Testando o SQL

Apresentação, instalação e outras dicas.

Este documento é um exercício sobre o SQL:1992. Você PODE testá-lo em qualquer SGBD relacional. Aqui fizemos nossos testes em um SGBD PostgreSQL, versão 9.2 (ou mais recente).
Este documento apresenta um exercício **VÁLIDO PARA NOTA**, este trabalho deve ser feito em partes ou etapas. Este trabalho é INDIVIDUAL, embora tenha uma etapa que pode ser feito de forma colaborativa para acelerar o ritmo do desenvolvimento.
Por favor, preencha o nome e matrícula do aluno no espaço abaixo disponível.

Nome e matrícula do aluno:

Matrícula	Nome
0210481722004	Matheus de Brito Vieira Martins

DEPOIS de resolvido o TESTESQL TROQUE o nome do arquivo para:**SQL(PG)-NomeDoAluno.pdf** e envie-o para a conta de e-mail divulgada no site da disciplina.

No assunto escreva: **LBDN-SQL(PG)-NomeDoAluno**, para ADS Noturno; ou **LBDT-SQL(PG)-NomeDoAluno**, para ADS Tarde.

(Escreva SEU como no exemplo: *José da Silva* → ***JosedaSilva***
Ou de outra forma:
– USE a primeira letra de cada parte do nome com maiúsculas e o resto com minúsculas e TODO o nome SEM espaços em branco.
– NÃO use apelido ou partes abreviadas do nome, ESCREVA como está na sua matrícula.

Ambiente de Trabalho.

O ambiente de software para executar este trabalho deverá ter:

- Servidor de Página: Usa-se o Apache (versão 2.0 ou mais recente), ainda é possível usar o TomCat ou o IIS.
- Interpretador de comandos da linguagem PHP
- Servidor de Banco de Dados: Usa-se o SGBD PostgreSQL (versão 9.2 ou mais recente).
- pgAdmin 4 (versão 3.6 ou mais recente).

Neste projeto usa-se o pacote XAMPP versão 1.8.2 (já existe versão mais recente, mas a 1.8.2 tem o servidor de páginas e é o que se precisa neste projeto). Você pode baixar o pacote neste link: https://sourceforge.net/projects/xampp/files/XAMPP%20Windows/1.8.2/ . Lá existem versões para vários Sistemas Operacionais e formatos.

Neste projeto não se usa o MySQL (ou o MariaDB). Instale o XAMP somente com o servidor de página, a linguagem PHP e o banco dados (MariaDB ou MySQL), mas neste projeto não usaremos o Banco de Dados do pacote XAMP. Usaremos o Servidor de Banco de Dados ***PostgreSQL***.

Depois de baixado o pacote XAMPP instale-o seguindo as instruções de instalação padrão. O procedimento coloca o XAMPP no diretório **C:\XAMPP**.

Eu aconselho instalar o XAMPP e **depois** o PostgreSQL. Seu computador fica com dois SGBDs. O PostgreSQL se instala como serviço do Windows mas o MySQL só é configurado para isso se você quiser. Use o painel de controle do XAMP e deixe configurado para o APACHE ser iniciado juntamente com o Windows. Observe os requisitos de hardware para rodar estes programas no seu computador.

Para o PHP conseguir acessar o PostgreSQL edite o arquivo PHP.INI que fica no diretório C:\XAMPP\PHP (use um editor de texto simples tipo notetab ou notepad). Neste arquivo procure a linha: **“;extension=phppgsql.dll”**, tire o ponto-e-vírgula do início da linha, ela fica escrita assim: **extension=phppgsql.dll**.

Grave a alteração. Reinicie seu computador.

Pronto para este trabalho é o que precisamos: do Apache, PHP e o PostgreSQL rodando.

Bem, agora começa a carga de trabalho.

O que devemos fazer?

Este trabalho é dividido em 6 (SEIS) partes:

1. Acerto do Dicionário de Dados da base (acessando o SGBD PostgreSQL usando o aplicativo PgAdmin 4, que vem no instalador do PostgreSQL, ou pode ser atualizado separadamente).
   - Você deverá desenvolver uma base com 101 tabelas e duas visões!
     Para acelerar o trabalho será disponibilizado um script em SQL que cria umas... 90 tabelas (ou um pouco mais) ...
     O Acerto do dicionário de dados da base **DEVE SER FEITO considerando CERTO** o dicionário que está disponível no acesso: http://localhost/FATEC/online/LBD/ecasoproj/DD-SQLPG-20192.php .
     De outro modo, a Base de Dados que vocês receberam está com **DEFEITOS**, ora falta campo em tabela ora a definição de um campo está errada ou então falta uma tabela inteira ou tem tabela que não precisa estar na base de dados.
     Novamente afirmo: o que está CORRETO é o que está escrito no acesso: http://localhost/FATEC/online/LBD/ecasoproj/DD-SQLPG-20192.php . Portanto vocês devem acertar a base de dados
2. Tradução de Álgebra Relacional → SQL (opcionalmente com comandos SQL melhorados).
   - Na segunda etapa você deve executar e INTERPRETAR o resultado dos comandos SQL. Os comandos foram escritos para uma base com o dicionário de dados CERTO. Muitos destes comandos devem ser executados em sequência, pois o grau de dificuldade é crescente e, portanto, a escrita fica mais complexa, mas pode ser reaproveitada de uma questão para outra. PORTANTO NÃO PULE COMANDOS. Execute-os na ordem proposta.
     Na etapa: Tradução AR→SQL, faça primeiro a tradução literal do comando proposto na Álgebra Relacional para o SQL correspondente. DEPOIS tente aperfeiçoar o comando, MAS DEIXE O QUE FOI ESCRITO CERTO NO LUGAR. O que será avaliado é tradução AR→SQL.
3. Resolução de Perguntas com Álgebra Relacional e SQL (ambos os itens são obrigatórios, cada comando de AR com sua tradução - opcionalmente com comandos SQL melhorados).
   - Nesta etapa vocês recebem somente a pergunta e DEVEM desenvolver a AR e a seguir o SQL. Note: Se a sua AR estiver errada, seu SQL não estará certo. Na etapa anterior a AR já está montada. Agora você terá que montar a sua AR e a seguir o SQL. Seu SQL deve ser tradução da SUA AR. Não é preciso otimizar seu SQL.
4. Desenvolvimento de comandos SQL (O comando SQL é o obrigatório e a AR é sugerida que você faça – para orientar sua resposta.).
   - Neste item é feita a execução de comandos. É sugerido que você escreva primeiro a AR e a seguir seu SQL. DEPOIS que seu SQL estiver OK você pode otimizá-lo (se QUISER e se tiver tempo!). Concentre-se em fazer o que é pedido na etapa se sobrar tempo otimize seus comandos, mas lembre-se: será corrigido seu comando que resolve a questão. Por isso somente tente otimizar seu SQL SE SOBRAR TEMPO. Outro alerta: MANTENHA SEU SQL ANTES DA TENTATIVA DE OTIMIZAR. É esta sequência de comandos que será corrigida.

Estas partes devem ser feitas na ordem que estão apresentadas, ***acredite***, é mais produtivo e efetivo desenvolver este trabalho cumprindo cada parte inteiramente antes de partir para seguinte. Use o item de inclusão de CAIXA DE TEXTO dos editores de PDF. NÃO use o item de incluir caixas de comentários.

