# SQL Statements - Syntax, Examples, and Descriptions

## SQL CREATE DATABASE

*Syntax:*

```
CREATE DATABASE database_name;
```

Example:

```
CREATE DATABASE EmployeeDB;
```

Description:

This statement creates a new database named EmployeeDB. The database will store tables and other objects like views and procedures.

## SQL DROP DATABASE

*Syntax:*

```
DROP DATABASE database_name;
```

Example:

```
DROP DATABASE EmployeeDB;
```

Description:

This statement deletes the database EmployeeDB. All tables and data within this database will be permanently removed.

## SQL CREATE TABLE

*Syntax:*

```
CREATE TABLE table_name (
    column1 datatype,
    column2 datatype,
    ...
);
```

Example:

```
CREATE TABLE Employees (
    Emp_ID INT PRIMARY KEY,
    First_Name VARCHAR(30),
    Last_Name VARCHAR(30),
    Salary DECIMAL(10, 2)
);
```

Description:

This statement creates a table called Employees with four columns: Emp_ID, First_Name, Last_Name, and Salary. The Emp_ID column is set as the primary key, and Salary is set to store decimal values.

## SQL INSERT INTO

*Syntax:*

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

Example:

```
INSERT INTO Employees (Emp_ID, First_Name, Last_Name, Salary) VALUES (1, 'John', 'Doe', 50000.00);
```

Description:

This statement inserts a new row into the Employees table with Emp_ID = 1, First_Name = 'John', Last_Name = 'Doe', and Salary = 50000.00.

## SQL SELECT

*Syntax:*

```
SELECT column1, column2, ... FROM table_name;
```

Example:

```
SELECT * FROM Employees;
```

Description:

This statement selects all columns from the Employees table. The * is a wildcard that selects all fields.

## SQL UPDATE

*Syntax:*

*UPDATE table_name SET column1 = value1, column2 = value2, ... WHERE condition;*

Example:

UPDATE Employees SET Salary = 55000.00 WHERE Emp_ID = 1;

Description:

This statement updates the Salary of the employee with Emp_ID = 1 to 55000.00. The WHERE clause ensures that only the specified employee's salary is changed.

## SQL DELETE

*Syntax:*

*DELETE FROM table_name WHERE condition;*

Example:

DELETE FROM Employees WHERE Emp_ID = 1;

Description:

This statement deletes the employee with Emp_ID = 1 from the Employees table.

## SQL ALTER TABLE

*Syntax:*

*ALTER TABLE table_name ADD column_name datatype;*

Example:

ALTER TABLE Employees ADD Hire_Date DATE;

Description:

This statement adds a new column Hire_Date of type DATE to the Employees table.

## SQL DROP TABLE

*Syntax:*

*DROP TABLE table_name;*

Example:

DROP TABLE Employees;

Description:

This statement deletes the Employees table and all the data within it from the database.

## SQL TRUNCATE TABLE

*Syntax:*

*TRUNCATE TABLE table_name;*

Example:

TRUNCATE TABLE Employees;

Description:

This statement removes all rows from the Employees table but keeps the table structure intact for future use.

## SQL SELECT DISTINCT

*Syntax:*

*SELECT DISTINCT column1, column2, ... FROM table_name;*

Example:

SELECT DISTINCT Last_Name FROM Employees;

Description:

This statement selects unique (distinct) values from the Last_Name column of the Employees table, eliminating duplicates.

## SQL COUNT

*Syntax:*

*SELECT COUNT(column_name) FROM table_name WHERE condition;*

Example:

SELECT COUNT(*) FROM Employees WHERE Salary > 40000;

Description:

This statement counts the number of rows in the Employees table where the Salary is greater than 40000.

## SQL GROUP BY

*Syntax:*

*SELECT column1, COUNT(*) FROM table_name GROUP BY column1;*

Example:

SELECT Last_Name, COUNT(*) FROM Employees GROUP BY Last_Name;

Description:

This statement groups employees by their Last_Name and counts how many employees have the same last name.

## SQL ORDER BY

*Syntax:*

*SELECT column1, column2, ... FROM table_name ORDER BY column1 [ASC|DESC];*

Example:

SELECT * FROM Employees ORDER BY Salary DESC;

Description:

This statement selects all rows from the Employees table and orders the results by Salary in descending order.

## SQL JOIN

*Syntax:*

*SELECT columns FROM table1 INNER JOIN table2 ON table1.column = table2.column;*

Example:

SELECT Employees.First_Name, Departments.Dept_Name FROM Employees INNER JOIN Departments ON Employees.Emp_ID = Departments.Emp_ID;

Description:

This statement performs an inner join between the Employees and Departments tables, showing the first name of the employee and their department name, where Emp_ID matches in both tables.

## SQL LEFT JOIN

*Syntax:*

*SELECT columns FROM table1 LEFT JOIN table2 ON table1.column = table2.column;*

Example:

SELECT Employees.First_Name, Departments.Dept_Name FROM Employees LEFT JOIN Departments ON Employees.Emp_ID = Departments.Emp_ID;

Description:

This statement performs a left join, selecting all employees and their department names. If an employee doesn't belong to any department, NULL will be returned for the department name.

## SQL CREATE INDEX

*Syntax:*

*CREATE INDEX index_name ON table_name (column_name);*

Example:

CREATE INDEX idx_salary ON Employees(Salary);

Description:

This statement creates an index named idx_salary on the Salary column of the Employees table, which speeds up searches on Salary.

## SQL DROP INDEX

*Syntax:*

*DROP INDEX index_name ON table_name;*

Example:

DROP INDEX idx_salary ON Employees;

Description:

This statement deletes the index idx_salary from the Employees table, removing the performance optimization on the Salary column.

## SQL CREATE VIEW

*Syntax:*

*CREATE VIEW view_name AS SELECT columns FROM table_name WHERE condition;*

Example:

CREATE VIEW high_earners AS SELECT First_Name, Last_Name, Salary FROM Employees WHERE Salary > 50000;

Description:

This statement creates a view called high_earners, which displays only employees with a salary greater than 50000.

## SQL SUBQUERY

*Syntax:*

*SELECT column1, column2, ... FROM table_name WHERE column_name = (SELECT column_name FROM table_name WHERE condition);*

Example:

SELECT First_Name, Last_Name FROM Employees WHERE Salary > (SELECT AVG(Salary) FROM Employees);

Description:

This statement selects employees whose salary is greater than the average salary of all employees using a subquery that calculates the average salary.