PROVAS DE TI
TUDO PARA VOCÊ PASSAR

BANCO DE DADOS
SQL DML – Parte 2 ☺

Prof. Eduardo Neves
edumneves@gmail.com

Dúvidas de Banco de Dados:
Fórum do curso ☺

Artigos do Blog:
http://goo.gl/Sm6JgV

- Curso de Banco de Dados
- https://goo.gl/Q2LwS6

Mini CV do Prof. Eduardo Neves

Analista de Sistemas do BNDES – Banco Nacional de Desenvolvimento Econômico e Social. Bacharel em Ciência da Computação pela Universidade Federal do Rio de Janeiro (UFRJ). Pós-graduado em Gestão de Projetos. Atuou como Analista de Sistemas em empresas privadas e em empresas públicas: BR Distribuidora e Petrobras. Principais aprovações: 4º BNDES/2012 – Desenvolvimento; 30º Petrobras/2011 – Engenharia de Software; 12º FINEP/2011; 3º Transpetro/2011; 42º TRT-RJ/2011; 7º Petrobras Macaé/2010; 1º Caixa Econômica Federal/2010 – 18º BR Distribuidora/2010 – Analista SAP. Leciona a disciplina de Banco de Dados em cursos preparatórios para concurso público no RJ.

7 Item(s) Exibir 30 por página

Classificar por Posição Menor para o maior

4 Comentário(s)
~~R$70,00~~
R$49,50

Adicionar ao Carrinho

1 Comentário(s)
~~R$70,00~~
R$49,50

Adicionar ao Carrinho

1 Comentário(s)
~~R$70,00~~
R$49,50

Adicionar ao Carrinho

3 Comentário(s)
~~R$70,00~~
R$49,50

Adicionar ao Carrinho

Qconcursos – estatística

	CESPE	FCC	CESGRANRIO	OUTRAS	TOTAL
SQL	173	131	66	153	523
Data warehouse	96	70	54	50	270
Modelagem	85	62	28	65	240
Conceitos Básicos	112	41	20	62	235
Formas Normais	69	22	27	71	189
Transações	24	18	22	22	86

BD – Distribuição das aulas

- Aula 01
  - Group by
  - Count(*), Count(Campo), Count(Distinct Campo)
- Aula 02
  - SUM, MAX, MIN, AVG
- Aula 03
  - Exemplos e exercícios
- Aula 04
  - Filtro com where e having
  - Order by
- Aula 05
  - Joins avançados

BD – Distribuição das aulas

- Aula 06 e 07
  - Selects avançados
- Aula 08 e 09
  - Exercícios
- Aula 10, 11, 12 e 13
  - Exists e not exists de dois níveis
- Aula 14 e 15
  - Exercícios

Master Mind de Concursos BD / ES

- Bônus para os alunos fiéis ☺
- Professores:
  - Eduardo Neves – Banco de Dados
  - Lúcio Camilo – Engenharia de Software
- Canal de comunicação (telegram) onde teremos diversas vantagens:
  - Tratamento premium, ou seja, nas filas de resposta de emails, dúvidas, vcs serão os primeiros ;-)
  - Compartilhar materiais
  - Direcionamento de estudos
  - Resolução e discussão de discursivas
  - Direcionamento no uso do ANKI
  - Simulados
- Condições de entrada
  - https://goo.gl/pCgM7J

https://www.youtube.com/user/edumartinsneves

E

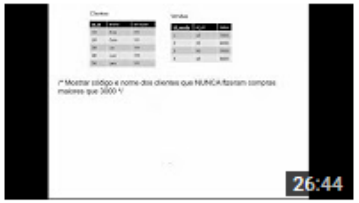
Eduardo Neves

Inscrever-se 48

Envios

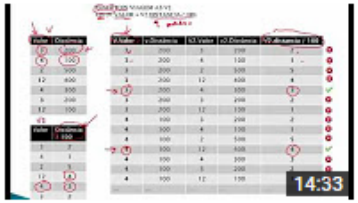
Pegadinha da Semana - 3 - 20161001
55 visualizações • 1 semana atrás

Pegadinha da semana - SQL Exists - parte 2
19 visualizações • 1 semana atrás



Pegadinha da semana - SQL Exists - parte 1
10 visualizações • 1 semana atrás

Pegadinha da Semana - 2 - 20160925
159 visualizações • 2 semanas atrás



Pegadinha da semana Banco de Dados - 1
100 visualizações • 2 semanas atrás

Playlists criadas

Anki
Atualizada hoje

Pegadinha da Semana

Dúvidas de Banco de Dados

Fórum Geral de Dúvidas

Fórum Geral de Dúvidas

Acrescentar um novo tópico de discussão

Tópico	Autor	Comentários	Última mensagem
Dúvida Questão Normalização FNBC	Danniel Albuquerque Araújo	3	Eduardo Neves Dom, 14 Ago 2016, 07:41
Dúvida Questão FGV sobre normalização FNBC	Danniel Albuquerque Araújo	3	Eduardo Neves Dom, 14 Ago 2016, 07:40
Dúvida questão	André Felipe Mendonça Andrade	3	Eduardo Neves Dom, 14 Ago 2016, 07:40
Dúvida questão	Gustavo Mantuan	2	Gustavo Mantuan Dom, 10 Abr 2016, 10:24
Dúvida Aula 5 Questão 32 fgv	Gustavo Mantuan	3	Eduardo Neves Sáb, 9 Abr 2016, 12:36

Gostou? Avalie o curso ☺

Excelente **Comentado por DHEISON**

Qualidade ★★★★★
Preço ★★★★★
Conteúdo ★★★★★
Professor ★★★★★

Após ver aulas de 3 professores diferentes tava começando a achar que o problema era comigo, mas finalmente uma boa didática para entender 100% o conteúdo proposto, recomendado! (Postado em 24/02/16)

Excelente Curso de BD **Comentado por RAFAEL NOGUEIRA**

Qualidade ★★★★★
Preço ★★★★★
Conteúdo ★★★★★
Professor ★★★★★

Curso muito didático com explicações na medida! :)
Melhor explicação que ja vi sobre conceitos confusos e abstratos como o ansi-spark e DBA vs AD!

5 Estrelas fácil! (Postado em 03/02/16)

Muito bom! **Comentado por Leonardo**

Qualidade ★★★★★
Preço ★★★★★
Conteúdo ★★★★★
Professor ★★★★★

Excelente Professor!

Teoria na medida certa e muitos exercícios!

Só em Arquitetura ANSI/SPARC foram mais de 30 exercícios!

Recomendado!

Grande Abraço! (Postado em 03/11/15)

Apresentação

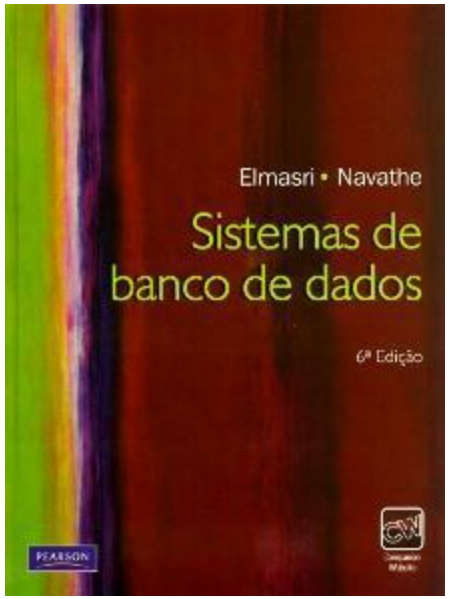
- Formação
  - Bacharel em Ciência da Computação/UFRJ
  - Pós-graduação – Gestão de Projetos
- Analista de Sistemas – BNDES – 2013
- Principais aprovações
  - 4º – BNDES/2012 – CESGRANRIO
  - 19º – BNDES/2011 – CESGRANRIO
  - 30º – Petrobras/ 2011 – CESGRANRIO
  - 12º – FINEP/2011 – CESGRANRIO
  - 3º – Transpetro/2011 – CESGRANRIO
  - 42º – TRT-RJ/2011 – FCC
  - 7º – Petrobras Macaé/2010 – CESGRANRIO
  - 1º – Caixa Econômica Federal/2010 – Nível médio informática – CESPE
  - 18º – BR Distribuidora/2010 – Analista SAP – CESGRANRIO

Dicas de estudo Gerais

- **NÃO DESISTIR!!!!**
- Aprender com os próprios erros
- Estudar as matérias em ciclos
  - https://goo.gl/1sxeTE
- Ter uma forma de revisão
  - Anki
    - https://goo.gl/SP1tJt
  - Mapa Mental
  - Resumos
- Fazer muitos exercícios
  - Da mesma banca primeiro da mais recente para a mais antiga
  - De outras bancas
  - Usar sites de questões
    - (gabaritou, qconcursos.com, mapadaprova, tecconcursos, ...)
- Timasters
  - Tirar dúvidas, compartilhar conhecimento.

