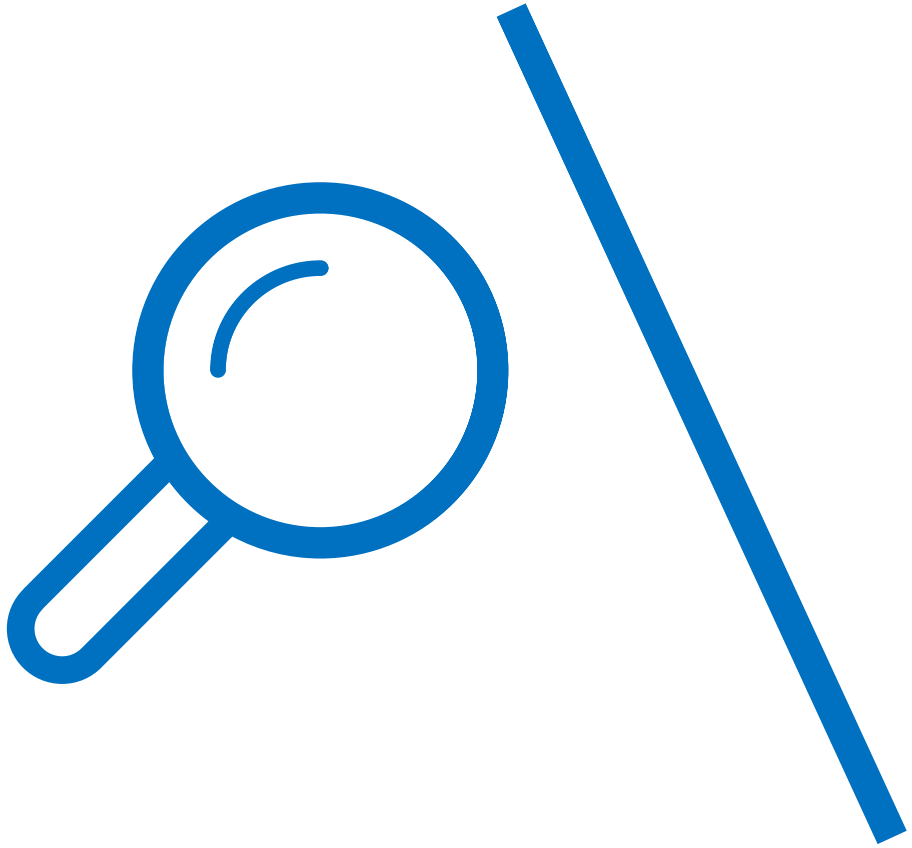
Start-Tech Academy

PostgreSQL

SQL

What is SQL



- ➢ Computer Language used for
  - Storing
  - Manipulating
  - Retrieving Data
- ➢ Invented by **IBM**
- ➢ SQL stands for **Structured Query Language**

