***MY SQL***

===============================================================================================

Data : Collection OF RAW facts : student marks, customer names

Database : collection of data/info Ex: SSMS

Data store ex: paper, books, flat files, Database

DBMS : DATABASE MANAGEMENT SYSTEM

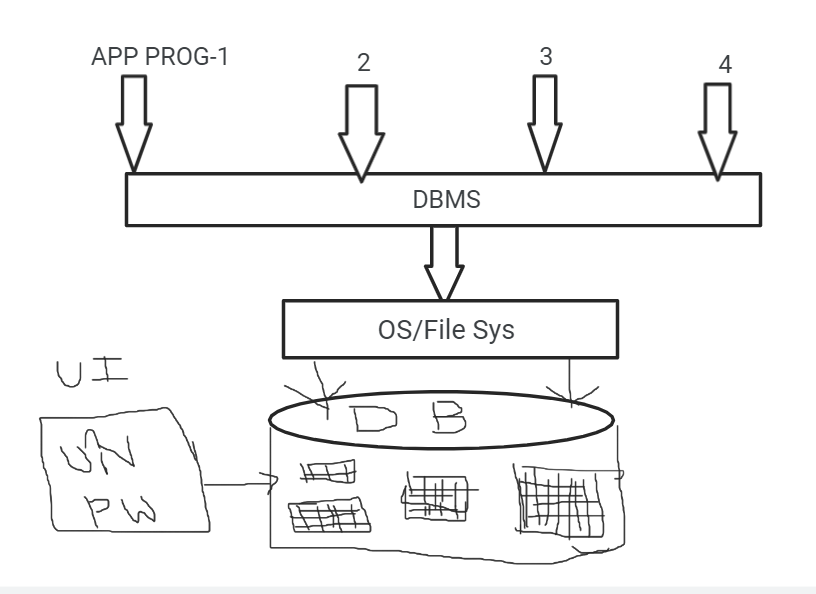
Exampls : oracle, teradata,sql server,MySql etc....

Database : collection of data/info

It is an organised collection of interrelated data.

Or

Collection of structured data



Every database having 2 types of structers

1. Logical
2. Physical

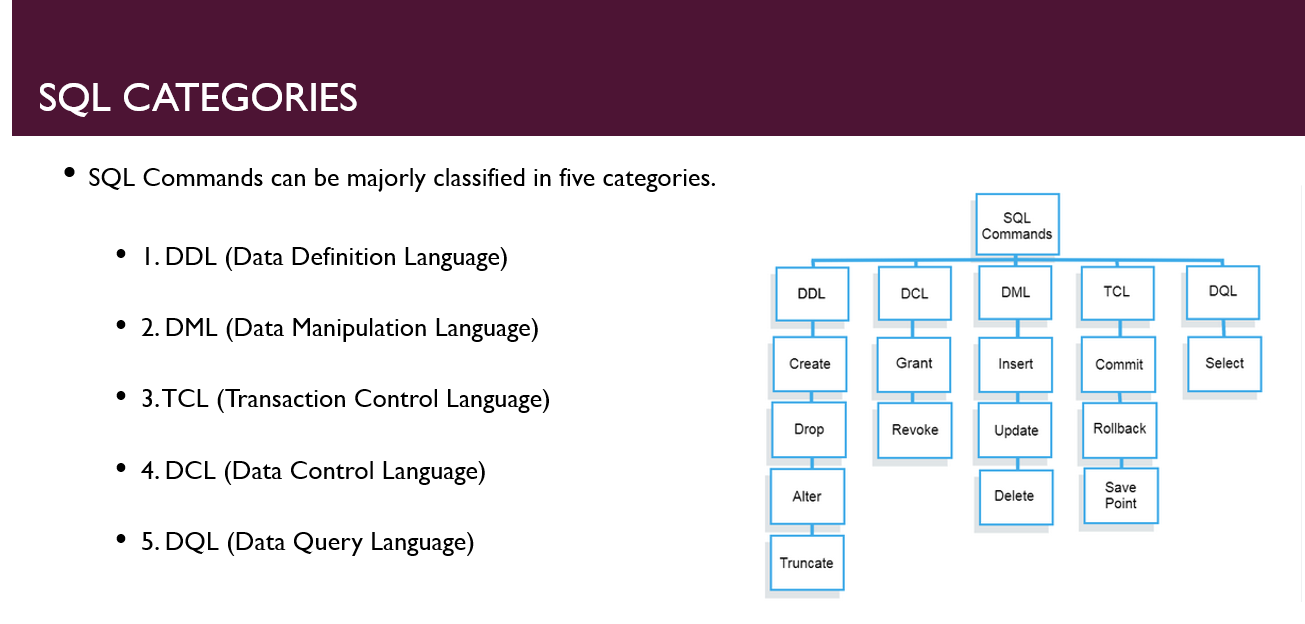
**Logical Structure:** which is not visible in O.S.