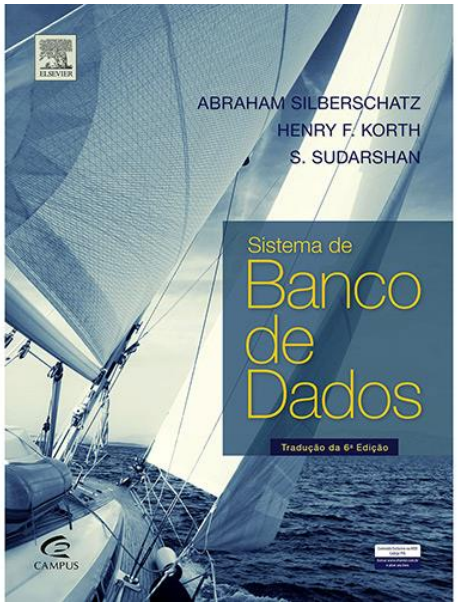
Bibliografia

Sistemas de Banco de Dados – 6ª edição
Ramez E. Elmasri, Shamkant B. **Navathe**
Editora Pearson, 2011



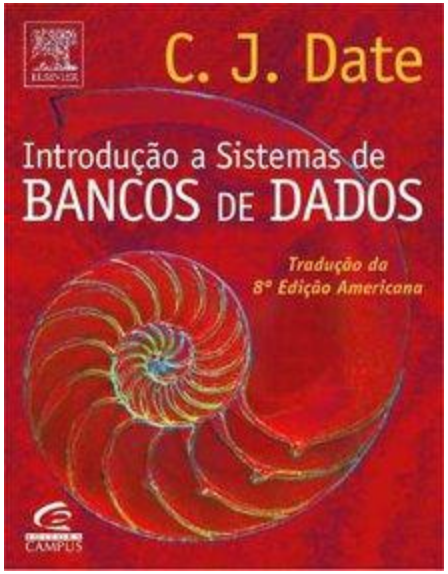
Bibliografia

Sistemas de Banco de Dados – 6ª edição
Abraham **Silberschatz**, Henry F. **Korth**
Editora Campus, 2012



Bibliografia

Introdução a Sistemas de Banco de Dados
C.J. **Date**
Editora Campus, 2004



PROVAS DE TI
TUDO PARA VOCÊ PASSAR

BANCO DE DADOS
SQL DML – Parte 2 ☺

Prof. Eduardo Neves
edumneves@gmail.com
http://www.itnerante.com.br/profile/EduardoNeves

Dúvidas de Banco de Dados:
http://www.itnerante.com.br/group/bancodedados

Artigos do Blog:
http://goo.gl/Sm6JgV

GROUP BY

Organiza a tabela em grupos que possuam o mesmo valor de atributo.

Esta organização é lógica, e não se dá no nível físico do banco de dados.

Vendas

id_venda	id_cli	Data	Valor
1	10	23/01/14	7000
2	20	17/04/13	6000
3	50	20/05/13	3000
4	10	17/02/14	4000

```
/* Quantidade de vendas por cliente */
SELECT id_cli, count(*) as quant
FROM vendas
GROUP BY id_cli
```

id_cli	quant
10	2
20	1
50	1

Funções Agregadas

```
-- Quantidade de linhas na tabela pessoa
SELECT COUNT(*) FROM PESSOA;

/*Quantidade de linhas com o valor nome preenchido na tabela pessoa
Não considera valores nulos no campo na contagem.
*/
SELECT COUNT(NOME) FROM PESSOA;

--Quantidade de nomes distintos na tabela pessoa
SELECT COUNT(DISTINCT NOME) FROM PESSOA;
```

Funções Agregadas

-- Somatório dos valores dos campos idade de todas as pessoas
SELECT **SUM**(idade) FROM PESSOA;

-- Média de idades de todas as pessoas. Não considera valores nulos.
SELECT **AVG**(idade) FROM PESSOA;

-- Idade máxima de todas as pessoas.
SELECT **MAX**(idade) FROM PESSOA;

-- Idade mínima de todas as pessoas.
SELECT **MIN**(idade) FROM PESSOA;

Vendas

id_venda	id_cli	Data	Valor
1	10	23/01/14	7000
2	20	17/04/13	6000
3	50	20/05/13	3000
4	10	17/02/14	4000

```
/* Quantidade total de vendas */
SELECT count(*) as quant
FROM vendas
```

Quant
4

```
/* Máximo valor de compra */
SELECT max(valor) as max_valor
FROM vendas
```

Max_valor
7000

```
/* Valor médio de compras */
SELECT avg(valor) as med_valor
FROM vendas
```

Med_valor
5000

Vendas

id_venda	id_cli	Data	Valor
1	10	23/01/14	7000
2	20	17/04/13	6000
3	50	20/05/13	3000
4	10	17/02/14	4000

```
/* Quantidade de vendas por cliente */
SELECT id_cli, count(*) as quant
FROM vendas
GROUP BY id_cli
```

id_cli	quant
10	2
20	1
50	1

```
/* Máximo valor de compra por cliente */
SELECT id_cli, max(valor) as max_valor
FROM vendas
GROUP BY id_cli
```

id_cli	Max_valor
10	7000
20	6000
50	3000

```
/* Valor médio de compra por cliente */
SELECT id_cli, avg(valor) as med_valor
FROM vendas
GROUP BY id_cli
```

id_cli	Med_valor
10	5500
20	6000
50	3000

Vendas

id_venda	id_cli	Data	Valor
1	10	23/01/14	7000
2	20	17/04/13	6000
3	50	20/05/13	3000
4	10	17/02/14	4000

```
/* Quantidade de vendas total, valor total, menor data de venda */
SELECT
        count(*) as quant,
        sum(valor) as soma,
        min(data) as menor
FROM vendas
```

Quant	Soma	Menor
4	20000	17/04/13

Vendas

id_venda	id_cli	Data	Valor
1	10	23/01/14	7000
2	20	17/04/13	6000
3	50	20/05/13	3000
4	10	17/02/14	4000

```
/* Quantidade de vendas, valor total, menor data por cliente */
SELECT
        id_cli,
        count(*) as quant,
        sum(valor) as soma,
        min(data) as menor
FROM vendas
GROUP BY id_cli
```

id_cli	quant	Soma	Menor
10	2	11000	23/01/14
20	1	6000	17/04/13
50	1	3000	20/05/13

Vendas

id_venda	id_cli	Data	Valor
1	10	23/01/14	7000
2	20	17/04/13	6000
3	50	20/05/13	3000
4	10	17/02/14	4000

```
/* Quantidade de vendas, valor total, menor data por cliente */
SELECT
        count(*) as quant,
        sum(valor) as soma,
        min(data) as menor
FROM vendas
GROUP BY id_cli
```

quant	Soma	Menor
2	11000	23/01/14
1	6000	17/04/13
1	3000	20/05/13

Q1 – FGV – TJSC – Analista de Sistemas – 2015

João escreveu os dois comandos SQL abaixo.

I. select * from T where x>5

II. select * from T where not x>5

Curiosamente, os dois comandos produziram resultados com zero linha. Intrigado, pois sabia que a instância de T não estava vazia, João escreveu outros dois comandos:

III. select count(*) from T

IV. select count(x) from T

Quando executados, os comandos III e IV, necessariamente, produziram resultados r1 e r2, tal que:

(A) r1 = 0 e r2 = 0;

(B) r1 é um número ≥ 1 e r2 = 0;

(C) r1 é um número ≥ 0 e r2 é NULL;

(D) r1 é NULL e r2 é NULL;

(E) r1 = 0 e r2 é NULL.

I. select * from T where x>5

II. select * from T where not x>5

III. select count(*) from T

IV. select count(x) from T

Q1 – FGV – TJSC – Analista de Sistemas – 2015

João escreveu os dois comandos SQL abaixo.

