SQL – Structured Query Language.

RDBMS – Relational Data Base Management System.

Some of The Most Important SQL Commands

* SELECT - extracts data from a database
* UPDATE - updates data in a database
* DELETE - deletes data from a database
* INSERT INTO - inserts new data into a database
* CREATE DATABASE - creates a new database
* ALTER DATABASE - modifies a database
* CREATE TABLE - creates a new table
* ALTER TABLE - modifies a table
* DROP TABLE - deletes a table
* CREATE INDEX - creates an index (search key)
* DROP INDEX - deletes an index

|  |  |
| --- | --- |
| **DDL Commands** |  |
| To create a database | **CREATE DATABASE** databasename; |
| To create a table (The name of the database in which the table is to be created should be used)  USE DATEBASE databasename; | **CREATE TABLE** tablename (  Column1 datatype constraints,  Column2 datatype constraints,  Column3 datatype constraints  ); |
| To create a table | **CREATE** **TABLE** table\_name (col1, col2, …coln) |
| To copy a table with data into a new table  (**All attributes will not be copied**) | **CREATE** **TABLE** new\_table\_name  SELECT \* FROM old\_table\_name; |
| To copy the structure of a table with all attributes | **CREATE** **TABLE** new\_table\_name  **LIKE**  old\_table\_name; |
| To drop an existing database | DROP DATABASE databasename; |
| To drop a table | DROP TABLE tablename; |
| To rename a table | **RENAME TABLE** old\_table\_name **TO** new\_table\_name**;** |
| To add column in an existing table | **ALTER TABLE** tablename **ADD** columnname datatype; |
| To delete column in an existing table | **ALTER TABLE** tablename **DROP** **COLUMN** columnname; |
| To modify column in an existing table | **ALTER TABLE** tablename **MODIFY** columnname datatype; |
| To change column name | **ALTER TABLE** tablename **CHANGE** old\_column\_name new\_column\_name datatype(size); |
| To add constraint after creating the table | **ALTER TABLE** tablename **ADD CONSTRAINT** constraint\_name(column\_name) **REFERENCES** table\_name(field\_name);  **ALTER TABLE** **orderdetails ADD FOREIGN KEY (ProductID) REFERENCES products(ProductID);** |
| Drop a **FOREIGN KEY** constraint | **ALTER TABLE** table\_name  **DROP FOREIGN KEY** FK\_column\_name |
| To see the structure of the table | **DESCRIBE table\_name;** |
| To modify data in a table | **UPDATE** tablename **SET** column1 = value1,  column2 = value2….WHERE condition; |
| To enter new record | **INSERT INTO** tablename (field1, field2….fieldn)  **VALUES** (value1,value2,……valuen); |
| To delete a record from a table | **DELETE FROM** tablename **WHERE** condition; |
| To delete all the records in a table, but not the table itself | TRUNCATE TABLE tablename;(DML) |
|  |  |

|  |  |
| --- | --- |
| To create backup of an existing database | BACKUP DATABASE databasename  TO DISK = ‘filepath’; |
| To create differential backup of an existing database | BACKUP DATABASE databasename  TO DISK = ‘filepath’  WITH DIFFERENTIAL; |

**ON DELETE CASCADE (A constraint assigned to a foreign key)**

**ON DELETE CASCADE** constraint is used to delete the rows from child table automatically when the rows from the parent table are deleted.

Syntax - FOREGIN KEY (column\_name)

REFERENCES table\_name(column\_name)

ON DELETE CASCADE;

Operators in the WHERE clause

SELECT \* FROM Products WHERE Price BETWEEN 50 AND 60;

SELECT \* FROM Customers WHERE City LIKE ‘s%’;

SELECT \* FROM Customers WHERE City IN (“Paris”,”London”);

SELECT \* FROM Customers WHERE Country =’Germany’ AND City=’Berlin’;

SELECT \* FROM Customers WHERE City=’Paris’ OR City=’London’;

SELECT \* FROM Customers WHERE NOT Country=’Germany’;

SELECT \* FROM Customers WHERE Country=’Brazil’ AND (City=’Campinas’ OR City=’Resende’);

