**MS SQL Server**

* **First, Download *MS SQL Server* from Microsoft website> choose developer version**

Note: MS SQL Server is actual database Engine.

* **Next, again download *MS SQL Server Management Studio***

Note: MS SQL Server Management studio is a editor where we write SQL command.

Integrity Constraints:

1. Not Null : restricting NULL values.
2. UNIQUE : restricting duplicate values (but it will accept NULL values.
3. PRIMARY KEY : restricting duplicate and NULL (Unique + Not NULL).
4. FOREIGN KEY : using for he child table and referring Parent table ( PRIMARY KEY)

**SQL Primary key:**

CREATE TABLE table\_name (id number(5)PRIMARY KEY, name char(20),age number(2));

**or**

CREATE TABLE table\_name (id number(5) CONSTRAINT constraint\_name PRIMARY KEY, name char(20),age number(2));

**SQL Foreign key or Referential Integrity:**

CREATE TABLE product\_id (p\_id number(5) CONSTRINT pk\_id PRIMARY KEY, name char(20),price number(2)……);

CREATE TABLE order\_items (item\_id number(5) CONSTRINT pk\_id PRIMARY KEY, p\_id number(5) CONSTRINT pk\_id REFERENCES, name char(20),age number(2));

**SQL Commands:**

SQL commands are instructions, coded into SQL statements, which are used to communicate with the data to perform **specific tasks, work, functions** and **queries** with data.

**Data Definition Language (DDL) –** These SQL commands are used for creating, modifying and dropping the structure of database objects.

Commands are **CREATE, ALTER, DROP, RENAME and TRUNCATE.**

**Data Manipulation Language (DML) –** These SQL commands are used for storing, retrieving, modifying and deleting data.

Commands are **SELECT, INSERT, UPDATE** and **DELETE.**

**Transaction Control Language (TCL) –** These SQL commands are used for managing changes affecting the data.

Commands are **COMMIT, ROLLBACK** and **SAVEPOINT.**

**Data Control Language (DCL)-** These SQL commands are used for providing security to database objects.

Commands are **GRANT** and **REVOKE.**

**Data types:**

Integer: **int, bigint, smallint, tinyint.**

Real numbers: **float, real.**

**date, time.**

**char(x) , varchar(x) and x is number of characters to store.**

**blob: used to store binary large data such as images and other files.**

**binary: used to store information in binary string format.**

**money: some databases also termed as currency. Used to store money or currency.**

**Comparison Operators:**

|  |  |
| --- | --- |
| **Comparison Operators** | **Description** |
| = | equal to |
| <>, != | is not equal to |
| < | less than |
| > | greater than |
| >= | greater than or equal to |
| <= | less than or equal to |

**Logical Operators:**

There are three Logical Operators namely **AND, OR** and **NOT**.

* **ALTER –** lets us adds columns to a table in database

ALTER TABLE table\_name ADD column\_name datatype;

* **AND –** is an **operator** that combines two conditions. Both conditions ust be true for the row to be included in the result set.

SELECT column\_name(s) FROM table\_name WHERE column\_1 = value\_1 AND column\_2 = value\_2;;

* **AS-** is a key word in SQL that allows to rename a **column or table** using ***alias.***

SELECT column\_name AS 'Alias' FROM table\_name;;

* **AUTO INCREMENT-** are used for auton generating values for particular column whenever new is being inserted. Very often the primary key of table needs to created automatically.

CREATE TABLE table\_name (columns1\_name int(4) PRIMARY KEY AUTOINCREMENT,column2\_name varchar(30)……);

**[Oracle uses SQUENCE to create AUTOINCREMENT]**

* **AVG() –** is an aggregate function that returns average value of the numeric column.

SELECT AVG(column\_name) FROM table\_name;;

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

SELECT column\_name(s) FROM table\_name WHERE column\_name BETWEEN value\_1 AND value\_2;;

* **CASE –** used to statements are used to create outputs( usually in the SELECT statements). It is SQL’S way of handling if -then logic.

SELECT column\_name, CASE WHEN condition THEN 'Result\_1' WHEN condition THEN 'Result\_2' ELSE 'Result\_3' END FROM table\_name;;

* **COUNT-** is a function takes the name of the column as an argument and counts the number of rows where the column is not NULL

SELECT COUNT(column\_name) FROM table\_name;

* **CREATE TABLE –** creates a new table in the database. It allows you specify the name of the

table and name of each column in the table.

CREATE TABLE table\_name ( column\_1 datatype, column\_2 datatype, column\_3 datatype );

;

* **CREATE DATABASE-** is used to create a database. ( After creating database we create create several database objects like tables, views, procedures)

CREATE DATABASE database\_name;

* **DELETE –** delete statements are used to remove rows from a table.

DELETE FROM table\_name WHERE some\_column = some\_value;

;

* **TRUNCATE-** deletes all the rows from the table

TRUNCATE TABLE table\_name;

* **DROP –** is used to remove an object from the database, if you DROP a table all the rows in table and the table structure is removed from database once table is dropped we cannot get it back.

Drop TABLE table\_name;

* **GROUP BY –** is a clause in SQL that is only used with aggregate functions. It is used in collaboration with SELECT to arrange identical data into groups.

SELECT column\_name, COUNT(\*) FROM table\_name GROUP BY column\_name,

;

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

SELECT column\_name, COUNT(\*) FROM table\_name GROUP BY column\_name HAVING COUNT(\*) > value;

;

* **INDEX -**  index in SQL is created on existing tables to retieve the rows quickly.