Nota IMPORTANTE: O desenvolvimento de Gatilhos e Relatórios NÃO FAZ MAIS PARTE DESTE TRABALHO. O desenvolvimento destes itens foram desmembrados em um novo projeto apresentado no site da disciplina.

Este é um trabalho extenso. Faça-o com cuidado e atenção.

Vamos ao trabalho!

Acerto do Dicionário de Dados (no SGBD PostgreSQL)

Modelo de dados

Todos os modelos de dados foram desenvolvimento com a notação de Peter Chen. Todos os modelos estão disponibilizados no site da disciplina.

Dicionário de dados

O dicionário de dados desta base é apresentado no item na tabela 1. Para agilizar a implementação da base de dados, no site da disciplina existe um acesso para o download de um script que gera muitas das tabelas desta base de dados (**lbdpgSQL-20192.SQL**). Estas tabelas (geradas pelo script) NÃO estão como especificadas no dicionário de dados. Por conta disso o trabalho sobre esta base deve ser realizado em etapas:

- Crie uma base de dados com nome **LBDPG**. Faça o download do Script **lbdpgSQL-20192.SQL** e execute o script (criando as tabelas iniciais do exercício)
- Faça os ajustes de estrutura em todas as tabelas desta base. **Note**: os campos devem estar definidos como neste dicionário de dados. Entretanto, não necessariamente na mesma ordem como os campos aparecem nestas tabelas. Ou seja, os campos não precisam estar na mesma ordem como está no dicionário.
- Depois de executado o **INSERTS.SQL**, CRIE TODAS AS CHAVES PRIMÁRIAS, SE quiser, nesta etapa crie também as Chaves Candidatas (combinação de campo ou campo que fora da CP podem ser candidatas à CP). Para as CCs crie uma *constraint* UNIQUE. Não se preocupe com as Chaves Estrangeiras.
- Depois de criadas todas as CPs, CRIE TODAS AS CHAVES ESTRANGEIRAS. Analise para cada par CP←→CE qual deve ser o comportamento para operações de INSERT, UPDATE e DELETE.
- As tabelas da base de dados já estãp populadas (contém tuplas). Durante o processo de ajuste os dados devem ser mantidos nas tabelas. Entretanto, SE acontecer de alguns dados se perderem, então, DEPOIS do ajuste das tabelas, edite o script e execute somente os comandos de insert. EXISTEM algumas operações que devem ser feitas antes da execução destes comandos, procure o professor de for necessário executar esta etapa.

Neste dicionário de dados tentou-se seguir a seguinte notação: **Negrito** é Chave Primária, *Itálico* é Chave Estrangeira, Chave Candidata não tem notação, somente são explicadas no campo **comentário**. Se ao executar a analise do Dicionário você perceber qualquer imperfeição na definição de campos das tabelas avise o professor. A alteração será analisada e atualizada no Dicionário (sendo destacada em cor **AZUL**) e uma nova data passará a constar na estrutura de menus que dá acesso ao TesteSQL.

Todo o Dicionário de Dados está disponibilizado no site da disciplina. Deste modo eventuais atualizações serão percebidas mais rapidamente.

Aqui TERMINA a parte do trabalho que pode ser feita em dupla.
Faça uma cópia da base (formato PLAIN com DATA, PREDATA e POSDATA, INSERT COMMAND) e passe para o ser par.

A partir deste ponto o trabalho DEVE ser individual. Terminadas todas estas tarefas iniciam-se a Prática dos Comandos.

Prática de Comandos SQL

Usando o pgAdmin 4, acesse a base de dados do exercício. Acesse a lista de tabelas e abre uma janela “QUERY TOOL” para execução dos comandos. A cada comando você deve usar esta janela para testar os comandos que pesquisar para tentar resolver a proposta da pergunta.

Como vocês estão trabalhando com a versão PDF aconselho que os comentários sejam escritos usando o programa FOXIT (http://cdn01.foxitsoftware.com/pub/foxit/reader/desktop/win/4.x/4.3/enu/FoxitReader431enuSetup.exe). Este programa permite colocar janela de texto sobre os PDF. Se você conhece outro que faça isso, tudo bem. Use-o, mas saiba que seu trabalho será corrigido com o programa FOXIT, ou seja:

- TESTE seu SQL de resposta antes de me enviar.

A Prática de comandos tem três partes:

- Tradução de AR para SQL – Você recebe uma pergunta, a AR que resolve a pergunta e deve montar a lista de comandos que é a tradução da AR para SQL. Nesta parte pondem acontecer que o SQL imponha alterações na AR pelo fato do SGBD ser ou não limitado no tratamento da tradução do SQL. POR ISSO, recomenda-se fazer a alteração necessária e depois escrever a sequencia de SQL que implementa a AR.
- Desenvolvimento de AR e SQL. – Aqui o aluno recebe somente a pergunta e deve escrever a AR e depois traduzi-la para SQL.
- Perguntas e Comandos de Resposta. – Aqui a proposta é analisar uma pergunta e escrever UM COMANDO em SQL que consiga resolver a pergunta.

Vocês devem testar estes comandos tanto em PostgreSQL. ACONSELHO que vocês façam o trabalho abrindo uma aba com o pgAdmin4, o arquivo PDF onde vai escrever as resposta e UM editor de programas. Analise a pergunta, escreva a resposta no editor de programas (teste sua resposta no pgAdmin 4), depois copie/cole a resposta no arquivo PDF. MANTENHA o PDF sempre salvo a cada resposta escrita.

Bem, vamos aos comandos em SQL.

Tradução Álgebra Relacional → SQL

Nota-se que: Pode haver alguma diferença no SQL dos dois

Agora que sabemos o que os comandos do SQL podem processar, vamos traduzir a Álgebra Relacional para comandos da SQL. As perguntas estão feitas e a Álgebra Relacional que apresenta a resposta está formulada. Você deve traduzir as expressões de AR para SQL.

Sugestão: traduza cada expressão da AR para um comando SQL. Teste sua resposta no banco de dados (comando por comando, e a seguir todos de uma vez). Sabemos que o SQL pode combinar em um comando mais de uma expressão da AR. **Se você quiser otimizar** o código SQL pode fazê-lo, ***MAS*** deixe na resposta a sua tradução expressão por expressão.

Para este contexto fizemos a especificação da Álgebra Relacional para resolver a algumas perguntas. Vamos apresentar as perguntas, as respostas em AR e a seguir você pratica a tradução para SQL.

```
1.   Quais ônibus fizeram viagens em todas as rotas viárias que tiveram viagens no ano de 2012?
-- Preparando as tabelas para a divisao
SEL : a  ← viagens [[dtsaidaviagem entre "2012-01-01" e "2012-12-31" ou dtchegadaviagem entre "2012-01-01" e "2012-12-31"]]
JUN : b  ← a [[ a.pkviagem=passagens.fkviagem ]] passagens
PRO : p  ← b [[fkrota]] – Códigos das rotas que tiveram viagens em 2010.
PRO : pd ← b [[fkrota, fkonibus]]
PRO : f ← onibus [[pkonibus→fkonibus ]]
-- Dividindo
PCT : s ← p X f
SUB : t ← s - pd
PRO : w ← t [[ onibusid ]]
SUB : r ← f - w
-- Completando a solução
JUN : z <- r [[ r.onibusid=onibus.id ]] onibus
PRO : z [[ukplaca,txapelido, nuanofabricacao, qtcapacidade]]
```

em sql:

```
-- Preparando as tabelas p/ divisão
create table a as ( select * from Viagens where dtsaidaviagem between '2012-01-01' and '2012-12-31');
create table b as ( select * from a, passagens where a.pkviagem=passagens.fkviagem );
create table p as (select fkrota from b);
create table pd as (select fkrota, fkonibus from b);
create table f as (select pkonibus as fkonibus from onibus);
-- Dividindo
create table s as ( select * from p, f);
create table t as (select * from s minus (select * from pd) );
create table w as (select onibusid from t);
create table r as ( select * from f minus (select * from w) );
-- Completando a solução
create table z as ( select * from r, onibus where r.onibusid=onibus.id);
select ukplaca,txapelido, nuanofabricacao, qtcapacidade from z;
drop table a, b, p, pd, f, s, t, w, r, z;
```

