LAB_03

3.1 → Criação da Database

Script pronto

3.2 → Consulta com cálculo

```
SELECT
nome, -- campo / coluna da tabela
dt_nascimento, -- campo / coluna da tabela
DATE_FORMAT(dt_nascimento, '%d/%d/%Y') AS 'Aniversário', -- formata data em dia/mês/ano com 4 dígitos
(
YEAR(NOW()) - YEAR(dt_nascimento) - -- vai SUBTRAIR de 0 ou 1, dependendo se já fez aniversário ou não
CASE
 WHEN (MONTH(NOW()) * 100 + DAY(NOW())) > (MONTH(dt_nascimento) * 100 + DAY(dt_nascimento))
 THEN 0 -- Valor de retorno para ser subtraído = 0
 ELSE 1 -- Valor de retorno para ser subtraído = 1
END
) AS Idade -- AS é a indicação de apelido de exibição para a coluna recém calculada
FROM Empregado;
```

1. **O que faz a função YEAR()?**

   É uma função utilizada para extrair o ano de uma data.

2. **O que faz a função MONTH()?**

   É uma função utilizada para extrair o mês de uma data.

3. **O que faz a função DAY()?**

   É uma função utilizada para extrair o dia de uma data.

4. **O que faz a função NOW()?**

   É uma função utilizada para extrair a data e hora do sistema no momento da inserção.

5. **Qual a sintaxe para o comando CASE ...?**

```
SELECT xxx, yyy, zzz
  CASE
      WHEN valor1 THEN resultado1
      WHEN valor2 THEN resultado2
      ...
      ELSE resultado_padrao
  END
FROM abc;
```

6. **Escreva o comando SQL para calcular a sua idade em anos, cuidando para o SQL verificar se já fez aniversário ou não.**

```
YEAR(NOW()) - YEAR(22/01/2005) - -- vai SUBTRAIR de 0 ou 1, dependendo se já fez aniversário ou não
CASE
  WHEN (MONTH(NOW()) * 100 + DAY(NOW())) > (MONTH(22/01/2005) * 100 + DAY(22/01/2005))
  THEN 0
  ELSE 1
END
```

3.3 → Associação JOIN ou INNER JOIN

```
-- Produto Cartesiano
SELECT * FROM empregado AS E, departamento AS D
WHERE E.ID_depto = D.ID_depto
ORDER BY E.nome;

-- INNER JOIN / JOIN
SELECT * FROM empregado AS E INNER JOIN departamento AS D -- INNER JOIN = JOIN
ON (E.ID_depto = D.ID_depto)
ORDER BY E.nome;
```

1. Houve diferença no resultado dos comandos de SELECT do exercício?

   Não, o resultado apresentado foi o mesmo.

2. Substitua no 2º SELECT o “INNER JOIN” por apenas “JOIN”. Qual a diferença entre esses dois comandos?

A diferença existe apenas na preferência do desenvolvedor, já que o resultado apresentado pelos dois é o mesmo.

```
-- Produto Cartesiano
SELECT E.nome AS Empregado, ES.nivel, S.nome
FROM Empregado AS E, EmpSkill AS ES, Skill AS S
WHERE E.ID_emp = ES.ID_emp AND S.ID_skill = ES.ID_skill
ORDER BY S.nome, E.nome;

-- Join (precedencia feita com (), juntando tabelas Empregado e EmpSkill, depois Skill)
SELECT E.nome AS Empregado, ES.nivel, S.nome
FROM
(Empregado AS E INNER JOIN EmpSkill AS ES ON (E.ID_emp = ES.ID_emp))
INNER JOIN Skill AS S ON (S.ID_skill = ES.ID_skill)
ORDER BY S.nome, E.nome;
```

1. Quantas e quais tabelas estão envolvidas nas consultas?

   São envolvidas 3 tabelas: Empregado, EmpSkill e Skill

2. É possível retirar alguma das tabelas indicadas nos comandos e obter o mesmo resultado? Qual tabela podemos retirar?

