SELECT e FROM

A parte SELECT(selecionar, em inglês) de uma consulta determina quais colunas dos dados serão exibidas nos resultados. Também há opções que você pode aplicar para mostrar dados que não são uma coluna da tabela. O exemplo abaixo mostra três colunas selecionadas na tabela "student" (aluno, em inglês), para isso usamos o comando FROM (vindo de, em inglês) e uma coluna calculada. O banco de dados armazena o studentID, FirstName e LastName do aluno (id do aluno, nome e sobrenome em inglês, respectivamente). Podemos combinar as colunas FirstName e LastName para criar a coluna calculada FullName. (nome completo, em inglês).

```
SELECT studentID, FirstName, LastName, FirstName + '' + LastName AS FullName
FROM student;
```

```
+-----------+-------------------+------------+------------------------+
| studentID | FirstName         | LastName   | FullName               |
+-----------+-------------------+------------+------------------------+
|         1 | Monique           | Davis      | Monique Davis          |
|         2 | Teri              | Gutierrez  | Teri Gutierrez         |
|         3 | Spencer           | Pautier    | Spencer Pautier        |
|         4 | Louis             | Ramsey     | Louis Ramsey           |
|         5 | Alvin             | Greene     | Alvin Greene           |
|         6 | Sophie            | Freeman    | Sophie Freeman         |
|         7 | Edgar Frank "Ted" | Codd       | Edgar Frank "Ted" Codd |
|         8 | Donald D.         | Chamberlin | Donald D. Chamberlin   |
|         9 | Raymond F.        | Boyce      | Raymond F. Boyce       |
+-----------+-------------------+------------+------------------------+
9 rows in set (0.00 sec)
```

CREATE TABLE

CREATE TABLE(criar tabela, em inglês) faz exatamente o que parece: cria uma tabela no banco de dados. Você pode especificar o nome da tabela e as colunas que devem estar nela.

```
CREATE TABLE nome_da_tabela (
    coluna_1 tipo_de_dados,
    coluna_2 tipo_de_dados,
    coluna_3 tipo_de_dados
);
```

ALTER TABLE

ALTER TABLE(alterar tabela, em inglês) altera a estrutura de uma tabela. Aqui está demonstrado como você adicionaria uma coluna a um banco de dados:

```
ALTER TABLE nome_da_tabela
ADD coluna tipo_de_dados;
```

CHECK

A instrução CHECK(verificar, em inglês) é usada para limitar o intervalo de valores que pode ser colocado em uma coluna.
Se você definir uma instrução CHECK em uma única coluna, ela permitirá apenas determinados valores para essa coluna. Se você definir uma instrução CHECK em uma tabela, ela poderá limitar os valores em determinadas colunas com base nos valores de outras colunas dessa linha.
O SQL a seguir cria uma instrução CHECK na coluna "Age" (idade, em inglês) quando a tabela "Persons" (pessoas, em inglês) é criada. CHECK garante que você não tenha pessoas com menos de 18 anos.

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    CHECK (Age>=18)
);
```

Para permitir a nomeação de uma instrução CHECK e para definir uma instrução CHECK em várias colunas, use a seguinte sintaxe no SQL:

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    City varchar(255),
    CONSTRAINT CHK_Person CHECK (Age>=18 AND City='Sandnes')
);
```

WHERE

(AND, OR, IN, BETWEEN e LIKE)