SELECT \* FROM Customers WHERE NOT Country=’Germany’ AND NOT Country=’USA’;

ORDER BY

SELECT column1, column2,….FROM tableName ORDER BY column1, column2,….DESC;

INSERT INTO tableName(column1, column2, column3, …….. )

VALUES (value1, value2, value3,…) /\*for specific columns\*/

INSERT INTO tableName VALUES(value1, value2, value3, …….) /\*for all columns\*/

**IS NULL**

SELECT column1, column2,….FROM tableName WHERE columnName IS NULL;

SELECT column1, column2,… FROM tableName WHERE columnName IS NOT NULL;

|  |  |
| --- | --- |
| SELECT TOP | |
| MySQL Syntax | SELECT columns FROM tableName WHERE condition LIMIT number;  Eg: SELECT FROM aadharTable WHERE City =’Trivandrum’ LIMIT 10000;  Eg: SELECT \* FROM Customers LIMIT 3; |
| **SQL Server / MS access Syntax:** | SELECT TOP *number*|percent *column\_name(s)* FROM *table\_name* WHERE *condition*; |
| **Oracle 12 Syntax:** | SELECT *column\_name(s)* FROM *table\_name* ORDER BY *column\_name(s)* FETCH FIRST *number* ROWS ONLY; |

**Min and Max**

SELECT MIN(columnName) FROM tableName;

SELECT MAX(columnName) FROM tableName;

**Count, Avg, Sum**

SELECT COUNT(CustomerID) FROM Customers;

SELECT AVG (Price) FROM Products;

SELECT SUM (Price) FROM Products;

**Aliases**

SELECT CustomerName AS ‘Customer Name’, ContactName AS [Contact Person] FROM Customers;

**Inner Join**



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

**Left Join**



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

**Right Join**



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

**Self Join**

A self join uses the [inner join](https://www.sqlservertutorial.net/sql-server-basics/sql-server-inner-join/) or [left join](https://www.sqlservertutorial.net/sql-server-basics/sql-server-left-join/) clause. Because the query that uses the self join references the same table, the [table alias](https://www.sqlservertutorial.net/sql-server-basics/sql-server-alias/) is used to assign different names to the same table within the query.

|  |  |
| --- | --- |
| SELECT column\_name(s)  FROM table1 T1  INNER JOIN  ON table1 T2  WHERE condition;  (T1 and T2 are different table aliases for the same table) | SELECT column\_name(s)  FROM table1 T1  LEFT JOIN  ON table1 T2  WHERE condition;  (T1 and T2 are different table aliases for the same table) |
| **create table** employees(  emp\_id **int auto\_increment**,  emp\_name **varchar(25)**,  age **int**,  manager\_id **int**,  **primary key (emp\_id)**,  **foreign key (manager\_id) references employees(emp\_id)**  **on delete no action on update no action**,  **check (age>=18)**  ); | SELECT T1.emp\_id, T1.emp\_name, T1.age, T2.emp\_name AS manager  FROM employees T1  LEFT JOIN employees T2  ON T1.manager\_id = T2.emp\_id; |

**Advanced Joins**

**To retrieve only the non matching rows from the left table.**

(Left join minus Inner join)

SELECT columnList

FROM Employee AS E

LEFT JOIN Department AS D

ON E.DepartmentID = D.ID

WHERE D.ID IS NULL;

**To retrieve only the non matching rows from the right table.**

(Right join minus inner join)

SELECT columnList

FROM Employee AS E

RIGHT JOIN Department AS D

ON E.DepartmentID = D.ID

WHERE E.DepartmentID IS NULL;

**To retrieve only the non matching rows from both the left and right table.**

(Full join minus inner join)

SELECT columnList

FROM Employee AS E

FULL JOIN Department AS D

ON E.DepartmentID = D.ID

WHERE E.DepartmentID IS NULL OR D.ID IS NULL;

