Resumo, prova 3 BD

Estrutura básica de um comando SELECT

SELECT <DISTINCT> <lista de atributos>
FROM <nome_tabela> <JOIN <tabela> ON <(atributo=atributo)>>
WHERE <condição>
GROUP BY <atributo> HAVING <condição>
ORDER BY <atributo> <ASC/DESC>

Operadores

- <>, !=, =, >, <, >=, <=
  - Diferente, diferente, igual, maior/menor que, maior/menor ou igual que.
- AND, OR, NOT
  - E, ou, negação.
- <NOT> BETWEEN
  - Substituem os operadores <= e >=.
    WHERE <nome_atributo> <NOT> BETWEEN <valor1> AND <valor2>;
- <NOT> LIKE
  - Permite que uma string seja comparada com um padrão.
  - O '%' representa que a partir dele, pode haver qualquer caractere.
  - O '_' representa que naquela posição, pode haver qualquer caractere.
  - WHERE <atributo> <NOT> LIKE 'M_R%';
- <NOT> IN
  - Procuram dados que estão (ou não) contidos em um dado conjunto de valores. Faz consulta relacionando tabelas sem junção.
  - WHERE <atributo> <NOT> IN <conjunto de valores>;
- IS <NOT> NULL
  - Identificam se o atributo tem valor nulo ou não.
  - WHERE <atributo> IS <NOT> NULL;

Expressões aritméticas

- **É correto ter expressões aritméticas escritas diretamente na cláusula SELECT.**
- **É correto ter expressões aritméticas escritas diretamente na cláusula FROM.**

O FROM é usado para especificar as tabelas a serem consultadas pela consulta, então não é espaço para realizar cálculos ou transformações nos dados.

- **É correto ter expressões aritméticas escritas diretamente na cláusula WHERE.**
- **É correto ter expressões aritméticas escritas diretamente na cláusula ORDER BY.**

O ORDER BY pode usar um valor para ordenar baseado na ordem do campo pelo SELECT. Por exemplo, se temos 3 campos no SELECT e botamos o número 2 no ORDER BY, será interpretado que deve-se fazer o ORDER BY pelo segundo campo.

- **É correto ter expressões aritméticas escritas diretamente na cláusula GROUP BY.**

O GROUP BY pode usar um valor para agrupar baseado na ordem do campo pelo SELECT. Por exemplo, se temos 3 campos no SELECT e botamos o número 2 no GROUP BY, será interpretado que deve-se fazer o GROUP BY pelo segundo campo.

- **É correto ter expressões aritméticas escritas diretamente na cláusula HAVING.**

ORDER BY

- ORDER BY <atributo> <ASC/DESC>
    - Vem depois de todos os outros trechos.
      Ordenar o resultado alfabeticamente ou numericamente.
    - ASC é o padrão, pode ser omitido. DESC é o contrário.
      Inserir mais de um atributo ordenador corresponde a critérios n-ários.
- **Para ordenar o resultado, deve-se repetir todos os campos do SELECT no ORDER BY.**

DISTINCT

- SELECT DISTINCT <atributo> FROM <tabela>
    - Elimina tuplas duplicadas no resultado de uma consulta.

GROUP BY e HAVING

- SELECT <atributo de agrupamento>, <funções de agregação>
  FROM <tabela>
  GROUP BY <atributos de agrupamento>
  HAVING <condição2>;
- **GROUP BY:** Organiza a seleção de dados em grupos.
    - Todos os atributos do GROUP BY, ou seja, os atributos que serão usados como condição para o agrupamento, devem aparecer no SELECT, com exceção das funções de agregação.
- **HAVING:** Agrupamento com condição, 'WHERE' de agrupamento.
    - Vem depois do GROUP BY e antes do ORDER BY.
- **Para gerar agrupamentos, todos os campos sem função de agregação no SELECT devem ser repetidos no GROUP BY.**
- **Para gerar agrupamentos, podem-se usar campos no GROUP BY que não aparecem no SELECT.**
- **É correto ter consultas com GROUP BY e sem HAVING.**
- **É correto ter consultas com HAVING e sem GROUP BY.**

Não tem como ter uma filtragem sobre um agrupamento sem o agrupamento. Já que o agrupamento é feito pelo GROUP BY, só existe HAVING com o GROUP BY.

