A.RENDA_TOTAL> 50*renda_media_familiar)
GROUP BY T.categoria,P.categoria;
```

# Subconsultas na cláusula FROM

- **Exemplo (9)** com Operadores EXISTS e NOT EXISTS
- A subconsulta pode retornar qualquer coisa no SELECT, mas ajudar a reduzir o custo de processamento pode-se usar um valor constante, neste caso usamos NULL. Logo isso não importa no EXISTS.

```
SELECT T.categoria, P.categoria TemPlanodeSaude
, AVG(TAMANHO_FAMILIA) Tamanho_Medio_Familia
, min(TAMANHO_FAMILIA) Mínimo
, max(TAMANHO_FAMILIA) Maximo
FROM Quest_morador A
INNER JOIN Tipo_situacao T ON T.tipo_situacao_reg=A.tipo_situacao_reg
INNER JOIN PlanoSaude P ON P.V0406 = A.V0406
WHERE EXISTS (SELECT NULL FROM T_UF
WHERE T_UF.COD_UF=A.UF AND A.RENDA_TOTAL> 50*renda_media_familiar)
GROUP BY T.categoria,P.categoria;
```

	categoria	TemPlanodeSaude	Tamanho_Medio_Familia	Mínimo	Maximo
1	Rural	Não	4	4	4
2	Rural	Sim	2.4	2	4
3	Urbana	Sim	3.4	1	4



# Subconsultas na cláusula FROM

- **Exemplo (9)** com Operadores EXISTS e NOT EXISTS
- Para fazer o oposto, ou seja, listar os casos que não estejam entre as famílias com renda acima da média da sua UF.

```
SELECT T.categoria, P.categoria TemPlanodeSaude
, AVG(TAMANHO_FAMILIA) Tamanho_Medio_Familia
, min(TAMANHO_FAMILIA) Mínimo
, max(TAMANHO_FAMILIA) Maximo
FROM Quest_morador A
INNER JOIN Tipo_situacao T ON T.tipo_situacao_reg=A.tipo_situacao_reg
INNER JOIN PlanoSaude P ON P.V0406 = A.V0406
WHERE NOT EXISTS (SELECT NULL FROM T_UF
WHERE T_UF.COD_UF=A.UF AND A.RENDA_TOTAL> 50*renda_media_familiar)
GROUP BY T.categoria,P.categoria;
```

	categoria	TemPlanodeSaude	Tamanho_Medio_Familia	Mínimo	Maximo
1	Rural	Não	4.06344899556483	1	17
2	Rural	Sim	3.41972409744643	1	11
3	Urbana	Não	3.91492695603986	1	16
4	Urbana	Sim	3.43210069335999	1	14

Obrigado