Contains tables,views,sequence,synonyms etc….

These are handled by Database developers or DB Admins

**Physical Structure:** which is visible in O.S.

These are handled by DBA only…



--create database ExceLR; --to create a database --commenting single lines

/\*create

database

ExceLR;\*/ --commenting multiple lines

--select \* from INFORMATION\_SCHEMA.TABLES; --to get the list of tables

***CREATE***

CREATE TABLE first (sno int,name varchar(40)); --A table creation command

select \* from dbo.employees;

create table employees (

emp\_id int primary key,

first\_name varchar(50),

last\_name varchar(50),

department\_id int

);

***ALTER***

***It is used to change existing table structure***

* ***add***

***🡪 modify***

* ***drop***

add:

--SYNTAX: Alter table TBL\_NM add COL\_NM data\_type;

Alter table first add salary bigint;

--create database ExceLR; --to create a database --commenting single lines

/\*create

database

ExceLR;\*/ --commenting multiple lines

--select \* from INFORMATION\_SCHEMA.TABLES; --to get the list of tables

CREATE TABLE first (sno int,name varchar(40)); --A table creation command

select \* from dbo.first;

create table employees (

emp\_id int primary key,

first\_name varchar(50),

last\_name varchar(50),

department\_id int

);

--SYNTAX: Alter table TBL\_NM add COL\_NM data\_type;

Alter table first add salary bigint;

--SYNTAX: Alter table TBL\_NM alter COL\_NM data\_type(size);

ALTER TABLE first ALTER COLUMN sno varchar(45);

--SYNTAX: Alter table TBL\_NM drop COL\_NM ;

Alter table first drop sno ;

DROP Table first;

drop database ExceLR\_2;

Insert into employees (emp\_id,first\_name,last\_name,department\_id) values (1,'vinny','valentain',100);

Insert into employees values (2,'mohan','reddy',101);

Insert into employees (emp\_id,first\_name,last\_name,department\_id) values (3,'rahul','dhravid',102);

Insert into employees (emp\_id,first\_name,last\_name,department\_id) values (4,'vivek','raj',103);

Update employees set department\_id = 110 where department\_id = 103;

Delete from employees where last\_name = 'reddy';

drop table employees;

begin transaction;

select \* from ExceLR.dbo.employees;

rollback transaction;

commit transaction;

***JOINS***

***Inner join/equi join/join***

--inner join/join

/\*

select reuired\_columns from table1

inner join/join/equi join table2

on table1.col\_name = table2.col\_name

note:

1) Based on equality condition we are retrieving data from multiple tables.

2) Here joining conditional columns must belongs to same data type.

3) Whenever the tables having common column then only we are allowed to use equi joins and also these common columns must belongs to same data type.