- **É correto ter consultas com HAVING e sem WHERE.**

ALIAS

- Para substituir nomes de tabelas em comandos SQL.
- São definidos na cláusula FROM.
- Também pode ser usado para renomear uma coluna, como "COUNT(*) AS 'QUANTIDADE'";
    - Através do alias você seria capaz de usar essa coluna como argumento de uma função de agregação, como no caso da consulta "Exibir os CPF e os nomes dos pesquisadores que escrevem mais artigos do que a média dos pesquisadores da sua instituição."
- Facilita compreensão em casos de junção de tabelas.
- SELECT **A**.Nome FROM Departamento **A** WHERE **A**.Numero = 15;

Funções de agregação

- São disparados a partir do SELECT.
- SELECT <atributo> <FUNÇÃO<(atributo)>> FROM <tabela>;
- Podem ser chamados APENAS no campo SELECT ou HAVING.
- AVG(), MIN(), MAX(), COUNT(), SUM()
- **É correto ter consultas com funções de agregação que têm outras funções de agregações como parâmetro.**

Não pode botar uma função de agregação dentro de outra, como AVG(COUNT()) ou MIN(COUNT()). Se for necessário algo do tipo, tem que usar subconsultas para obter o valor e então passar para a função de agregação.

Junção

- Obter informação de múltiplas tabelas.
    - Os nomes das tabelas envolvidas devem ser citados na cláusula FROM.
    - Uso de qualificadores de nomes para evitar ambiguidades.
- SELECT A.<atributo>, B.<atributo2>
  FROM <tabela> A, <tabela2> B
  WHERE A.<cod> = B.<cod>;
- Pode-se utilizar as cláusulas (NOT) LIKE, (NOT) IN, IS (NOT) NULL misturadas com operadores AND, OR, NOT (cláusula WHERE), além de ORDER BY e GROUP BY.

- **Tipos de junção:**
    - **INNER JOIN**
      Equivalente a uma junção natural;
      Retorna todas as linhas das tabelas onde a condição join é encontrada;
      Interseção entre as tabelas da esquerda e da direita.
      FROM <tabela> INNER JOIN <tabela2> ON <cod = cod2> ;
    - **LEFT JOIN**
      A tabela principal é a que está antes (à esquerda) da cláusula LEFT JOIN, essa é a tabela principal que definirá as linhas na tabela resultado, enquanto que a tabela que está depois (à direita) é uma tabela secundária;
      Retorna todos os registros da tabela_esquerda e apenas os registros da tabela_direita que se conectam aos da tabela_esquerda.
      FROM <tabela> LEFT JOIN <tabela2> ON <cod = cod2> ;
      ou WHERE cod = cod2 (+); <sintaxe implícita>
    - **RIGHT JOIN**
      A tabela principal é a que está depois (à direita) da cláusula RIGHT JOIN, essa é a tabela principal que definirá as linhas na tabela resultado, enquanto que a tabela que está antes (à esquerda) é uma tabela secundária;
      Retorna todos os registros da tabela_direita e apenas os registros da tabela_esquerda que se conectam aos da tabela_direita.
      FROM <tabela> RIGHT JOIN <tabela2> ON <cod = cod2> ;
      ou WHERE cod (+) = cod2; <sintaxe implícita>
    - **FULL JOIN**
      Retorna também todas as linhas das tabelas à esquerda e à direita com NULL

nos locais onde a condição não é satisfeita (não há correspondência entre as tabelas).
FROM <tabela> FULL JOIN <tabela2> ON <cod = cod2> ;

- **As operações de junção interna e interseção são equivalentes.**

A interseção é uma operação de conjunto, então parte do pressuposto que sintaticamente deve haver relações equivalentes em domínio. Os campos do primeiro SELECT devem ser equivalentes ao primeiro campo do segundo SELECT, com o mesmo tipo de dado, quantidade de colunas e ordem de visualização.
Já a junção não tem esse pressuposto, então pode ter relações não equivalentes em domínio.