2. Quais os nomes dos professores e os nomes das respectivas disciplinas atribuídas aos professores que ministram disciplinas da Área de Estudo "EXATAS"?

```
SEL : a  ← areadeestudo[[txnomearea="EXATAS"]]
JUN : b  ← a[[a. pkareaestudo = k.fkareaestudo]] areaestudodisciplinas →K
JUN : c  ← b[[b.fkdisciplina= disciplinas.pkdisciplina]]disciplinas
PRO : c1 ← c[[pkdisciplina, txnome→txnomedisciplina]]
JUN : d  ← c1[[C.pkdisciplina = atribuicoes.fkdisciplina]]atribuicoes
PRO : d1 ← d[[fkprofessor, txnomedisciplina]]
JUN : e  ← d1[[D.fkprofessor=professores.fkprofessor]]professores
PRO : r  ← e[[professores.txnomeprofessor,disciplinas.txnome]]
```

Em SQL:

```
create table a as (select * from areaestudos where txnomearea = 'EXATAS');
create table b as (select * from a, areaestudodisciplinas as k where a.pkareaestudo = k.fkareaestudos );
create table c as (select * from b, disciplinas where b.fkdisciplinas = disciplinas.pkdisciplina);
create table c1 as (select pkdisciplina, txnome as txnomedisciplina from c);
create table d as (select * from c1, atribuicoes where c1.pkdisciplina=atribuicoes.fkdisciplina );
create table d1 as (select fkprofessor, txnomedisciplina from d);
create table e as (select * from d, professores where d.fkprofessor=professores.pkprofessor);
create table r as (select txnomeprofessor, txnomedisciplina from e);
select * from r;
drop table a, b, c, c1, d, d1, e, r;
```

3. Qual é a quantidade de disciplina atribuídas a cada professor?

```
PRO : A ← atribuicoes[[CONTA(fkprofessor)→qtddisc,fkprofessor]]
                        {agrupado por fkprofessor}
                        {ordenado por fkprofessor}
JUN : B ← A[[A.fkprofessor = Pr.pkprofessor]]Professores→Pr
PRO : B[[qtddisc,id,txnomeprofessor]]
```

Em SQL:

```
create table A as(select count(fkprofessor) as qtddisc, fkprofessor from atribuicoes group by fkprofessor order by fkprofessor);
create table B as(select * from A, Professores as PR where A.fkprofessor=Pr.pkprofessor);
select qtddisc, id, txnomeprofessor from B;
```

4. Quais são os nomes dos professores que ministram **todas** disciplinas (note que uma parte da resposta especifica o procedimento da divisão)?

Preparando o procedimento da divisão

Projeção: P ← professores[[pkprofessor→fkprofessor]]

Projeção: D ← disciplinas[[pkdisciplina→fkdisciplina]]

Projeção: PD← atribuições[[fkprofessor, fkdisciplina]]

Estas três tabelas ficam prontas para o procedimento da divisão

Procedimento da Divisão:

A ← P X D - Esta tab. Fica UC com PD (atribuições)

B ← A – PD – A é maior que Atribuições

C ← B[[fkprofessor]] - Aqui temos os prof. Que não se ligam a pelo menos uma disciplina

E ← P – C - Aqui tenho os códigos de professores que ministram TODAS as disc.

Conclusão da Pergunta

Junção..: F ← E[[E.fkprofessor=Professores.pkprofessor]]Professores

Projeção: F[[pkprofessor, txnomeprofessor]]

Em SQL:

```
--Preparando o procedimento de divisão
create table P as (select pkprofessor as fkprofessor from professores );
create table D as(select pkdisciplina as fkdisciplina from disciplinas);
create table PD as(select fkprofessor, fk disciplina from atribuições);
--Procedimento da Divisão
create table A as( select * from P, D);
create table B as(select * from A minus(select * from PD));
create table C as(select fkprofessor from B);
create table E as(select * from P minus(select * from C));
--Conclusão da Pergunta
create table F as(select * from E, Professores where E.fkprofessor=Professores.pkprofessor);
select pkprofessor, txnomeprofessor from F;
```

5. Qual é a média da quantidade de disciplinas atribuídas aos professores para cada área de estudo? O que pode representar esta média (interprete o resultado), exiba a média com duas casas decimais.

```
W ← atribuicoes [[fkprofessor→idprof, fkdisciplina→ iddisc]]
S ← W [[W.iddisc = disciplinas.pkdisciplina][disciplinas
T ← S [[idprof,pkdisciplina→iddisc]]
K ← T [[T.iddisc= aed.fkdisciplina]] areaestudodisciplinas → aed
M ← K [[media(idprof)→QtdMedProf, fkareaestudo]] {agrupar por fkareaestudo} {ordernar por fkareaestudo}
L ← M[[M.fkareaestudo=areaestudo.pkareaestudo]]areaestudo
L[[txnomearea,ARREDONDAR(QtdMedProf,2)]]
```

Em SQL:

```
create table W as(select fkprofessor as idprof, fkdisciplina as iddisc);
create table S as(select * from W right join disciplinas on w.iddisc = disciplinas.pkdisciplina);
create table T as(select idprof, pkdisciplina as iddisc from S);
create table K as (select * from T, areaestudodisciplinas as aed where T.iddisc = aed.fkdisciplina);
create table M as(select AVG(idprof) as QtdMedProf, fkareaestudo from K group by fkareaestudo order by fkareaestudo);
create table L as(select * from M, pkareaestudo where M.fkareaestudo = areaestudo.pkareaestudo);
select txnomearea, round(QtdMedProf, 2);
```

6. Quantas disciplinas são atribuídas aos professores que ministram disciplinas da Área de Estudo "Humanas"?

```
A ← areaestudo[[txnomearea="Humanas"]]
B ← A[[ A.pkareaestudo = K.fkareaestudo ]]areaestudodisciplinas → K
C ← B[[ B.fkdisciplina = disciplinas.pkdisciplina ]]disciplinas
D ← C[[pkdisciplina]]
E ← D[[ D.pkdisciplina = atribuicoes.fkdisciplina ]]atribuicoes
     E[[fkprofessor , conta(fkdisciplina) ]] {agrupa por fkprofessor} {ordem por fkprofessor}
```

Em SQL:

```
create table A as(select * from areaestudo where txnomearea="Humanas");
create table B as(select * from A, areaestudodisciplinas as K where A.pkareaestudo = K.fkareaestudo);
create table C as(select * from B, disciplinas where B.fkdisciplina = disciplinas.pkdisciplina);
create table D as(select pkdisciplina from C);
create table E as(select * from D, atribuicoes where D.pkdisciplina = atribuicoes.fkdisciplina);
select fkprofessor, count(fkdisciplina) from E group by fkprofessor order by fkprofessor
```