**UNION**

|  |  |
| --- | --- |
| SELECT City FROM Customers  UNION  SELECT City FROM Suppliers  ORDER BY City; | This SQL statement will return the cities(only distinct values) from both the tables.  ie, if some customers or suppliers have the same city, each city will be listed only once. |
| SELECT City FROM Customers  UNION ALL  SELECT City FROM Suppliers  ORDER BY City; | All cities will be listed. |

|  |  |  |
| --- | --- | --- |
| **GROUP BY** |  | |
| SELECT COUNT (CustomerID), Country  FROM Customers  GROUP BY Country; | To find how many customers from each country. | |
| SELECT name AS Student Name, COUNT (courseID) AS Number of Courses  FROM studentTable  GROUP BY name; |  | |
| SELECT Country, COUNT(CustomerID)  FROM Customers  GROUP BY Country  ORDER BY COUNT(CustomerID) DESC; |  | |
| GROUP BY With JOIN Example | | |
| SELECT Shippers.ShipperName, COUNT(Orders.OrderID) AS NumberOfOrders  FROM Orders  LEFT JOIN Shippers  ON Orders.ShipperID = Shippers.ShipperID  GROUP BY ShipperName; | | ShipperName NumberOfOrders  Federal Shipping 68  Speedy Express 54  United Package 74 |

SQL HAVING Examples

The HAVING clause was added to SQL because the WHERE keyword cannot be used with aggregate functions such as COUNT(), MIN(), MAX(), AVG(), SUM().

|  |  |
| --- | --- |
| Write an SQL statement to extract the countries having more than 5 customers | SELECT Country, COUNT(CustomerID)  FROM Customers  GROUP BY Country  HAVING COUNT(CustomerID) > 5; |
| Find the employees who placed more than 10 orders. | SELECT Employees.FirstName,  COUNT (Orders.orderID) AS NumberofOrders  FROM Orders  INNER JOIN Employees  ON Orders.EmployeeID = Employees.EmployeeID;  GROUP BY FirstName  HAVING COUNT(Orders.orderID) > 10; |

**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

|  |  |
| --- | --- |
| Find the suppliers who has supplied products whose price is less than 20? | SELECT SupplierName FROM Suppliers  WHERE EXISTS (SELECT ProductName FROM Products WHERE  Products.SupplierID = Suppliers.SupplierID  AND Price < 20); |

**ANY,ALL**

|  |  |
| --- | --- |
| Get the list of ProductName where quantity = 10 | SELECT ProductName FROM Products  WHERE ProductID = Any (SELECT ProductID FROM OrderDetails WHERE Quantity = 10); |

**SELECT INTO**

|  |  |
| --- | --- |
| To copy all columns into a new table into external database. | SELECT \*  INTO *newTable* [IN externaldb]  FROM *oldTable*  WHERE condition; |
| To Copy only some columns into a new table into external database. | SELECT column1, column2, column3,….  INTO *newTable* [IN externaldb]  FROM *oldTable*  WHERE condition; |

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

The INSERT INTO SELECT statement requires that the data types in source and target tables match.

**Note:** The existing records in the target table are unaffected.

|  |  |
| --- | --- |
| Copy all columns from one table to another table | INSERT INTO table2  SELECT \* FROM table1  WHERE condition; |
| Copy only some columns from one table into another table | INSERT INTO table2 (column1, column2, …….)  SELECT column1, column2, …..  FROM table1  WHERE condition; |

**CASE**

CASE Syntax

CASE  
    WHEN condition1 THEN result1  
    WHEN condition2 THEN result2  
    WHEN conditionN THEN resultN  
    ELSE result  
END;

|  |  |
| --- | --- |
| The following SQL goes through conditions and returns a value when the first condition is met: | SELECT OrderID, Quantity, CASE     WHEN Quantity > 30 THEN 'The quantity is greater than 30'     WHEN Quantity = 30 THEN 'The quantity is 30'     ELSE 'The quantity is under 30' END AS QuantityText FROM OrderDetails; |

What is a Stored Procedure?

A stored procedure is a prepared SQL code that you can save, so the code can be reused over and over again.

So if you have an SQL query that you write over and over again, save it as a stored procedure, and then just call it to execute it.