\*/

Create table customers(

customer\_id int,

first\_name varchar(45)

);

Create table orders(

order\_id int,

amount bigint,

o\_cust\_id int

);

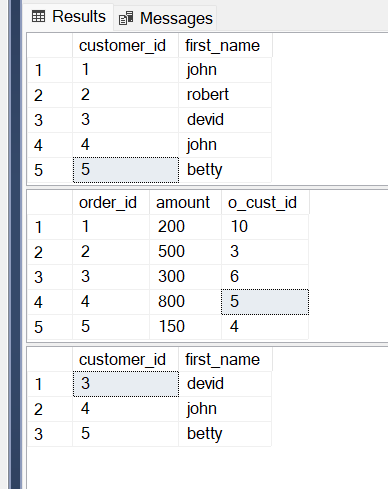
select \* from customers;

select \* from orders;

select customer\_id,first\_name from customers

inner join orders

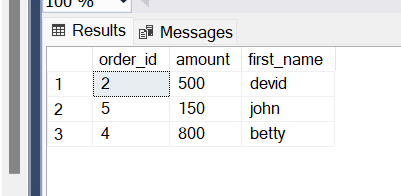
on customers.customer\_id = orders.o\_cust\_id;

******

select order\_id,amount,first\_name from orders

inner join customers

on customers.customer\_id = orders.o\_cust\_id;

******

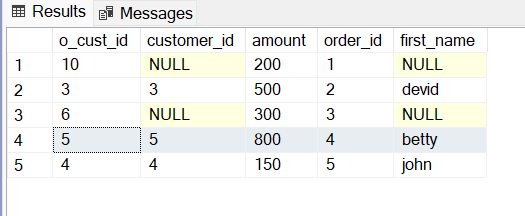
***Left join/ Left outer join:***

* ***In a LEFT JOIN, every record from the table on the left, the base table, will be returned. Then values from the right table, the table being joined, will be added where they exist.***
* ***The LEFT JOIN is also known as LEFT OUTER JOIN and you can use them interchangeably.***

select o\_cust\_id,customer\_id,amount,order\_id,first\_name from orders o

left outer join customers c

on c.customer\_id = o.o\_cust\_id;



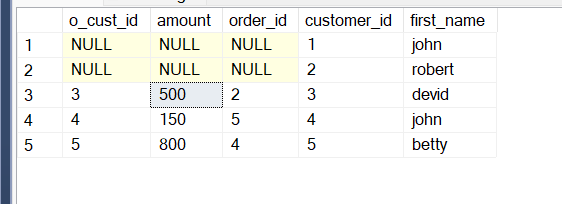
**Right join**

* The RIGHT JOIN keyword in SQL returns the all matching records(or rows) and the records(or rows) which are present in the right table but not in the left table .That means that, if a certain row is present in the right table but not in the left, the result will include this row but with a NULL value in each column from the left . If a record from the left table is not in the right, it will not be included in the result.

select o\_cust\_id,amount,order\_id,customer\_id,first\_name from orders o

right outer join customers c

on c.customer\_id = o.o\_cust\_id;



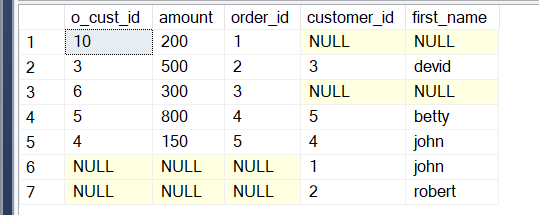
***Full join***

* ***SQL Full Join creates a new table by joining two tables as a whole. The joined table contains all records from both the tables and fills NULL values for missing matches on either side. In short, full join is a type of outer join that combines the result-sets of both left and right joins.***

select o\_cust\_id,amount,order\_id,customer\_id,first\_name from orders o

full outer join customers c

on c.customer\_id = o.o\_cust\_id;

******

***Aggregate functions***

select \* from customers;