7. Quais são, em ordem decrescente, as disciplinas com mais professores atribuídas?

```
A ← atribuicoes [[ disciplinasid, conta(professoresid) → qtdprof ]]
                {agrupa por disciplinasid tendo conta(professoresid)>3}
                {ordem por conta(professoresid) decrescente}
B ← A [[ A.disciplinasID = disciplinas.id ]] disciplinas
B [[id, txnome, qtdprof,  txementa, qthoras, txcriterioavaliacao, dtcaddisciplina]]
```

Em SQL:

```
create table A as(select fkdisciplina, count(fkprofessor) as qtdprof from atribuicoes group by fkdisciplina having count(fkdisciplina)>3 order  by count(fkprofessor) desc);
create table B as(select * from A, disciplinas where A.fkdisciplina = disciplinas.pkdisciplina);
select pkdisciplina, txnome, qtdprof, txementa,qthoras, txcriterioavaliacao, dtcaddisciplina from B;
drop table A, B;
```

8. Quais são todas as rotas viárias que passam pelas cidades "São Paulo" e "Rio de Janeiro"?

```
A <- cidades [[ txnomecidade="São Paulo" ou txnome="Rio de Janeiro" ]]
B1 <- A [[ A.pkcidade = rotasviarias.fkcidadeorigem]] rotasviarias
B2 <- A [[ A.pkcidade = rotasviarias.fkcidadedestino]] rotasviarias
C <- B1 U B2
C [[ pkrota, txnomerota, txdescrperiodo, fkcidadeorigem, fkcidadedestino ]]
```

Em SQL:

```
create table A as(select * from cidades where txnomecidade='São Paulo' or txnomecidade='Rio de Janeiro');
create table B1 as(select  * from A, rotasviarias where A.pkcidade = rotasviarias.fkcidadeorigem);
create table B2 as(select  * from A, rotasviarias where A.pkcidade = rotasviarias.fkcidadedestino);
create table C as(select * from B1 union(select * from B2));
drop table A, B1, B2, C;
```

9. Quais são os nomes de funcionários que passaram por consultas de todas as especialidades médicas nos últimos 180 dias

```
a ← consultas [[ dthoraconsulta >= hoje-180 ]]
b ← a [[ a.fkmedico = medicos.pkmedico ]] medicos
PD← b [[ fkfuncionario, fkespecialidade ]]
P ← funcionarios [[ pkfuncionario→fkfuncionario ]]
D ← especmedicas [[ pkespecialidade→fkespecialidade ]]

S ← P X D
T ← S - PD
W ← T [[ fkfuncionario ]]
R ← P – W

V ← R[[R.fkfuncionario=f.pkfuncionario ]] funcionários→f
V[[txprenomes,txsobrenome]]
```

Em SQL:

```
create table a as(select * from consultas where dthoraconsulta >= NOW()- interval '180');
create table b as(select * from a, medicos where a.fkmedico = medicos.pkmedico);
create table PD as(select fkfuncionario, fkespecialidade from b);
create table P as(select pkfuncionario as fkfuncionario from funcionarios);
create table D as(select pkespecialidade as fkespecialidade);

create table S as(select * from P, D);
create table T as(select * from S
  minus(select * from PD);
create table W as(select fkfuncionario from T);
create table R as(select * from P minus(select * from W));
create table V as(select * from R, funcionarios as f where R.fkfuncionario=f.pkfuncionario);

select txprenomes, txsobrenome from V;
drop table a, b, PD, P, D, S, T, W, R, V;
```

Agora podemos implementar os comandos que respondem a algumas perguntas sobre esta base de dados.

Resolução de Perguntas com Álgebra Relacional e SQL

Agora podemos implementar os comandos que respondem a algumas perguntas sobre esta base de dados.

Faça esta etapa da seguinte maneira: Analise a pergunta e formule os comandos em Álgebra relacional que respondem a pergunta. Depois de formulada a resposta em AR, escreva os comandos em SQL usando o máximo possível de aproximação com a Álgebra Relacional. Se desejar usar alguma técnica da linguagem SQL para melhorar o comando faça COMENTÁRIOS explicando sua implementação.

1. Quais foram os funcionários (cite: primeiro nome e sobrenome) que participaram nos projetos feitos pelo departamento onde o gerente é o funcionário "HEATHER"?

```
a <- funcionarios [[ txprenomes = 'HEATHER']]
b <- a [[ a.pkfuncionario = departamentos.fkfuncionariogerente ]] departamentos
c <- b [[ b.pkdepto = projetos.fkdepto ]] projetos
a1 <- funcionarios [[ funcionarios.fkdepto = c.pkdepto ]] c
a1 [[ txprenome, txsobrenome ]]
```

```
Create table a as( select * from funcionarios where txprenomes 'Heather' );
Create table b as( select * from a, departamentos where a.pkfuncionario = departamentos.fkfuncionariogerente );
Create table c as( select * from b, projetos where b.pkdepto = projetos.fkdepto );
Create table a1 as( select * from funcionarios, c where funcionarios.fkdepto = c.pkdepto );
select txprenome, txsobrenome from a1;
```

2. Quais foram os nomes das tarefas que foram executadas em projetos que são dos departamentos onde o departamento superior é o departamento "A01"?

```
a <- departamentos [[ fkdeptosuperior = 'A01' ]]
b <- a [[ a.fkdepto = projetos.fkdeptosuperior ]] projetos
c <- b [[ b.pkprojetos = tarefasprojetos.fkprojetos ]] tarefasprojetos
d <- c [[ c.fktarefas = tarefas.pktarefa ]] tarefas
d [[ txnometarefa ]]
```

```
Create table a as(select * from departamentos where fkdeptosuperior = 'A01');
Create table b as(select * from a, projetos where a.fkdepto = projetos.fkdeptosuperior);
Create table c as(select * from b, tarefasprojetos where b.pkprojetos = tarefasprojetos.fkprojetos);
Create table d as(select * from c, tarefas where c.fktarefas = tarefas.pktarefa);
select txnometarefa from d;
```

3. Quais são os nomes dos funcionários que registraram vendas para os clientes da cidade de "Bauru"?

```
a <- cidades [[ txnomecidade = 'Bauru' ]]
b <- a [[ a.pkcidade = logradouros.fkcidades  ]] logradouros
c <- b [[ b.pklogradouro = clientes.fklogradouros ]] clientes
d <- c [[ c.pkcliente = nfvendas.fkcliente ]] nfvendas
e <- d [[ d.fkfuncionario = funcionarios.pkfuncionario ]] funcionarios
```

```
Create table a as(select * from cidades where txnomecidade = 'Bauru');
Create table b as(select * from a, logradouros where a.pkcidade = logradouros.fkcidade);
Create table c as(select * from b, clientes where b.pklogradouro = clientes.fklogradouros);
Create table d as(select * from c, nfvendas where c.pkcliente = nfvendas.fkcliente);
Create table e as(select * from d, funcionarios where d.fkfuncionario = funcionario.pkfuncionario);
select txprenome from e;
```

4. Quais são os nomes dos projetos que utilizaram todas as tarefas?

a <- projetos [[pkprojeto, txnomeprojeto]]
b <- tarefas [[pktarefas]]
c <- a [[a.pkprojeto = tarefasprojetos.fkprojeto]] tarefasprojetos
d <- c[[c.fktarefas = tarefas.pktarefas]] tarefas
d [[txnomeprojeto]]

```
create table a as(select pkprojeto, txnomeprojeto from projetos);
create table b as(select pktarefas from tarefas);
create table c as(select * from a, tarefasprojetos where a.pkprojeto = tarefasprojetos.fkprojeto );
create table d as(select * from c, tarefas where c.fktarefas = tarefas.pktarefas);
select txnomeprojeto from d;
```

