1) Refaça todas as consultas SQL vistas em aula.

```
CREATE DATABASE academico;
CREATE TABLE aluno(
id_aluno SERIAL NULL,
nome_aluno VARCHAR(50) NOT NULL,
cpf CHAR(11),
data_nascimento DATE,
media FLOAT DEFAULT 0.0,
CONSTRAINT aluno_pk PRIMARY KEY (id_aluno),
CONSTRAINT aluno_cpf_key UNIQUE (cpf)
);
CREATE TABLE professor (
id_professor INT NOT NULL,
nome_professor VARCHAR(50) NOT NULL,
sobrenome VARCHAR(50) NOT NULL,
area VARCHAR(20),
CONSTRAINT professor_pk PRIMARY KEY (id_professor)
);
CREATE TABLE disciplina(
id_disciplina INT NOT NULL,
nome_disciplina VARCHAR(30) NOT NULL,
carga_horaria INT NOT NULL,
id_professor INT,
CONSTRAINT disciplina_pk PRIMARY KEY (id_disciplina),
CONSTRAINT disciplina_fk_professor FOREIGN KEY (id_professor) REFERENCES
professor(id_professor)
);
CREATE TABLE matriculado(
id_disciplina INT NOT NULL,
id_aluno INT NOT NULL,
nota FLOAT,
CONSTRAINT matriculado_pk PRIMARY KEY (id_disciplina, id_aluno),
CONSTRAINT matriculado_fk_aluno FOREIGN KEY (id_aluno) REFERENCES
aluno(id_aluno),
CONSTRAINT matriculado_fk_disciplina FOREIGN KEY (id_disciplina)
REFERENCES disciplina(id_disciplina)
);
INSERT INTO aluno(nome_aluno, cpf, data_nascimento, media)
VALUES ('José', NULL , '1990-01-20', 0.0);

INSERT INTO aluno(nome_aluno, data_nascimento)
VALUES
('João', '1993-09-10'),
('Maria', '1989-05-15'),
('Ana', '1992-04-21');

SELECT * FROM aluno;

DELETE FROM aluno
```

```
WHERE id_aluno = 2;

UPDATE aluno
SET cpf='01234567890', data_nascimento='1991-12-23'
WHERE id_aluno = 3;

SELECT nome_aluno, data_nascimento FROM aluno;

SELECT a.nome_aluno AS aluno,
a.data_nascimento AS nascimento
FROM aluno AS a;

SELECT 102 * 30 as conta;

SELECT nome_aluno AS aluno,
media * 0.8 + 25 AS media
FROM aluno;

SELECT * FROM aluno
WHERE data_nascimento >= '1991-01-01';

SELECT * FROM disciplina
WHERE carga_horaria IN (40,60);

SELECT * FROM aluno
WHERE data_nascimento BETWEEN '1980-01-01' AND '1989-12-31';

SELECT * FROM aluno
WHERE data_nascimento >= '1990-01-01'
AND media > 80;

SELECT * FROM aluno
WHERE cpf IS NULL;

SELECT * FROM disciplina
ORDER BY carga_horaria DESC, nome_disciplina;

SELECT nome_professor AS nome
FROM professor
UNION
SELECT nome_aluno AS nome
FROM aluno;

SELECT id_aluno, AVG(nota) AS media
FROM matriculado
WHERE id_disciplina = 100 OR id_disciplina = 200
GROUP BY id_aluno
HAVING AVG(nota) > 80;

SELECT p.nome_professor,
```