   Não é possivel retirar nenhuma das tabelas, já que pegamos o nome do empregado da tabela Empregado, nome da skill da tabela Skill e o nível dos empregados em cada skill da tabela EmpSkill.

3.4 → LEFT OUTER JOIN

```
-- Left Outer Join (precedencia feita com (), juntando tabelas Empregado e EmpSkill, depois Skill)
SELECT E.nome AS Empregado, ES.nivel, S.nome
FROM (Empregado AS E LEFT OUTER JOIN EmpSkill AS ES ON (E.ID_emp = ES.ID_emp))
LEFT OUTER JOIN Skill AS S ON (S.ID_skill = ES.ID_skill)
ORDER BY S.nome, E.nome;
```

1. Para o LEFT OUTER JOIN, qual a tabela à esquerda do comando? E qual a tabela à direita do comando?

   Na primeira junção, a tabela à esquerda é “Empregado”, e a da direita é “EmpSkill”. Na segunda junção, a tabela à esquerda é o resultado da primeira junção, e a da direita é “Skill”.

A tabela esquerda vai ser a que sempre será a principal, com todos os registros incluídos, mesmo que a tabela secundária (direita) complete as correspondências com nulo.

2. Quais dados foram apresentados, mesmo sem correspondência?

   São apresentados os dados: nome do empregado, nível na skill e o nome da skill. Alguns Empregados estão com as colunas “nível” e “skill” vazias.

3.5 → RIGHT OUTER JOIN

```
-- Right Outer Join (precedencia feita com (), juntando tabelas Empregado e EmpSkill, depois Skill)
SELECT E.nome AS Empregado, ES.nivel, S.nome
FROM (Empregado AS E RIGHT OUTER JOIN EmpSkill AS ES ON (E.ID_emp = ES.ID_emp))
RIGHT OUTER JOIN Skill AS S ON (S.ID_skill = ES.ID_skill)
ORDER BY S.nome, E.nome;
```

1. Para o RIGHT OUTER JOIN, qual a tabela à direita do comando? E qual a tabela à esquerda do comando?

   Na primeira junção, a tabela à direita é “EmpSkill”, e à esquerda é “Empregado”. Na segunda junção, a tabela à direita é “Skill”, e a da esquerda é o resultado da primeira junção.

   Como no Left Outer Join, existe uma tabela principal (direita) com todos os registros incluídos, e uma secundária (esquerda), com registros inexistentes completos por null.

2. Quais dados foram apresentados, mesmo sem correspondência?

   São apresentados os dados: nome do empregado, nível na skill e o nome da skill. Algumas linhas das colunas “Empregado” (E.nome) e “nível” estão vazias.

3.5 → FULL OUTER JOIN

```
(SELECT E.nome AS Empregado, ES.nivel, S.nome
FROM (Empregado AS E LEFT OUTER JOIN EmpSkill AS ES ON (E.ID_emp = ES.ID_emp))
LEFT OUTER JOIN Skill AS S ON (S.ID_skill = ES.ID_skill))
UNION
(SELECT E.nome AS Empregado, ES.nivel, S.nome
FROM (Empregado AS E RIGHT OUTER JOIN EmpSkill AS ES ON (E.ID_emp = ES.ID_emp))
RIGHT OUTER JOIN Skill AS S ON (S.ID_skill = ES.ID_skill))
ORDER BY nivel;
```

1. Quantos registros são retornados no SELECT do LEFT OUTER JOIN?

   Foram retornados 5 registros a mais em cada coluna. (12 total)

2. Quantos registros são retornados no SELECT do RIGHT OUTER JOIN?

   Foi retornado 1 registro a mais em cada coluna. (8 total)

3. Pesquise quais as condições para o comando UNION ser realizado. Dê um exemplo de uma utilização incorreta do UNION e sua respectiva correção explicada.