A instrução WHERE(onde, em inglês) é usada para limitar o número de linhas retornadas.
Como exemplo, primeiro mostraremos uma instrução SELECT e resultados *sem* declaração WHERE. Em seguida, adicionaremos uma instrução WHERE que usa todos os cinco qualificadores acima.

```
SELECT studentID, FullName, sat_score, rcd_updated FROM student;
```

```
+-----------+------------------------+-----------+---------------------+
| studentID | FullName               | sat_score | rcd_updated         |
+-----------+------------------------+-----------+---------------------+
|         1 | Monique Davis          |       400 | 2017-08-16 15:34:50 |
|         2 | Teri Gutierrez         |       800 | 2017-08-16 15:34:50 |
|         3 | Spencer Pautier        |      1000 | 2017-08-16 15:34:50 |
|         4 | Louis Ramsey           |      1200 | 2017-08-16 15:34:50 |
|         5 | Alvin Greene           |      1200 | 2017-08-16 15:34:50 |
|         6 | Sophie Freeman         |      1200 | 2017-08-16 15:34:50 |
|         7 | Edgar Frank "Ted" Codd |      2400 | 2017-08-16 15:35:33 |
|         8 | Donald D. Chamberlin   |      2400 | 2017-08-16 15:35:33 |
|         9 | Raymond F. Boyce       |      2400 | 2017-08-16 15:35:33 |
+-----------+------------------------+-----------+---------------------+
9 rows in set (0.00 sec)
```

Agora, repetiremos a consulta SELECT, mas limitaremos as linhas retornadas usando uma instrução WHERE.

```
STUDENT studentID, FullName, sat_score, recordUpdated
FROM student
WHERE (studentID BETWEEN 1 AND 5 OR studentID = 8)
        AND
        sat_score NOT IN (1000, 1400);
```

```
+-----------+----------------------+-----------+---------------------+
| studentID | FullName             | sat_score | rcd_updated         |
+-----------+----------------------+-----------+---------------------+
|         1 | Monique Davis        |       400 | 2017-08-16 15:34:50 |
|         2 | Teri Gutierrez       |       800 | 2017-08-16 15:34:50 |
|         4 | Louis Ramsey         |      1200 | 2017-08-16 15:34:50 |
```

```
|         5 | Alvin Greene         |      1200 | 2017-08-16 15:34:50 |
|         8 | Donald D. Chamberlin |      2400 | 2017-08-16 15:35:33 |
+-----------+----------------------+-----------+---------------------+
5 rows in set (0.00 sec)
```

UPDATE

Para atualizar um registro em uma tabela, você usa a instrução UPDATE (atualizar, em inglês).
Use a instrução WHERE para especificar quais registros você deseja atualizar. É possível atualizar uma ou mais colunas por vez. A sintaxe é:

```
UPDATE nome_da_tabela
SET coluna1 = valor1,
    coluna2 = valor2, ...
WHERE condição;
```

Aqui está um exemplo atualizando Name (nome, em inglês) do registro com Id 4:

```
UPDATE Person
SET Name = "Elton John"
WHERE Id = 4;
```

Você também pode atualizar colunas em uma tabela usando valores de outras tabelas. Use a instrução JOIN(unir, em inglês) para obter dados de várias tabelas. A sintaxe é:

```
UPDATE nome_da_tabela1
SET nome_da_tabela1.coluna1 = nome_da_tabela2.colunaA
    nome_da_tabela1.coluna2 = nome_da_tabela2.colunaB
FROM nome_da_tabela1
JOIN nome_da_tabela2 ON nome_da_tabela1.ChaveEstrangeira = nome_da_tabela2.Chave
```

Aqui está um exemplo de atualização de Manager (gerenciador, em inglês) de todos os registros:

```
UPDATE Person
SET Person.Manager = Department.Manager
FROM Person
JOIN Department ON Person.DepartmentID = Department.ID
```

GROUP BY

GROUP BY(agrupar por, em inglês) permite combinar linhas e agregar dados.
Aqui está a sintaxe de GROUP BY:

```
SELECT nome_da_coluna, COUNT(*)
FROM nome_da_tabela
GROUP BY nome_da_coluna;
```

HAVING

HAVING(contendo, em inglês) permite filtrar os dados agregados pela instrução GROUP BY para que o usuário obtenha um conjunto limitado de registros para visualização.
Aqui está a sintaxe de HAVING:

```
SELECT nome_da_coluna, COUNT(*)
FROM nome_da_tabela
GROUP BY nome_da_coluna
HAVING COUNT(*) > valor;
```

AVG()