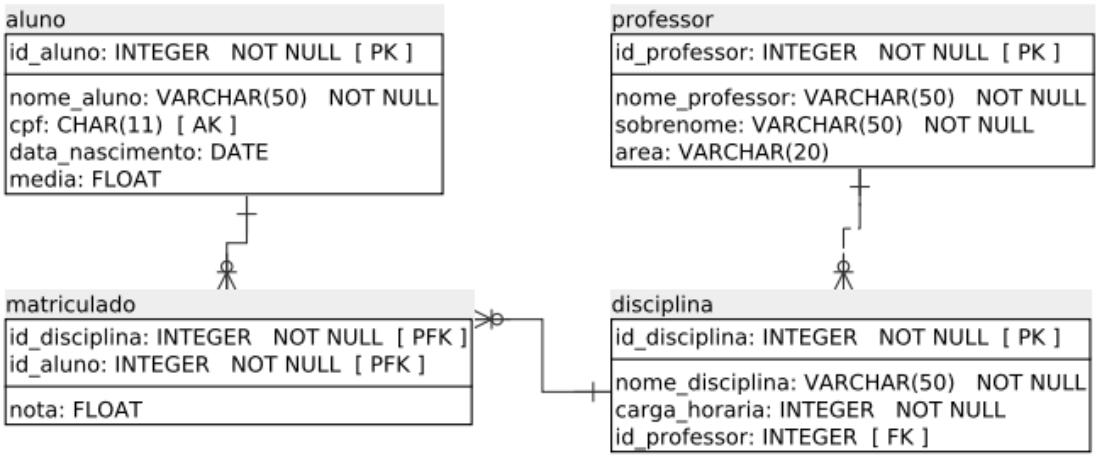
```
d.nome_disciplina
FROM professor AS p,
disciplina AS d
WHERE p.id_professor = d.id_professor;

SELECT a.nome_aluno,
d.nome_disciplina,
m.nota
FROM aluno AS a
JOIN matriculado AS m
ON a.id_aluno = m.id_aluno
JOIN disciplina AS d
ON d.id_disciplina = m.id_disciplina;

SELECT a.nome_aluno,
d.nome_disciplina,
m.nota
FROM aluno AS a
NATURAL JOIN matriculado AS m
NATURAL JOIN disciplina AS d;

SELECT d1.nome_disciplina, p.nome_professor
FROM disciplina AS d1,
professor AS p,
disciplina AS d2
WHERE d1.id_professor = p.id_professor
AND d1.id_professor = d2.id_professor
AND d2.id_disciplina <> d1.id_disciplina;

SELECT a.nome_aluno
FROM aluno AS a
WHERE NOT EXISTS(
SELECT 1
FROM matriculado AS m,
disciplina as D
WHERE m.id_disciplina = d.id_disciplina
AND d.id_professor = 30
AND m.id_aluno = a.id_aluno);
```



2) Considere agora o banco de dados Acadêmico mostrado na Figura 1. Escreva as instruções SQL para executar as seguintes ações:

a. Obter o nome do aluno, o nome da disciplina e a nota obtida pelo aluno na disciplina;

```
SELECT nome_aluno FROM aluno;
SELECT nome_disciplina FROM disciplina;
SELECT nota FROM matriculado;
```

b. Obter a quantidade de disciplinas ministradas por cada professor;

```
SELECT p.nome_professor,
d.nome_disciplina
FROM professor AS p,
disciplina AS d
WHERE p.id_professor = d.id_professor;
SELECT p.nome_professor,
d.nome_disciplina
FROM professor AS p
INNER JOIN disciplina AS d
ON p.id_professor = d.id_professor;
```

c. Obter os nomes completos de todos os professores com sua carga horária total;

```
SELECT p.nome_professor,
d.carga_horaria
FROM professor AS p
NATURAL JOIN disciplina AS d;
```

d. Obter a nota média para cada disciplina;

```
SELECT a.media,
d.nome_disciplina
FROM aluno AS a
```

```
NATURAL JOIN disciplina AS d;
```

e. Obter a maior e a menor nota para cada uma das disciplinas;

```
SELECT MAX(nota)
FROM matriculado;m
SELECT MIN(nota)
FROM matriculado;m
```

f. Obter as disciplinas que o aluno José está matriculado e que possuam pelo menos 2 alunos matriculados.

```
SELECT d.nome_disciplina,
COUNT(*)
FROM matriculado AS m,
disciplina AS d
WHERE m.id_disciplina = d.id_disciplina
AND EXISTS (
SELECT id_disciplina
FROM matriculado AS m2
WHERE m2.id_aluno = 1
AND m2.id_disciplina = d.id_disciplina)
GROUP BY d.nome_disciplina
HAVING COUNT(*) >= 2;
```

g. Obter os alunos matriculados nas disciplinas com carga horária maior ou igual a 60;

```
SELECT a.nome_aluno
FROM aluno AS a
WHERE carga_horaria >= 60;
```

h. Obter a média das notas de cada aluno em ordem decrescente pela média;

```
SELECT * FROM aluno
ORDER BY media DESC;
```

i. Atualizar a média dos alunos;

```
CREATE TEMPORARY TABLE media AS
SELECT id_aluno, AVG(nota) AS media
FROM matriculado
GROUP BY id_aluno;
UPDATE aluno AS a
SET media = m.media
FROM media AS m
WHERE a.id_aluno = m.id_aluno;
```

j. Obter os nomes dos alunos matriculado em disciplinas de professores da área de Computação;

```
SELECT a.nome_aluno
FROM aluno AS a
WHERE id_disciplina IN (
        SELECT id_professor
        FROM disciplina AS d
        WHERE d.nome_disciplina = computacao;
```

k. Obter a carga horária total de cada professor de acordo com as disciplinas ministradas;

```
SELECT id_professor, SUM(carga_horaria)
FROM disciplina
GROUP BY id_professor;
```

