Prova P2

1) Explique com detalhes como é feito o projeto de um banco de dados, suas etapas e a importância de cada uma delas. Em seguida, descreva o modelo entidade relacionamento e defina os seguintes termos (10 Pontos):

Para criação de um banco de dados, é necessário antes levantar os requisitos do projeto, depois criar um modelagem conceitual (MER). Após esses passos, podemos criar uma modelagem lógica para um SGBD e depois disso já criar o que foi mapeado nos passos anteriores.

- **Visões:** é uma consulta salva que pode ser tratada como uma tabela, permitindo simplificar consultas complexas, fornecendo segurança e encapsulando a lógica da consulta.
- **Tuplas:** seria um registro da tabela de dados.
- **Cardinalidade mínima:** define o número mínimo de instâncias de uma entidade que pode se relacionar com outra entidade. Sendo a cardinalidade mínima 0 ou 1.
- **Cardinalidade máxima:** define um número máximo de instâncias de uma entidade que pode se relacionar com outra entidade. Sendo a cardinalidade máxima 1 ou N.

2) Você foi contratado para construir um banco de dados de uma empresa que faz massas para vender no atacado. Com base no cenário descrito, construa o modelo lógico que irá representar o banco (10 Pontos).

- **Funcionários (Tipo, CPF, Nome, Telefone, Especialidade ou Veículo)**
  - Tipo: Enum('Padeiro', 'Entregador')
  - CPF: Primária
  - Nome
  - Telefone
  - Especialidade (se Tipo = 'Padeiro')

    - Veículo (se Tipo = 'Entregador')
- **Massas (ID, Nome, Tipo, Data)**
    - ID: Primária
    - Nome
    - Tipo
    - Data
- **Clientes (ID, Nome, Endereço, Telefone)**
    - ID: Primária
    - Nome
    - Endereço
    - Telefone
- **Vendas (ID, ClienteID, MassaID, Data)**
    - ID: Primária
    - ClienteID: Estrangeira
    - MassaID: Estrangeira
    - Data

3) Considerando o modelo o MER estendido a seguir , faça o código SQL para criar as Tabelas Batalha e Item. Em seguida descreva os comandos necessários para realizar as seguintes consultas (25 Pontos).

- **CREATE TABLES**

```
CREATE TABLE Heroi (
    id INT PRIMARY KEY AUTO_INCREMENT,
    nome VARCHAR(30),
    superpoder ENUM('outro não sei'),
    nivel FLOAT
)

CREATE TABLE Vilao (
    id INT PRIMARY KEY AUTO_INCREMENT,
```

```
    nome VARCHAR(30),
    superpoder ENUM('outro não sei'),
    nivel FLOAT
)

CREATE TABLE Batalha (
    id INT PRIMARY KEY AUTO_INCREMENT,
    heroi_id INT,
    vilao_id INT,
    vencedor VARHCAR(35),
    resultado FLOAT,

    FOREIGN KEY (heroi_id) REFERENCES Heroi(id),
    FOREIGN KEY (vilao_id) REFERENCES Vilao(id)
)

CREATE TABLE Item (
    id INT PRIMARY KEY AUTO_INCREMENT,
    tipo ENUM('não sei', 'eu sei'),
    heroi_id INT,
    poder INT,

    FOREIGN KEY (heroi_id) REFERENCES Heroi(id)
)
```

- **CONSULTAS**

A) Quais os itens do herói chamado Batman;

```
SELECT
    Heroi.nome,
    Item.tipo,
    Item.poder

FROM Heroi Heroi

INNER JOIN Item Item
    ON Heroi.id = Item.heroi_id
```

```
WHERE Heroi.nome = 'Batman'
```

B) Quais heróis batalharam com um vilão chamado Coringa;

```
SELECT
    Heroi.nome,
    Vilao.nome

FROM Heroi Heroi

INNER JOIN Batalha Batalha
    ON Heroi.id = Batalha.heroi_id

INNER JOIN Vilao Vilao
    ON Batalha.vilao_id = Vilao.id

WHERE Vilao.nome = 'Coringa'

GROUP BY
    Heroi.nome,
    Vilao.nome
```