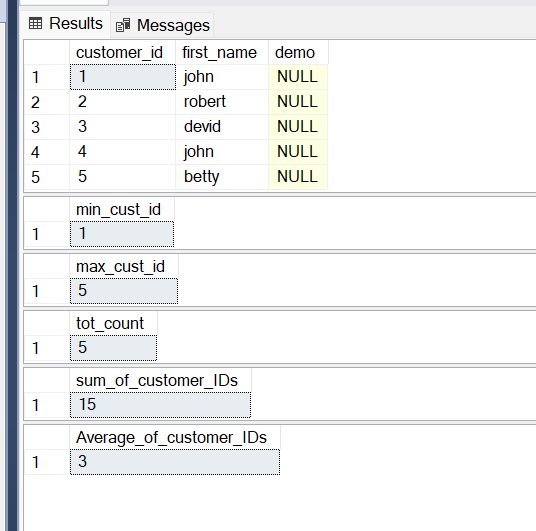
SELECT MIN(customer\_id) AS min\_cust\_id FROM customers;

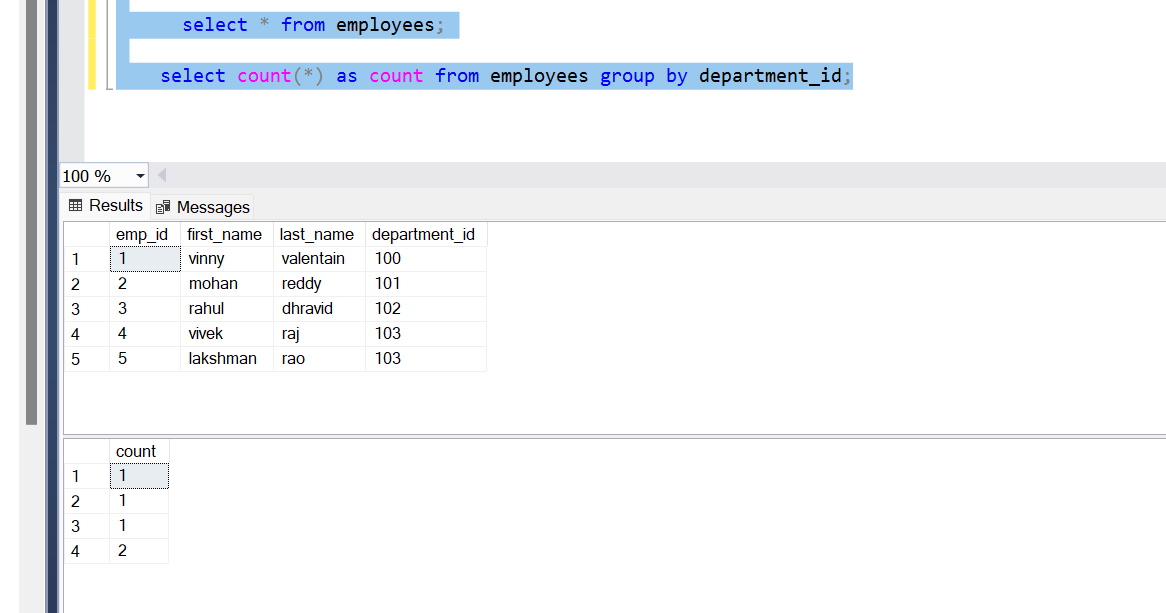
SELECT MAX(customer\_id) AS max\_cust\_id FROM customers;

select count(\*) as tot\_count from customers;

select sum(customer\_id) as sum\_of\_customer\_IDs from customers;

select avg(customer\_id) as Average\_of\_customer\_IDs from customers;





Example:

Create table student

(

Roll\_No integer,

Student\_Name varchar(20),

Gender Varchar(1),

Class varchar(1),

Marks integer

);

select \* from student;

insert into STUDENT values (1,'MOHAN','M','5',90);

insert into STUDENT values (2,'vivek','M','9',91);

insert into STUDENT values (3,'vaishnavi','6','6',80);

insert into STUDENT values (4,'rahul','M','9',67);

insert into STUDENT values (5,'kiran','M','8',96);