You can also pass parameters to a stored procedure, so that the stored procedure can act based on the parameter value(s) that is passed.

|  |  |
| --- | --- |
| Create a procedure. | DELIMITER &&  CREATE PROCEDURE test()  BEGIN  Sql qwery;  END &&  DELIMITER |
| Call the above procedure | CALL test(); |

|  |
| --- |
| SQL CONSTRAINTS |
| The following constraints are commonly used in SQL:  [NOT NULL](https://www.w3schools.com/sql/sql_notnull.asp) - Ensures that a column cannot have a NULL value  [UNIQUE](https://www.w3schools.com/sql/sql_unique.asp) - Ensures that all values in a column are different  [PRIMARY KEY](https://www.w3schools.com/sql/sql_primarykey.asp) - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table  [FOREIGN KEY](https://www.w3schools.com/sql/sql_foreignkey.asp) - Prevents actions that would destroy links between tables  [CHECK](https://www.w3schools.com/sql/sql_check.asp) - Ensures that the values in a column satisfies a specific condition  [DEFAULT](https://www.w3schools.com/sql/sql_default.asp) - Sets a default value for a column if no value is specified  [CREATE INDEX](https://www.w3schools.com/sql/sql_create_index.asp) - Used to create and retrieve data from the database very quickly |

|  |
| --- |
| CREATE TABLE Persons (     ID int NOT NULL,     LastName varchar(255) NOT NULL,     FirstName varchar(255) NOT NULL,     Age int ); |
| CREATE TABLE Persons (     AadharNumber int UNIQUE NOT NULL,     LastName varchar(255) NOT NULL,     FirstName varchar(255) NOT NULL,     Age int ); |
| CREATE TABLE Persons (     ID int NOT NULL,     LastName varchar(255) NOT NULL,     FirstName varchar(255),     Age int,     PRIMARY KEY (ID) ); |
| CREATE TABLE Orders (     OrderID int NOT NULL,     OrderNumber int NOT NULL,     PersonID int,     PRIMARY KEY (OrderID),     FOREIGN KEY (PersonID) REFERENCES Persons(PersonID) ); |
| CREATE TABLE Persons (     ID int NOT NULL,     LastName varchar(255) NOT NULL,     FirstName varchar(255),     Age int,     CHECK (Age>=18) ); |
| CREATE TABLE Persons (     ID int NOT NULL,     LastName varchar(255) NOT NULL,     FirstName varchar(255),     Age int,     City varchar(255) DEFAULT 'Trivandrum' ); |
| **To get the current date by default**  CREATE TABLE Ration (     ID int NOT NULL,  Item1Name varchar(255) NOT NULL,      Item1Quantity int NOT NULL,     IssusedOn date DEFAULT GETDATE() ); |

VIEW

A view is a virtual table based on the result set of an SQL statement.

CREATE VIEW viewName AS

SELECT column1, column2, column3, …..FROM tableName;

CREATE VIEW employee\_list AS

SELECT T1.emp\_id, T1.emp\_name, T1.age, T2.emp\_name AS manager

FROM employees T1

LEFT JOIN employees T2

ON T1.manager\_id = T2.emp\_id;

To view the records - SELECT \* FROM viewName;

INDEX

* Index is used to retrieve data very fast.
* The users cannot see the index, they are just used to speed up searches/queries.

|  |  |
| --- | --- |
| A table with index | Retrieval is faster. Updating is slower. |
| A table without index |  |

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

ALTERTABLEtable\_nameDROP INDEXindex\_name*;*

**TCL – Transaction Control Language (COMMIT & ROLLBACK)**

**Java Database Connectivity with MySQL**

In order to establish connection between java with mySQL, we need to download JDBC-mySQL driver.

Now we need to add the driver file into this project.

1. Right click on the project and select properties.
2. Select java bulid path.
3. Add the external jar file.

**Date**

To convert string to date format

UPDATE table\_name SET Date\_field = str\_to\_date(Date\_field , “%y/%m/%d”);

This method was not successful. So we executed the following command and changed the date format successfully.

ALTER TABLE orderscsv CHANGE OrderDate OrderDate DATE;