- **Todas as vezes que existem duas ou mais tabelas no FROM, tem-se uma operação de junção.**

Duas ou mais tabelas no FROM é apenas o produto cartesiano entre as tabelas, enquanto a junção é uma seleção sobre produto cartesiano, então falta o WHERE para que seja feita a junção de fato.

- **Toda junção é feita sobre as chaves primárias e estrangeiras das tabelas envolvidas.**

Podemos também ter joins que não necessariamente envolvem as chaves primárias e estrangeiras. O WHERE pode ter qualquer campo, não obrigatoriamente uma chave, apesar disso ser mais comum e fazer mais sentido para juntar uma tabela e outra (PK = FK). Mas também pode fazer outro tipo de comparação, como aluno.idade > professor.idade.

- **Toda junção interna retorna apenas as linhas que satisfazem a condição de junção.**
- **Toda junção externa retorna apenas as linhas da tabela à esquerda (Left Join), à direita (Right Join) ou de ambas (Full Join) que não correspondem à condição de junção.**

Ele retorna obrigatoriamente os dados da junção interna mais as linhas que não atendem a condição de junção. Para saber quais são os que não atendem vai depender do tipo de JOIN usado, se for LEFT, RIGHT ou FULL.

Subconsultas

- Resultado de uma consulta é utilizado por outra consulta;
- Resultado do SELECT mais interno é utilizado pelo mais externo para obter o resultado final;
- ORDER BY não pode ser usado em subconsulta;
- Os atributos especificados no SELECT devem estar associados às tabelas listadas na cláusula FROM da mesma;
    - É possível referir-se a uma tabela da cláusula FROM da consulta mais externa utilizando Qualificadores de atributos

- **Tipos de subconsultas**
    - Escalar = retorna um único valor
    - Linha = retorna várias colunas, mas apenas uma única linha
    - Tabela = retornam uma ou mais colunas e múltiplas linhas

- **EXISTS e NOT EXISTS (Semijoin e Antijoin)**
    - Projetadas para uso apenas com subconsultas;
    - Sempre são mais eficientes que operações de join;

    - IMPORTANTE: Só é possível quando precisa exibir informação de 1 tabela manipulando outras;
    - EXISTS retorna TRUE se existe pelo menos uma linha produzida pela subconsulta;
    - Logo retorna FALSE se a subconsulta produz uma tabela vazia, ou seja, nenhuma linha é produzida pela subconsulta.

- **Para exibir os nomes apenas dos pesquisadores que já escreveram artigos, é eficiente escrever um semi-join.**

Como não temos o nome do pesquisador na tabela escreve, precisamos fazer um JOIN entre escreve e pesquisador. Para otimizar esse processo o SEMIJOIN pode ser usado para partir para o próximo assim que encontrasse um artigo escrito.

- **Para exibir as siglas apenas dos eventos que ainda não têm artigos publicados, é eficiente escrever um anti-join.**

Já que evento não tem chave estrangeira de artigo, precisamos manipular as duas tabelas. A melhor forma de fazer isso nesse caso é com o ANTIJOIN, porque basta encontrar uma tupla que atenda a condição e a procura é encerrada e passa para o próximo.

```
SELECT E.SIGLA FROM EVENTO E
WHERE NOT EXISTS (SELECT * FROM ARTIGO A
                  WHERE A.COD = E.COD);
```

ou

```
SELECT E.SIGLA FROM EVENTO E
WHERE E.COD NOT IN (SELECT A.COD FROM ARTIGO A
                    WHERE A.COD IS NOT NULL);
```

- **Para exibir os nomes apenas dos pesquisadores que já escreveram artigos, não é necessário escrever nenhum tipo de join.**

Pode usar JOIN, mas não é obrigatório pois também podemos usar subconsulta ou SEMIJOIN.

```
SELECT NOME FROM PESQUISADOR
WHERE CPF IN (SELECT CPF FROM ESCREVE);
```