l. Obter a quantidade de alunos matriculados em cada disciplinas;

```
SELECT id_disciplina, SUM(id_aluno)
FROM matriculado
GROUP BY id_aluno;
```

m. Obter os alunos que cursaram mais de 100 horas de disciplinas;

```
SELECT a.nome_aluno
FROM aluno AS a
WHERE EXISTS(
SELECT 1
FROM disciplina AS d
WHERE carga_horaria > 100
AND a.id_aluno = m.id_aluno);
```

n. Obter o nome dos alunos, nome das disciplinas, nome dos professores das disciplinas e a nota obtida pelos alunos;

```
SELECT a.nome_aluno,
d.nome_disciplina,
p.nome_professor,
d.nome_disciplina,
m.nota
FROM aluno AS a,
disciplina AS d,
matricula AS m,
professor AS p
matriculado AS m
WHERE a.id_aluno = m.id_aluno
AND d.id_disciplina = m.id_disciplina
AND d.id_professor = p.id_professor;
```

o. Obter as disciplinas sem professor;

```
SELECT nome_disciplina
FROM disciplina AS d
WHERE d.id_professor IS NULL ;
```

~~o.~~p. Obter os professores sem disciplina;

```
SELECT id_professor
FROM disciplina AS d
WHERE d.id_disciplina IS NULL ;
```

~~p.~~q. Obter possíveis duplas combinando todos os nomes de alunos, mas sem combinar um aluno com ele mesmo;

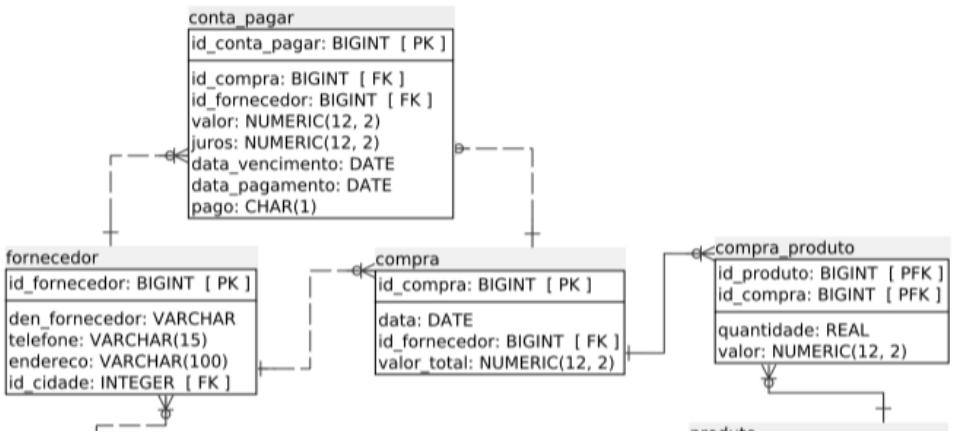
```
SELECT a1.nome_aluno, a2.nome_aluno
FROM aluno AS a1,
aluno AS a2
WHERE a1.id_aluno < a2.id_aluno;
```

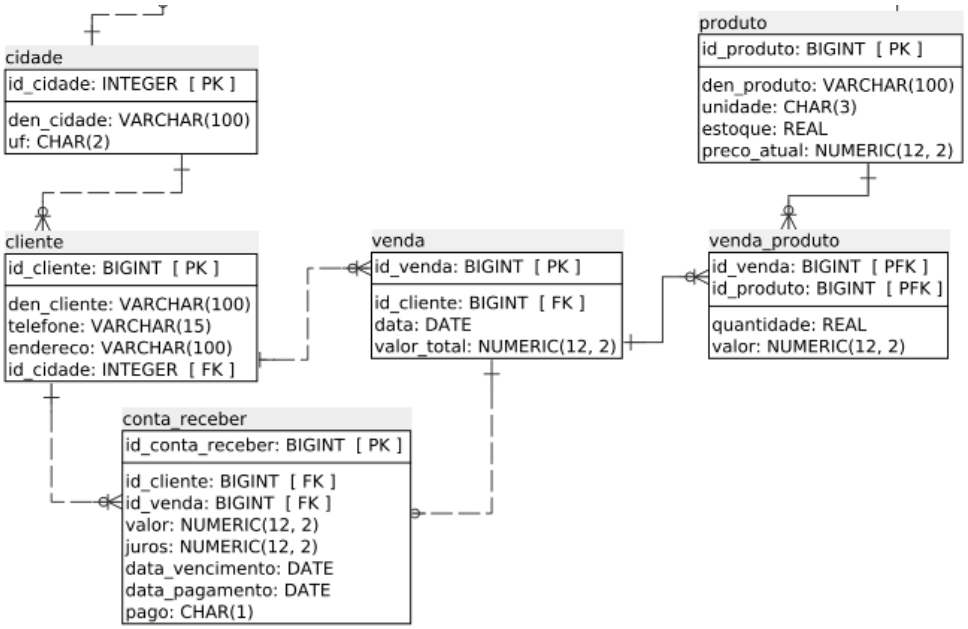
~~q.~~r. Obter as disciplinas sem nenhum aluno matriculado;

```
SELECT id_disciplina
FROM matriculado AS m
WHERE m.id_disciplina IS NULL ;
```

