1. Considere o seguinte esquema relacional (as chaves primárias estão sublinhadas):

Pessoa(Cod, Nome, DataNasc)
Publicacao(Cod, Titulo, CodArea)
CodArea referencia Area
Autor(CodAutor, CodPublicacao)
CodAutor referencia Pessoa
CodPublicacao referencia Publicacao
Avaliacao(CodAvaliador, CodPublicacao, Nota)
CodAvaliador referencia Pessoa,
CodPublicacao referencia Publicacao
Area(Cod, Nome, CodAreaGenerica)
CodAreaGenerica referencia Area

(a) Escreva os comandos SQL necessários para criar as relações Área e Publicação acima, incluindo as restrições de chave primária e chave estrangeira. As seguintes restrições de integridade devem ser garantidas:

1. Ao excluir uma Área, todas as suas subáreas devem ser excluídas.

2. Uma área não pode ser excluída se houverem Publicações associadas a ela.

```
CREATE TABLE Area (
cod INTEGER NOT NULL,
nome VARCHAR(30),
codAreaGenerica INTEGER,
PRIMARY KEY (cod),
FOREIGN KEY (codAreaGenerica) references Area (cod) ON DELETE CASCADE
)
```

```
CREATE TABLE Publicacao (
cod INTEGER NOT NULL,
titulo VARCHAR(50),
codArea INTEGER,
PRIMARY KEY (cod),
FOREIGN KEY (codArea) REFERENCES Area (cod) ON DELETE RESTRICT
)
```

(b) Escreva a instrução SQL necessária para incluir uma nova Publicação com código 122, título "XML e Banco de Dados", da área de código 11.

```
INSERT INTO Publicação VALUES (122, "XML e Banco de Dados", 11)
```

(c) Escreva a instrução SQL necessária para excluir todas as publicações da área de "Banco de Dados".

```
DELETE FROM Publicacao
WHERE codArea IN (SELECT cod
                  FROM AREA
                  WHERE nome = "Banco de Dados"
                  )
```

(d) Escreva uma instrução SQL para modificar a nota de todas as avaliações do avaliador João para 10.

```
UPDATE Avaliacao
SET Nota=10
WHERE codAvaliador IN (SELECT cod FROM Pessoa WHERE nome = ”João”)
```

(e) Escreva uma instrução SQL que retorna o número de publicações de cada área. No resultado deve aparecer o nome de cada área, seu código e a quantidade de publicações associada a ela.

```
SELECT COUNT(p.cod) AS NumPublicacoes, a.nome, a.cod
FROM Publicacao p, Area a
WHERE p.CodArea=a.Cod
GROUP BY a.cod, a.nome
```

(f) Escreva uma instrução SQL que retorna o código de todas as áreas que têm mais de 2 publicações.

```
SELECT a.cod
FROM Area a
WHERE (
        SELECT COUNT(*), p.cod
        FROM Publicacao p
        WHERE a.cod=p.codArea
        GROUP BY a.codArea
        ) > 2
```

(g) Escreva uma instrução SQL que retorna a média de notas obtidas pela publicação "XML e Banco de Dados".

```
SELECT AVG(nota)
FROM Avaliacao v, Publicacao p
WHERE p.cod=v.codPublicacao
AND v.titulo="XML e Banco de Dados"
```

(h) Escreva uma instrução SQL que retorna o título da publicação que obteve a maior nota de avaliação, e o nome do Avaliador responsável por esta avaliação.

```
SELECT MAX(v.nota), r.Nome, p.Titulo
FROM Avaliacao v, Publicacao p, Pessoa r
WHERE p.cod=v.codPublicacao
AND v.codAvaliador=r.cod
```

(i) Escreva uma instrução SQL para mostrar o efeito de aumentar em 10% a nota de todas as avaliações do avaliador João.

```
SELECT v.codAvaliador, v.codPublicacao, 1.1*v.nota
FROM Avaliacao v, Pessoa r
WHERE v.codAvaliador=r.cod
AND nome = "João"
```