ANY, SOME e ALL

- Cláusulas usadas com subconsulta que produzem uma única coluna de números.
- **ANY/SOME**
    - Pelo menos um valor
    - Com = é equivalente ao IN
    - WHERE atributoN > ANY(...) == WHERE atributoN > SOME(...)
    - Significa que atributoN deve ser > pelo menos um dos valores em (...).
- **ALL**
    - Todos os valores
    - Com <> é equivalente ao NOT IN
    - WHERE atributoN > ALL (...)
    - Significa que o atributoN deve ser maior que todos os valores em (...)

- **O resultado de uma subconsulta do tipo escalar pode ser testado usando operadores como >, < ou =.**
- **O resultado de uma subconsulta do tipo linha pode ser testado usando operadores como >, < ou =.**

Nesse caso, já que temos mais de uma coluna, deve haver a comparação de vários valores com vários valores (compara todos os valores de todas as colunas ou de algumas colunas em específico). Mesmo sendo vários valores, ainda é elemento a elemento, só feito várias vezes.

- **O resultado de uma subconsulta do tipo tabela pode ser testado usando operadores como >, < ou =.**

Esse tipo não permite essas comparações pois agora nós temos várias colunas e várias linhas, então não temos como fazer elemento a elemento. Para isso vamos usar o ANY, ALL IN ou NOT IN.

Consultas SQL

CONSULTA PARA EXIBIR TODOS OS DADOS DO BANCO:

```
SELECT *
FROM PESQUISADOR P
FULL JOIN ESCREVE ESC ON (ESC.CPF = P.CPF)
FULL JOIN ARTIGO A ON (ESC.MAT = A.MAT)
FULL JOIN EVENTO E ON (A.COD = E.COD);
```

- **Exibir os CPF dos pesquisadores que escrevem mais artigos do que a média geral.**

```
SELECT Q1.CPF
FROM (SELECT E.CPF, COUNT(*) AS QTD1
        FROM ESCREVE E
        GROUP BY E.CPF) Q1
WHERE Q1.QTD1 > (SELECT AVG(QTD2)
                    FROM (SELECT E2.CPF, COUNT(*) AS QTD2
                            FROM ESCREVE E2
                            GROUP BY E2.CPF));

SELECT P.CPF FROM PESQUISADOR P
WHERE (SELECT COUNT(*) FROM ESCREVE E1 WHERE E1.CPF = P.CPF) >
  (SELECT AVG(QTD) FROM (
  SELECT E.CPF, COUNT(*) AS QTD FROM ESCREVE E GROUP BY E.CPF
  ));
```

- **Exibir os títulos dos artigos que têm a mesma nota e o mesmo idioma do artigo 1234.**

```
SELECT A.TITULO
FROM ARTIGO A
WHERE (A.NOTA, A.IDIOMA) = (SELECT A2.NOTA, A2.IDIOMA
                            FROM ARTIGO A2
                            WHERE A2.MAT = 1234);
```

- **Exibir os CPF e os nomes dos pesquisadores que escrevem mais artigos do que a média dos pesquisadores da sua instituição.**

```
SELECT P.CPF, P.NOME
FROM PESQUISADOR P
WHERE
        (SELECT COUNT(*)
        FROM ESCREVE E
        WHERE E.CPF = P.CPF) >=
  (SELECT AVG(CONTAGEM)
  FROM
  (SELECT COUNT(*) AS CONTAGEM
```

```
                FROM ESCREVE E1
                WHERE E1.CPF IN
                (SELECT P1.CPF
                        FROM PESQUISADOR P1
                        WHERE P.INSTITUICAO = P1.INSTITUICAO)
                GROUP BY E1.CPF))
```

- **Exibir os CPF dos pesquisadores que escreveram o artigo com título 'XPTO'.**

```
SELECT E.CPF
FROM ESCREVE E, ARTIGO A
WHERE E.MAT = A.MAT
AND A.TITULO = "XPTO";
ou
SELECT E.CPF
FROM ESCREVE E
WHERE E.MAT = (SELECT A.MAT
                FROM ARTIGO A
                WHERE A.TITULO = "XPTO");
```

