QUESTIONÁRIO 4

Agrupamentos

1. **Para gerar agrupamentos, todos os campos sem função de agregação no SELECT devem ser repetidos no GROUP BY.**

Para fazer GROUP BY as funções de agregação (min max sum avg) se mantêm no SELECT junto dos outros campos enquanto esses campos se repetem no GROUP BY.

2. **Para gerar agrupamentos, podem-se usar campos no GROUP BY que não aparecem no SELECT.**

Você pode colocar um campo no GROUP BY para fazer um agrupamento onde não vai ser visto no SELECT (projeção). Pode ter coisas no GROUP BY sem estar no SELECT, mas não pode ter coisa no SELECT (que não é função de agregação) sem estar no GROUP BY.

3. **Para ordenar o resultado, deve-se repetir todos os campos do SELECT no ORDER BY.**

Podemos ter 3 campos no SELECT (nome idade e sexo) e então só ordenamos com um campo (só idade por exem plo ou passar todos então não é imposição).

4. **É correto ter consultas com GROUP BY e sem HAVING.**

O HAVING é só impor um filtro depois do agrupamento, então ele é opcional já que é um filtro que nós decidimos depois.

5. **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.

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

O WHERE é uma filtragem antes do agrupamento, então eles não tem uma ligação direta. É só agrupar primeiro e então impor uma condição no agrupamento (o WHERE pode estar lá ou não, ou seja, um não impede o outro).

Expressões aritméticas

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

Pode fazer somas no SELECT, ou cálculos de campos com números (como nota-1).

8. **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.

9. **É correto ter expressões aritméticas escritas diretamente na cláusula WHERE.**

Podemos escrever condições que envolvem campos e valores aritméticos, como números. Por exemplo, se você for procurar notas >= 7.

10. **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.

11. **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.

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

Assim como WHERE, é natural usar números e expressões aritméticas ao escrever os filtros.

Funções de agregação

13. **É correto ter consultas com funções de agregação escritas diretamente na cláusula SELECT.**
14. **É correto ter consultas com funções de agregação escritas diretamente na cláusula WHERE.**
15. **É correto ter consultas com funções de agregação escritas diretamente na cláusula HAVING.**

Essas funções de agregação só podem ser usadas nos campos SELECT e HAVING, já que servem para projeção. É bem comum o uso de AVG(), COUNT(), SUM() nesses campos.
Não faz sentido ter agregação em outras partes da consulta, como FROM ou WHERE.

16. **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.

Junções

17. **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.

18. **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.

19. **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.

20. **Toda junção interna retorna apenas as linhas que satisfazem a condição de junção.**

O INNER JOIN sempre retorna as linhas que atendam a condição de junção depois do produto cartesiano. Caso queira algo que está além da condição de junção, então já estamos trabalhando com junções externas.

21. **Toda junção externa retorna apenas as linhas da tabela à esquerda (Left Outer Join), à direita (Right Outer Join) ou de ambas (Full Outer 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.

Esquema Pesquisador-Artigo-Evento

22. **Considerando o esquema Pesquisador-Escreve-Artigo-Evento usado nas aulas, para exibir os CPF dos pesquisadores que escreveram o artigo com título 'XPTO', é eficaz tanto escrever uma junção quanto uma subconsulta.**

No SELECT temos o CPF do pesquisador, mas já que o título do artigo só se encontra na outra tabela fazemos o JOIN para obter essa informação. Também é possível substituir a junção por uma subconsulta, que tem uma performance melhor mas pode ser menos utilizada.

```
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”);
```

23. Considerando o esquema Pesquisador-Escreve-Artigo-Evento usado nas aulas, para exibir os nomes dos pesquisadores que escreveram o artigo com título ‘XPTO’, é eficaz tanto escrever duas junções quanto duas subconsultas.