CREATE INDEX index\_name ON TABLE table\_name(column\_name1, column\_name2);

* **INNER JOIN –** inner join will combine rows from different table if the join condition is true.

SELECT column\_name(s) FROM table\_1 JOIN table\_2 ON table\_1.column\_name = table\_2.column\_name;

;

* **INSERT -** statements are used to add row to a table.

INSERT INTO table\_name (column\_1, column\_2, column\_3) VALUES (value\_1, 'value\_2', value\_3);

;

* **IS NULL / IS NOT NULL -**IS NULL and IS NOT NULL are opertors used with the WHERE clause to test for empty values.

SELECT column\_name(s) FROM table\_name WHERE column\_name IS NULL;;

* **LIKE –** is a special operator used with the WHERE clause to search for a specific pattern in a column.

SELECT column\_name(s) FROM table\_name WHERE column\_name LIKE pattern;;

* **LIMIT-** is a clause that lets us to specify the maximum number of rows the result set will have

SELECT column\_name(s) FROM table\_name LIMIT number;;

* **MAX() –** is a function that takes the name of a column as an argument and returns the largest value in the column.

SELECT MAX(column\_name) FROM table\_name;;

* **MIN()-** is a function that takes the name of a column as an argument and returns the smallest value in that column.

SELECT MIN(column\_name) FROM table\_name;;

* **OR –** is an operator that filters the results set to only includes rows where either condition ia true.

SELECT column\_name FROM table\_name WHERE column\_name = value\_1 OR column\_name = value\_2;;

* **ORDER BY –** is a cluse that indicates you want to sort the result set by a particular column either alphabetically or numerically.

SELECT column\_name FROM table\_name ORDER BY column\_name ASC | DESC;;

* **OUTER JOIN -** an outer join will combine from different tables even if the join condition is not met Every row in the le table is returned in the result set, and if the join is not met, then NULL values are used to fill in the columns from right table.

SELECT column\_name(s) FROM table\_1 LEFT JOIN table\_2 ON table\_1.column\_name = table\_2.column\_name;

* **RENAME-** statement is used to rename a table.

RENAME TABLE table\_name TO table\_name;

**[ for Oracle we use ALTER TABLE table\_name RENAME TO table\_name]**

* **ROUND() –** is function that takes a column name and integer as an argument. It rounds the values in the column to the number of decimal places specified by the Integer.

SELECT ROUND(column\_name, integer) FROM table\_name;;

* **SELECT –** statements are used to fetch data from a database. Every query will begins with SELECT.

SELECT column\_name FROM table\_name;

* **SELECT DISTINCT –** specifies that the statement is going to be a query that returns unique values in the specified column(s).

SELECT DISTINCT column\_name FROM table\_name FROM table\_name;

* **SELECT INTO-** statement copies data from one table ans insert it into a new table.

SELECT column\_name(s)INTO new table\_name;;

**[ SELECT \* INTO new\_table\_name FROM table\_name;]**

* **SUM() –** is function that takes the name of a column as an argument and returns the sum of all values in that column.

SELECT SUM(column\_name) FROM table\_name;;

* **VIEW –** is a virtual table, through which a selective portion of the data from one or more tables can be seen. Views do not contain data of their own. They are used to restrict access to the database or to hide complex data complexity. A view is stored as a SELECT statement in database. DML operations on view like INSERT, UPDATE, DELETE affects the data in original table.

CREATE VIEW view\_name AS SELECT COLUMN\_LIST FROM table\_name [ WHERE condition]

* **UPDATE -**statements allow us to edit rows in a table.

UPDATE table\_name SET some\_column = some\_value WHERE some\_column = some\_value;

;

* **WHERE -**  is a clause that indicates you want to filter set to includw only rows where the following condition is true.

SELECT column\_name(s) FROM table\_name WHERE column\_name operator value;;

* **WITH -** clause lets you store the result of a query in a temporary table using an alias. You can also define multiple temporary tables using a comma and with one instance of the WITH keyword.

WITH temporary\_name AS ( SELECT \* FROM table\_name) SELECT \* FROM temporary\_name WHERE column\_name operator value;

**Subquery** or **Inner query** or **Nested query** is a query in a query. SQL subquery is usually added in the [WHERE](https://beginner-sql-tutorial.com/sql-where-clause.htm) Clause of the SQL statement. Most of the time, a subquery is used when you know how to search for a value using a SELECT statement, but do not know the exact value in the database.

**Subqueries** are an alternate way of returning data from multiple tables.

Subqueries can be used with the following SQL statements along with the comparision operators like =, <, >, >=, <= etc.

* [SELECT](https://beginner-sql-tutorial.com/sql-select-statement.htm)
* [INSERT](https://beginner-sql-tutorial.com/sql-insert-statement.htm)
* [UPDATE](https://beginner-sql-tutorial.com/sql-update-statement.htm)
* [DELETE](https://beginner-sql-tutorial.com/sql-delete-statement.htm)

**Eg:**

SELECT first\_name, last\_name, subject   
FROM student\_details   
WHERE games NOT IN ('Cricket', 'Football');

SELECT id, first\_name   
FROM student\_details   
WHERE first\_name IN (SELECT first\_name  
FROM student\_details   
WHERE subject= 'Science');