5. Quantas notas fiscais de vendas foram emitidas para cada cliente no primeiro **semestre** de 2016? Para cliente sem nota fiscal emita o valor "ZERO".

```
a <- nfvendas [[ dtemissao = ' 01-01-2016' entre '31-12-2016' ]]
a [[ soma(dtemissao)]]
b <- a  [[ a.fkcliente = clientes.pkcliente ][ clientes
c <- b [[distinct vltotalnfvenda = nulo]]
d <- c [[ vltotalnfvenda = '0']]
```

```
create table a as(select sum(dtemissao) from nfvendas where dtemissao between ' 01-01-2016' and  '31-12-2016');
create table b as(select * from a, cliente where a.fkcliente = clientes.pkcliente);
create table c as(select distinct from b where vltotalnfvenda = null);
insert into c (vltotalnfvenda) values (0);
```

6. Quantos funcionários atuaram em todos os projetos no ano de 2018?

```
a <- projetos [[ dtcadprojeto entre '01-01-2018' até '31-12-2018' ]]
b <- a [[ a.fkfuncionarioresponsavel = funcionarios.pkfuncionarios ]] funcionarios
b [[count(pkfuncionarios)]]
```

```
create table a as(select * from projetos where between '01-01-2018' and '31-12-2018');
create table b as(select * from a, funcionarios where a.fkfuncionarioresponsavel = funcionarios.pkfuncionario);
select count(pkfuncionarios) from b;
```

7. Exiba os nomes das Instituições de ensino que oferecem mais do que 5 cursos de capacitação na Área de Estudo: "Ciência da Computação"

```
a <- areaestudos [[ txnomearea = 'Ciência da computação" ]]
b <- a [[ a.pkareaestudo = areaestudocursos.fkareaestudos ]] areaestudocursos
c <- b [[ b.fkcursos = cursos.pkcurso ]] cursos
d <- c [[ c.fkinstituicao = instituicoesensino.pkinstituicao ]] instituicoesensino
e <- d [[ se pkcurso > 5 ]]
e [[ txnomeinstituicao ]]
```

```
create table a as(select * from areaestudos where txnomearea = 'Ciência da Computação');
create table b as(select * from a, areaestudocursos where a.pkareaestudo = areaestudocursos.fkareaestudos);
create table c as(select * from b, cursos where b.fkcursos = cursos.pkcurso);
create table d as(select * from c, instituicoesensino where c.fkinstituicao = instituicoesensino.pkinstituicao);
create table e as(select * from d where pkcurso > 5);
select txnomeinstituicao from e;
```

8. Exiba os títulos de livros do acervo de livros que NÃO foram movimentados no ano de 2018 e calcule a percentagem destes livros em relação ao total de livros do acervo. O que pode representar este número?

```
a <- movimentos [[ movimentos.fklivro = livros.pklivro ]] livros]
b <- a [[dtmovimento entre '01-01-2018' até '31-12-2018']]
c <- a - b
c [[ txtituloacervo ]]
p1 < - livros [[ count(pklivros) -> vT1]]
p2 <- c [[ sum(count (pklivros) * 100) - vR2 ]]
p3 <- p1 U p2
p4 <- p3[[ vR2/vT1 -> total  ]]
p4 [[ total ]]


create table a as(select * from movimentos, livros where movimentos.fklivro = livros.pklivro);
create table b as(select * from a where dtmovimento between '01-01-2018' and '31-12-2018');
create table c as(select * from a minus(select * from b));
select txtituloacervo from c;
create table p1 as(select count(pklivro) as vT1);
create table p2 as(select sum(count(pklivro) * 100) as vR2);
create table p3 as(select * from p1 union(select * from p2));
create table p4 as(select sum(vT2/vT1) as total);
select total from p4;
```

9. Qual a percentagem de funcionários que NÃO tem carro?

```
a <- carros [[ carros.fkfuncionario = funcionarios.pkfuncionarios ]] funcionarios
b <- funcionarios - a
c1 <- funcionarios [[count(pkfuncionario)-> vT1]]
c2 <- b [[ sum(count (pkfuncionarios)->vR2 * 100)]
c3  <- c1 U c2
c4 <- c3[[sum(vR2/vT1)-> total]
c4 [[ total]]
```

```
create table a as(select * from carros, funcionarios where carros.fkfuncionario = funcionarios.pkfuncionarios  );
create table b as(select * from funcionarios minus(select * from a);
create table c1 as(select count(pkfuncionario) as vT1 from funcionarios );
create table c2 as(select sum(count(pkfuncionario) * 100) as vR2);
create table c3 as(select * from c1 union(select * from c2));
create table c4 as(select sum(vT2/vT1) as total);
select total from c4;
```

10. Qual a quantidade de consultas realizadas no ano de 2010 que foi coberta por planos de saúde? Qual a percentagem de funcionários que usaram estas consultas? Se o custo por consulta é de R$40,00 e os planos de saúde custam R$350,00 por funcionário, responda: Qual o número mínimo de consultas que torna o pagamento dos planos de saúde com uma relação custo/benefício positivo?

```
a <- consultas [[ conta(pkconsulta) -> tc dthorarealizada entre '01-01-2010' até '31-12-2010'  e fkplanosaude não nulo]]
a [[ tc ]]
c1 <- funcionarios [[ conta(pkfuncionario) -> vT1]]
b <- a[[ a.fkfuncionario = funcionarios.pkfuncionario]]funcionarios
c2 <- b [[ sum(count (pkfuncionarios)->vR2 * 100)]
c3  <- c1 U c2
c4 <- c3[[sum(vR2/vT1)-> total]
c4 [[ total ]]

 sum(planoSaude/custoPConsulta) -> r1
 ceil(r1) -> r2
 [[r2]]


 create table a as(select count(pkconsulta) as tc where dthorarealizada between '01-01-2010' and '31-12-2010' and  fkplanosaude is not null);
 select tc from a;
 create table c1 as(select count(pkfuncionario)as vT1) ;
 create table b as(select * from a, funcionarios where a.fkfuncionario = funcionario.pkfuncionario);
 create table c2 as(select sum(count(pkfuncionarios) as vR2 * 100));
 create table c3 as(select * from c1 union(select * from c2);
 create table c4 as(select sum(vR2/vT1)as total);
 select total from c4;

 select sum(planoSaude/custoPConsulta) as r1;
 select ceil(r1) as r2;
 select r2;
```