~~r.~~s. Matricular todos os alunos nas disciplinas sem nenhum aluno matriculado.

```
INSERT INTO matriculado
(id_disciplina, id_aluno, nota)
SELECT 101, id_aluno, 0 FROM aluno;
```





3) Considere o banco de dados de uma empresa de varejo cujo esquema lógico é apresentado na Figura 2. Escreva as instruções SQL para executar as seguintes ações:

a. Obter o valor total comprado de cada fornecedor;

b. O valor total vendido para cada cidade;

c. O valor total de cada produto vendido para cada cidade;

```
SELECT cli.id_cidade,
                cid.den_cidade,
                p.den_produto AS produto,
                SUM((vp.valor * vp.quantidade)) AS "valor total"
FROM cliente AS cli, venda AS v,cidade AS cid, venda_produto AS vp,produto as p
WHERE cli.id_cidade=cid.id_cidade AND v.id_cliente=cli.id_cliente AND
                vp.id_venda= v.id_venda AND vp.id_produto=p.id_produto
GROUP BY cli.id_cidade,cid.den_cidade,p.den_produto;
```

d. A quantidade, o valor total e o valor médio de cada produto comprado de cada estado (UF);

```
SELECT p.den_produto AS produto,
c.uf AS estado,
SUM(cp.quantidade) AS quantidade,
SUM(cp.quantidade * cp.valor) AS valor_total,
AVG(cp.quantidade * cp.valor) AS valor_medio
```

```
FROM cidade AS c,
fornecedor AS f,
compra AS co,
compra_produto AS cp,
produto AS p
WHERE c.id_cidade = f.id_cidade
AND f.id_fornecedor = co.id_fornecedor
AND co.id_compra = cp.id_compra
AND p.id_produto = cp.id_produto
GROUP BY p.den_produto, c.uf
ORDER BY produto, estado;
```

~~d.~~e. Listar as cidades com suas respectivas quantidades de cadastros (um cadastro pode ser um cliente ou um fornecedor);

~~e.~~f. Listar as contas a pagar vencidas até dezembro de 2011 e que não foram pagas;

~~f.~~g. Listar os fornecedores que possuem mais de 10 contas a pagar;

~~g.~~h. Listar o total devido por cada cliente;

~~h.~~i. Listar os 10 produtos com maior movimentação (considerando compras e vendas);

```
CREATE TEMPORARY TABLE quantidade AS
SELECT vp.id_produto, SUM(vp.quantidade) quant
FROM venda_produto AS vp
GROUP BY vp.id_produto
UNION
SELECT cp.id_produto, SUM(cp.quantidade) quant
FROM compra_produto AS cp
GROUP BY cp.id_produto;

SELECT q.id_produto, SUM(q.quant) AS quant
FROM quantidade AS q
GROUP BY q.id_produto
ORDER BY quant DESC;
```

~~i.~~j. Listar o faturamento (vendas) mensal de todos os meses;

```
SELECT vp.id_produto, SUM(vp.quantidade) quant
FROM venda_produto AS vp
```

```
GROUP BY vp.id_produto
UNION
SELECT cp.id_produto, SUM(cp.quantidade) quant
FROM compra_produto AS cp
GROUP BY cp.id_produto
ORDER BY quant DESC
LIMIT 10;
```

~~j.~~k. Crie um campo de estoque mínimo na tabela produto e atualize com 50% da média mensal de venda do produto;

~~k.~~l. Listar os produtos que foram vendidos, mas não foram comprados em janeiro de 2011;

~~l.~~m. Listar os produtos que foram comprados em 2010, sem repetições;

~~m.~~n. Listar as vendas de 2011 contendo mais de 5 produtos que ainda não foram pagas;

~~n.~~o. Listar o total vendido e o total comprado de cada produto (pode acontecer de um produto ter sido comprado e não ter sido vendido e vice-versa);

~~o.~~p. Listar o mês e ano com a maior quantidade de vendas de cada produto.