Já que temos o nome do pesquisador em uma tabela, o título do artigo em outra e a conexão entre as duas coisas numa terceira, precisamos fazer dois JOIN para juntar tudo. Também pode resolver com subconsultas, o que é mais otimizado, mas menos direto.

```
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”));
```

24. Considerando o esquema Pesquisador-Escreve-Artigo-Evento usado nas aulas, para exibir os títulos dos artigos publicados no evento cujo código é 1234, não é necessário escrever junções nem subconsultas.

O código do evento já se encontra na tabela do artigo através de chave estrangeira, então não precisa de JOIN ou subconsulta.

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

25. Considerando o esquema Pesquisador-Escreve-Artigo-Evento usado nas aulas, para exibir os títulos dos artigos publicados por código do evento, não é necessário escrever junções nem subconsultas.

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.

26. Considerando o esquema Pesquisador-Escreve-Artigo-Evento usado nas aulas, para exibir apenas os títulos dos artigos publicados, não é necessário escrever junções nem subconsultas.

É só fazer um SELECT normal com o título do artigo, FROM artigo e no WHERE botar se código não é nulo, já que um artigo publicado necessariamente terá o código do evento no qual ele foi publicado.

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

27. Considerando o esquema Pesquisador-Escreve-Artigo-Evento usado nas aulas, para exibir apenas os títulos dos artigos publicados no evento cuja sigla é 'SBBD', não é necessário escrever junções nem subconsultas.

Já que essas informações não estão na mesma tabela, precisamos fazer um JOIN ou subconsulta. Com a junção, unimos as tabelas com a chave estrangeira e procuramos o evento com a sigla SBBD.

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

28. Considerando o esquema Pesquisador-Escreve-Artigo-Evento usado nas aulas, para exibir as siglas dos eventos com artigos publicados, não é necessário escrever junções nem subconsultas.

Já que a tabela de evento não tem nada de artigo, então precisamos fazer uma junção, subconsulta ou semijunção. Vai procurar algum artigo publicado no evento e quando achar para, aí damos um SELECT na sigla do evento FROM evento WHERE exists (retornar valor em que a.cod=e.cod).

```
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);
```

29. Considerando o esquema Pesquisador-Escreve-Artigo-Evento usado nas aulas, para exibir os CPF dos pesquisadores que escreveram artigos, não é necessário escrever junções ou subconsultas.

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.

30. Considerando o esquema Pesquisador-Escreve-Artigo-Evento usado nas aulas, para exibir os artigos que têm mais de 5 autores, é necessário escrever a cláusula WHERE.

Como estamos querendo uma filtragem de quantidade, vamos precisar usar uma função de agregação. Já que ela só pode estar no SELECT ou no HAVING, o WHERE não é necessário. Tudo que precisamos está na tabela escreve, então é só dar um SELECT com matrícula de artigo e um COUNT, agrupando por matrícula no GROUP BY e COUNT > 5 no HAVING.

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

31. Considerando o esquema Pesquisador-Escreve-Artigo-Evento usado nas aulas, para exibir os eventos com mais de 10 artigos em inglês, não é necessário escrever a cláusula WHERE.

Num caso assim vamos precisar tanto do WHERE quanto do HAVING, porque o WHERE vai ser usado para fazer a filtragem do idioma dos artigos e o HAVING vai ser usado para verificar se a contagem é

maior de 10 depois do GROUP BY. E também vamos precisar da junção para ligar artigo, onde está o idioma, e o evento.

```
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;
```

Subconsultas

32. Considerando o esquema Pesquisador-Escreve-Artigo-Evento usado nas aulas, para exibir os CPF dos pesquisadores que escrevem mais artigos do que a média geral, deve-se escrever uma subconsulta correlacionada e do tipo escalar.

Depende de qual subconsulta você leva em consideração, mas se for a subconsulta mais interna ela não é correlacionada, é simples; e também não é escalar, é tabela. Apenas a subconsulta pra encontrar a média é escalar.

