Realizando Consultas com o SELECT

➔ Para realizar consultas em SQL, utiliza-se o comando SELECT acrescido das colunas, das tabelas as quais estas colunas fazem parte e condições para filtrar a consulta;
➔ SELECT [colunas]
FROM [tabelas]
[condições];
➔ Para selecionar todas as colunas de uma tabela na consulta, utiliza-se o asterisco;
Exemplo:
SELECT * FROM Pessoas;

Ordenando as Consultas com o ORDER BY

➔ É possível ordenar os resultados gerados nas consultas através do comando ORDER BY [coluna];
➔ Por exemplo, para selecionar todos os cursos, ordenando pelo nome:
SELECT * FROM Cursos
ORDER BY nome;

➔ A ordenação é feita em ordem ascendente (no caso do nome, ordem alfabética) por padrão, mesmo assim, existe o parâmetro ASC para isto;
➔ Porém, para que a ordenação seja feita em ordem descendente ou decrescente, acrescenta-se o parâmetro DESC ao final do comando;
➔ Por exemplo, para selecionar todos os cursos, ordenados pelo nome, em ordem alfabética inversa:
SELECT * FROM Cursos
ORDER BY nome DESC;

➔ Ademais, pode-se fazer a ordenação por mais de uma coluna;

➔ Por exemplo, selecionar as colunas nome, carga e ano dos cursos, ordenados pelo ano e pelo nome:

```
SELECT nome, carga, ano
FROM Cursos
ORDER BY ano, nome;
```

Filtrando as Consultas com a Cláusula WHERE

➔ Através da cláusula WHERE são especificados os critérios que os valores da coluna devem cumprir, para que os registros que contêm estes valores sejam incluídos nos resultados da consulta;
➔ Chamamos de expressão lógica o conjunto de uma ou mais condições a serem verificadas pela cláusula WHERE;
➔ Essas expressões são construídas utilizando operadores relacionais e lógicos.

Operadores Relacionais

➔ O primeiro operador relacional é o de **Igualdade**;
➔ Este operador realiza a operação de igualdade entre dois valores;
➔ Com ele, é verificado se o valor à direita do sinal **=** é igual ao valor contido na coluna informada à esquerda;
➔ Por exemplo, selecionar os cursos que são do ano de 2016:

```
SELECT * FROM Cursos
WHERE ano = “2016”;
```

➔ É possível ainda selecionar dados em uma tabela utilizando o operador **<>** ou **!=**, o qual aplica a operação de **Diferença/Desigualdade**;
➔ Por exemplo, selecionar o nome, descrição e ano dos cursos que não são de 2016:

```
SELECT nome, descricao, ano FROM Cursos
WHERE ano <> “2016”;
```

➔ O operador **Menor Que** (**<**) verifica se o valor informado à esquerda é menor do que aquele à direita;
➔ Também pode-se utilizar esse operador em conjunto com o **=**, formando um único operador **<=**, chamado **Menor ou Igual**, para verificar se o valor é menor ou igual ao esperado;
➔ Por exemplo, selecionar o nome, descrição e ano dos cursos que são de 2015 e anos anteriores:

```
SELECT nome, descricao, ano FROM Cursos
WHERE ano <= 2015;
```

➔ O operador **Maior Que** (**>**) verifica se o valor informado à direita é maior do que aquele à direita;
➔ Também é possível combinar esse operador com o **=**, formando um único operador **>=**, chamado **Maior ou Igual**, o qual verifica se o valor é maior ou igual ao esperado;
➔ Por exemplo, selecionar o nome, descrição e ano dos cursos que são de 2016 e anos posteriores:

```
SELECT nome, descricao, ano FROM Cursos
WHERE ano >= 2016;
```

➔ O operador **BETWEEN** é utilizado quando é necessário buscar registro de acordo com um intervalo de valores;
➔ O operador recebe um valor mínimo e um valor máximo e retorna os dados da coluna que atendem a esse critério;
➔ Por exemplo, selecionar o nome e o ano dos cursos que são entre 2014 e 2016:

```
SELECT nome, ano FROM Cursos
WHERE ano BETWEEN 2014 AND 2016;
```

➔ Com o operador **IN**, pode-se criar uma lista de valores que serão comparados com o valor na coluna;
➔ Dessa forma, a linha a qual o valor avaliado pertence será ignorada, exceto se qualquer um dos valores na lista corresponder ao valor da coluna;
➔ Por exemplo, selecionar o nome e o ano dos cursos que são de 2018 e 2020:

```
SELECT nome, ano FROM Cursos
WHERE ano IN (2018, 2020);
```

➔ O operador **LIKE** permite a comparação entre o valor de uma coluna e uma sequência de caracteres;
➔ Com ele é possível verificar se o texto na coluna contém uma palavra ou parte dela ou até mesmo uma sentença completa;
➔ Este operador permite o uso do sinal **%** sendo utilizado como parâmetro, o que substitui um ou mais caracteres no início, meio ou final do texto;
➔ Por exemplo, selecionar os cursos que começam com a letra P:

```
SELECT * FROM Cursos
WHERE nome LIKE "P%";
```

➔ Outro exemplo, selecionar os cursos que terminam com a letra A:

```
SELECT * FROM Cursos
WHERE nome LIKE "%A";
```

➔ Mais um exemplo, selecionar os alunos que possuem a letra "e" no nome:

```
SELECT * FROM Alunos
WHERE nome LIKE "%e%";
```