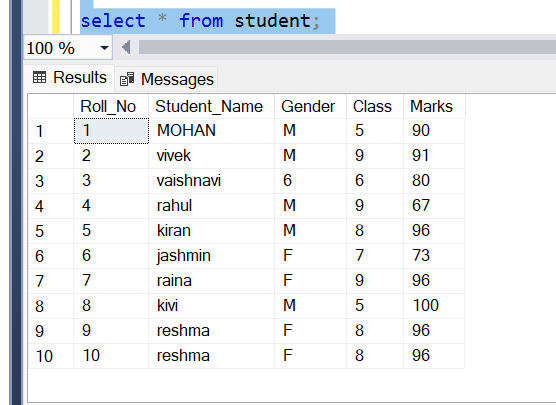
insert into STUDENT values (6,'jashmin','F','7',73);

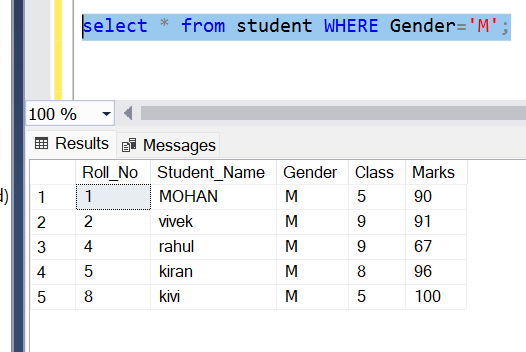
insert into STUDENT values (7,'raina','F','9',96);

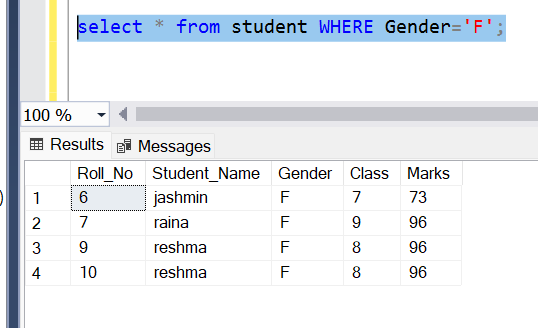
insert into STUDENT values (8,'kivi','M','5',100);

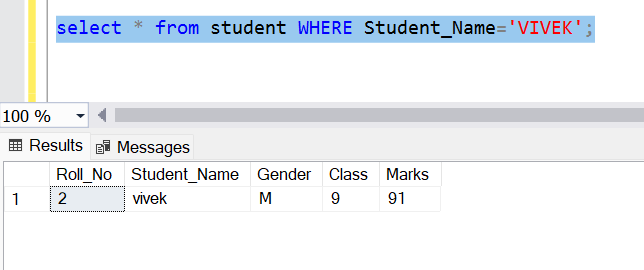
insert into STUDENT values (9,'reshma','F','8',96);

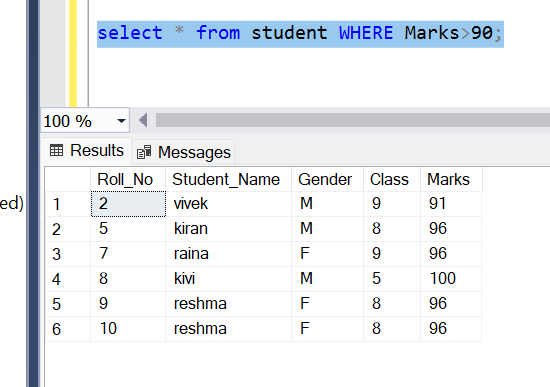
insert into STUDENT values (10,'reshma','F','8',96);







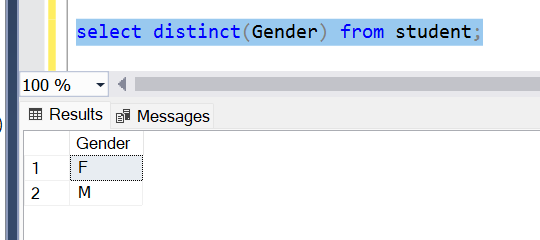


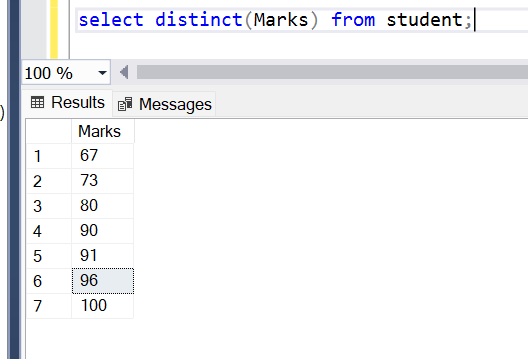


**Eliminating Redundant Data**

--For removing the duplicates, we will use keyword ‘distinct’