Na verdade, vai fazer um COUNT dos pesquisadores por agrupamento. Vai agrupar por pesquisador, ver a quantidade de artigos que cada pesquisador tem e se é maior que a média.

Já que não tem como fazer uma média de uma contagem, tem que fazer subconsultas. Dá SELECT no CPF vindo da subconsulta que retorna a quantidade de artigos escritos pelo CPF e usa o WHERE para filtrar os maiores que a média.

A subconsulta do WHERE vai verificar se qtd1 > AVG(qtd2) e esse AVG(qtd2) vem de um FROM com outra subconsulta para pegar a quantidade de artigos escritos pelos outros.

```
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));
```

33. Considerando o esquema Pesquisador-Escreve-Artigo-Evento usado nas aulas, exibir os títulos dos artigos que têm a mesma nota e o mesmo idioma do artigo 1234, deve-se escrever uma subconsulta simples e do tipo linha.

Vai precisar de subconsulta para poder comparar a nota de um artigo com os outros artigos, e ela é simples porque não vai se relacionar com a consulta externa.

É de linha pois a subconsulta só vai retornar 2 colunas (nota e idioma) e com apenas uma linha (já que só quer 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);
```

34. Considerando o esquema Pesquisador-Escreve-Artigo-Evento usado nas aulas, para exibir os CPF e os nomes dos pesquisadores que escrevem mais artigos do que a média dos pesquisadores da sua instituição, deve-se escrever uma subconsulta correlacionada e do tipo tabela.

Depende de qual subconsulta você leva em consideração. A subconsulta mais interna de fato é correlacionada porque na subconsulta nós temos a comparação da instituição de um evento com a de outro, comparando sempre uma com a externa.

A subconsulta mais interna também é tabela (já que retorna várias linhas e tem 2 colunas), mas a da média é simplesescalar.

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

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

Como a subconsulta escalar retorna um único valor dá para usar os operadores para fazer a comparação elemento a elemento. Nos outros tipos já que retorna mais de um valor deve haver a comparação entre conjuntos ou olhar o conjunto completo.

36. **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.

37. **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.

Semijoin e antijoin

38. **Considerando o esquema Pesquisador-Escreve-Artigo-Evento usado nas aulas, 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.

O SEMIJOIN é usado quando queremos exibir informações de uma única tabela, mas precisamos manipular duas... Basta encontrar 1 caso true para parar de analisar e passar para o próximo.

39. **Considerando o esquema Pesquisador-Escreve-Artigo-Evento usado nas aulas, para exibir os títulos apenas dos Artigos já publicados, é eficiente escrever um semi-join.**

Essa informação está disponível manipulando apenas a tabela artigos, então usar um JOIN ou SEMIJOIN é desnecessário.

40. **Considerando o esquema Pesquisador-Escreve-Artigo-Evento usado nas aulas, 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);
```

CONCLUSAO: semi-join e anti-join sempre são mais eficientes que operações de join (mas lembrar que precisam exibir informação de 1 tabela manipulando outras 2...)

41. Considerando o esquema Pesquisador-Escreve-Artigo-Evento usado nas aulas, 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. Com subconsulta, por exemplo, fazemos SELECT NOME FROM PESQUISADOR WHERE CPF IN (SELECT CPF FROM ESCREVE).

Consultas SQL

42. Considerando o esquema Pesquisador-Escreve-Artigo-Evento usado nas aulas, é eficaz escrever a junção "FROM (EVENTO E INNER JOIN ARTIGO A ON A.MAT = E.MAT)" para exibir os artigos publicados e as siglas dos seus eventos.

A condição da junção está errada já que está tentando fazer A.MAT = E.MAT, no lugar de A.COD = E.COD. O código de evento vai para artigo, não o contrário.