➔ Também é possível utilizar o operador lógico **NOT** em conjunto com o **LIKE** para retornar os registros que não atendem à condição estabelecida;
➔ Por exemplo, selecionar os cursos que não possuem a letra "a" no nome:

```
SELECT * FROM Cursos
WHERE nome NOT LIKE "%a%";
```

Operadores Lógicos

➔ Há casos em que são necessários unir duas ou mais condições, é possível realizar isso através de operadores lógicos como o **AND**, o qual aplica a operação lógica **E**;

➔ Além do operador lógico **OR**, o qual aplica a operação lógica **OU**;
➔ E o operador **NOT**, citado no exemplo anterior, que aplica a operação lógica **NÃO/Negação**;
➔ Por exemplo, selecionar os cursos que possuem carga horária maior do que 35 E o total de aulas menor do que 30:

```
SELECT * FROM Cursos
WHERE carga > 35 AND totaulas < 30;
```

➔ Outro exemplo, selecionar os cursos que possuem carga horária maior do que 35 OU o total de aulas menor do que 30:

```
SELECT * FROM Cursos
WHERE carga > 35 OR totaulas < 30;
```

Eliminando Repetições com a Função DISTINCT

➔ O **DISTINCT** trata-se de uma função responsável por eliminar as repetições que podem aparecer no resultado de uma consulta;
➔ Ela é aplicada a algum atributo e também pode ser usada dentro de uma função de agregação;
➔ Por exemplo, selecionar a nacionalidade dos alunos (sem repetições):

```
SELECT DISTINCT(nacionalidade)
FROM Alunos
ORDER BY nacionalidade;
```

Funções de Agregação

➔ Uma função de agregação processa um conjunto de valores contidos em uma única coluna de uma tabela e retorna um único valor como resultado;
➔ Sua sintaxe é semelhante aquela encontrada em várias linguagens de programação;
➔ Entretanto, o valor informado é sempre a coluna cujos valores serão processados.

COUNT

➔ A função **COUNT** retorna a quantidade de linhas selecionadas;
➔ Ela pode receber por parâmetro o nome de alguma coluna da tabela ou um asterisco;
➔ Quando é informado o nome de alguma coluna, valores do tipo **NULL** são ignorados;
➔ Porém, quando informado * todas as linhas serão contabilizadas;
➔ Por exemplo, selecionar a quantidade de cursos inseridos no banco de dados:

```
SELECT COUNT(*) FROM Cursos;
```

➔ Outro exemplo, selecionar a quantidade de cursos que possuem carga horária maior do que 40:

```
SELECT COUNT(*)
FROM Cursos
WHERE carga > 40;
```

MAX

➔ A função **MAX** analisa um conjunto de valores e retorna o maior entre eles;
➔ Por exemplo, selecionar a maior carga horária dos cursos:

```
SELECT MAX(carga) FROM Cursos;
```

➔ Outro exemplo, selecionar o maior total de aulas dos cursos de 2016:

```
SELECT MAX(totaulas)
FROM Cursos
WHERE ano = 2016;
```

MIN

➔ A função **MIN** analisa um conjunto de valores e retorna o menor entre eles;

➔ Por exemplo, selecionar o menor total de aulas dos cursos de 2016:

```
SELECT MIN(totaulas)
FROM Cursos
WHERE ano = 2016;
```

SUM

➔ A função **SUM** realiza a soma dos valores de uma única coluna e retorna esse resultado;
➔ Por exemplo, somar o total de aulas dos cursos de 2016:

```
SELECT SUM(totaulas)
FROM Cursos
WHERE ano = 2016;
```

AVG

➔ A função **AVG** é responsável por calcular a média aritmética dos valores de uma única coluna;
➔ Por exemplo, calcular a média do total de aulas dos cursos de 2016:

```
SELECT AVG(totaulas)
FROM Cursos
WHERE ano = 2016;
```

Agrupando Valores com o GROUP BY

➔ A cláusula **GROUP BY** é responsável por agrupar os registros de acordo com o valor da coluna especificada e retorna uma linha de resultados para cada uma;
➔ Pode ser aplicado junto com alguma função de agregação;
➔ Por exemplo, agrupar o total de aulas de todos os cursos, exibindo a quantidade de cada grupo:

```
SELECT totaulas, COUNT(*)
FROM Cursos
```

```
GROUP BY totaulas
ORDER BY totaulas;
```

➔ Outro exemplo, agrupar os cursos que possuem um total de 30 aulas, pela carga horária:

```
SELECT carga, COUNT(nome)
FROM Cursos
WHERE totaulas = 30
GROUP BY carga;
```

HAVING

➔ O **HAVING** trata-se de uma cláusula a qual é utilizada em conjunto com o **GROUP BY**;

➔ Ela é responsável por filtrar os resultados após o processamento da função de agregação;

➔ Por exemplo, agrupar os anos que possuem 5 ou mais cursos:

```
SELECT ano, COUNT(*)
FROM Cursos
GROUP BY ano
HAVING COUNT(ano) >= 5
ORDER BY ano;
```

➔ Outro exemplo, selecionar a carga horária dos cursos que são depois de 2015, agrupados pela carga horária e possuem essa maior do que a média da carga horária de todos os cursos:

```
SELECT carga, COUNT(*)
FROM Cursos
WHERE ano > 2015
GROUP BY carga
HAVING carga > (SELECT AVG(carga) FROM Cursos);
```