Why SQL

Large Amount of Data

Controlled access

Data Manipulation

Business Insights

Start-Tech ACADEMY

Who uses SQL

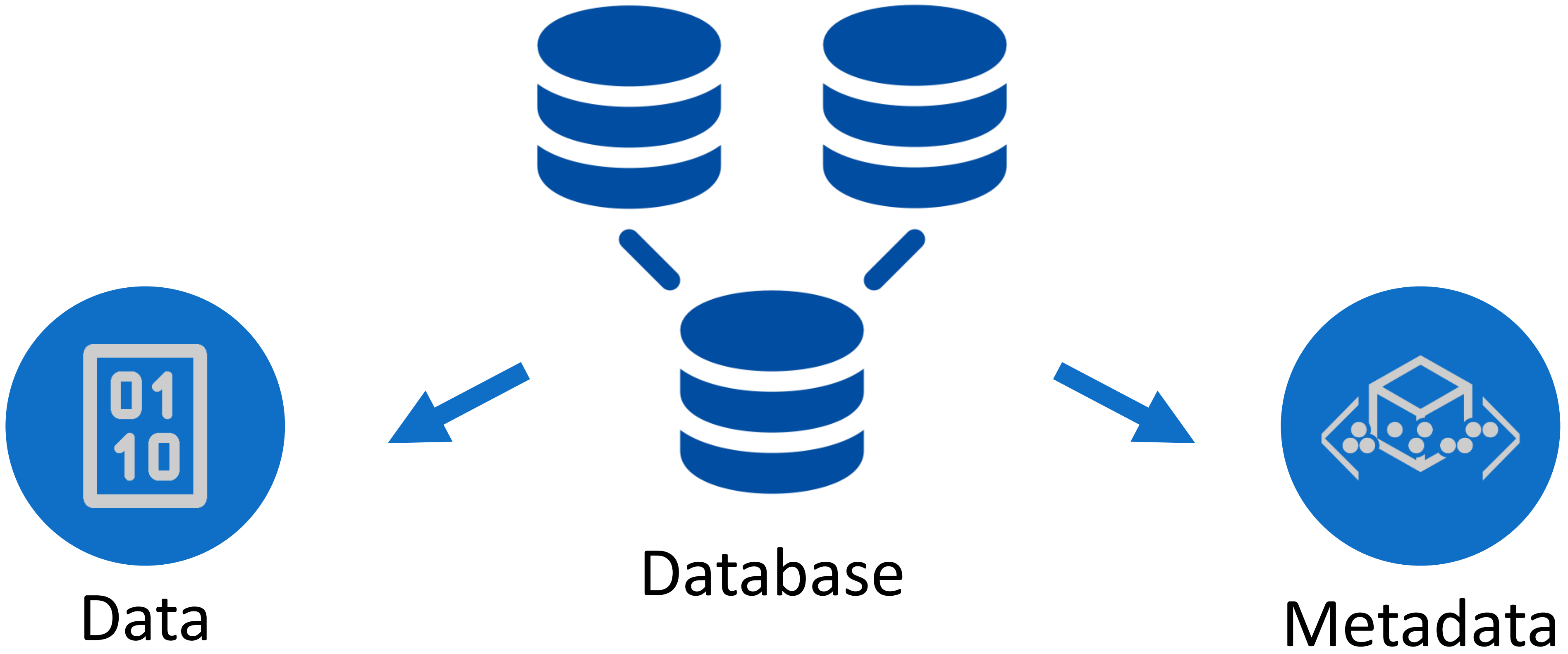
Software developers

Database Managers

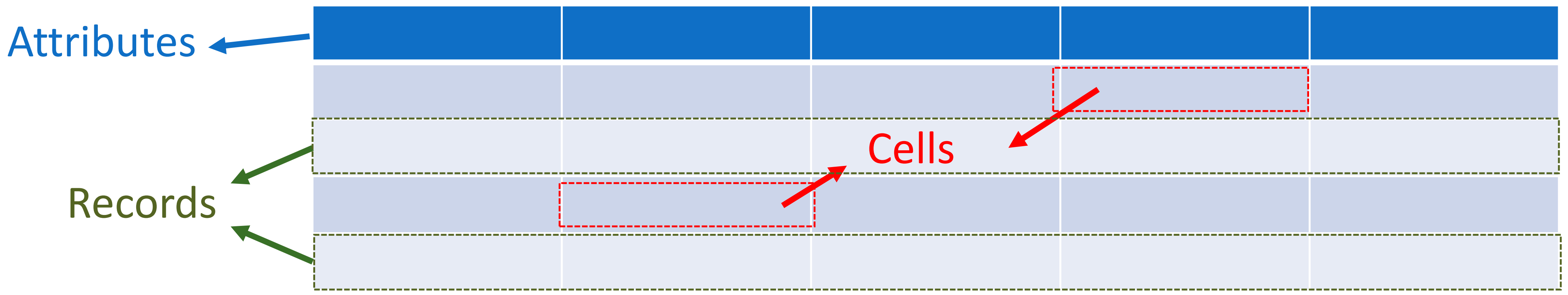
Business Managers

SQL : Lecture 2

Database



Tables



Example

Student	Age	Gender	Score	Rank
Student 1	18	M	89	2
Student 2	17	F	90	1
Student 3	19	M	72	3
Student 4	20	F	54	4

DBMS

DBMS
Database Management Systems

- ➢ It allows creation of new DB and their data structures
- ➢ Allows modification of data
- ➢ Allows retrieval of Data
- ➢ Allows Storage over long period of time
- ➢ Enables recovery in times of failure
- ➢ Control access to users

Start-Tech Academy

PostgreSQL

SQL : Lecture 3

SQL Queries

SQL Queries

1. DDL – Data Definition Language

   CREATE, ALTER, DROP
2. DML – Data Manipulation Language

   INSERT, UPDATE, DELETE
3. DQL – Data Query Language

   SELECT, ORDER BY, GROUP BY
4. DCL – Data Control Language

   GRANT, REVOKE
5. TCC – Transactional Control Commands

   COMMIT, ROLLBACK

Start-Tech Academy

Create

Creating a basic table involves naming the table and defining its columns and each column's data type.