2. Considere o seguinte esquema relacional (as chaves primárias estão sublinhadas):

Emp (*eid*: integer, *ename*: string, *idade*: integer, *salario*: real)
Trabalha (*eid*: integer, *did*: integer, *cargahoraria*: integer)
Dept (*did*: integer, *dnome*: string, *orçamento*: real, *gerenteid*: integer)

(a) Escreva os comandos SQL necessários para criar as relações Trabalha e Dept acima, incluindo as restrições de chave primária e chave estrangeira. As seguintes restrições de integridade devem ser garantidas:

1. Ao excluir um Departamento, todas as tuplas relacionadas em Trabalha devem ser excluídas.

2. Um empregado não pode ser excluído se ele for gerente de algum departamento.

```
CREATE TABLE Trabalha (
        eid INTEGER NOT NULL,
        did INTEGER NOT NULL,
        cargahoraria INTEGER,
        PRIMARY KEY (eid, did),
        FOREIGN KEY (eid) references Emp (eid)
        FOREIGN KEY (did) references Dept (did) ON DELETE CASCADE
        )
```

```
CREATE TABLE Dept (
        did INTEGER NOT NULL,
        dnome VARCHAR(50),
        orçamento: REAL,
        gerenteid INTEGER,
        PRIMARY KEY (did),
        FOREIGN KEY (gerenteid) REFERENCES Emp (eid) ON DELETE
RESTRICT
        )
```

(b) Escreva a instrução SQL necessária para incluir um novo empregado de nome João Silva, eid 1234 e salário 1200.

```
INSERT INTO Emp VALUES (1234, “João Silva”, 1200)
```

(c) Escreva a instrução SQL necessária para excluir todos os empregados que trabalham no departamento cujo did é igual a 122.

```
DELETE FROM Emp
WHERE eid IN (SELECT eid FROM Trabalha WHERE did = 122)
```

(d) Escreva uma instrução SQL para modificar o salário de todos os empregados que trabalham no departamento 122. O aumento no salário deve ser de 20%.

```
UPDATE Emp
SET salario=salario*1,2
WHERE eid IN (SELECT eid FROM Trabalha WHERE did = 122)
```

(e) Escreva uma instrução SQL que retorna a média dos salários do departamento cujo id é 122

```
SELECT AVG(salario)
FROM Emp e, Trabalha t
WHERE e.eid=t.eid
AND t.did=122
```

(f) Escreva uma instrução SQL que retorna o número de empregados de cada departamento. No resultado deve aparecer o nome do departamento, seu did e o número de empregados que nele trabalham.

```
SELECT d.dnome, d.eid, COUNT(t.eid) AS NUMEMP
FROM Dept d, Trabalha t
WHERE t.did=d.did
GROUP BY d.dnome, d.eid
```

(g) Escreva uma consulta SQL que retorne, para cada departamento, o nome do departamento e maior salário dentre seus empregados.

```
SELECT d.dnome, MAX(SALARIO)
FROM Emp e, Dept d, Trabalha t
WHERE t.did=d.did
AND e.eid=t.eid
GROUP BY d.dnome, d.eid
```

3. Considere o seguinte esquema relacional (as chaves primárias estão sublinhadas)

Dept (*deptId*: integer, *dnome*: string)

Professor (*pid*: integer, *pnome*: string, *salario*: real, *deptId*: integer)

deptId referencia Dept

Disciplina (*did*: integer, *dnome*: string)

OfertaDisciplina (*did*: integer, *anoSemestre*: string, *pid*: integer)

did referencia Disciplina

pid referencia Professor

(a) Escreva os comandos SQL necessários para criar a relação OfertaDisciplina acima, incluindo as restrições de chave primária e chave estrangeira. As seguintes restrições de integridade devem ser garantidas: [0,5 ponto]