Perguntas e Comandos de Resposta

1. A base de dados apresentada tem a necessidade de ser atualizada com alguns registros para ter efeito alguns comandos que serão escritos mais adiante neste exercício. Escreva UM comando que atualize a tabela produtos inserindo as seguintes linhas

id	txnome	vlpreco	vlprecopromocao	vlprecominimo	qtestoque	qtdiaspromocao	txdescricaocompleta	dtcadproduto
001003001002	Dihidróxido de Enxofre	512.23	470	460	25000	5	Ácido Sulfúrico usado para quebrar moléculas de material orgânico combinado com Titânio	2019-02-16
001003001003	Dióxido de Titânio	4089.9	3900	3700	34500	5	Reagente para combinar com corantes e garantir a aderência em superfícies diversas	2019-02-16
001003001011	Bisulfídrio de Amônia	2512.15	2200	1900	12500	10	Usado para disolver resíduos na fabricação do TiO2	2019-02-16
001003001021	Hidróxido de Amônia	2150	1900	1500	10000	0	Usado no processo de oxidação do Ti	2019-02-16
001003001301	Cloreto de Sódio	1745.52	1600	1300	85000	2	Sal. Dissolvido no processo de separação do Ti. Resulta em Cl e Na.	2019-02-16

```
insert into tabelaexemplo
                    (id, txnome, vlpreco, vlprecopromocao, vlprecominimo, qtestoque, qtdiaspromocao, txdescricaocompleta, dtcadproduto)
            values ('001003001002', 'Dihidróxido de Enxofre', '512.23', '470', '460', '25000', '5', 'Ácido Sulfúrico usado para quebrar moléculas de
material orgânico combinado com Titânio', '2019-02-16'),
            values ('001003001003', 'Dióxido de Titânio', '4089.9', '3900', '3700', '34500', '5', 'Reagente para combinar com corantes e garantir a
aderência em superfícies diversas', '2019-02-16'),
            values ('001003001011', 'Bisulfídrio de Amônia', '2512.15', '2200', '1900', '12500', '10', 'Usado para disolver resíduos na fabricação do TiO2', '2019-02-16'),
            values ('001003001021', 'Hidróxido de Amônia', '2150', '1900', '1500', '10000', '0', 'Usado no processo de oxidação do Ti', '2019-02-16'),
            values ('001003001301', 'Cloreto de Sódio', '1745.52', '1600', '1300', '85000', '2', 'Sal. Dissolvido no processo de separação do Ti', '2019-02-16');
```

2. O seguinte comando deve ser executado na base de dados:

```
INSERT INTO feitospor
       VALUES
         (1,10,5, 'Ajuste do parachoque',  2,'2019-02-05'),
         (2,10,10,'Ajuste do parabrisa',   2,'2019-02-05'),
         (3,10,15,'Troca da Porta Direita',2,'2019-02-05'),
         (4,10,20,'Pintura dos ajustes',   2,'2019-02-05'),
         (5,15,5, 'Ajuste do parachoque',  2,'2019-02-05'),
         (6,15,10,'Ajuste do parabrisa',   2,'2019-02-05'),
         (7,20,15,'Troca da Porta Direita',2,'2019-02-05'),
         (8,20,20,'Pintura dos ajustes',   2,'2019-02-05');
```

```
ERROR:  invalid input syntax for type date: "Ajuste do parachoque"
LINE 3: (1,10,5, 'Ajuste do parachoque', 2,'2019-02-05'),
                 ^
SQL state: 22007
Character: 39
```

3. Escreva um comando que atualize todos os registros das notas fiscais de venda segundo a regra:
o valor unitário deve ser 2.95 se o produto for 001001001001,
3.66 se o produto for 001001001002
2.45 se o produto for 001001001003,
2.50 se o produto for 001001001004 e
1.75 se o produto for 001001001005,
e se o produto não for nenhum valor entre 001001001001 e 001001001005 então o valor unitário deve ser 1.90.

```
if (id == '001001001001') then insert into tabelaexemplo (valorUnitario) value ('2.95');
else if (id == '001001001002') then insert into tabelaexemplo (valorUnitario) value  ('3.66');
else if (id == '001001001003') then insert into tabelaexemplo (valorUnitario) value ('2.45');
else if (id == '001001001004') then insert into tabelaexemplo (valorUnitario) value ('2.50');
else if (id == '001001001005') then insert into tabelaexemplo (valorUnitario) value ('1.75');
else insert into tabelaexemplo (valorUnitario) value ('1.90');
```

4. Os professores que têm residência em logradouros das cidades de 'Osasco' ou 'Barueri' devem estar habilitados a ministrar aulas em cursos com carga horária maior ou igual a 100 horas. A partir do dia 01 de novembro de 2018 (sendo que se considere esta data como data de habilitação, de capacitação e de cadastramento na tabela habilitações). As combinações possíveis para esta condição podem ser geradas a partir do produto cartesiano entre professores e cursos. Escreva o comando que gere o conjunto de dados (linhas) na tabela habilitações supondo que em habilitações o valor máximo do campo pkhabilitacao seja 70 (já gravado na tabela) e deva ser incrementado de 10 em 10.

```
create function incrementaTeste()
returns trigger as $BODY$
begin
	update habilitações
		set increment = increment + 10
	 where pkhabilitacao = '70'
end;
$BODY$
```

5. Um analista de sistemas trabalhava na administração de um banco de dados e recebeu a ordem de alterar os valores unitários dos registros de itens de notas fiscais. Depois de executar o comando ele precisa atualizar o valor total das notas de vendas. Escreva um comando que atualize a tabela das notas de vendas calculando o valor total das notas como sendo a somatória dos valores unitários multiplicados pelas respectivas quantidades em cada item de nota correspondente a uma nota fiscal.

```
begin
    update NotasVendas set valorTotalNotas = (sum(valorUnitario)* count(itemNota = fknotafiscal))
end;
```

6. Exiba as linhas da tabela clientes onde o campo txrazaosocial contenha a palavra “Tratoria” escrito com letras maiúsculas ou minúsculas.

```
select * from clientes where txrazaosocial ilike '%tratoria%';
```

7. Exiba uma lista com os nomes dos funcionarios de cada departamento e para cada departamento em ordem por funções desempenhadas pelos funcionarios. Esta lista deve ter os dados dos departamentos (exiba Código e nome do departmento), onde o Código do departamento esteja entre A01 e D01, também devem ser exibidos o nome da função (ou cargo) ao lado do Código da função:

```
create table a as(select * from funcionarios, departamentos);
select txprenomes, txsobrenome,  pkdepto, txnomedepto from a  where fkdepto between 'A01'  and 'D01' order by (txnomedepto);
drop table a;
```

8. Escreva o comando que cria um novo campo com nome cechefe na tabela funcionários que permita fazer um relacionamento reflexivo entre as linhas desta tabela. Este campo deve ser uma chave estrangeira na tabela funcionários apontando para a chave primária da mesma tabela (relacionamento reflexivo). Se uma linha tiver a PK alterada a alteração deve ser propagada para as FKs correspondentes. Se uma linha for excluída de funcionários seus correspondentes valores de FKs devem ficar nulos:

```
ALTER TABLE public.funcionarios
   ADD COLUMN cechefe integer;
ALTER TABLE public.funcionarios
   ADD CONSTRAINT funcionariosfk3 FOREIGN KEY (cechefe)
   REFERENCES public.funcionarios (pkfuncionario) MATCH SIMPLE
   ON UPDATE CASCADE
   ON DELETE CASCADE
   DEFERRABLE INITIALLY DEFERRED;
```

9. Exiba os dados dos funcionários que são gerentes de departamentos

Resposta:

```
create table a as(select * from funcionarios, departamentos where departamentos.fkfuncionariogerente = funcionarios.pkfuncionario);
select pkfuncionario, nucpf, txprenomes, txsobrenome, fkdepto, fkfuncao, fkniveleducacao, fklogradouro, txcomplemento, nuramal, dtcontratacao, aosexo, dtnascimento, txresenha, vlsalario, vlbonus, vlcomissao, nucep, dtcadfuncionario from a;
drop table a;
```