- **Exibir os nomes dos pesquisadores que escreveram o artigo com título 'XPTO'.**

```
SELECT P.NOME
FROM PESQUISADOR P, ESCREVE E, ARTIGO A
WHERE P.CPF = E.CPF AND E.MAT = A.MAT AND A.TITULO = "XPTO";
ou
SELECT P.NOME
FROM PESQUISADOR P
WHERE P.CPF = (SELECT E.CPF
                FROM ESCREVE E
                WHERE E.MAT = (SELECT A.MAT
                                FROM ARTIGO A
                                WHERE A.TITULO = "XPTO"));
```

- **Exibir os títulos dos artigos publicados no evento cujo código é 1234.**

```
SELECT A.TITULO
FROM ARTIGO A
WHERE A.COD = "1234";
```

- **Exibir os títulos dos artigos publicados por código do evento.**

O código do evento já se encontra na tabela do artigo através de chave estrangeira, então não precisa de JOIN ou subconsulta. Para fazer o agrupamento por código de evento vamos ter só que usar um GROUP BY.

- **Exibir apenas os títulos dos artigos publicados.**

```
SELECT A.TITULO
FROM ARTIGO A
WHERE A.COD IS NOT NULL;
```

- **Exibir apenas os títulos dos artigos publicados no evento cuja sigla é 'SBBD'.**

```
SELECT A.TITULO
FROM ARTIGO A, EVENTO E
WHERE A.COD = E.COD AND E.SIGLA = "SBBD";
```

- **Exibir as siglas dos eventos com artigos publicados.**

```
SELECT E.SIGLA
FROM EVENTO E
```

```
WHERE EXISTS (SELECT *
                FROM ARTIGO A
                WHERE A.COD = E.COD);
```

ou

```
SELECT E.SIGLA
FROM EVENTO E
WHERE E.COD IN (SELECT A.COD
                FROM ARTIGO A
                WHERE A.COD IS NOT NULL);
```

- **Exibir os CPF dos pesquisadores que escreveram artigos.**

Não precisa fazer essas coisas pois a tabela escreve já possui todas as informações necessárias, é só dar um SELECT no CPF em escreve, já que os pesquisadores que estão lá necessariamente escreveram artigos.

- **Exibir os artigos que têm mais de 5 autores.**

```
SELECT E.MAT, COUNT(*)
FROM ESCREVE E
GROUP BY E.MAT HAVING COUNT(*) > 5.
```

- **Exibir os eventos com mais de 10 artigos em inglês.**

```
SELECT E.DOG, COUNT(*)
FROM ARTIGO A INNER JOIN EVENTO E ON A.COD = E.COD
WHERE A.IDIOMTA = “INGLÊS”
GROUP BY E.COD HAVING COUNT(*) > 10;
```

- **Exibir nome dos pesquisadores com artigos publicados no evento 'SBBD21'.**

```
SELECT P.NOME
FROM PESQUISADOR P
WHERE P.CPF IN (
  SELECT ESC.CPF
  FROM ESCREVE ESC
  WHERE ESC.MAT IN (
        SELECT A.MAT
        FROM ARTIGO A
        WHERE A.COD IN(
                SELECT E.COD
                FROM EVENTO E
                WHERE E.SIGLA = 'SBBD21')));
```

- **Exibir sigla dos eventos nos quais 'RUI' publicou artigo.**

```
SELECT E.SIGLA FROM EVENTO E
WHERE E.COD IN (
  SELECT A.COD FROM ARTIGO A
  WHERE A.MAT IN (
        SELECT ESC.MAT FROM ESCREVE ESC
        WHERE ESC.CPF IN (
                SELECT P.CPF FROM PESQUISADOR P
                WHERE P.NOME = 'RUI')));
```