--keyword ‘distinct’ Fetches the Unique Records





***Using Column Aliases***

**The columns that we can select in query can be given a different name, i.e; column alias name for output purpose.**

**Syntax:**

**=======**

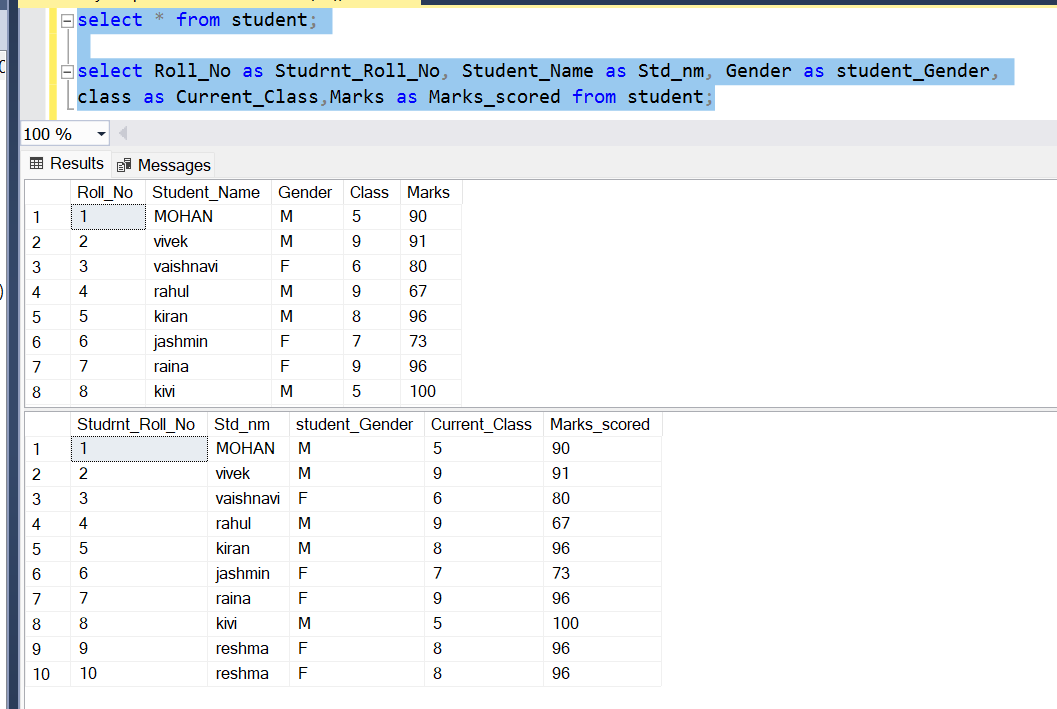
**Select col\_1 AS col\_alias,col\_2 AS col\_alias,……….**

**From table\_nm;**

select \* from student;

select Roll\_No as Studrnt\_Roll\_No, Student\_Name as Std\_nm, Gender as student\_Gender,

class as Current\_Class,Marks as Marks\_scored from student;