I. select * from T where x>5

II. select * from T where not x>5

Curiosamente, os dois comandos produziram resultados com zero linha. Intrigado, pois sabia que a instância de T não estava vazia, João escreveu outros dois comandos:

III. select count(*) from T

IV. select count(x) from T

Quando executados, os comandos III e IV, necessariamente, produziram resultados r1 e r2, tal que:

(A) r1 = 0 e r2 = 0;

(B) r1 é um número ≥ 1 e r2 = 0;

(C) r1 é um número ≥ 0 e r2 é NULL;

(D) r1 é NULL e r2 é NULL;

(E) r1 = 0 e r2 é NULL.

Vendas

id_venda	id_cli	Data	Valor
1	10	23/01/14	7000
2	20	17/04/13	6000
3	50	20/05/13	3000
4	10	17/02/14	4000

```
/* Filtrar antes do agrupamento deve ser feito com o WHERE */
SELECT
        id_cli,
        count(*) as quant,
        sum(valor) as soma,
        min(data) as menor
FROM vendas
WHERE valor < 5000
GROUP BY id_cli
```

id_cli	quant	Soma	Menor
50	1	3000	20/05/13
10	1	4000	17/02/14

Vendas

id_venda	id_cli	Data	Valor
1	10	23/01/14	7000
2	20	17/04/13	6000
3	50	20/05/13	3000
4	10	17/02/14	4000

```
/* Filtrar depois do agrupamento deve ser feito com o HAVING */
SELECT
        id_cli,
        count(*) as quant,
        sum(valor) as soma,
        min(data) as menor
FROM vendas
GROUP BY id_cli
HAVING sum(valor) > 7000
```

id_cli	quant	Soma	Menor
10	2	11000	23/01/14
20	1	6000	17/04/13
50	1	3000	20/05/13

Vendas

id_venda	id_cli	Data	Valor
1	10	23/01/14	7000
2	20	17/04/13	6000
3	50	20/05/13	3000
4	10	17/02/14	4000

```
/* Filtrar depois do agrupamento deve ser feito com o HAVING */
SELECT
        id_cli,
        count(*) as quant,
        sum(valor) as soma,
        min(data) as menor
FROM vendas
WHERE id_cli in (10,20)
GROUP BY id_cli
HAVING count(*) > 1
```

id_cli	quant	Soma	Menor
10	2	11000	23/01/14
20	1	6000	17/04/13

Vendas

id_venda	id_cli	Data	Valor
1	10	23/01/14	7000
2	20	17/04/13	6000
3	50	20/05/13	3000
4	10	17/02/14	4000

```
/* Filtrar depois do agrupamento deve ser feito com o HAVING */
SELECT
        count(*) as quant,
        min(data) as menor
FROM vendas
GROUP BY id_cli
HAVING sum(valor) > 7000
ORDER BY quant
```

id_cli	quant	Soma	Menor
10	2	11000	23/01/14
20	1	6000	17/04/13
50	1	3000	20/05/13

quant	Menor
2	23/01/14

OBS: No order by podemos usar alias de colunas
Alias de colunas não podem ser usados no WHERE e HAVING
OBS2: No MySQL podemos usar no HAVING

```
/*
• Podemos agrupar por coluna que não está no select
• Podemos usar no Having e no order by funções de agregação que não
  estão no select
• Podemos usar WHERE e HAVING na mesma consulta
*/
SELECT
        count(*) as quant,
        min(data) as menor
FROM vendas
WHERE   year(data) < 2015
GROUP BY id_cli
HAVING sum(valor) > 7000
ORDER BY avg(valor)
```

OBS: O order by é feito no final, então só podemos usar colunas e funções que fazem sentido no grupamento lógico.
No exemplo: id_cli, quant, menor e qualquer função de agregação

Q2 – FGV – PGE-RO – Micro – 2015

Nos comandos SQL que empregam funções de agregação em conjunto com a cláusula group by, é possível filtrar linhas do resultado depois da aplicação das funções de agregação por meio da cláusula:

(A) Case;
(B) Distinctrow;
(C) Full outer join;
(D) Having;
(E) Where.

Q2 – FGV – PGE-RO – Micro – 2015

Nos comandos SQL que empregam funções de agregação em conjunto com a cláusula group by, é possível filtrar linhas do resultado depois da aplicação das funções de agregação por meio da cláusula:

(A) Case;
(B) Distinctrow;
(C) Full outer join;
(D) Having;
(E) Where.

Vendas

id_venda	id_cli	Data	Valor
1	10	23/01/14	7000
2	20	17/04/13	6000
3	50	20/05/13	3000
4	10	17/02/14	4000

```
SELECT count(*) as a, sum(valor) as b, max(data) as c
FROM vendas
Where 1 = 1
```

a	b	c
4	20000	17/02/14

```
SELECT count(*) as a, sum(valor) as b, max(data) as c
FROM vendas
Where 1 = 2
```

a	b	c
0	null	null

Vendas

id_venda	id_cli	Data	Valor
1	10	23/01/14	7000
2	20	17/04/13	6000
3	50	20/05/13	3000
4	10	17/02/14	4000

Clientes

id_cli	Nome	DataNasc	Endereco	IDCidade
10	Ana	01/01/80	X	101
20	Caio	17/04/82	Z	101
30	Lia	13/03/79	A	103
40	Luiz	22/09/90	C	103
50	Lais	10/11/70	E	102

```
/* Junção de tabelas, mostre o nome dos clientes que fizeram compras */
SELECT nome
FROM Clientes, vendas
WHERE clientes.id_cli = vendas.id_cli

/* Podemos usar alias (apelidos) para as tabelas assim como para as colunas */
SELECT
        nome 'Nome Cliente',
        v.valor as VendaValor
FROM Clientes c, vendas as v
WHERE c.id_cli = v.id_cli
```

Obs: O "AS" pode ser omitido, tanto na coluna quanto na tabela.

Vendas

id_venda	id_cli	Data	Valor
1	10	23/01/14	7000
2	20	17/04/13	6000
3	50	20/05/13	3000
4	10	17/02/14	4000

Clientes

id_cli	Nome	DataNasc	Endereco	IDCidade
10	Ana	01/01/80	X	101
20	Caio	17/04/82	Z	101
30	Lia	13/03/79	A	103
40	Luiz	22/09/90	C	103
50	Lais	10/11/70	E	102

```
/* Mesma consulta usando JOIN  */
SELECT nome
FROM   Clientes JOIN
       vendas ON (clientes.id_cli = vendas.id_cli)

/* O termo INNER é opcional */
SELECT
       nome 'Nome Cliente',
       v.valor as VendaValor
FROM   Clientes c INNER JOIN
       vendas as v ON (c.id_cli = v.id_cli)
```

Vendas

id_venda	id_cli	Data	Valor
1	10	23/01/14	7000
2	20	17/04/13	6000
3	50	20/05/13	3000
4	10	17/02/14	4000

Clientes

id_cli	Nome	DataNasc	Endereco	IDCidade
10	Ana	01/01/80	X	101
20	Caio	17/04/82	Z	101
30	Lia	13/03/79	A	103
40	Luiz	22/09/90	C	103
50	Lais	10/11/70	E	102

/* A junção pode ser feita especificando a coluna com o USING
Só pode ser feita quando a coluna tem o mesmo nome nas duas tabelas
*/

```
SELECT nome
FROM   Clientes INNER JOIN
       vendas USING (id_cli)
```

Funcionario

Id	Nome	DataNasc	Endereco	Id_gerente
10	Ana	01/01/80	X	20
20	Caio	17/04/82	Z	NULL
30	Lia	13/03/79	A	20
40	Luiz	22/09/90	C	50
50	Lais	10/11/70	E	NULL

Gerente

Id	Nome	DataNasc	Endereco	Id_gerente
10	Ana	01/01/80	X	20
20	Caio	17/04/82	Z	NULL
30	Lia	13/03/79	A	20
40	Luiz	22/09/90	C	50
50	Lais	10/11/70	E	NULL

/* Podemos fazer junção com a mesma tabela, mostre o nome do funcionário e seu gerente */

SELECT func.nome as nome_func, ger.nome as nome_gerente

FROM funcionario func, funcionario ger

WHERE func.id_gerente = ger.id

Nome_func	Nome_Gerente
Ana	Caio
Lia	Caio
Luiz	Lais

Funcionario

Id	Nome	DataNasc	Endereco	Id_gerente
10	Ana	01/01/80	X	20
20	Caio	17/04/82	Z	NULL
30	Lia	13/03/79	A	20
40	Luiz	22/09/90	C	50
50	Lais	10/11/70	E	NULL