1. Ao excluir uma Disciplina, todas as ofertas daquela disciplina devem ser excluídas.
2. Ao alterar o id de um professor, a tabela OfertaDisciplina deve ser atualizada automaticamente.
3. O id de uma disciplina não poderá ser alterado se já houver alguma oferta para aquela disciplina registrada no banco de dados.

```
CREATE TABLE OfertaDisciplina (
        did INTEGER NOT NULL,
        anoSemestre VARCHAR(6) NOT NULL,
        pid INTEGER NOT NULL,
        cargahoraria INTEGER,
        PRIMARY KEY (did, anoSemestre, pid),
        FOREIGN KEY (did) references Disciplina (did) ON DELETE CASCADE ON
UPDATE  RESTRICT,
        FOREIGN KEY (pid) references Professor (pid) ON UPDATE CASCADE
        )
```

(b) Escreva a instrução SQL necessária para incluir um novo professor de nome João Silva, pid 1234, salário 1200, que trabalha no departamento 122.

```
INSERT INTO Professor (pid, pnome, salário, deptId) VALUES (1234, “João Silva”,
1200, 122)
```

(c) Escreva a instrução SQL necessária para excluir todas as ofertas da disciplina chamada “Algoritmos”.

```
DELETE FROM OfertaDisciplina
WHERE did IN (SELECT did FROM Disciplina WHERE dnome = “Algoritmos”)
```

(d) Escreva uma instrução SQL para modificar o salário de todos os professores que trabalham no departamento 122 e que ministraram alguma disciplina no anoSemestre 2008/1. O aumento no salário deve ser de 15%.

```
UPDATE Professor
SET salario=salario*1,15
WHERE pid IN (SELECT pid FROM OfertaDisciplina WHERE anoSemestre = “2008/1”)
```

(e) Escreva uma instrução SQL que retorna o maior salário dos professores que trabalham no departamento cujo deptId é 123

```
SELECT MAX(salario)
FROM Professor p, Dept d
WHERE p.did=d.deptId
AND d.deptId=123
```

(f) Escreva uma instrução SQL que retorna o número de disciplinas oferecidas em 2008/1. No resultado deve aparecer o anoSemestre e o número de disciplinas oferecidas.

```
SELECT o.anoSemestre, COUNT(o.did) AS NumDisc
FROM OfertaDisciplina o
GROUP BY o.anoSemestre
```

(g) Escreva uma consulta SQL que retorne, para cada departamento, o nome do departamento e a média de salário dos professores daquele departamento.

```
SELECT d.dnome, AVG(p.salario)
FROM Professor p, Dept d
WHERE p.did=d.deptId
GROUP BY d.dnome
```

4. Considere o esquema relacional abaixo. As chaves primárias estão sublinhadas.

Aluno (CPF, Nome, Curso, DataI)
Disciplina (NumDiscipl, Dnome, Depto)
Matricula (CPF, NumDiscipl, Semestre, Nota)
    CPF referencia Aluno
    NumDiscipl referencia Disciplina
LivroAdotado (NumDiscipl, Semestre, ISBNLivro)
    NumDiscipl referencia Disciplina
    ISBNLivro referencia Texto
Texto (ISBNLivro, TituloLivro, Editora, Autor)

(a) Escreva os comandos SQL para criar as tabelas Aluno, Disciplina e Matrícula, incluindo as restrições de integridade que se aplicam. Assuma que:

