SQL Queries

1. Create Statement

1.1 Create new database

```
CREATE DATABASE database_name;
```

Example:

```
CREATE DATABASE testDB;
```

1.2 Create new table

```
CREATE TABLE table_name (
   column1_name datatype,
   column2_name datatype,
   column3_name datatype,
  ....
);
```

Example:

```
CREATE TABLE tb_students (
   ID int NOT NULL PRIMARY KEY,
   firstname varchar(255) NOT NULL,
   lastname varchar(255),
   age int
);
```

---

2. Insert into table

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

```
INSERT INTO tb_students (firstname, lastname, age)
VALUES (Ali, Ahmed, 25);
```

---

3. Drop Statement

3.1 Drop database

```
DROP DATABASE database_name;
```

Example:

```
DROP DATABASE testDB;
```

3.2 Drop Table

```
DROP TABLE table_name;
```

Example:

```
DROP TABLE tb_students;
```

---

4. Delete Statement

4.1 Delete specific rows from table

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

Example:

```
DELETE FROM tb_students WHERE id = 1;
```

4.2 Delete all rows from table

```
DELETE FROM table_name;
```

Example:

```
DELETE FROM tb_students;
```

---

5. Alter Table

5.1 Alter table – Add new column

```
ALTER TABLE table_name
ADD column_name datatype;
```

Example:

```
ALTER TABLE tb_students
ADD email varchar(255);
```

5.2 Alter table – delete column in a table

```
ALTER TABLE table_name
DROP COLUMN column_name;
```

Example:

```
ALTER TABLE tb_students
DROP COLUMN email;
```

5.3 Alter table – alter/modify a column

```
ALTER TABLE table_name
ALTER COLUMN column_name datatype;
```

Example:

```
ALTER TABLE tb_students
ALTER COLUMN firstname varchar(55);
```

---

6. Update

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

Example:

```
UPDATE tb_students
SET firstname = 'Sara'
WHERE id=2;
```

---

7. Select Statement

7.1 Select specific columns

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

Example:

```
SELECT firstname, age
FROM tb_students;
```

7.2 Select all columns in table

```
SELECT * FROM table_name;
```

7.3 WHERE clause

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

Example:

```
SELECT firstname, lastname
FROM tb_students
WHERE name='Sara';
```

7.4 Distinct

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

Example:

```
SELECT DISTINCT firstname
FROM tb_students;
```

7.5 Top

```
SELECT TOP number column1, column2, ...
FROM table_name
WHERE condition;
```

Example:

```
SELECT TOP 3 * FROM tb_students;
```

7.6 MIN() and MAX() Functions

```
SELECT MIN(column_name)
FROM table_name
WHERE condition;
```

```
SELECT MAX(column_name)
FROM table_name
WHERE condition;
```

Example:

```
SELECT MIN(age)
FROM tb_students
```

```
SELECT MAX(age)
FROM tb_students
```

7.7 COUNT(), AVG() and SUM() Functions

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

```
SELECT AVG(column_name)
FROM table_name
WHERE condition;
```

```
SELECT SUM(column_name)
FROM table_name
WHERE condition;
```

Example:

```
SELECT COUNT(firstname)
FROM tb_students
WHERE firstname = 'Ali';
```

```
SELECT AVG(age)
FROM tb_students
WHERE age > 18;
```

```
SELECT SUM(age)
FROM tb_students
WHERE age < 30;
```

7.8 Order by keyword

```
SELECT column1, column2, ...
FROM table_name
ORDER BY column1, column2, ... ASC|DESC;
```

Example:

```
SELECT firstname, lastname
FROM tb_students
ORDER BY age DESC;
```

7.9 LIKE Operator

(%) represents zero, one, or multiple characters

(_) represents a single character

```
SELECT column1, column2, ...
FROM table_name
WHERE column LIKE pattern;
```

Example:

```
SELECT firstname
FROM tb_students
WHERE firstname LIKE %a;
```

7.10 SQL Aliases

SQL aliases are used to give a table or a column in a table, a temporary name.

7.10.1 Alias Column

```
SELECT column_name AS alias_name
FROM table_name;
```

Example:

```
SELECT firstname AS student_name
FROM tb_students;
```

7.10.2 Alias Table

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

Example:

```
SELECT firstname,lastname,age
FROM tb_students AS Students;
```

7.11 SQL AND, OR and NOT Operators

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 AND condition2 AND condition3 ...;
```

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 OR condition2 OR condition3 ...;
```

```
SELECT column1, column2, ...
FROM table_name
WHERE NOT condition;
```

Example:

```
SELECT firstname
FROM tb_students
WHERE lastname='Ahmed' AND age>30;
```

```
SELECT firstname
FROM tb_students
WHERE lastname='Ahmed' OR lastname='Ali';
```

```
SELECT firstname
FROM tb_students
WHERE NOT age = 30;
```

7.12 BETWEEN Operator

The BETWEEN operator is inclusive: begin and end values are included.

```
SELECT column1, column2, ...
FROM table_name
WHERE column_name BETWEEN value1 AND value2;
```

Example:

```
SELECT firstname
FROM tb_students
WHERE age BETWEEN 15 AND 30;
```

7.13 IN Operator

```
SELECT column1, column2, ...
FROM table_name
WHERE column_name IN (value1, value2, ...);
```

Or:

```
SELECT column1, column2, ...
FROM table_name
WHERE column_name IN (SELECT STATEMENT);
```

Example:

```
SELECT firstname,lastname
FROM tb_students
WHERE firstname IN (‘Ahmed’, ‘Ali, ‘Sara’);
```

```
SELECT firstname,lastname
FROM Customer
WHERE Country IN (SELECT Country FROM Supplier);
```

7.14 NULL Values

7.14.1 IS NULL

```
SELECT column1, column2, ...
FROM table_name
WHERE column_name IS NULL;
```

Example:

```
SELECT firstname
FROM tb_students
WHERE age IS NULL;
```

7.14.2 IS NOT NULL Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE column_name IS NOT NULL;
```

Example:

```
SELECT firstname
FROM tb_students
WHERE age IS NOT NULL;
```

---

8. GROUP BY and HAVING

8.1 GROUP BY Statement

The GROUP BY statement is often used with aggregate functions (COUNT, MAX, MIN, SUM, AVG) to group the result-set by one or more columns.

```
SELECT column1, column2, ...
FROM table_name
WHERE condition
GROUP BY column_name(s)
ORDER BY column_name(s);
```

Example:

```
SELECT COUNT(id),firstname,lastname
FROM tb_students
WHERE age>15
GROUP BY age
ORDER BY firstname;
```

8.2 HAVING Clause

The HAVING clause was added to SQL because the WHERE keyword could not be used with aggregate functions.

```
SELECT column1, column2, ...
FROM table_name
WHERE condition
GROUP BY column_name(s)
HAVING condition
ORDER BY column_name(s);
```

Example:

```
SELECT COUNT(id),firstname,lastname
FROM tb_students
GROUP BY age
HAVING COUNT(id) >5;
```