"Average" (média, em inglês) é usado para calcular a média de uma coluna numérica do conjunto de linhas retornado por uma instrução SQL.
Aqui está a sintaxe para usar a função:

```
SELECT campoAgrupamento, AVG(num_campo)
FROM tabela1
GROUP BY campoAgrupamento
```

Aqui está um exemplo usando a tabela de alunos:

```
SELECT studentID, FullName, AVG(sat_score)
FROM student
GROUP BY studentID, FullName;
```

AS

AS (como, em inglês - similar a *like*, que vemos posteriormente) permite renomear uma coluna ou tabela usando um alias.

```
SELECT user_only_num1 AS AgeOfServer, (user_only_num1 - warranty_period) AS
NonWarrantyPeriod FROM server_table
```

Isso resulta na saída abaixo.

```
+-------------+------------------------+
| AgeOfServer | NonWarrantyPeriod      |
+-------------+------------------------+
|          36 |                     24 |
|          24 |                     12 |
|          61 |                     49 |
|          12 |                      0 |
|           6 |                     -6 |
|           0 |                    -12 |
|          36 |                     24 |
|          36 |                     24 |
|          24 |                     12 |
+-------------+------------------------+
```

Você também pode usar AS para atribuir um nome a uma tabela para facilitar a referência em junções.

```
SELECT ord.product, ord.ord_number, ord.price, cust.cust_name, cust.cust_number
FROM customer_table AS cust

JOIN order_table AS ord ON cust.cust_number = ord.cust_number
```

Isso resulta na saída como abaixo.

```
+-------------+------------+-----------+-----------------+--------------+
| product     | ord_number | price     | cust_name       | cust_number  |
+-------------+------------+-----------+-----------------+--------------+
|     RAM     |   12345    |       124 | John Smith      |  20          |
|     CPU     |   12346    |       212 | Mia X           |  22          |
|     USB     |   12347    |        49 | Elise Beth      |  21          |
|     Cable   |   12348    |         0 | Paul Fort       |  19          |
|     Mouse   |   12349    |        66 | Nats Back       |  15          |
|     Laptop  |   12350    |       612 | Mel S           |  36          |
|     Keyboard|   12351    |        24 | George Z        |  95          |
|     Keyboard|   12352    |        24 | Ally B          |  55          |
|     Air     |   12353    |        12 | Maria Trust     |  11          |
+-------------+------------+-----------+-----------------+--------------+
```

ORDER BY

ORDER BY (ordenar por, em inglês) nos dá uma maneira de classificar o conjunto de resultados por um ou mais

itens na seção SELECT. Aqui está um SQL classificando os alunos por FullName em ordem decrescente. A ordem de classificação padrão é crescente (ASC), mas, para classificar na ordem oposta (decrescente), você usa DESC.

```
SELECT studentID, FullName, sat_score
FROM student
ORDER BY FullName DESC;
```

COUNT

COUNT (contar, em inglês) contará o número de linhas e retornará essa contagem como uma coluna no conjunto de resultados.
Aqui estão alguns exemplos onde você usaria COUNT:

- Para contar todas as linhas em uma tabela (GROUP BY não é obrigatório)
- Para contar os totais de subconjuntos de dados (requer um GROUP BY na instrução)

Essa instrução SQL fornece uma contagem de todas as linhas. Observe que você pode dar um nome à coluna COUNT resultante usando "AS".

```
SELECT count(*) AS studentCount FROM student;
```

DELETE

DELETE(excluir, em inglês) é usado para excluir um registro em uma tabela.
Tome cuidado. Você pode excluir todos os registros da tabela ou apenas alguns deles. Use a condição WHERE para especificar quais registros você deseja excluir. A sintaxe é:

```
DELETE FROM table_name
WHERE condition;
```

Aqui está um exemplo de exclusão do registro com Id 3 na tabela Person:

```
DELETE FROM Person
WHERE Id = 3;
```

INNER JOIN

JOIN(junção, em inglês), também chamado de *inner join*(junção interna, em inglês), seleciona registros que têm valores correspondentes em duas tabelas.

```
SELECT * FROM A x JOIN B y ON y.aId = x.Id
```