1. Quando um aluno é excluído, todas as suas matrículas são excluídas automaticamente.
2. Uma disciplina não pode ser excluída se houver alguma informação sobre matrícula associada a ela.
3. Ao alterar o número de uma disciplina, todas as referências na tabela Matrícula devem ser atualizadas.

```
CREATE TABLE ALUNO (
        CPF VARCHAR(9) NOT NULL,
        NOME VARCHAR(30),
        CURSO VARCHAR(30),
        DATAI DATE,
        PRIMARY KEY (CPF)
        )
CREATE TABLE DISCIPLINA (
        NumDiscipl INTEGER NOT NULL,
        Dnome VARCHAR(30),
        Depto VARCHAR(30),
        PRIMARY KEY (NumDiscipl)
        )
CREATE TABLE MATRICULA (
        CPF VARCHAR(9) NOT NULL,
        NumDiscipl INTEGER NOT NULL,
        Semestre VARCHAR(7) NOT NULL,
        Nota DOUBLE,
        PRIMARY KEY (CPF, NUMDISCIPL, SEMESTRE)
        FOREIGN KEY CPF references ALUNO (CPF) ON DELETE CASCADE,
        FOREIGN KEY NumDiscipl references DISCIPLINA (NumDiscipl) ON DELETE RESTRICT,
ON UPDATE CASCADE )
```

(b) Escreva o comando SQL necessário para incluir um novo aluno no banco de dados, com os seguintes dados: CPF 000000000-01, nome “Maria Lima”, curso “Arquitetura” e Data de Início 01/05/2005.

```
INSERT INTO ALUNO
VALUES (“00000000001”, “Maria Lima”, “Arquitetura”, 01/05/2005);
```

(b) Escreva o comando SQL necessário para incluir um novo aluno no banco de dados, com os seguintes dados: CPF 123456789-01, nome “Alexandre Silva”, curso “Computação” e Data de Início 01/03/2007.

```
INSERT INTO ALUNO
VALUES (“12345678901”, “Alexandre Silva”, “Computação”, 01/03/2007);
```

(

(c) Escreva um comando SQL para excluir todas as adoções de livros (tabela LivroAdotado) do autor “Aluísio de Azevedo”.

```
DELETE FROM LIVROADOTADO
WHERE ISBNLivro IN (SELECT ISBNLivro
                    FROM TEXTO
                    WHERE Autor=”Aluísio de Azevedo”)
```

(c) Escreva um comando SQL para excluir todas as adoções de livros da editora “XYZ”.

```
DELETE FROM LIVROADOTADO
WHERE ISBNLivro IN (SELECT ISBNLivro
                    FROM TEXTO
                    WHERE Editora=”XYZ”)
```

(d) Crie uma visão que contenha o nome, o CPF, o curso do aluno e o número disciplinas que ele se matriculou no semestre de 01/2010.

```
CREATE VIEW V (CPF, Nome, Curso, NumDisciplinas) AS
SELECT a.CPF, a.Nome, a.Curso, COUNT(*)
FROM Aluno a, Matricula m
WHERE a.CPF = m.CPF
AND m.Semestre=”01/2010”
GROUP BY a.CPF, a.Nome, a.Curso
```

(d) Crie uma visão que contenha o nome, o CPF do aluno e o número disciplinas que ele se matriculou no semestre de 02/2007.

```
CREATE VIEW V (CPF, Nome, NumDisciplinas) AS
SELECT a.CPF, a.Nome, COUNT(*)
FROM Aluno a, Matricula m
WHERE a.CPF = m.CPF
AND m.Semestre=”02/2007”
GROUP BY a.CPF, a.Nome
```

(e) Escreva uma instrução SQL que dado o aluno de CPF 00000000001 retorna o nome das disciplinas do semestre “01/2010” em que ele está matriculado e o nome dos livros-texto de cada disciplina.

```
SELECT d.Dnome, t.TituloLivro
FROM Matricula m, Disciplina d, LivroAdotado l, Texto t
WHERE m.NumDiscipl = d.NumDiscipl
AND l.NumDiscipl = d.NumDiscipl
AND t.ISBNLivro = l.ISBNLivro
AND m.Semestre = l. Semestre
AND m.CPF = 00000000001
AND m.Semestre=”01/2010”
(HAVING COUNT(*) > 5
```