Syntax

```
CREATE TABLE      “table_name”(

“column 1“        "data type for column 1"     [column 1 constraint(s)],

“column 2“        "data type for column 2"     [column 2 constraint(s)],
...
“column n “
[table constraint(s)] );
```

Start-Tech
ACADEMY

Create

Creating a basic table involves naming the table and defining its columns and each column's data type.

Constraints

- NOT NULL Constraint: Ensures that a column cannot have NULL value.
- DEFAULT Constraint: Provides a default value for a column when none is specified.
- UNIQUE Constraint: Ensures that all values in a column are different.
- CHECK Constraint: Makes sure that all values in a column satisfy certain criteria.
- Primary Key Constraint: Used to uniquely identify a row in the table.
- Foreign Key Constraint: Used to ensure referential integrity of the data.

Create

Creating a basic table involves naming the table and defining its columns and each column's data type.

Keys

- A primary key is used to uniquely identify each row in a table.
- A primary key can consist of one or more columns on a table.
- When multiple columns are used as a primary key, they are called a composite key.
- A foreign key is a column (or columns) that references a column (most often the primary key) of another table.
- The purpose of the foreign key is to ensure referential integrity of the data.

Create

Creating a basic table involves naming the table and defining its columns and each column's data type.

Keys

Customer Table

Column Name	Characteristic
Cust_ID	Primary Key
Last_Name	
First_Name	

Order Table

Column Name	Characteristic
Order_ID	Primary Key
Order_Date	
Customer_SID	Foreign Key
Amount	

Start-Tech ACADEMY

Start-Tech Academy

PostgreSQL

SQL

What is PostgreSQL

PostgreSQL

PostgreSQL is an advanced object-relational database management system that supports an extended subset of the SQL standard, including transactions, foreign keys, subqueries, triggers, user-defined types and functions.

Companies using PostgreSQL

Instagram	Netflix	Instacart
Spotify	Uber Technologies	reddit

Start-Tech ACADEMY

Why PostgreSQL

Why PostgreSQL

- Completely Open source
- Complete ACID Compliance
- Comprehensive documentation and active discussion forums
- PostgreSQL performance is utilized best in systems requiring execution of complex queries
- PostgreSQL is best suited for Data Warehousing and data analysis applications that require fast read/write speeds
- Supported by all major cloud service providers, including Amazon, Google, & Microsoft

Start-Tech Academy

INSERT

The INSERT INTO statement is used to add new records into a database table

Syntax

```
INSERT INTO "table_name" ("column1", "column2", ...)
VALUES ("value1", "value2", ...);
```

Start-Tech ACADEMY

INSERT

The INSERT INTO statement is used to add new records into a database table

Example

- *Single row (without column names specified)*

  ```
  INSERT INTO customer_table
  VALUES (1, ‘bee', ’cee’, 32, ‘bc@xyz.com’ );
  ```

- *Single row (with column names specified)*

  ```
  INSERT INTO customer_table ( cust_id, first_name, age, email_id)
  VALUES (2, ‘dee', 22, ‘d@xyz.com’);
  ```

- *Multiple rows*

  ```
  INSERT INTO customer_table
  VALUES (1, ‘ee', ’ef’, 35, ‘ef@xyz.com’ ),
  (1, ‘gee', ’eh’, 42, ‘gh@xyz.com’ ),
  (1, ‘eye', ’jay’, 62, ‘ij@xyz.com’ ),
  (1, ‘kay', ’el’, , ‘el@xyz.com’ );
  ```

Start-Tech
ACADEMY

COPY

The basic syntax to import data from CSV file into a table using COPY statement is as below

Syntax

```
COPY "table_name" ("column1", "column2", ...)
FROM 'C:\tmp\persons.csv' DELIMITER ',' CSV HEADER;
```

Another option is to use PG Admin

Start-Tech Academy

SELECT

The SELECT statement is used to fetch the data from a database table which returns this data in the form of a result table. These result tables are called **result-sets**.