Gerente

Id	Nome	DataNasc	Endereco	Id_gerente
10	Ana	01/01/80	X	20
20	Caio	17/04/82	Z	NULL
30	Lia	13/03/79	A	20
40	Luiz	22/09/90	C	50
50	Lais	10/11/70	E	NULL

/* Podemos fazer junção com a mesma tabela, mostre o nome do funcionário e seu gerente */

```
SELECT func.nome as nome_func, ger.nome as nome_gerente
FROM   funcionario func JOIN
       funcionario ger ON (func.id_gerente = ger.id)
```

Nome_func	Nome_Gerente
Ana	Caio
Lia	Caio
Luiz	Lais

Funcionario

Id	Nome	DataNasc	Endereco	Id_gerente
10	Ana	01/01/80	X	20
20	Caio	17/04/82	Z	NULL
30	Lia	13/03/79	A	20
40	Luiz	22/09/90	C	50
50	Lais	10/11/70	E	NULL

Gerente

Id	Nome	DataNasc	Endereco	Id_gerente
10	Ana	01/01/80	X	20
20	Caio	17/04/82	Z	NULL
30	Lia	13/03/79	A	20
40	Luiz	22/09/90	C	50
50	Lais	10/11/70	E	NULL

```
SELECT func.nome as nome_func, ger.nome as nome_gerente
FROM   funcionario func LEFT JOIN
       funcionario ger ON (func.id_gerente = ger.id)
```

Nome_func	Nome_Gerente
Ana	Caio
Caio	Null
Lia	Caio
Luiz	Lais
Lais	NULL

Vendas

id_venda	id_cli	Valor
1	10	7000
2	20	6000
3	50	3000
4	10	4000

Clientes

id_cli	Nome	IDCidade
10	Ana	101
20	Caio	101
30	Lia	103
40	Luiz	103
50	Lais	102

Cidades

ID Cid	Nome	UF
101	Rio de Janeiro	RJ
102	Cataguases	MG
103	Vitória	ES
104	Petrópolis	RJ

/* Junção de várias tabelas, mostre o valor da venda, o nome e a cidade dos clientes de todas as compras com valor acima de 3000 reais */

```
SELECT
        v.valor,
        cli.nome as 'Nome Cliente',
        cid.nome as Cidade
FROM
        clientes cli, vendas v, cidades cid
WHERE
        cli.id_cli = v.id_cli AND
        cli.idCidade = cid.id_cid AND
        v.valor > 3000
```

Valor	Nome Cliente	Cidade
7000	Ana	Rio de Janeiro
6000	Caio	Rio de Janeiro
4000	Ana	Rio de Janeiro

Vendas

id_venda	id_cli	Valor
1	10	7000
2	20	6000
3	50	3000
4	10	4000

Clientes

id_cli	Nome	IDCidade
10	Ana	101
20	Caio	101
30	Lia	103
40	Luiz	103
50	Lais	102

Cidades

ID Cid	Nome	UF
101	Rio de Janeiro	RJ
102	Cataguases	MG
103	Vitória	ES
104	Petrópolis	RJ

/* Junção de várias tabelas, mostre o valor da venda, o nome e a cidade dos clientes de todas as compras com valor acima de 3000 reais */

```
SELECT
        v.valor,
        cli.nome as 'Nome Cliente',
        cid.nome as Cidade
FROM
        clientes cli JOIN
        vendas v ON (cli.id_cli = v.id_cli) JOIN
        cidades cid ON (cli.idCidade = cid.id_cid)
WHERE
        v.valor > 3000
```

Valor	Nome Cliente	Cidade
7000	Ana	Rio de Janeiro
6000	Caio	Rio de Janeiro
4000	Ana	Rio de Janeiro

Vendas

id_venda	id_cli	Data	Valor
1	10	23/01/14	7000
2	20	17/04/13	6000
3	50	20/05/13	3000
4	10	17/02/14	4000

Clientes

id_cli	Nome	DataNasc	Endereco	IDCidade
10	Ana	01/01/80	X	101
20	Caio	17/04/82	Z	101
30	Lia	13/03/79	A	103
40	Luiz	22/09/90	C	103
50	Lais	10/11/70	E	102

```
/* Sub-consultas */
SELECT
        nome,
        (select sum(valor) from vendas) as soma,
        (select count(*) from vendas) as quant
FROM clientes
```

nome	soma	Quant
Ana	20000	4
Caio	20000	4
Lia	20000	4
Luiz	20000	4
Lais	20000	4

Vendas

id_venda	id_cli	Data	Valor
1	10	23/01/14	7000
2	20	17/04/13	6000
3	50	20/05/13	3000
4	10	17/02/14	4000

Clientes

id_cli	Nome	DataNasc	Endereco	IDCidade
10	Ana	01/01/80	X	101
20	Caio	17/04/82	Z	101
30	Lia	13/03/79	A	103
40	Luiz	22/09/90	C	103
50	Lais	10/11/70	E	102

```
/* Sub-consultas
Valor total de vendas por cliente e quantidade de compras  */
SELECT
        nome,
        (select sum(valor) from vendas v where v.id_cli = c.id_cli) as soma,
        (select count(*) from vendas v where v.id_cli = c.id_cli) as quant
FROM clientes c
```

Nome	soma	Quant
Ana	11000	2
Caio	6000	1
Lia	Null	0
Luiz	Null	0
Lais	3000	1

Vendas

id_venda	id_cli	Data	Valor
1	10	23/01/14	7000
2	20	17/04/13	6000
3	50	20/05/13	3000
4	10	17/02/14	4000

Clientes

id_cli	Nome	DataNasc	Endereco	IDCidade
10	Ana	01/01/80	X	101
20	Caio	17/04/82	Z	101
30	Lia	13/03/79	A	103
40	Luiz	22/09/90	C	103
50	Lais	10/11/70	E	102

/* Não é o mesmo resultado, devido ao JOIN */

```
SELECT
        c.nome,
        sum(v.valor) as soma,
        count(*) as quant
FROM clientes c, vendas v
WHERE v.id_cli = c.id_cli
GROUP BY c.nome
```

C.Nome	v.Valor
Ana	7000
Ana	4000
Caio	6000
Lais	3000

Nome	soma	Quant
Ana	11000	2
Caio	6000	1
Lais	3000	1

Vendas

id_venda	id_cli	Data	Valor
1	10	23/01/14	7000
2	20	17/04/13	6000
3	50	20/05/13	3000
4	10	17/02/14	4000

Clientes

id_cli	Nome	DataNasc	Endereco	IDCidade
10	Ana	01/01/80	X	101
20	Caio	17/04/82	Z	101
30	Lia	13/03/79	A	103
40	Luiz	22/09/90	C	103
50	Lais	10/11/70	E	102

```
/* Usando LEFT JOIN  */
SELECT
        c.nome,
        sum(v.valor) as soma,
        count(*) as quant
FROM clientes c LEFT JOIN
      vendas v on (v.id_cli = c.id_cli)
GROUP BY c.nome
```

C.Nome	v.Valor
Ana	7000
Ana	4000
Caio	6000
Lia	NULL
Luiz	NULL
Lais	3000

Nome	soma	Quant
Ana	11000	2
Caio	6000	1
Lia	NULL	1
Luiz	NULL	1
Lais	3000	1

Se quisesse contar a quantidade de vendas
Deveria utilizar count(id_venda)
por exemplo

Vendas

id_venda	id_cli	Data	Valor
1	10	23/01/14	7000
2	20	17/04/13	6000
3	50	20/05/13	3000
4	10	17/02/14	4000

Clientes

id_cli	Nome	DataNasc	Endereco	IDCidade
10	Ana	01/01/80	X	101
20	Caio	17/04/82	Z	101
30	Lia	13/03/79	A	103
40	Luiz	22/09/90	C	103
50	Lais	10/11/70	E	102

```
/* Usando LEFT JOIN  */
SELECT
        c.nome,
        sum(v.valor) as soma,
        count(v.valor) as quant
FROM clientes c LEFT JOIN
      vendas v on (v.id_cli = c.id_cli)
GROUP BY c.nome
```

C.Nome	v.Valor
Ana	7000
Ana	4000
Caio	6000
Lia	NULL
Luiz	NULL
Lais	3000

Nome	soma	Quant
Ana	11000	2
Caio	6000	1
Lia	NULL	0
Luiz	NULL	0
Lais	3000	1

Q3 – FGV – COMPESA – 2014

Considere as tabelas T1 e T2, com as respectivas instâncias, a seguir.