   Condições:

   - Número de colunas nas tabelas combinadas deve ser o mesmo;
   - Os tipos de dados devem ser compatíveis;
   - As colunas correspondentes devem ter a mesma ordem, porém não o mesmo nome.

   ```
   -- Incorreto -> conlfito entre os nomes das colunas
   SELECT nome FROM Clientes
   UNION
   SELECT nome FROM Fornecedores;

   -- Correção -> diferencia cada tipo de nome
   SELECT nome AS NomeCliente FROM Clientes
   UNION
   SELECT nome AS NomeFornecedor FROM Fornecedores;
   ```

4. A operação de UNION do exercícios retornou quantos registros no total? Esse valor corresponde à soma dos resultados do LEFT e do OUTER JOIN? Sim ou não? Por que?

   São retornados todos os registro das colunas “Empregado“, “nível” e “nome”, somando 13 registros de cada coluna.

   A soma não corresponde se contarmos que cada um dos comandos gerou a lista inteira + os sem correspondência. Porém, se contarmos os registros com todas as colunas preenchidas como padrão e somarmos os registros a mais de cada comando, a soma corresponde com o número total.

3.6 → Associação x Subconsulta

```
-- Primeiro Comando
SELECT E.ID_emp, E.nome
FROM Empregado AS E JOIN Departamento AS D
ON (E.ID_depto = D.ID_depto)
WHERE D.sigla = 'CTB' OR D.sigla ='VND'
ORDER BY E.nome;

-- Segundo Comando
SELECT E.ID_emp, E.nome
FROM Empregado AS E
WHERE E.ID_depto IN
(SELECT D.ID_depto
FROM Departamento AS D
WHERE D.sigla = 'CTB' OR D.sigla ='VND')
ORDER BY E.nome;
```

1. Qual a diferença entre os comando de SELECT passados?

   Eles retornam o mesmo resultado, mas o primeiro usa o comando JOIN e o segundo uma subconsulta dentro do SELECT.

2. Qual comando possui uma subconsulta?

   O segundo comando possui uma subconsulta.

3. Como funciona o comando WHERE ... IN?

   É um comando que filtra as linhas de uma tabela com base em valores fornecidos (ex: where (coluna) IN (valores))

```
-- Primeiro Comando
SELECT E.ID_depto, E.ID_emp, E.nome FROM Empregado AS E
WHERE E.ID_depto NOT IN
(SELECT D.ID_depto
FROM Departamento AS D
WHERE D.sigla = 'CTB' OR D.sigla ='VND')
ORDER BY E.nome;

-- Segundo Comando
SELECT E.ID_depto, E.ID_emp, E.nome FROM Empregado AS E
WHERE E.ID_depto <> ALL
(SELECT D.ID_depto
FROM Departamento AS D
```

```
WHERE D.sigla = 'CTB' OR D.sigla ='VND')
ORDER BY E.nome;
```

1. Qual a diferença entre os comando passados?

   O primeiro utiliza o NOT IN, que analisa a lista e seleciona os valores não condizentes com a solução. O segundo comando utiliza o <> ALL, comando que verifica cada valor individualmente.

2. Como funciona o comando WHERE ... <> ALL?

   Ele serve para filtrar linhas de uma coluna com base nos dados NÃO correspondentes a subconsulta. (ex: WHERE (coluna) <> ALL (SELECT…))

3.7: Associação x Subconsulta

```
SELECT E.ID_depto, E.ID_emp, E.nome, E.dt_nascimento
FROM Empregado AS E
WHERE YEAR(E.dt_nascimento) >= 1998 AND E.ID_depto IN
(SELECT D.ID_depto
FROM Departamento AS D, Empregado AS E1
WHERE D.ID_depto = E1.ID_depto AND (D.sigla = 'CTB' OR D.sigla ='VND'))
ORDER BY E.nome;

-- com ANY
SELECT E.ID_depto, E.ID_emp, E.nome, E.dt_nascimento
FROM Empregado AS E
WHERE E.ID_depto = ANY
(SELECT D.ID_depto FROM Departamento AS D, Empregado AS E1
WHERE D.ID_depto = E1.ID_depto AND (D.sigla = 'CTB' OR D.sigla ='VND'))
AND YEAR(E.dt_nascimento) >= 1998
ORDER BY E.nome;
```

1. Como funcionou o comando WHERE ... = ANY nesta consulta?

   O comando funcionou filtrando resultados da tabela “Empregado” com base numa subconsulta, que filtra os resultados da tabela “Departamento” onde a "sigla" do departamento é 'CTB' ou 'VND’.