C) Qual o id da batalha, o vilão e o herói que participaram de uma batalha onde o resultado foi igual a alguma batalha de resultado 10 (aninhada);

```
SELECT
    Batalha.id,
    Vilao.nome,
    Heroi.nome

FROM Batalha Batalha, Vilao Vilao, Heroi Heroi

WHERE Batalha.heroi_id = Heroi.id
    AND Batalha.vilao_id = Vilao.id
    AND Batalha.resultado = 10
```

D) Qual a média do nível entre os vilões;

```
SELECT AVG(Vilao.nivel) FROM Vilao Vilao
```

E) Qual o nível máximo entre os heróis:

```
SELECT MAX(Heroi.nivel) FROM Heroi Heroi;
```

F) Mostre o nome do Vilão, cujo resultado da batalha seja pior que algum herói chamado Batman:

```
SELECT
    Vilao.nome

FROM Vilao Vilao

INNER JOIN Batalha
    ON Vilao.id = Batalha.vilao_id

WHERE Batalha.resultado < ANY (
    SELECT
        Batalha.resultado

    FROM Batalha Batalha

    INNER JOIN Heroi Heroi
        ON Batalha.heroi_id = Heroi.id

    WHERE Heroi.nome = 'Batman'
)
```

G) Mostre o nome e o número de batalhas do Herói que batalhou mais de uma vez:

```
SELECT
    Heroi.nome,
    COUNT(1) AS quantidade_batalha
```

```
FROM Batalha Batalha

INNER JOIN Heroi Heroi
    ON Batalha.heroi_id = Heroi.id

GROUP BY
    Heroi.id

HAVING COUNT(1) > 1
```

4) **O que são visões e por que são importantes? Faça 3 visões envolvendo as 3 tabelas abaixo e explique o propósito de cada uma delas (25 Pontos).**

Visões são consultas salvas que representam tabelas e são importantes porque simplificam consultas complexas, melhoram a segurança ao restringir o acesso direto às tabelas e encapsulam as lógicas facilitando assim a manutenção de consultas repetitivas.

1. Visão para listar todos os prontuários com detalhes de pacientes médicos:

```
CREATE VIEW vw_prontuarios_detalhados AS
SELECT
    Prontuarios.id AS prontuario_id,
    Prontuarios.dia AS prontuario_dia,
    Prontuarios.tipo AS prontuario_tipo,

    Pacientes.nome AS paciente_nome,
    Pacientes.endereco AS paciente_endereco,

    Medicos.crm AS medico_crm,
    Medicos.nome AS medico_nome,
    Medicos.especialidade AS medico_especialidade

FROM Prontuarios Prontuarios

INNER JOIN Pacientes Pacientes
    ON Prontuarios.id_paciente = Pacientes.id_paciente
```

```
INNER JOIN Medicos Medicos
    ON Prontuarios.crm = Medicos.crm
```

2. Visão para listar pacientes com prontuários específicos:

```
CREATE VIEW vw_pacientes_prontuarios_consulta AS
SELECT
    Pacientes.id_paciente,


FROM Pacientes Pacientes

INNER JOIN Prontuarios Prontuarios
    ON Pacientes.id_paciente = Prontuarios.id_paciente

WHERE Prontuarios.tipo = 'consulta'
```

3. Visão para listar médicos e o número de prontuários associados a eles:

```
CREATE VIEW vw_medicos_prontuarios AS
SELECT
    Medicos.crm,
    Medicos.nome AS medico_nome,

    COUNT(Prontuarios.id_paciente) AS quantidade_prontuarios

FROM Medicos

INNER JOIN Prontuarios
    ON Medicos.crm = Prontuarios.crm

GROUP BY
    Medicos.crm,
    Medicos.nome
```