LEFT JOIN

A LEFT JOIN(junção à esquerda, em inglês) retorna todas as linhas da tabela da esquerda e as linhas correspondentes da tabela da direita. As linhas na tabela da esquerda serão retornadas mesmo que não haja correspondência na tabela da direita. As linhas da tabela à esquerda sem correspondência na tabela à direita terão os valores da tabela à direita null.

```
SELECT * FROM A x LEFT JOIN B y ON y.aId = x.Id
```

RIGHT JOIN

A RIGHT JOIN (junção à direita, em inglês) retorna todas as linhas da tabela da direita e as linhas correspondentes da tabela da esquerda. Ao contrário de uma junção à esquerda, isso retornará todas as linhas da tabela à direita, mesmo quando não houver correspondência na tabela à esquerda. As linhas na tabela à direita que não

têm correspondência na tabela à esquerda terão valores null para as colunas da tabela à esquerda.

```
SELECT * FROM A x RIGHT JOIN B y ON y.aId = x.Id
```

FULL OUTER JOIN

A FULL OUTER JOIN(junção externa completa, em inglês) retorna todas as linhas para as quais haja uma correspondência em qualquer uma das tabelas. Portanto, se houver linhas na tabela da esquerda que não tenham correspondência na tabela da direita, elas serão incluídas. Além disso, se houver linhas na tabela à direita que não tenham correspondência na tabela à esquerda, elas também o serão.

```
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
FULL OUTER JOIN Orders
ON Customers.CustomerID=Orders.CustomerID
ORDER BY Customers.CustomerName
```

INSERT

INSERT(inserir, em inglês) é uma maneira de inserir dados em uma tabela.

```
INSERT INTO nome_da_tabela (coluna_1, coluna_2, coluna_3)
VALUES (valor_1, 'valor_2', valor_3);
```

LIKE

LIKE(como, em inglês - comparar com "*as*" acima) é usado em uma instrução comWHERE ou HAVING(como parte deGROUP BY) para limitar as linhas selecionadas aos itens quando uma coluna possui um determinado padrão de caracteres contido nela.

Este SQL selecionará os alunos que tem FullName começando com "Monique" ou que terminam com "Greene".

```
SELECT studentID, FullName, sat_score, rcd_updated
FROM student
WHERE
    FullName LIKE 'Monique%' OR
    FullName LIKE '%Greene';
```

```
+-----------+---------------+-----------+---------------------+
| studentID | FullName      | sat_score | rcd_updated         |
+-----------+---------------+-----------+---------------------+
|         1 | Monique Davis |       400 | 2017-08-16 15:34:50 |
|         5 | Alvin Greene  |      1200 | 2017-08-16 15:34:50 |
+-----------+---------------+-----------+---------------------+
2 rows in set (0.00 sec)
```

Você pode colocar NOT antes de LIKE para excluir as linhas com o padrão de string em vez de selecioná-las. Esse SQL exclui registros que contenham "encer Pauti" e "Ted" na coluna FullName.

```
SELECT studentID, FullName, sat_score, rcd_updated
FROM student
WHERE FullName NOT LIKE '%encer Pauti%' AND FullName NOT LIKE '%"Ted"%';
```

```
+-----------+----------------------+-----------+---------------------+
| studentID | FullName             | sat_score | rcd_updated         |
+-----------+----------------------+-----------+---------------------+
|         1 | Monique Davis        |       400 | 2017-08-16 15:34:50 |
|         2 | Teri Gutierrez       |       800 | 2017-08-16 15:34:50 |
|         4 | Louis Ramsey         |      1200 | 2017-08-16 15:34:50 |
|         5 | Alvin Greene         |      1200 | 2017-08-16 15:34:50 |
|         6 | Sophie Freeman       |      1200 | 2017-08-16 15:34:50 |
|         8 | Donald D. Chamberlin |      2400 | 2017-08-16 15:35:33 |
|         9 | Raymond F. Boyce     |      2400 | 2017-08-16 15:35:33 |
+-----------+----------------------+-----------+---------------------+
7 rows in set (0.00 sec)
```