(e) Escreva uma instrução SQL que dado o aluno de CPF 99999999999 retorna o nome das disciplinas do semestre = “02/2007” em que ele está matriculado e o nome dos livros-texto de cada disciplina.

```
SELECT d.Dnome, t.TituloLivro
FROM Matricula m, Disciplina d, LivroAdotado l, Texto t
WHERE m.NumDiscipl = d.NumDiscipl
AND l.NumDiscipl = d.NumDiscipl
AND t.ISBNLivro = l.ISBNLivro
AND m.Semestre = l. Semestre
AND m.CPF = 99999999999
AND m.Semestre=”02/2007”
```

(f) Escreva uma instrução SQL que retorna o número de disciplinas do semestre “01/2010” com mais de 5 livros-texto. No resultado deve aparecer o nome e número de cada disciplina e a quantidade de livros a ela alocados.

```
SELECT d.Dnome, d.NumDiscipl, COUNT(*)
FROM Disciplina d, LivroAdotado l
WHERE d.NumDiscipl = l.NumDiscipl
AND l.Semestre=”01/2010”
GROUP BY d.NumDiscipl, d.Dnome
HAVING COUNT(*) > 5
```

(f) Escreva uma instrução SQL que retorna o número de disciplinas do semestre = “02/2007” com mais de 2 livros-texto. No resultado deve aparecer o nome e número de cada disciplina e a quantidade de livros a ela alocados.

```
SELECT d.Dnome, d.NumDiscipl, COUNT(*)
FROM Disciplina d, LivroAdotado l
WHERE d.NumDiscipl = l.NumDiscipl
AND l.Semestre=”02/2007”
GROUP BY l.NumDiscipl, d.Dnome
HAVING COUNT(*) > 2
```

(g) Escreva uma instrução SQL aninhada correlacionada que mostre o CPF dos alunos que não estão matriculados no semestre “01/2010”.

```
SELECT a.CPF
FROM Aluno a
WHERE a.CPF NOT IN (
                    SELECT CPF
                    FROM Matricula m
                   WHERE a.CPF = m.CPF
                    AND m.Semestre=”01/2010”
                    )
```

(g) Escreva uma instrução SQL aninhada correlacionada que mostre o nome dos alunos que não estão matriculados no semestre = “02/2007”.

```
SELECT a.Nome
FROM Aluno a
WHERE a.CPF NOT IN (
                    SELECT CPF
                    FROM Matricula m
                    WHERE a.CPF = m.CPF
                    AND m.Semestre=”02/2007”
                    )
```

5. Considere o esquema relacional abaixo. As chaves primárias estão sublinhadas.

(a) Escreva os comandos SQL para criar as tabelas, incluindo as restrições de integridade que se aplicam.

Assuma que:

1. Um proprietário não pode ser excluído se houver alguma informação associada a ele.

2. Ao alterar o número de um apartamento, todas as referências na tabela Cobranca devem ser atualizadas.

3. Um Apartamento não pode ser excluído caso hajam Cobranças associadas.

Proprietário (Nome, CPF)
Apartamento (Número, Andar, CPF)
CPF referencia Proprietário (CPF)
Cobranca (NumApartamento, Data, Valor)
NumApartamento referencia Apartamento (Numero)

```
CREATE TABLE PROPRIETARIO (
        CPF VARCHAR(9) NOT NULL,
        NOME VARCHAR(30)
        PRIMARY KEY (CPF)
        )

CREATE TABLE APARTAMENTO (
        Numero INTEGER NOT NULL,
        Andar INTEGER,
        CPF VARCHAR(9),
        PRIMARY KEY (Numero),
        FOREIGN KEY CPF references PROPRIETARIO (CPF) ON DELETE RESTRICT
        )

CREATE TABLE COBRANCA (
        NumApartamento INTEGER NOT NULL,
        Data DATE NOT NULL,
        Valor DOUBLE,
        PRIMARY KEY (NumApartamento, Data)
        FOREIGN KEY NumApartamento references APARTAMENTO (Numero) ON UPDATE CASCADE
ON DELETE RESTRICT
        )
```

