SQL Queries

# CREATE DATABASE

The **SQL CREATE DATABASE** statement is used to create a database.

CREATE DATABASE database\_name;

# CREATE TABLE

SQL CREATE TABLE statement is used to create table in a database.

* create table "tablename"
* ("column1" "data type",
* "column2" "data type",
* "column3" "data type",
* ...
* "columnN" "data type");

I have created the following table

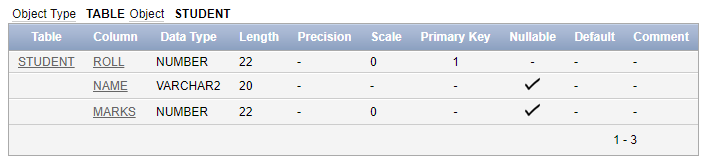
create table student(

roll int,

name varchar(20),

marks int

)



# INSERT

The INSERT statement inserts data from into the table.

* INSERT requires that data types in table matches the given values.

INSERT INTO table\_name(column1, column2,….,coloumnN)

VALUES ( value1, value2....valueN);

# SELECT

The SELECT statement is used to select data from a database.

The data returned is stored in a result table, called the result-set.

SELECT column1, column2, ...  
FROM table\_name;

# SELECT \*

Select all the fields available in the table

SELECT \* FROM table\_name;

# SELECT DISTINCT

The SELECT DISTINCT statement is used to return only distinct (different) values.

SELECT DISTINCT column1, column2, ...  
FROM table\_name;

# WHERE

The WHERE clause is used to filter records.

The WHERE clause is used to extract only those records that fulfill a specified condition.

SELECT column1, column2, ...  
FROM table\_name  
WHERE condition;

# UPDATE

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

UPDATE *table\_name*  
SET *column1* = *value1*, *column2* = *value2*, ...  
WHERE *condition*;

# DELETE

The DELETE statement is used to delete existing records in a table.

DELETE FROM table\_name  
WHERE condition;

# TRUNCATE

The TRUNCATE TABLE statement is used to delete the data inside a table, but not the table itself.

TRUNCATE TABLE table\_name;

# DROP DATABASE

The following SQL statement drops the existing database "testDB":

DROP DATABASE testDB;

# INSERT INTO

The INSERT INTO SELECT statement copies data from one table and inserts it into another table.

* INSERT INTO SELECT requires that data types in source and target tables match
* The existing records in the target table are unaffected.

INSERT INTO table2  
SELECT \* FROM table1WHERE condition;

# SELECT INTO

The SELECT INTO statement copies data from one table into a new table.

SELECT *column1*, *column2*, *column3*, ...  
INTO *newtable* [IN *externaldb*]  
FROM *oldtable*WHERE *condition;*

# SELECT TOP

The SELECT TOP clause is used to specify the number of records to return.

SELECT TOP number|percent column\_name(s)  
FROM table\_nameWHERE condition;

# AND/OR

The SQL AND/OR condition is used in SQL query to create two or more conditions to be met.

* SELECT columns FROM tables WHERE condition1 AND condition2;
* SELECT columns FROM tables WHERE condition1 OR condition2;

# ALTER TABLE

The ALTER TABLE statement is used to add, modify or delete columns in an existing table. It is also used to rename a table.

* ALTER TABLE table\_name ADD column\_name column-definition;
* ALTER TABLE table\_name MODIFY column\_name column\_type;
* ALTER TABLE table\_name DROP COLUMN column\_name;
* ALTER TABLE table\_name RENAME COLUMN old\_name to new\_name;

# AS

**SQL AS** is used to assign temporarily a new name to a table column or table.

* SELECT column AS “Alias” FROM table;
* SELECT column FROM table AS “Alias”

# BETWEEN

The BETWEEN operator is used to filter the result set within a certain range. The values can be numbers, text or dates.

SELECT column\_name(s)

FROM table\_name

WHERE column\_name BETWEEN value\_1 AND value\_2;

# CREATE INDEX

The CREATE INDEX statement is used to create indexes in tables.