10. Um executivo quer receber um relatório onde se vejam os funcionários contratados antes do ano 2000 ou que tenham o nível de escolaridade acima ou igual ao Ensino Superior, estes funcionários também devem ter uma renda anual superior à 40000. Os dados devem ser ordenados pelo nome de departamento ao qual o funcionário pertence:

```
create table a as(select * from funcionarios, grausdeescolaridade where txnomecomum ilike '%Ensino Superior%'  or dtcontratacao between '01-01-1700' and '31-12-2000');
create table b as(select * from a, departamentos where SUM(vlsalario) * 12 > 4000);
select txnomedepto, ptxprenomes from b order by (txnomedepto);
drop table a, b;
```

11. Exiba os dados de funcionários (código e nome completo e departamentos) onde o código do funcionário seja igual a 10 ou 80 ou 170 ou 280:

```
create table a as(select * from funcionarios, departamentos where fkdepto = pkdepto);
select pkfuncionario, txprenomes, txsobrenome, txnomedepto from a where pkfuncionario = '10' or '80' or '170' and '280';
```

12. Escreva os comandos que criem três tabelas (não temporárias). A primeira com nome func1 com os dados da tabela funcionários que perteçam aos departamentos com nome 'Pesquisa de ferramentas de arquitetura computacional' ou 'Desenvolvimento de Software'. A segunda tabela deve ter o nome de func2 e deve ter os dados de funcionários que foram contratados antes do ano 2000. A terceira tabela deve ter o nome de dept e deve conter os dados de departamento onde o nome do departamento seja 'Oceanografia aplicada em des. de software' ou 'Desenvolvimento de Software'

```
create table func1 as(select * from funcionarios, departamentos where txnomedepto = 'Pesquisa de ferramentas de arquitetura computacional' or 'Desenvolvimento de Software');
create table func2 as(select * from funcionarios where dtcontratacao between '01-01-1555' and '31-12-1999' );
create table dept as(select * from departamentos where txnomedept = 'Oceanografia aplicada em des. de software' or 'Desenvolvimento de Software' );
```

13. Escreva um comando que exiba os dados dos funcionários, os nomes de suas funções e os nomes dos departamentos onde trabalham de todos os funcionários que trabalhem em departamentos onde o departamento superior seja o 'Pesquisa em Biociências'.

```
create table a as(select * from funcionarios, departamentos where txnomedept='Pesquisa em Biociências');
select  from pkfuncionario, nucpf, txprenomes, txsobrenome, fkdepto, fkfuncao, fkniveleducacao, fklogradouro, txcomplemento, nuramal, dtcontratacao, aosexo, dtnascimento, txresenha, vlsalario, vlbonus, vlcomissao, nucep, dtcadfuncionario, txnomedepto;
```

14. Escreva um comando que exiba os códigos e nomes de funções, os códigos, prenomes e sobrenomes e códigos e nomes dos departamentos dos funcionários da tabela func1.

```
create table func1 as(select * from departamentos, funcionarios);
select pkfuncionario, pkdepto, txprenomes, txsobrenome from func1;
```

15. Escreva um comando em SQL usando o operador JOIN que seja equivalente a este comando:

```
select * from autores, autorias, livros, bibliografia, disciplinas
          where autores.pkautor=autorias.fkautor and
                autorias.fklivro=livros.pklivro and
                livros.pklivro=bibliografia.fklivro and
                bibliografia.fkdisciplina=disciplinas.pkdisciplina;
```

```
select autores.pkautor, autorias.fkautor, autorias.fklivro, livros.pklivro, bibliografia.fklivro, bibliografia.fkdisciplina, disciplinas.pkdisciplina
    from autores
    join autorias on autores.pkautor = autorias.fkautor
    join livros on autorias.fklivro = livros.pklivros
    join bibliografia on livros.pklivro = bibliografia.fklivro
    join disciplinas on bibliografia.fkdisciplina = disciplinas.pkdisciplina;
```

16. Escreva pelo menos dois comandos que exibam os códigos, prenomes, nomes e datas de nascimento do resultado da união dos registros das tabelas func1 e func2 exibindo todas linhas (mesmo as repetidas)
(*) O operador UNION deverá ser usado na resposta. Este operador pode ser usado com o complemento ALL. O que acontece se tiramos a palavra ALL (complemento do operador)?

```
select cod1, cod2, prenomes, nomes, dtnasc from func1, func2;

select cod1, cod2, prenomes, nomes, dtnasc from func1 union all(select * from func2);
```

Union ALL trás a união de todas as tuplas
Já o Union ele não trás valores duplicados

17. Escreva pelo menos dois comandos que exibam os códigos, prenomes, nomes e datas de nascimento dos registros que estão na tabela funcionários e que não estejam na tabela func1 nem na tabela func2.

```
select * from funcionarios minus(select * from func1, fun2)

select * from funcionarios except(select * from func1, fun2)
```

18. Escreva pelo menos dois modos de uso do operador JOIN para realizar uma junção fechada entre funcionários e a união entre func1 e func2.

```
select * from funcionario, func1, func2  where funcionario.pkfuncionario = func1.pkfuncionario and funcionario.pkfuncionario = func2.pkfuncionario;

select * from funcionario join func1 on funcionario.pkfuncionario = func1.pkfuncionario
                        join func2 on func1.pkfuncionario = func2.pkfuncionario;
```

19. Escreva um comando que exiba uma lista das especialidades médicas que não tem relacionamento com médicos.

```
select * from especmedicas right join medicos especmedicas.pkespecialidade = medicos.fkespecialidade;
```

20. Escreva um comando que exiba uma lista com nomes dos cursos e os respectivos nomes dos professores relacionados ordenando esta lista pelo Código do curso e dentro desta sequencia, pelos nomes dos professores. Neste mesmo comando o resultado também deve ser a lista dos professores que NÃO TEM cursos relacionados.

```
create table a as(select * from professores, habilitações where professores.pkprofessor = habilitacoes.fkprofessor);
create table b as(select * from a right join cursos on cursos.pkcursos = a.fkcurso);
select txnomeprofessor, txnomecurso from order by(pkcurso);
```

21. Escreva um comando que exiba uma lista com os códigos dos projetos, seus nomes e a quantidade de tarefas que são (e foram) realizadas no projeto.

```
create table a as(select * from tarefas, tarefasprojetos);
create table b as(select * from projetos, a);
select pkprojeto, txnomeprojeto, txnometarefa, sum(pktarefas) from b;
```

22. Escreva um comando que exiba uma lista com os códigos e nomes dos departamentos com os dados agrupados e ordenados por departamento e sexo e mostre os valores do salário anual mínimo, médio e máximo, bem como a quantidade de funcionários e o salário total de cada departamento (separados por sexo). Devem ser exibidos somente os departamentos com salário médio anual acima de 150000.
Dica: para emitir um título para cada valor do sexo de cada departamento pesquise e use o sub-comando CASE...END.

```
select * from departamentos
```

23. Escreva um comando que exiba uma lista com os códigos dos departamentos e os valores totais das remunerações pagas aos seus funcionários sendo que somente devem ser exibidos os dapartamentos com remuneração total superior à 80000/ano.

```
create table a as(select * from departamentos, funcionarios);
create table b as(select sum(vlsalario) as total, total*12);
select total from b where total>80000;
```