T1

a	B
2	4
3	2
1	2

T2

a	c
2	8
2	4
3	1
3	5
1	7

Sabe-se que o comando

```
select x.a,count(*) x from T1 x, T2 y
where x.a=y.a
group by x.a
order by . . .
```

produziu

a	x
3	2
2	2
1	1

Nesse caso, é correto afirmar que a cláusula *order by* usada no comando foi

(A) **order by 2**
(B) **order by avg(y.c)**
(C) **order by x.b**
(D) **order by sum(y.c)**
(E) **order by count(y.c)**

T1

a	B
2	4
3	2
1	2

T2

a	c
2	8
2	4
3	1
3	5
1	7

```
select x.a,count(*) x from T1 x, T2 y
where x.a=y.a
group by x.a
order by . . .
```

produziu

a	x
3	2
2	2
1	1

x.a	x.b	y.a	y.c
2	4	2	8
2	4	2	4
3	2	3	1
3	2	3	5
1	2	1	7

x.a	Count(*)
2	2
3	2
1	1

Order by 2

a	X
1	1
2	2
3	2

T1

a	B
2	4
3	2
1	2

T2

a	c
2	8
2	4
3	1
3	5
1	7

```
select x.a,count(*) x from T1 x, T2 y
where x.a=y.a
group by x.a
order by . . .
```

produziu

a	x
3	2
2	2
1	1

x.a	x.b	y.a	y.c
2	4	2	8
2	4	2	4
3	2	3	1
3	2	3	5
1	2	1	7

x.a	count(*)	avg(y.c)
2	2	6
3	2	3
1	1	7

order by avg(y.c)

x.a	count(*)	Avg(y.c)
3	2	3
2	2	6
1	1	7

T1

a	B
2	4
3	2
1	2

T2

a	c
2	8
2	4
3	1
3	5
1	7

```
select x.a,count(*) x from T1 x, T2 y
where x.a=y.a
group by x.a
order by . . .
```

produziu

a	x
3	2
2	2
1	1

x.a	x.b	y.a	y.c
2	4	2	8
2	4	2	4
3	2	3	1
3	2	3	5
1	2	1	7

x.a	Count(*)
2	2
3	2
1	1

order by x.b

ERRO!!!
x.b não faz sentido no grupamento lógico.

T1

a	B
2	4
3	2
1	2

T2

a	c
2	8
2	4
3	1
3	5
1	7

```
select x.a,count(*) x from T1 x, T2 y
where x.a=y.a
group by x.a
order by . . .
```

produziu

a	x
3	2
2	2
1	1

x.a	x.b	y.a	y.c
2	4	2	8
2	4	2	4
3	2	3	1
3	2	3	5
1	2	1	7

x.a	count(*)	sum(y.c)
2	2	12
3	2	6
1	1	7

order by sum(y.c)

x.a	count(*)	sum(y.c)
3	2	6
1	1	7
2	2	12

T1

a	B
2	4
3	2
1	2

T2

a	c
2	8
2	4
3	1
3	5
1	7

```
select x.a,count(*) x from T1 x, T2 y
where x.a=y.a
group by x.a
order by . . .
```

produziu

a	x
3	2
2	2
1	1

x.a	x.b	y.a	y.c
2	4	2	8
2	4	2	4
3	2	3	1
3	2	3	5
1	2	1	7

x.a	count(*)	count(y.c)
2	2	2
3	2	2
1	1	1

order by count(y.c)

x.a	count(*)	count(y.c)
1	1	1
2	2	2
3	2	2

Q3 – FGV – COMPESA – 2014

51 Considere as tabelas T1 e T2, com as respectivas instâncias, a seguir.

T1

a	B
2	4
3	2
1	2

T2

a	c
2	8
2	4
3	1
3	5
1	7

Sabe-se que o comando

```
select x.a,count(*) x from T1 x, T2 y
where x.a=y.a
group by x.a
order by . . .
```

produziu

a	x
3	2
2	2
1	1

Nesse caso, é correto afirmar que a cláusula *order by* usada no comando foi

(A) **order by 2**
(B) **order by avg(y.c)**
(C) **order by x.b**
(D) **order by sum(y.c)**
(E) **order by count(y.c)**

Q3 – FGV – COMPESA – 2014

51 Considere as tabelas T1 e T2, com as respectivas instâncias, a seguir.

T1

a	B
2	4
3	2
1	2

T2

a	c
2	8
2	4
3	1
3	5
1	7

Sabe-se que o comando

```
select x.a,count(*) x from T1 x, T2 y
where x.a=y.a
group by x.a
order by . . .
```

produziu

a	x
3	2
2	2
1	1

Nesse caso, é correto afirmar que a cláusula *order by* usada no comando foi

(A) **order by 2**

(B) **order by avg(y.c)**

(C) **order by x.b**

(D) **order by sum(y.c)**

(E) **order by count(y.c)**

Q4 – FGV – TCMSP –2015

Considere a tabela relacional criada pelo comando

```
create table xx
(a int null, b int null, c int null)
```

Depois de instanciada com um conjunto de registros, os seguintes comandos foram executados:

```
select count(*) from XX
select count(distinct A) from XX
select count(distinct B) from XX
select count(*) from XX where C>10
select count(*) from XX where not C>10
```

Sabendo-se que esses comandos produziram como resultado, respectivamente, os números 10, 10, 0, 0 e 5, analise as quarto alternativas para a definição da tabela XX:

A lista com todos os comandos que são válidos e compatíveis com a instância corrente da tabela é:

```
10  select count(*) from XX
10  select count(distinct A) from XX
0   select count(distinct B) from XX
0   select count(*) from XX where C>10
5   select count(*) from XX where not C>10
```

I.
```
CREATE TABLE XX(
        A int NULL,
        B int NULL,
        C int NULL )
```

II.
```
CREATE TABLE XX(
        A int primary key,
        B int NULL,
        C int NULL )
```

III.
```
CREATE TABLE XX(
        A int NULL,
        B int NULL,
        C int )
```

IV.
```
CREATE TABLE XX(
        A int,
        B int primary key,
        C int NULL )
```

(A) I, II;
(B) I, II, III;
(C) II, IV;
(D) I, III;
(E) IV.

```
10  select count(*) from XX
10  select count(distinct A) from XX
0   select count(distinct B) from XX
0   select count(*) from XX where C>10
5   select count(*) from XX where not C>10
```

A	B	C
1	null	1
2	Null	1
3	Null	0
4	Null	0
5	Null	5
6	Null	Null
7	Null	Null
8	Null	Null
9	Null	Null
10	Null	Null

I.
```
CREATE TABLE XX(
        A int NULL,
        B int NULL,
        C int NULL )
```

II.
```
CREATE TABLE XX(
        A int primary key,
        B int NULL,
        C int NULL )
```

III.
```
CREATE TABLE XX(
        A int NULL,
        B int NULL,
        C int )
```

IV.
```
CREATE TABLE XX(
        A int,
        B int primary key,
        C int NULL )
```

```
10  select count(*) from XX
10  select count(distinct A) from XX
0   select count(distinct B) from XX
0   select count(*) from XX where C>10
5   select count(*) from XX where not C>10
```

I.
```
CREATE TABLE XX(
        A int NULL,
        B int NULL,
        C int NULL )
```

II.
```
CREATE TABLE XX(
        A int primary key,
        B int NULL,
        C int NULL )
```

III.
```
CREATE TABLE XX(
        A int NULL,
        B int NULL,
        C int )
```

IV.
```
CREATE TABLE XX(
        A int,
        B int primary key,
        C int NULL )
```

(A) I, II;
(B) I, II, III;
(C) II, IV;
(D) I, III;
(E) IV.

```
10  select count(*) from XX
10  select count(distinct A) from XX
0   select count(distinct B) from XX
0   select count(*) from XX where C>10
5   select count(*) from XX where not C>10
```

I.

```
CREATE TABLE XX(
        A int NULL,
        B int NULL,
        C int NULL )
```

II.

```
CREATE TABLE XX(
        A int primary key,
        B int NULL,
        C int NULL )
```

III.

```
CREATE TABLE XX(
        A int NULL,
        B int NULL,
        C int )
```

IV.

```
CREATE TABLE XX(
        A int,
        B int primary key,
        C int NULL )
```

(A) I, II;
(B) I, II, III;
(C) II, IV;
(D) I, III;
(E) IV.