Syntax

```
SELECT "column_name1", "column_name2", "column_name3" FROM
"table_name";

SELECT * FROM "table_name";
```

Start-Tech
ACADEMY

SELECT

The SELECT statement is used to fetch the data from a database table

Example

- Select one column

  ```
  SELECT first_name FROM customer_table;
  ```

- Select multiple columns

  ```
  SELECT first_name, last_name FROM customer_table;
  ```

- Select all columns

  ```
  SELECT * FROM customer_table;
  ```

Start-Tech ACADEMY

SELECT DISTINCT

The DISTINCT keyword is used in conjunction with the SELECT statement to eliminate all the duplicate records and fetching only unique records.

Syntax

```
SELECT DISTINCT "column_name"
FROM "table_name";
```

SELECT DISTINCT

The DISTINCT keyword is used in conjunction with the SELECT statement to eliminate all the duplicate records and fetching only unique records.

Example

- Select one column
```
SELECT DISTINCT customer_name FROM customer_table;
```
- Select multiple columns
```
SELECT DISTINCT customer_name, age FROM customer_table;
```

Start-Tech Academy

WHERE

The SQL WHERE clause is used to specify a condition while fetching the data from a single table or by joining with multiple tables. If the given condition is satisfied, then only it returns a specific value from the table.

Syntax

```
SELECT "column_name"
FROM "table_name"
WHERE "condition";
```

WHERE

The SQL WHERE clause is used to specify a condition while fetching the data from a single table or by joining with multiple tables. If the given condition is satisfied, then only it returns a specific value from the table.

Example

- Equals to condition
  ```
  SELECT first_name FROM customer_table WHERE age = 25;
  ```
- Less than/ Greater than condition
  ```
  SELECT first_name, age FROM customer_table WHERE age>25;
  ```
- Matching text condition
  ```
  SELECT * FROM customer_table WHERE first_name = “John”;
  ```

Start-Tech Academy

AND & OR

The SQL AND & OR operators are used to combine multiple conditions to narrow data in an SQL statement. These two operators are called as the conjunctive operators.

Syntax

```
SELECT "column_name"
FROM "table_name"
WHERE "simple condition"
{ [AND|OR] "simple condition"}+;
```

Start-Tech
ACADEMY

AND & OR

The SQL AND & OR operators are used to combine multiple conditions to narrow data in an SQL statement. These two operators are called as the conjunctive operators.

Example

```
SELECT first_name, last_name, age
FROM customer_table
WHERE age>20
AND age<30;

SELECT first_name, last_name, age
FROM customer_table
WHERE age<20
OR age>30
OR first_name = ‘John’;
```

NOT

NOT condition is used to negate a condition in a SELECT, INSERT, UPDATE, or DELETE statement.

Syntax

```
SELECT "column_name"
FROM "table_name"
WHERE NOT "simple condition"
```

NOT

NOT condition is used to negate a condition in a SELECT, INSERT, UPDATE, or DELETE statement.

Example

```
SELECT first_name,last_name, age
FROM employee
WHERE  NOT age=25

SELECT first_name,last_name, age
FROM employee
WHERE  NOT age=25
AND NOT first_name = ‘JAY’;
```

Start-Tech Academy

UPDATE

The SQL UPDATE Query is used to modify the existing records in a table.

Syntax

```
UPDATE "table_name"
SET column_1 = [value1], column_2 = [value2], ...
WHERE "condition";
```

UPDATE

The SQL UPDATE Query is used to modify the existing records in a table.

Example

- *Single row (with column names specified)*

```
UPDATE Customer_table
SET Age = 17, Last_name = ‘Pe'
WHERE Cust_id = 2;
```

- *Multiple rows*

```
UPDATE Customer_table
SET email_id = ‘gee@xyz.com
WHERE First_name = ‘Gee’ or First_name = ‘gee’;
```

Start-Tech
ACADEMY

DELETE

The DELETE Query is used to delete the existing records from a table.

Syntax

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

Start-Tech
ACADEMY

DELETE

The DELETE Query is used to delete the existing records from a table.