- **Exibir nome dos pesquisadores com artigos publicados em eventos com média maior que a nota do artigo.**

```
SELECT P.NOME FROM PESQUISADOR P
WHERE P.CPF IN (
  SELECT ESC.CPF FROM ESCREVE ESC
```

```
   WHERE ESC.MAT IN (
         SELECT A.MAT FROM ARTIGO A
         WHERE A.NOTA < (
               SELECT AVG(A1.NOTA) FROM ARTIGO A1
               WHERE A1.COD = A.COD
               GROUP BY A.COD)));
```

- **Para cada pesquisador, projetar seu nome e os nomes dos pesquisadores mais novos do que ele.**

```
SELECT P.NOME, P1.NOME
FROM PESQUISADOR P, PESQUISADOR P1
WHERE P.NASCIMENTO < P1.NASCIMENTO;

SELECT P1.NOME, P2.NOME
FROM PESQUISADOR P1 INNER JOIN PESQUISADOR P2 ON (P1.NASCIMENTO < P2.NASCIMENTO);
```

- **Projetar a média de notas de artigos por evento (sigla).**

```
SELECT A.COD, AVG(A.NOTA)
FROM ARTIGO A
WHERE A.COD IS NOT NULL
GROUP BY A.COD;

SELECT E.SIGLA, AVG (A.NOTA) AS MEDIA_NOTA
FROM ARTIGO A INNER JOIN EVENTO E ON (A.COD = E. COD)
```

- **Projetar as siglas dos eventos cujas médias das notas dos artigos são maiores do que 8.**

```
SELECT E.SIGLA, AVG(A.NOTA)
FROM EVENTO E, ARTIGO A
WHERE E.COD = A.COD
GROUP BY E.SIGLA HAVING AVG(NOTA) > 8;
```

- **Projetar os nomes dos pesquisadores que publicaram mais de 3 artigos.**

```
SELECT P.NOME, COUNT(*)
FROM PESQUISADOR P JOIN ESCREVE E ON (P.CPF = E.CPF)
GROUP BY P.NOME HAVING COUNT(*) > 3;
```

- **Projetar os títulos dos artigos com nota acima da média.**

```
SELECT A.TITULO, A.NOTA
FROM ARTIGO A
WHERE A.NOTA > (
   SELECT AVG(NOTA)
   FROM ARTIGO)
```

- **Projetar os nomes dos pesquisadores que possuem artigos sem nota.**

```
SELECT P.NOME FROM PESQUISADOR P
WHERE P.CPF IN (
   SELECT E.CPF FROM ESCREVE E
   WHERE E.MAT IN (
         SELECT A.MAT FROM ARTIGO A
         WHERE A.NOTA IS NULL));
```

```
SELECT P.NOME
FROM PESQUISADOR P INNER JOIN ESCREVE E ON (P.CPF = E.CPF)
        INNER JOIN ARTIGO A ON (A.MAT = E.MAT)
WHERE A.NOTA IS NULL;
```

- **Projetar os nomes dos pesquisadores que nasceram no mesmo ano e trabalham na mesma instituição do pesquisador '1111'.**

```
SELECT NOME
FROM PESQUISADOR
WHERE (INSTITUICAO,TO_CHAR(NASCIMENTO,'YY')) = (SELECT
INSTITUICAO,TO_CHAR(NASCIMENTO,'YY')
                FROM PESQUISADOR
                WHERE CPF = '1111');
```

- **Por instituição, projetar o nome e a instituição dos pesquisadores mais velhos.**

```
SELECT P.NOME, P.INSTITUICAO
FROM PESQUISADOR P
WHERE P.NASCIMENTO IN (
  SELECT MIN(P1.NASCIMENTO)
  FROM PESQUISADOR P1
  GROUP BY P1.INSTITUICAO);
```