2. Qual a diferença entre WHERE ... = ANY e WHERE ... IN.

WHERE… = ANY → compara os valores com os resultados de uma subconsulta

WHERE… IN → compara os valores com uma lista digitada manualmente

3.8 → Views

```
-- Deleta a VIEW, se existir
DROP VIEW IF EXISTS CompetenciasEmpregados;

-- Cria a VIEW
CREATE VIEW CompetenciasEmpregados AS (
SELECT D.sigla AS Depto, S.nome AS Competencia, ES.nivel AS Nivel, E.nome AS Empregado
FROM ((Empregado AS E INNER JOIN EmpSkill AS ES ON (E.ID_emp = ES.ID_emp))
INNER JOIN Skill AS S ON (S.ID_skill = ES.ID_skill))
INNER JOIN Departamento AS D ON (D.ID_depto = E.ID_depto));

-- Retorna a VIEW criada
SELECT * FROM CompetenciasEmpregados
ORDER BY Depto, Competencia, Empregado;
```

1. Pesquise uma ou mais vantagens em se usar uma VIEW.
    - Simplificação de consultas complexas.
    - Melhor segurança de dados.
    - Reutilização de consultas.
    - Abstração de tabelas.

2. Pesquise uma ou mais desvantagens em se usar uma VIEW
    - Possível impacto no desempenho.
    - Dificuldade de manutenção em sistemas complexos.
    - Restrições na atualização de dados através de Views.
    - Consumo de recursos do sistema em sistemas com muitas Views.

3.9 → Funções de Agregação

SUM (N) → soma
AVG (N) → média
MIN (EXP) → menor valor
MAX (EXP) → maior valor
COUNT (EXP) → total de valores

```
-- Primeiro Comando
SELECT
COUNT(*) AS 'Número de Empregados',
AVG(salario) AS 'Salário Médio',
MIN(salario) AS 'Menor Salário' ,
MAX(salario) AS 'Maior Salário' ,
SUM(salario) AS 'Total Salários'
FROM Empregado;

-- Segundo Comando
SELECT
COUNT(*) AS 'Número de Empregados',
CONVERT(AVG(salario), DECIMAL(8,2)) AS 'Salário Médio',
MIN(salario) AS 'Menor Salário' ,
MAX(salario) AS 'Maior Salário' ,
SUM(salario) AS 'Total Salários'
FROM Empregado;
```

1. Qual a diferenças entre os comandos?

   Os dois comandos geram a mesma tabela, porém na segunda o ‘Salário Médio’ é utilizada a função DECIMAL( ), que limita o número de casas de um número, nesse caso, um total de 8 casas, sendo 2 delas partes fracionárias, e a função CONVERT( ) que converte a primeira condição para a segunda.

2. Descreva cada um dos resultados obtidos no segundo comando de SELECT, explicando os comandos de AGREGAÇÃO executados.

   Segundo:

Número de Empregados	Salário Médio	Menor Salário	Maior Salário	Total Salários
8	2210.63	1500	3005	17685

   Número de empregados → COUNT( ) → total de empregados

   Salário médio → AVG( ) → média entre os salários

   → CONVERT( ) → converte o resultado do primeiro comando no segundo

→ DECIMAL ( ) → limita o número de casas de um número

Menor salário → MIN( ) → —

Maior salário → MAX( ) → —

Total salários → SUM( ) → soma entre todos os valores dos salários