Example

- *Single row*

```
DELETE FROM CUSTOMERS
WHERE ID = 6;
```

- *Multiple rows*

```
DELETE FROM CUSTOMERS
WHERE age>25;
```

- *All rows*

```
DELETE FROM CUSTOMERS;
```

Start-Tech Academy

ALTER

The ALTER TABLE statement is used to change the definition or structure of an existing table

Syntax

ALTER TABLE "table_name"
[Specify Actions];

Following actions can be performed

- Columns – Add, Delete (Drop), Modify or Rename
- Constraints – Add, Drop
- Index – Add, Drop

COLUMN – ADD & DROP

The basic syntax of an ALTER TABLE command to add/drop a **Column** in an existing table is as follows.

Syntax

```
ALTER TABLE "table_name"
ADD "column_name" "Data Type";

ALTER TABLE "table_name"
DROP "column_name";
```

COLUMN – MODIFY & RENAME

The basic syntax of an ALTER TABLE command to Modify/Rename a **Column** in an existing table is as follows.

Syntax

```
ALTER TABLE "table_name"

ALTER COLUMN "column_name" TYPE "New Data Type";
```

```
ALTER TABLE "table_name"

RENAME COLUMN "column 1" TO "column 2";
```

CONSTRAINT – ADD & DROP

The basic syntax of an ALTER TABLE command to add/drop a **Constraint** on a existing table is as follows.

Syntax

1. ALTER TABLE "table_name“ ALTER COLUMN “column_name” SET NOT NULL;
2. ALTER TABLE "table_name“ ALTER COLUMN “column_name” DROP NOT NULL;
3. ALTER TABLE "table_name" ADD CONSTRAINT “column_name” CHECK (“column_name”>=100);
4. ALTER TABLE "table_name" ADD PRIMARY KEY (“column_name”);
5. ALTER TABLE “child_table" ADD CONSTRAINT “child_column”
   FOREIGN KEY (“parent column”) REFERENCES “parent table”;

Start-Tech Academy

IN

IN condition is used to help reduce the need to use multiple OR conditions in a SELECT, INSERT, UPDATE, or DELETE statement.

Syntax

```
SELECT "column_name"
FROM "table_name"
WHERE "column_name" IN ('value1', 'value2', ...);
```

IN

IN condition is used to help reduce the need to use multiple OR conditions in a SELECT, INSERT, UPDATE, or DELETE statement.

Example

```
SELECT *
FROM customer
WHERE city IN ('Philadelphia', 'Seattle')

SELECT *
FROM customer
WHERE city = 'Philadelphia' OR city = 'Seattle';
```

Start-Tech Academy

BETWEEN

The BETWEEN condition is used to retrieve values within a range in a SELECT, INSERT, UPDATE, or DELETE statement.

Syntax

```
SELECT "column_name"
FROM "table_name"
WHERE "column_name" BETWEEN 'value1' AND 'value2';
```

Start-Tech
ACADEMY

BETWEEN

The BETWEEN condition is used to retrieve values within a range in a SELECT, INSERT, UPDATE, or DELETE statement.

Example

```
SELECT * FROM customer
WHERE age BETWEEN 20 AND 30;
```

Which is same as

```
SELECT * FROM customer
WHERE age>= 20 AND age<= 30;
```

```
SELECT * FROM customer
WHERE age NOT BETWEEN 20 and 30;
```

```
SELECT * FROM sales
WHERE ship_date BETWEEN ‘2015-04-01' AND ‘2016-04-01';
```

Start-Tech ACADEMY

Start-Tech Academy

LIKE

The PostgreSQL LIKE condition allows you to perform pattern matching using Wildcards.

Syntax

```
SELECT "column_name"
FROM "table_name"
WHERE "column_name" LIKE {PATTERN};
```

{PATTERN} often consists of wildcards

WILDCARDS

The PostgreSQL LIKE condition allows you to perform pattern matching using Wildcards.

Example

Wildcard	Explanation
%	Allows you to match any string of any length (including zero length)
_	Allows you to match on a single character

A% means starts with A like ABC or ABCDE
%A means anything that ends with A
A%B means starts with A but ends with B

AB_C means string starts with AB, then there is one character, then there is C

Start-Tech ACADEMY

LIKE

The PostgreSQL LIKE condition allows you to perform pattern matching using Wildcards.

Example

```
SELECT * FROM customer_table
WHERE first_name LIKE 'Jo%’;

SELECT * FROM customer_table
WHERE first_name LIKE '%od%’;

SELECT first_name, last_name FROM customer_table
WHERE first_name LIKE 'Jas_n’;

SELECT first_name, last_name FROM customer_table
WHERE last_name NOT LIKE 'J%’;

SELECT * FROM customer_table
WHERE last_name LIKE 'G\%';
```

Start-Tech ACADEMY

Start-Tech Academy

ORDER BY

The ORDER BY clause is used to sort the records in result set. It can only be used in SELECT statements.