Mudou com recursos para B

Q5 – FGV – DPEMT – 2015

Analise o comando SQL a seguir.

```
select x,
       sum (nota) as soma,
       count (nota) as numero
from inscricao
group by x
having ???
```

Assinale a opção que indica a expressão que, ao ser utilizada para

substituir o trecho “???”, invalida o comando SQL acima.

(A) count (nota) > 1

(B) x=2

(C) max (nota) = 10

(D) nota > 7

(E) (select max(nota) from inscricao) > 5

Q5 – FGV – DPEMT – 2015

Analise o comando SQL a seguir.

```
select x,
       sum (nota) as soma,
       count (nota) as numero
from inscricao
group by x
having ???
```

Assinale a opção que indica a expressão que, ao ser utilizada para
substituir o trecho “???”, invalida o comando SQL acima.

(A) count (nota) > 1

(B) x=2

(C) max (nota) = 10

(D) nota > 7

(E) (select max(nota) from inscricao) > 5

Q6 – FGV – DPERO – 2015

Analise o comando SQL a seguir.

```
select max(A1) X, count(*) Y, sum(A1) Z
from T
```

Executado quando a instância da tabela T estiver vazia (com zero registros), esse comando produz como resultado:

(A)

X	Y	Z
NULL	0	NULL

(B)

X	Y	Z
NULL	NULL	NULL

(C)

X	Y	Z
0	0	0

(D)

X	Y	Z
0	NULL	0

(E)

X	Y	Z
NULL	NULL	0

Q6 – FGV – DPERO – 2015

Analise o comando SQL a seguir.

```
select max(A1) X, count(*) Y, sum(A1) Z
from T
```

Executado quando a instância da tabela T estiver vazia (com zero registros), esse comando produz como resultado:

(A)

X	Y	Z
NULL	0	NULL

(B)

X	Y	Z
NULL	NULL	NULL

(C)

X	Y	Z
0	0	0

(D)

X	Y	Z
0	NULL	0

(E)

X	Y	Z
NULL	NULL	0

Q7 – CFO/QC – 2014/2015

Considere a tabela W abaixo, seus campos VALOR e DADO, e a consulta em linguagem SQL:

W

VALOR	DADO
1	3
2	4
5	6
4	5

```
SELECT MAX (W.VALOR)
   FROM W LEFT JOIN W AS Z
      ON (W.VALOR = Z.DADO)
WHERE Z.DADO IS NULL;
```

Teremos como retomo da consulta SQL:

a) 1

b) 2

c) 3

d) 4

e) 5

```
SELECT MAX (W.VALOR)
   FROM W LEFT JOIN W AS Z
      ON (W.VALOR = Z.DADO)
   WHERE Z.DADO IS NULL;
```

Valor	Dado
1	3
2	4
5	6
4	5

Valor	Dado
1	3
2	4
5	6
4	5

w.Valor	w.dado	z.valor	z.dado	
1	3	null	Null	
2	4	null	Null	
5	6	4	5	
4	5	2	4	

w.Valor	w.dado	z.valor	z.dado
1	3	null	Null
2	4	null	Null

Q7 – CFO/QC – 2014/2015

Considere a tabela W abaixo, seus campos VALOR e DADO, e a consulta em linguagem SQL:

W

VALOR	DADO
1	3
2	4
5	6
4	5

```
SELECT MAX (W.VALOR)
   FROM W LEFT JOIN W AS Z
      ON (W.VALOR = Z.DADO)
WHERE Z.DADO IS NULL;
```

Teremos como retomo da consulta SQL:

a) 1

b) 2

c) 3

d) 4

e) 5

Q7 – CFO/QC – 2014/2015

Considere a tabela W abaixo, seus campos VALOR e DADO, e a consulta em linguagem SQL:

W

VALOR	DADO
1	3
2	4
5	6
4	5

```
SELECT MAX (W.VALOR)
   FROM W LEFT JOIN W AS Z
      ON (W.VALOR = Z.DADO)
WHERE Z.DADO IS NULL;
```

Teremos como retomo da consulta SQL:

a) 1

b) 2

c) 3

d) 4

e) 5

W

Valor	Dado
1	3
2	4
5	6
5	6
4	5

Z

Valor	Dado
1	3
2	4
5	6
5	6
4	5

```
SELECT Z.VALOR, SUM(W.DADO)
FROM
  W LEFT JOIN W AS Z
  ON (W.VALOR = Z.DADO)
GROUP BY Z.VALOR
```

w.Valor	w.dado	z.valor	z.dado
1	3	null	Null
2	4	null	Null
5	6	4	5
5	6	4	5
4	5	2	4

z.valor	sum(w.dado)
null	7
4	12
2	5

Gabarito

Q1 – B
Q2 – D
Q3 – B
Q4 – A
Q5 – D
Q6 – A
Q7 – B
Q8 – B
Q9 – A
Q10 – D
Q11 – E

PROVAS DE TI
TUDO PARA VOCÊ PASSAR

BANCO DE DADOS
SQL DML – Parte 2 ☺

Prof. Eduardo Neves
edumneves@gmail.com
http://www.itnerante.com.br/profile/EduardoNeves

Dúvidas de Banco de Dados:
http://www.itnerante.com.br/group/bancodedados

Artigos do Blog:
http://goo.gl/Sm6JgV

Subconsultas Correlacionadas

Funcionário

Maticula	Nome	DataNasc
10	Ana	23/01/74
11	Daniel	17/04/83
12	Lucia	20/05/63
13	Rafael	17/02/84

Participacao

Id_Proj	Matricula
P1	11
P2	11
P3	11
P1	13
P2	13
P1	12

Projeto

Id_Proj	Nome	Data Inicio
P1	PROJ1	23/01/14
P2	PROJ2	17/04/13
P3	PROJ3	20/05/13

Funcionários que participaram de pelo menos um projeto.

```
SELECT fun.nome
FROM funcionario fun
WHERE
        EXISTS (SELECT * FROM projeto prj
                WHERE EXISTS (SELECT * FROM participacao par
                              WHERE par.matricula = fun.matricula AND
                                    par.id_proj = prj.id_proj))
```

Nome
Daniel
Lucia
Rafael

Subconsultas Correlacionadas

Funcionário

Maticula	Nome	DataNasc
10	Ana	23/01/74
11	Daniel	17/04/83
12	Lucia	20/05/63
13	Rafael	17/02/84

Participacao

Id_Proj	Matricula
P1	11
P2	11
P3	11
P1	13
P2	13
P1	12

Projeto

Id_Proj	Nome	Data Inicio
P1	PROJ1	23/01/14
P2	PROJ2	17/04/13
P3	PROJ3	20/05/13

Funcionários que não participaram em nenhum projeto

```
SELECT fun.nome
FROM funcionario fun
WHERE
        NOT EXISTS (SELECT * FROM projeto prj
                WHERE EXISTS (SELECT * FROM participacao par
                        WHERE par.matricula = fun.matricula AND
                                par.id_proj = prj.id_proj))
```

Nome
Ana

Subconsultas Correlacionadas

Funcionário

Maticula	Nome	DataNasc
10	Ana	23/01/74
11	Daniel	17/04/83
12	Lucia	20/05/63
13	Rafael	17/02/84

Participacao

Id_Proj	Matricula
P1	11
P2	11
P3	11
P1	13
P2	13
P1	12

Projeto

Id_Proj	Nome	Data Inicio
P1	PROJ1	23/01/14
P2	PROJ2	17/04/13
P3	PROJ3	20/05/13

Funcionários que não participaram em pelo menos um projeto

```
SELECT fun.nome
FROM funcionario fun
WHERE
        EXISTS (SELECT * FROM projeto prj
                WHERE NOT EXISTS (SELECT * FROM participacao par
                                  WHERE par.matricula = fun.matricula AND
                                        par.id_proj = prj.id_proj))
```

Nome
Ana
Lucia
Rafael

Subconsultas Correlacionadas

Funcionário

Maticula	Nome	DataNasc
10	Ana	23/01/74
11	Daniel	17/04/83
12	Lucia	20/05/63
13	Rafael	17/02/84