Indexes are used to retrieve data from the database very fast. The users cannot see the indexes, they are just used to speed up searches/queries.

CREATE INDEX index\_name  
ON table\_name (column1, column2, ...);

# CREATE VIEW

In SQL, a view is a virtual table based on the result-set of an SQL statement.

A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.

CREATE VIEW view\_name AS  
SELECT column1, column2, ...  
FROM table\_name  
WHERE condition;

# DROP INDEX

The DROP INDEX statement is used to delete an index in a tablE.

DROP INDEX index\_name;

# DROP TABLE

The DROP TABLE statement is used to drop an existing table in a database.

DROP TABLE table\_name;

# EXISTS

The EXISTS operator is used to test for the existence of any record in a subquery.

The EXISTS operator returns true if the subquery returns one or more records.

SELECT column\_name(s)  
FROM table\_name  
WHERE EXISTS  
(SELECT column\_name FROM table\_name WHERE condition);

# GROUP BY

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 column\_name(s)  
FROM table\_name  
WHERE condition  
GROUP BY column\_name(s)ORDER BY column\_name(s);

# HAVING

The HAVING clause was added to SQL because the WHERE keyword could not be used with aggregate functions. It is used with the group by statement.

SELECT column\_name(s)  
FROM table\_name  
WHERE condition  
GROUP BY column\_name(s)HAVING conditionORDER BY column\_name(s);

# IN

The IN operator allows you to specify multiple values in a WHERE clause.

The IN operator is a shorthand for multiple OR conditions.

SELECT column\_name(s)  
FROM table\_name  
WHERE column\_name IN (value1, value2, ...);

# INNER JOIN

* The INNER JOIN keyword selects records that have matching values in both tables.

SELECT column\_name(s)  
FROM table1  
INNER JOIN table2 ON table1.column\_name = table2.column\_name;

# LEFT JOIN

The LEFT JOIN keyword returns all records from the left table (table1), and the matched records from the right table (table2). The result is NULL from the right side, if there is no match.

SELECT column\_name(s)  
FROM table1  
LEFT JOIN table2 ON table1.column\_name = table2.column\_name;

# RIGHT JOIN

The RIGHT JOIN keyword returns all records from the right table (table2), and the matched records from the left table (table1). The result is NULL from the left side, when there is no match.

SELECT column\_name(s)  
FROM table1  
RIGHT JOIN table2 ON table1.column\_name = table2.column\_name;

# FULL JOIN

The FULL OUTER JOIN keyword return all records when there is a match in either left (table1) or right (table2) table records.

SELECT column\_name(s)  
FROM table1  
FULL OUTER JOIN table2 ON table1.column\_name = table2.column\_name;

# ORDER BY

The ORDER BY keyword is used to sort the result-set in ascending or descending order.

The ORDER BY keyword sorts the records in ascending order by default. To sort the records in descending order, use the DESC keyword.

SELECT column1, column2, ...  
FROM table\_name  
ORDER BY column1, column2, ... ASC|DESC;

# LIKE

The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

There are two wildcards used in conjunction with the LIKE operator:

* % - The percent sign represents zero, one, or multiple characters
* \_ - The underscore represents a single character

SELECT column1, column2, ...  
FROM table\_name  
WHERE columnN LIKE pattern;

# UNION

The UNION operator is used to combine the result-set of two or more SELECT statements.

* Each SELECT statement within UNION must have the same number of columns
* The columns must also have similar data types
* The columns in each SELECT statement must also be in the same order.

SELECT column\_name(s) FROM table1  
UNION  
SELECT column\_name(s) FROM table2;

# UNION ALL

The UNION operator selects only distinct values by default. To allow duplicate values, use UNION ALL:

SELECT *column\_name(s)* FROM *table1*  
UNION ALL  
SELECT *column\_name(s)* FROM *table2*;