43. Considerando o esquema Pesquisador-Escreve-Artigo-Evento usado nas aulas, é eficaz escrever a junção "FROM (ESCREVE E INNER JOIN ARTIGO A ON A.MAT = E.MAT)" para exibir os CPF dos pesquisadores com artigos sem nota.

Está correto, já que queremos exibir o CPF dos pesquisadores (que está disponível em escreve) e a nota do artigo (que está em artigo). A informação de artigo que está em escreve é a matrícula, então a condição está correta.

44. Considerando o esquema Pesquisador-Escreve-Artigo-Evento usado nas aulas, é eficaz escrever a junção "FROM ((PESQUISADOR P INNER JOIN ESCREVE E ON P.CPF = E.CPF) INNER JOIN ARTIGO A ON E.MAT = A.MAT)" para exibir os nomes dos pesquisadores que têm artigos não publicados em eventos.

Para obter todas essas informações precisamos lidar com as três tabelas, pesquisador para o nome, escreve para a ligação e artigo para saber se foi publicado, então usamos dois INNERJOIN da maneira que é escrita no enunciado.

45. Considerando o esquema Pesquisador-Escreve-Artigo-Evento usado nas aulas, é eficaz escrever a seleção "WHERE NOTA = NULL" para exibir os artigos sem nota.

A sintaxe correta é WHERE NOTA IS NULL, pois o símbolo de igual trata NULL como uma palavra qualquer, e não a ausência de um valor. Essa consulta retornaria as notas que foram preenchidas com a string 'NULL'.

46. Considerando o esquema Pesquisador-Escreve-Artigo-Evento usado nas aulas, é eficaz escrever a seleção "WHERE NOTA > AVG (NOTA)" para filtrar os artigos com nota acima da média.

Não pode usar funções de agregação no WHERE, somente no SELECT e no HAVING, então essa consulta está errada.

47. **Considerando o esquema Pesquisador-Escreve-Artigo-Evento usado nas aulas, é eficaz escrever o agrupamento “GROUP BY ESCREVE.CPF” para computar a quantidade de CPF por artigo.**

O correto seria usar "GROUP BY ESCREVE.MAT" para agrupar por artigo e contar quantos CPFs estão relacionados a ele.

48. **Considerando o esquema Pesquisador-Escreve-Artigo-Evento usado nas aulas, é eficaz escrever a seleção “HAVING COUNT(*) < = 5” para filtrar os artigos com até 5 CPF.**

Sendo esse GROUP BY feito por matrícula, a função de agregação COUNT pra CPF pode ser usada no HAVING sem problemas, e nele mostrar que é menor ou igual a 5.

49. **Considerando o esquema Pesquisador-Escreve-Artigo-Evento usado nas aulas, é eficaz escrever a seleção “WHERE EXISTS (SELECT * FROM ESCREVE WHERE PESQUISADOR.CPF = ESCREVE.CPF) para filtrar os pesquisadores que já escreveram artigos.**

Esse é um exemplo de uso de subconsulta no lugar de INNERJOIN, ele consegue mostrar quais pesquisadores já escreveram artigos, mas não que artigos são esses.

50. **Considerando o esquema Pesquisador-Escreve-Artigo-Evento usado nas aulas, é eficaz escrever a seleção “WHERE A.NOTA = (SELECT MAX (AA.NOTA) FROM ARTIGO AA WHERE A.COD = AA.COD)” para filtrar os artigos cujas notas são iguais a maior nota do seu evento.**

Deve haver uma correlação sempre comparando o artigo externo com o artigo interno, então fazemos uma subconsulta comparando o A.NOTA com o maior valor do artigo que tem o mesmo código de fora, o que indica que faz parte do mesmo evento. Logo estamos comparando as notas do primeiro artigo com a maior nota de artigos que existem.

Primeiro comparamos a tabela com ela mesma, depois pegamos o maior valor de nota no evento (comparando A.COD com AA.COD). Aí projetamos os artigos que tem nota maior ou igual a ela.