Participacao

Id_Proj	Matricula
P1	11
P2	11
P3	11
P1	13
P2	13
P1	12

Projeto

Id_Proj	Nome	Data Inicio
P1	PROJ1	23/01/14
P2	PROJ2	17/04/13
P3	PROJ3	20/05/13

Funcionários que participaram de todos os projetos

```
SELECT fun.nome
FROM funcionario fun
WHERE
        NOT EXISTS (SELECT * FROM projeto prj
                WHERE NOT EXISTS (SELECT * FROM participacao par
                        WHERE par.matricula = fun.matricula AND
                                par.id_proj = prj.id_proj))
```

Nome
Daniel

Q8 – Considere um banco de dados relacional que possua as tabelas a seguir, e que em cada tabela os atributos sublinhados formem a chave primária:

PRODUTO (idP, descricaoP)
LOJA (idL, nomeL, endereçoL)
OFERTA (idP, idL, preçoO)

Nesse banco de dados a tabela Oferta contém informações sobre as lojas onde cada produto é vendido e o respectivo preço, e todas as tabelas possuem pelo menos um registro. Sendo assim, o comando SQL

```
select p.descricaoP
from produto p
where not exists
        (select *
         from loja l
         where not exists
                (select *
                 from oferta o
                 where o.idP=p.idP and o.idL=l.idL))
```

```
select p.descricaoP
from produto p
where not exists
        (select *
         from loja l
         where not exists
                 (select *
                  from oferta o
                  where o.idP=p.idP and o.idL=l.idL))
```

PRODUTO (idP, descricaoP)
LOJA (idL, nomeL, endereçoL)
OFERTA (idP, idL, preçoO)

produz a lista de produtos que:

a) nenhuma das lojas vende;
b) todas as lojas vendem;
c) são vendidos apenas em lojas que não vendem outros produtos;
d) são vendidos por somente uma loja que vende outros produtos;
e) não são vendidos por somente uma loja.

```
select p.descricaoP
from produto p
where not exists
        (select *
         from loja l
         where not exists
                 (select *
                  from oferta o
                  where o.idP=p.idP and o.idL=l.idL))
```

PRODUTO (idP, descricaoP)
LOJA (idL, nomeL, endereçoL)
OFERTA (idP, idL, preçoO)

produz a lista de produtos que:

a) nenhuma das lojas vende;

b) todas as lojas vendem;

c) são vendidos apenas em lojas que não vendem outros produtos;

d) são vendidos por somente uma loja que vende outros produtos;

e) não são vendidos por somente uma loja.

Q9 - FGV - TCMSP - 2015

Considere as tabelas relacionais, e respectivas instâncias, mostradas a seguir. O comando SQL

```
select A from X1 where
  not exists
  (select * from X3 where
      not exists
      (select * from X2 where
         X1.A=X2.C and X3.B=X2.D))
```

X1

A
1
2
3
4
5

X2

C	D
1	1
1	3
1	5
2	1
2	3
3	1
3	1

X3

B
1
3
5

produz um resultado com apenas uma coluna, cujo(s) valor(es) é/são:

(A) 1
(B) 1,2,3
(C) 4,5
(D) 1,2,3,4,5
(E) NULL

Q9 - FGV - TCMSP - 2015

Considere as tabelas relacionais, e respectivas instâncias, mostradas a seguir. O comando SQL

```
select A from X1 where
  not exists
  (select * from X3 where
      not exists
      (select * from X2 where
         X1.A=X2.C and X3.B=X2.D))
```

X1

A
1
2
3
4
5

X2

C	D
1	1
1	3
1	5
2	1
2	3
3	1
3	1

X3

B
1
3
5

produz um resultado com apenas uma coluna, cujo(s) valor(es) é/são:

(A) 1
(B) 1,2,3
(C) 4,5
(D) 1,2,3,4,5
(E) NULL

CESGRANRIO – BNDES – Desenvolvimento – 2011

Q10 – As tabelas *Projeto, Funcionario e Participacao_Projeto* participam da base de dados de um sistema de controle de projetos. A estrutura dessas tabelas é a seguinte:

Projeto (IDProjeto, Nome, DataInicio, DataFim)
Funcionario (Matricula, Nome, DataNascimento)
Participacao_Projeto (**IDProjeto, Matricula**)

Colunas sublinhadas participam da chave primária
Colunas em negrito participam de chaves estrangeiras

O comando SQL que retorna o nome somente dos funcionários que participaram de **TODOS** os projetos é:

```
(A) SELECT DISTINCT F.NOME FROM FUNCIONARIO F
        INNER JOIN PARTICIPACAO_PROJETO PP
        ON F.MATRICULA = PP.MATRICULA
            INNER JOIN PROJETO P
            ON P.IDPROJETO = PP.IDPROJETO
```

Projeto (IDProjeto, Nome, DataInicio, DataFim)
Funcionario (Matricula, Nome, DataNascimento)
Participacao_Projeto (**IDProjeto, Matricula**)

Colunas sublinhadas participam da chave primária
Colunas em negrito participam de chaves estrangeiras

O comando SQL que retorna o nome somente dos funcionários que participaram de **TODOS** os projetos é:

```
(B) SELECT DISTINCT F.NOME FROM FUNCIONARIO F
        INNER JOIN PARTICIPACAO_PROJETO PP
            ON F.MATRICULA = PP.MATRICULA
        WHERE EXISTS (SELECT 1 FROM PROJETO P
            WHERE P.IDPROJETO = PP.IDPROJETO)

(C) SELECT F.NOME FROM FUNCIONARIO F
        WHERE MATRICULA IN
        (SELECT MATRICULA FROM PARTICIPACAO_PROJETO)
```

Projeto (IDProjeto, Nome, DataInicio, DataFim)
Funcionario (Matricula, Nome, DataNascimento)
Participacao_Projeto (**IDProjeto, Matricula**)

Colunas sublinhadas participam da chave primária
Colunas em negrito participam de chaves estrangeiras

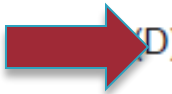
O comando SQL que retorna o nome somente dos funcionários que participaram de **TODOS** os projetos é:

```
(D) SELECT F.NOME FROM FUNCIONARIO F
        WHERE NOT EXISTS
            (SELECT 1 FROM PROJETO P
            WHERE NOT EXISTS (SELECT 1 FROM
                    PARTICIPACAO_PROJETO PP
                    WHERE PP.IDPROJETO = P.IDPROJETO
                    AND PP.MATRICULA = F.MATRICULA))

(E) SELECT F.NOME FROM FUNCIONARIO F
        WHERE NOT EXISTS
            (SELECT 1 FROM PARTICIPACAO_PROJETO  PP
            WHERE PP.MATRICULA = F.MATRICULA
            AND NOT EXISTS (SELECT 1 FROM PROJETO P
                    WHERE PP.IDPROJETO = P.IDPROJETO))
```

Projeto (IDProjeto, Nome, DataInicio, DataFim)
Funcionario (Matricula, Nome, DataNascimento)
Participacao_Projeto (**IDProjeto, Matricula**)

Colunas sublinhadas participam da chave primária
Colunas em negrito participam de chaves estrangeiras

O comando SQL que retorna o nome somente dos funcionários que participaram de **TODOS** os projetos é:



```
(D) SELECT F.NOME FROM FUNCIONARIO F
        WHERE NOT EXISTS
            (SELECT 1 FROM PROJETO P
            WHERE NOT EXISTS (SELECT 1 FROM
                    PARTICIPACAO_PROJETO PP
                    WHERE PP.IDPROJETO = P.IDPROJETO
                    AND PP.MATRICULA = F.MATRICULA))

(E) SELECT F.NOME FROM FUNCIONARIO F
        WHERE NOT EXISTS
            (SELECT 1 FROM PARTICIPACAO_PROJETO  PP
            WHERE PP.MATRICULA = F.MATRICULA
            AND NOT EXISTS (SELECT 1 FROM PROJETO P
                    WHERE PP.IDPROJETO = P.IDPROJETO))
```

CESGRANRIO – BNDES – Desenvolvimento – 2012

Considere as tabelas a seguir para responder às questões de nos 38 e 39. Essas tabelas pertencem ao esquema de um banco de dados de uma locadora de veículos.

```
CREATE TABLE VEICULO (
   PLACA      CHAR(7)     NOT NULL,
   MODELO     VARCHAR2(50) NOT NULL,
   COD_CAT    CHAR(2)     NOT NULL,
   CONSTRAINT  VEICULO_PK PRIMARY KEY (PLACA),
   CONSTRAINT  VEICULO_FK FOREIGN KEY (COD_CAT) REFERENCES
     CATEGORIA (COD_CAT))

CREATE TABLE CATEGORIA (
   COD_CAT    CHAR(2)     NOT NULL,
   DESCR      VARCHAR2(80) NOT NULL,
   VAL_DIARIA  NUMBER(7,2),
   CONSTRAINT  CATEGORIA_PK PRIMARY KEY (COD_CAT))

CREATE TABLE ALUGUEL (
   PLACA      CHAR(7)  NOT NULL,
   DATA_DEV   NUMBER(6),
   DATA_ALG   NUMBER(6)  NOT NULL,
   CONSTRAINT  ALUGUEL_PK PRIMARY KEY (PLACA, DATA_ALG),
   CONSTRAINT  ALUGUEL_FK FOREIGN KEY (PLACA) REFERENCES
     VEICULO (PLACA))
```