- **Projetar os títulos dos artigos com nota maior do que todos os artigos do evento 'SBBD'.**

```
SELECT A.TITULO, A.NOTA
FROM ARTIGO A
WHERE A.NOTA > (
  SELECT MAX(A1.NOTA)
  FROM ARTIGO A1 JOIN EVENTO E ON (E.COD = A1.COD)
  WHERE E.SIGLA = 'SBBD21');

SELECT TITULO, NOTA
FROM ARTIGO
WHERE NOTA > ALL
  (SELECT NOTA
  FROM ARTIGO A INNER JOIN EVENTO E ON (A.COD = E.COD)
  WHERE  E.SIGLA = 'SBBD21');
```

- **Projetar os títulos dos artigos com nota acima da média geral, mostrando quanto a nota está acima da média.**

```
SELECT A.TITULO, A.NOTA - (SELECT AVG(A1.NOTA) FROM ARTIGO A1)
FROM ARTIGO A
WHERE A.NOTA > (
  SELECT AVG(A1.NOTA)
  FROM ARTIGO A1);
```

- **Projetar o percentual de pesquisadores por instituição.**

```
SELECT P.INSTITUICAO, (COUNT(*))/(SELECT COUNT(*) FROM PESQUISADOR)*100 AS PERCENT
FROM PESQUISADOR P
GROUP BY P.INSTITUICAO;
```

- **Projetar quais são as instituições que têm mais pesquisadores do que a média.**

```
SELECT P.INSTITUICAO, COUNT(*)
FROM PESQUISADOR P
GROUP BY P.INSTITUICAO
```

```
HAVING COUNT(*) >
(SELECT AVG(COUNT(*))
FROM PESQUISADOR P
GROUP BY P.INSTITUICAO);

SELECT CONTAGEM.INSTITUICAO, CONTAGEM. QTD
FROM (SELECT P.INSTITUICAO, COUNT(*) AS QTD
                FROM PESQUISADOR P
                GROUP BY P.INSTITUICAO) CONTAGEM
WHERE (SELECT AVG(QTD1)
                FROM (SELECT P1.INSTITUICAO, COUNT(*) AS QTD1
                        FROM PESQUISADOR P1
                        GROUP BY P1.INSTITUICAO)) < CONTAGEM.QTD;
```

- **Projetar os títulos dos artigos com nota abaixo da média dos artigos publicados no mesmo evento.**

```
SELECT A1.TITULO, A1.NOTA
FROM ARTIGO A1
WHERE A1.NOTA < (
  SELECT AVG(A.NOTA)
        FROM ARTIGO A
        WHERE A1.COD = A.COD);
```

- **QUESTÕES DA PROVA:**
- **1 Imprimir matrícula de artigos com mais de 5 pesquisadores.**

```
SELECT ESC.MAT
FROM ESCREVE ESC
GROUP BY ESC.MAT HAVING COUNT(*) > 5;
```

- **2 Imprimir sigla de eventos q tiveram artigos publicados por pesquisadores da ufpe.**

```
SELECT DISTINCT E.SIGLA
FROM EVENTO E
JOIN ARTIGO A ON (A.COD = E.COD)
JOIN ESCREVE ESC ON (ESC.MAT = A.MAT)
JOIN PESQUISADOR P ON (P.CPF = ESC.CPF)
WHERE P.INSTITUICAO = 'UFPE';
```

- **3 Imprimir titulo de artigos com nota igual a maior nota do seu evento.**

```
SELECT A.TITULO
FROM ARTIGO A
WHERE A.NOTA = (
  SELECT MAX(A1.NOTA)
  FROM ARTIGO A1
  WHERE A1.COD = A.COD
);
```

- **4 Imprimir siglas dos eventos com a maior quantidade de artigos em inglês.**

```
SELECT E.SIGLA
FROM EVENTO E
WHERE E.COD IN (
  SELECT Q1.COD
```

```
FROM (
   SELECT A.COD, COUNT(*) AS QTD
        FROM ARTIGO A
   WHERE A.IDIOMA = 'INGLES'
   GROUP BY A.COD
        ) Q1
WHERE QTD =
                    (SELECT MAX(QTD)
   FROM (
      SELECT COUNT(*) AS QTD
      FROM ARTIGO A2
      WHERE A2.IDIOMA = 'INGLES'
      GROUP BY A2.COD )
        )
        );
```