24. Escreva um comando em SQL que emita uma lista com o seguinte cabeçalho: idprojeto, txnomeprojeto, txnometarefa, txprenomes, txsobrenome.

```
create table a(select pkprojeto as idprojeto, txnomeprojeto, txnometarefa, txprenomes, txsobrenome from tarefas
                join tarefasprojetos  on  tarefa.pktarefa = tarefasprojetos.fktarefa
                join projetos on tarefasprojetos.fkprojeto = projetos.pkprojeto
                join funcionarios on projetos.fkfuncionarioresponsavel = funcionarios.pkfuncionario);
```

25. Um Gerente de RH quer fazer uma previsão para a correção de salário prevista para o mês seguinte. Sabendo que o índice de correção será de 23,14521% escreva um comando que emita uma lista com os códigos dos funcionários, seu salário e uma coluna com os valores de salários corrigidos por este índice.

```
create table a as(select vlsalario as correcao from funcionarios);
select correcao * 23.14521;
create table b as(select sum(vlsalario + correcao)as vlsalcor from a);
select pkfuncionario, vlsalario, vlsalcor from b;
```

26. Um gerente de produção procura você e te faz esse pedido: “Preciso de uma lista simples com os códigos e nomes dos projetos e com as quantidades de tarefas que foram executadas nos projetos sempre nos últimos 45 dias”. Escreva um comando SQL que consiga atender a este requisito do gerente de produção.

```
create table a as(select * from tarefas
                                join tarefasprojetos  on  tarefa.pktarefa = tarefasprojetos.fktarefa
                                join projetos on tarefasprojetos.fkprojeto = projetos.pkprojeto);
select count(pktarefas)as qtdtarefas where dttermino age (timestamp) interval(45) from a;
```

27. Escreva um comando em SQL que exiba uma lista com os nomes e sobrenomes de funcionarios que sejam chefes de departamentos, bem como os códigos e nomes dos departamentos e nomes e sobrenomes dos funcionários que trabalham nos respectivos departamentos.

```
create table a as(select * from funcionarios, departamentos);
select txprenome, txsobrenome from a;
```

28. Escreva um comando em SQL que exiba uma lista com os códigos e nomes dos departamentos que não tem funcionários relacionados.

```
create table a as(select * from funcionarios right join departamentos on funcionarios.fkdepto = departamentos.pkdepto);
create table b as(select * from funcionarios, departamentos where funcionarios.fkdepto = departamentos.pkdepto);
create table c as(select * from a except(select * from b);
select pkdepto, txnomedepto from c;
```

29. Escreva um comando que exiba uma lista com nome e sobrenome dos funcionários com mais de 40 anos.

```
select txprenome, txsobrenome from funcionarios where age(current_date,  dtnascimento)as idade and idade > 40;
```

30. Escreva um comando em SQL que exiba uma lista com os dados de departamentos, seus funcionários e as funções que estes exercem, mas somente para os funcionários que foram copiados para a tabela func1. Esta lista deve estar ordenada por nome do departemento, nome da função e nome do funcionário.

```
create table func1 as(select * from funcionarios);
select * from func1, departamentos, funcoes order by(txnomedepto);
```

31. Escreva um comando em SQL que exiba uma lista com os códigos, prenomes e sobrenomes dos funcionários que trabalham nos departamentos 'A01' ou 'E21' unidos aos funcionários que trabalham nos departamentos 'D01' ou 'D11' ou 'E21' onde os nomes (de todos contenham a letra 'O' em qualquer posição.

```
create table a as(select * from funcionarios where fkdepto =  'A01' or 'E21');
create table b as(select * from funcionarios where fkdepto = 'D01' or 'D11' or 'E21');
create table c as(select * from a union all(select * from b));
select fkdepto, txprenomes, txsobrenomes from c where txprenomes ilike '%o%' or txsobrenomes ilike '%o%';
```

32. Escreva um comando em SQL (somente UM) que realize a divisão entre projetos, tarefasprojetos e tarefas e que exiba uma lista com os códigos de tarefas que foram realizadas em todos os projetos. Para testar (e obter sucesso) no seu comando, você poderá atualizar as tabelas envolvidas nesta pergunta. SE você fizer isso, escreva no espaço complementar os comandos que você executou para atualizar as tabelas do banco de dados.

```
select * from projetos except(select * from tarefasprojetos except(select * from tarefas)));
```

Espaço complementar para os eventuais comandos de atualização das tabelas do banco de dados.

33. Escreva um comando que exiba uma lista com dados dos funcionários (id, prenomes e sobrenome), a soma de seus ganhos e a percentagem em relação à soma dos ganhos de todos os funcionários que tenham mais de 40 anos.

```
select sum(vlsalario, vlcomissao, vlbonus)as soma from funcionaros where age(current_date,  dtnascimento)as idade and idade > 40
```

Desculpa professor eu não entendi a parte da porcentagem

34. Um analista de sistemas tentou escrever um comando para realizar a divisão entre as tabelas: Oficinas, Feitospor e Serviços para responder a seguinte pergunta quais os nomes das oficinas (razão social e apelido), que prestam todos os serviços; para isso ele escreveu o seguinte comando:

```
CREATE TEMPORARY TABLE A AS (SELECT id as oficinasid FROM oficinas);
CREATE TEMPORARY TABLE B AS (SELECT id as servicosdemanutencaoid FROM servicos);
CREATE TEMPORARY TABLE AB AS (SELECT * FROM A,B);
CREATE TEMPORARY TABLE NAOEXISTE AS
       (SELECT * FROM AB WHERE CONCAT(oficinasid,servicosdemanutencaoid) NOT IN
       (SELECT CONCAT(oficinasid,servicosdemanutencaoid) FROM feitospor));
CREATE TEMPORARY TABLE ACAOSEMP AS (SELECT DISTINCT oficinasid FROM NAOEXISTE);
SELECT oficinas.txrazaosocial,oficinas.txapelido
       FROM oficinas, (SELECT * FROM A EXCEPT (SELECT * FROM ACAOSEMP)) AS TodasOficinas
                              WHERE oficinas.id=TodasOficinas.oficinasid;
```

Entretanto, este comando apresenta erros (alguns). Então pede-se: Ache os erros e escreve o comando correto.

```
CREATE TEMPORARY TABLE A AS (SELECT pkoficina as oficinasid FROM oficinas);
CREATE TEMPORARY TABLE B AS (SELECT pkservico as servicosdemanutencaoid FROM servicos);
CREATE TEMPORARY TABLE AB AS (SELECT * FROM A,B);
CREATE TEMPORARY TABLE NAOEXISTE AS
(SELECT * FROM AB WHERE CONCAT(oficinasid,servicosdemanutencaoid) NOT IN
(SELECT CONCAT(oficinasid,servicosdemanutencaoid) FROM feitospor));
CREATE TEMPORARY TABLE ACAOSEMP AS (SELECT DISTINCT oficinasid FROM NAOEXISTE);
SELECT oficinas.txrazaosocial,oficinas.txapelido
FROM oficinas, (SELECT * FROM A EXCEPT (SELECT * FROM ACAOSEMP)) AS TodasOficinas
WHERE oficinas.pkoficina=TodasOficinas.oficinasid;

drop table a, b, ab, naoexiste, acaosemp;
```

35. Escreva um comando que exiba uma lista com os departamentos e seus departamentos subordinados em primeiro nível (somente o departamento principal e seu subdepartamento, sem mostrar subdepartamento de outros subordinados). Esta lista orienta a destinação de verba de investimento no desenvolvimento dos projetos destes departamentos.

```
select pkdepto, txnomedepto, fkdeptosuperior from departamentos distinct order by(fkdeptosuperior) group by(fkdeptosuperior)
```