CESGRANRIO – BNDES – Desenvolvimento – 2012

Observações:

- A tabela VEICULO contém as informações sobre os veículos que a locadora dispõe para aluguel. Ela possui uma coluna chamada COD_CAT, que contém a categoria à qual um veículo pertence.
- A tabela CATEGORIA representa a tabela de preços da locadora. Ela registra o valor que será cobrado por um dia de aluguel de um veículo de uma determinada categoria.
- A tabela ALUGUEL é usada para registrar todas as operações de aluguel.
- A coluna DATA_ALG guarda a data na qual um veículo foi alugado, enquanto a coluna DATA_DEV guarda a data na qual o veículo foi devolvido. Ela é informada ao sistema quando o cliente devolve o veículo à locadora. Ambas as datas estão no formato AAMMDD. Dessa forma, a data 05/02/2011 será armazenada como 110205.

Q11 – Qual comando SQL será executado com sucesso, independente do estado das tabelas que compõem a base de dados da locadora de veículos?

```
CREATE TABLE VEICULO (
   PLACA      CHAR(7)     NOT NULL,
   MODELO     VARCHAR2(50) NOT NULL,
   COD_CAT    CHAR(2)     NOT NULL,
   CONSTRAINT  VEICULO_PK PRIMARY KEY (PLACA),
   CONSTRAINT  VEICULO_FK FOREIGN KEY (COD_CAT) REFERENCES
     CATEGORIA (COD_CAT))

CREATE TABLE CATEGORIA (
   COD_CAT    CHAR(2)     NOT NULL,
   DESCR      VARCHAR2(80) NOT NULL,
   VAL_DIARIA  NUMBER(7,2),
   CONSTRAINT  CATEGORIA_PK PRIMARY KEY (COD_CAT))

CREATE TABLE ALUGUEL (
   PLACA      CHAR(7)  NOT NULL,
   DATA_DEV   NUMBER(6),
   DATA_ALG   NUMBER(6)  NOT NULL,
   CONSTRAINT  ALUGUEL_PK PRIMARY KEY (PLACA, DATA_ALG),
   CONSTRAINT  ALUGUEL_FK FOREIGN KEY (PLACA) REFERENCES
     VEICULO (PLACA))
```

(A)
```
INSERT INTO CATEGORIA (DESCR,VAL_DIARIA,COD_CAT)
   VALUES ('sedan compacto',90.00,'uc')
```

(B)
```
DELETE FROM CATEGORIA X WHERE
   NOT EXISTS (SELECT COUNT(*)
   FROM VEICULO V,ALUGUEL A
   WHERE V.COD_CAT=X.COD_CAT AND V.PLACA=A.PLACA AND
   A.DATA_DEV IS NOT NULL
   GROUP BY V.COD_CAT)
```

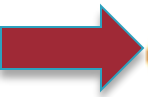
```
CREATE TABLE VEICULO (
   PLACA      CHAR(7)     NOT NULL,
   MODELO     VARCHAR2(50) NOT NULL,
   COD_CAT    CHAR(2)     NOT NULL,
   CONSTRAINT  VEICULO_PK PRIMARY KEY (PLACA),
   CONSTRAINT  VEICULO_FK FOREIGN KEY (COD_CAT) REFERENCES
     CATEGORIA (COD_CAT))

CREATE TABLE CATEGORIA (
   COD_CAT    CHAR(2)     NOT NULL,
   DESCR      VARCHAR2(80) NOT NULL,
   VAL_DIARIA  NUMBER(7,2),
   CONSTRAINT  CATEGORIA_PK PRIMARY KEY (COD_CAT))

CREATE TABLE ALUGUEL (
   PLACA      CHAR(7)  NOT NULL,
   DATA_DEV   NUMBER(6),
   DATA_ALG   NUMBER(6)  NOT NULL,
   CONSTRAINT  ALUGUEL_PK PRIMARY KEY (PLACA, DATA_ALG),
   CONSTRAINT  ALUGUEL_FK FOREIGN KEY (PLACA) REFERENCES
     VEICULO (PLACA))
```

(C)
```
INSERT INTO ALUGUEL VALUES ('LJJ2222',120618)
```

(D)
```
DELETE FROM VEICULO X WHERE
    NOT EXISTS (SELECT COUNT(*)
    FROM ALUGUEL A
    WHERE X.PLACA=A.PLACA AND
    A.DATA_DEV IS NOT NULL
    GROUP BY A.PLACA)
```

```
CREATE TABLE VEICULO (
   PLACA      CHAR(7)     NOT NULL,
   MODELO     VARCHAR2(50) NOT NULL,
   COD_CAT    CHAR(2)     NOT NULL,
   CONSTRAINT  VEICULO_PK PRIMARY KEY (PLACA),
   CONSTRAINT  VEICULO_FK FOREIGN KEY (COD_CAT) REFERENCES
     CATEGORIA (COD_CAT))

CREATE TABLE CATEGORIA (
   COD_CAT    CHAR(2)     NOT NULL,
   DESCR      VARCHAR2(80) NOT NULL,
   VAL_DIARIA  NUMBER(7,2),
   CONSTRAINT  CATEGORIA_PK PRIMARY KEY (COD_CAT))

CREATE TABLE ALUGUEL (
   PLACA      CHAR(7)  NOT NULL,
   DATA_DEV   NUMBER(6),
   DATA_ALG   NUMBER(6)  NOT NULL,
   CONSTRAINT  ALUGUEL_PK PRIMARY KEY (PLACA, DATA_ALG),
   CONSTRAINT  ALUGUEL_FK FOREIGN KEY (PLACA) REFERENCES
     VEICULO (PLACA))


(E) UPDATE VEICULO X SET COD_CAT='xs' WHERE
    EXISTS (SELECT COUNT(*)
    FROM VEICULO V,ALUGUEL A
    WHERE V.COD_CAT='xs' AND V.PLACA=A.PLACA AND
    A.DATA_DEV IS NOT NULL
    GROUP BY V.COD_CAT)
```

```
CREATE TABLE VEICULO (
   PLACA      CHAR(7)     NOT NULL,
   MODELO     VARCHAR2(50) NOT NULL,
   COD_CAT    CHAR(2)     NOT NULL,
   CONSTRAINT  VEICULO_PK PRIMARY KEY (PLACA),
   CONSTRAINT  VEICULO_FK FOREIGN KEY (COD_CAT) REFERENCES
     CATEGORIA (COD_CAT))

CREATE TABLE CATEGORIA (
   COD_CAT    CHAR(2)     NOT NULL,
   DESCR      VARCHAR2(80) NOT NULL,
   VAL_DIARIA  NUMBER(7,2),
   CONSTRAINT  CATEGORIA_PK PRIMARY KEY (COD_CAT))

CREATE TABLE ALUGUEL (
   PLACA      CHAR(7)  NOT NULL,
   DATA_DEV   NUMBER(6),
   DATA_ALG   NUMBER(6)  NOT NULL,
   CONSTRAINT  ALUGUEL_PK PRIMARY KEY (PLACA, DATA_ALG),
   CONSTRAINT  ALUGUEL_FK FOREIGN KEY (PLACA) REFERENCES
     VEICULO (PLACA))
```



```
(E) UPDATE VEICULO X SET COD_CAT='xs' WHERE
     EXISTS (SELECT COUNT(*)
     FROM VEICULO V,ALUGUEL A
     WHERE V.COD_CAT='xs' AND V.PLACA=A.PLACA AND
     A.DATA_DEV IS NOT NULL
     GROUP BY V.COD_CAT)
```

gabaritos

Gabarito

Q1 – B
Q2 – D
Q3 – B
Q4 – A
Q5 – D
Q6 – A
Q7 – B
Q8 – B
Q9 – A
Q10 – D
Q11 – E