(b) Escreva o comando SQL necessário para incluir um novo proprietário no banco de dados, com os seguintes dados: CPF 000000000-01, nome “Maria Lima”.

```
INSERT INTO PROPRIETARIO
VALUES (“00000000001”, “Maria Lima”);
```

(c) Escreva um comando SQL para excluir todas as cobranças dos apartamentos do proprietário “João Paulo” .

```
DELETE FROM COBRANCA
WHERE numApartamento IN
                        (SELECT numero
                        FROM apartamento a, proprietário p
                        WHERE a.cpf=p.cpf AND p.nome = “João Paulo”)
```

(d) Crie uma visão que contenha o nome, o CPF e o numero de apartamentos que cada proprietário possui.

```
CREATE VIEW V (Nome, CPF, NumApartamentos) AS
SELECT p.Nome, p.CPF, COUNT(*)
FROM Proprietario p, Apartamento a
WHERE p.CPF = a.CPF
GROUP BY p.Nome, p.CPF
```

(e) Escreva uma instrução SQL que retorna o numero do apartamento, o andar, a data e o valor das cobranças dos apartamentos cujo proprietário tem CPF 00000000001.

```
SELECT a.Numero, a.Andar, c.Data, c.Valor
FROM apartamento a, cobrança c
WHERE a.numero = c.numApartamento
AND a.cpf= “00000000001”
```

6. Considere o esquema relacional abaixo. As chaves primárias estão sublinhadas.

Pessoa (CIC,NomePess,Sexo,DataNasc,EsposaDeCIC)
(EsposaDeCIC) referencia Pessoa
(Tabela com dados dos clientes. Em caso de clientes casados em comunhão de bens, ambos estão registrados na tabela, sendo que as esposas contêm uma referência para seus maridos)

Imóvel (CodImovel,DescricaoImovel,PrecoImovel,AnoImovel)
(Tabela com dados dos imóveis a venda e vendidos)

Venda (CodImovel,CIC,DataVenda,CodCorretor)
(CodImovel) referencia Imovel
(CIC) referencia Pessoa
(CodCorretor) referencia Corretor
(Tabela que contém dados das vendas. Quando a venda for para um casal, apenas um deles aparece nesta tabela)

Corretor (CodCorretor,NomeCorretor)
(Tabela que contém dados dos corretores)

(a) Escreva um comando SQL para excluir a tabela *Venda*

```
DROP TABLE VENDA;
```

(b) Escreva um comando SQL para incluir uma coluna nova, chamada *Tipo, do tipo inteiro, na tabela Imóvel.*

```
ALTER TABLE Imovel
ADD Tipo INTEGER;
```

(c) Escreva o comando SQL necessário para excluir todas as vendas que foram realizadas no dia 10/10/2008.

```
DELETE FROM Venda
WHERE DataVenda = ”10/10/2008”
```

(d) Escreva um comando SQL para inserir uma pessoa chamada “Maria”, do sexo Feminino, nascida em “01/01/1980”, que é solteira e cujo CIC é “000000000-00”.

```
INSERT INTO PESSOA (CIC,NomePess,Sexo,DataNasc,EsposaDeCIC)
VALUES (“000000000-00”, “Maria”, “F”, “01/01/1980”, NULL)
```

(e) Crie uma visão que contenha o CIC, nome da pessoa, Cod do imóvel, descrição do imóvel das vendas que foram realizadas no dia 10/10/2008.

```
CREATE VIEW V (CIC, NomePess, CodImovel, DescricaoImovel) AS
SELECT p.CIC, p.NomePess, i.CodImovel, i.DescricaoImovel
FROM Pessoa p, Imovel i, Venda v
WHERE p.CIC=v.CIC
AND v.DataVenda=”10/10/2008”
```