Syntax

```
SELECT "column_name"
FROM "table_name"
[WHERE "condition"]
ORDER BY "column_name" [ASC, DESC];
```

It is possible to order by more than one column.

```
ORDER BY "column_name1" [ASC, DESC], "column_name2" [ASC, DESC]
```

ORDER BY

The ORDER BY clause is used to sort the records in result set. It can only be used in SELECT statements.

Example

```
SELECT * FROM customer
WHERE state = 'California’ ORDER BY Customer_name;
```

Same as

```
SELECT * FROM customer
WHERE state = 'California’ ORDER BY Customer_name ASC;

SELECT * FROM customer
ORDER BY 2 DESC;

SELECT * FROM customer
WHERE age>25 ORDER BY City ASC, Customer_name DESC;

SELECT * FROM customer
ORDER BY age;
```

Start-Tech
ACADEMY

Start-Tech Academy

LIMIT

LIMIT statement is used to limit the number of records returned based on a limit value.

Syntax

```
SELECT "column_names"
FROM "table_name"
[WHERE conditions]
[ORDER BY expression [ ASC | DESC ]]
LIMIT row_count;
```

LIMIT

LIMIT statement is used to limit the number of records returned based on a limit value.

Example

```
SELECT * FROM customer
WHERE age >= 25
ORDER BY age DESC
LIMIT 8;

SELECT * FROM customer
WHERE age >=25
ORDER BY age ASC
LIMIT 10;
```

Start-Tech Academy

AS

The keyword **AS** is used to assign an alias to the column or a table. It is inserted between the column name and the column alias or between the table name and the table alias.

Syntax

```
SELECT column_name" AS "column_alias"
FROM "table_name“;
```

AS

The keyword **AS** is used to assign an alias to the column or a table. It is inserted between the column name and the column alias or between the table name and the table alias.

Example

```
SELECT Cust_id AS “Serial number”, Customer_name as name, Age as
Customer_age
FROM Customer ;
```

Start-Tech Academy

COUNT

Count function returns the count of an expression

Syntax

```
SELECT "column_name1", COUNT ("column_name2")
FROM "table_name"
```

Start-Tech
ACADEMY

COUNT

Count function returns the count of an expression

Example

```
SELECT COUNT(*) FROM sales;

SELECT COUNT (order_line) as "Number of Products Ordered",
 COUNT (DISTINCT order_id) AS "Number of Orders"
FROM sales WHERE customer_id = 'CG-12520';
```

Start-Tech Academy

SUM

Sum function returns the summed value of an expression

Syntax

```
SELECT sum(aggregate_expression)
FROM tables
[WHERE conditions];
```

Start-Tech ACADEMY

SUM

Sum function returns the summed value of an expression

Example

```
SELECT sum(Profit) AS "Total Profit"
FROM sales;

SELECT sum(quantity) AS “Total Quantity"
FROM orders where product_id = ‘FUR-TA-10000577’;
```

Start-Tech ACADEMY

Start-Tech Academy

AVERAGE

AVG function returns the average value of an expression.

Syntax

```
SELECT avg(aggregate_expression)
FROM tables
[WHERE conditions];
```

AVERAGE

AVG function returns the average value of an expression.

Example

```
SELECT avg(age) AS "Average Customer Age"
FROM customer;

SELECT avg(sales * 0.10) AS "Average Commission Value"
FROM sales;
```

Start-Tech
ACADEMY

Start-Tech Academy

MIN/MAX

MIN/MAX function returns the minimum/maximum value of an expression.

Syntax

```
SELECT min(aggregate_expression)
FROM tables
[WHERE conditions];

SELECT max(aggregate_expression)
FROM tables
[WHERE conditions];
```

MIN/MAX

MIN/MAX function returns the minimum/maximum value of an expression.

Example

```
SELECT MIN(sales) AS Min_sales_June15
FROM sales
WHERE order_date BETWEEN '2015-06-01' AND '2015-06-30';

SELECT MAX(sales) AS Min_sales_June15
FROM sales
WHERE order_date BETWEEN '2015-06-01' AND '2015-06-30';
```