(f) Escreva uma instrução SQL que retorna o nome das pessoas casadas que constam na tabela pessoa. A consulta deve retornar duas colunas: uma para o marido, outra para a esposa

```
SELECT p1.NomePess AS ESPOSA, p2.NomePess AS MARIDO
FROM Pessoa p1, Pessoa p2
WHERE p1.EsposaDeCIC=p2.CIC
```

(g) Escreva uma instrução SQL que retorna o número de imóveis comprados por cada pessoa.

```
SELECT p.CIC, COUNT(CodImovel) AS NUMIMOVEL
FROM Pessoa p, Venda v
WHERE p.CIC=v.CIC
GROUP BY v.CIC
```

(h) Escreva uma instrução SQL que retorne o nome do corretor, seu código e a média do preço dos imóveis que ele vendeu, mas apenas para os corretores que tenham vendido mais do que 5 imóveis.

```
SELECT c.CodCorretor, c.NomeCorretor, AVG(i.PrecoImovel) AS MEDIAPRECO
FROM Corretor c, Venda v, Imovel i
WHERE c.CodCorretor=v.CodCorretor
AND i.CodImovel=v.CodImovel
GROUP BY c.CodCorretor, c.NomeCorretor
HAVING COUNT(*) > 5
```

7. Considere o esquema relacional abaixo. As chaves primárias estão sublinhadas.

Pessoa(Cod,Nome,DataNasc)

Publicacao(Cod,Titulo,CodArea)
CodArea referencia Area (Cod)

Autor(CodAutor,CodPublicacao)
CodAutor referencia Pessoa (Cod),
CodPublicacao referencia Publicacao (Cod)

Revisao(CodRevisor,CodPublicacao,Nota)
CodRevisor referencia Pessoa (Cod),
CodPublicacao referencia Publicacao (Cod)

Area(Cod,Nome,CodAreaGenerica)
CodAreaGenerica referencia Area (Cod)

(a) Escreva um comando SQL para excluir a tabela *Autor*

```
DROP TABLE Autor;
```

(b) Escreva um comando SQL para incluir uma coluna nova, chamada *Ano, do tipo inteiro, na tabela Publicacao.*

```
ALTER TABLE Publicacao
ADD Ano INTEGER;
```

(c) Escreva o comando SQL necessário para excluir todas as revisões que tiveram nota 5.

```
DELETE FROM Revisao
WHERE Nota = 5
```

(d) Escreva um comando SQL para inserir uma pessoa chamada “Maria”, nascida em “01/01/1980”, com código 10.

```
INSERT INTO PESSOA (Codigo,Nome,DataNasc)
VALUES (10, “Maria”, “01/01/1980”)
```

(e) Crie uma visão que contenha o nome da pessoa e o título de todas as publicações das quais ela foi autora.

```
CREATE VIEW V (Nome, Titulo) AS
SELECT p.Nome, pub.Titulo
FROM Pessoa p, Publicacao pub, Autor a
WHERE p.Cod=a.CodAutor
```

(f) Escreva uma instrução SQL que retorna o número de revisões que cada pessoa efetuou.

```
SELECT p.Cod, COUNT(CodPublicacao) AS NumRevisoes
FROM Pessoa p, Revisao r
WHERE p.Cod=r.CodRevisor
GROUP BY p.Cod
```

(g) Escreva uma instrução SQL que retorne o nome da pessoa, seu código e a média das notas de suas revisões, mas apenas para as pessoas que tenham revisado mais do que 10 publicações.

```
SELECT p.Cod, p.Nome, AVG(r.Nota) AS NotaMedia
FROM Pessoa p, Revisao r
WHERE p.Cod=r.CodRevisor
GROUP BY p.Cod, p.Nome
HAVING COUNT(*) > 10
```

(h) Escreva uma instrução SQL que retorne o nome das pessoas que não trabalharam como revisores de nenhuma publicação.

```
SELECT p.Nome
FROM Pessoa p
WHERE p.Cod NOT IN (SELECT r.CodRevisor
FROM Revisao)
```