****

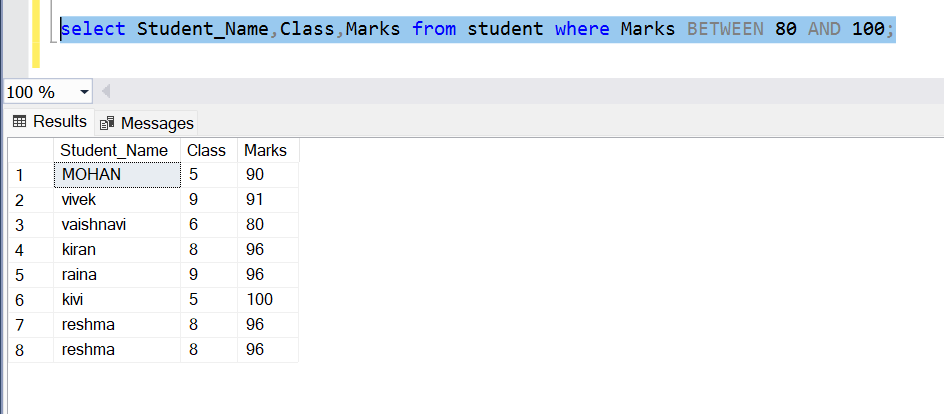
***Condition Based on a RANGE:***

**The BETWEEN operator defines a range of values that**

**The column values must fall in to make the condition true.**

**The range includes both lower value and upper value.**

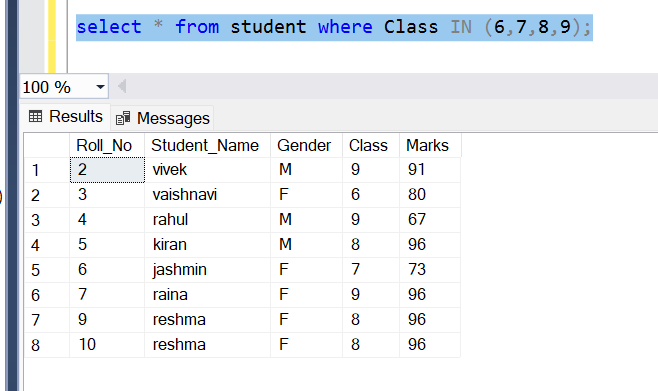
select Student\_Name,Class,Marks from student where Marks BETWEEN 80 AND 100;

****

***Condition Based on a list:***

**To Specify a list of values, We will use a ‘IN’ operator.**

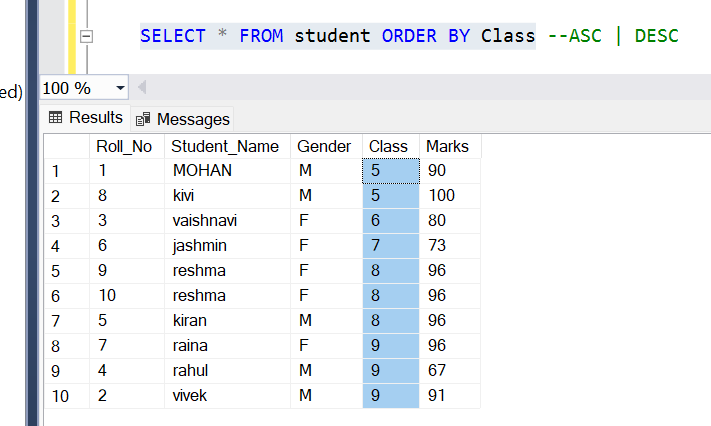
select \* from student where Class IN (6,7,8,9);

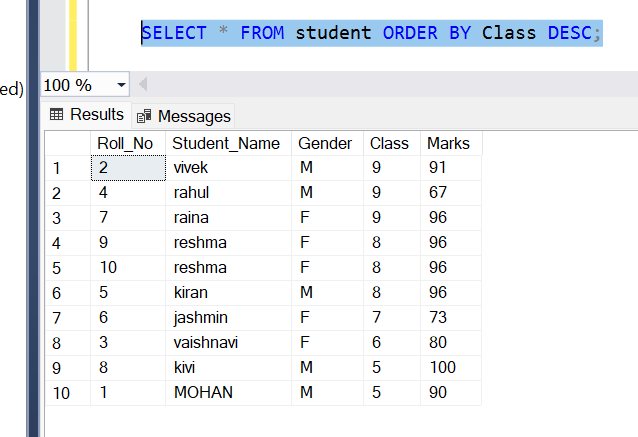
******

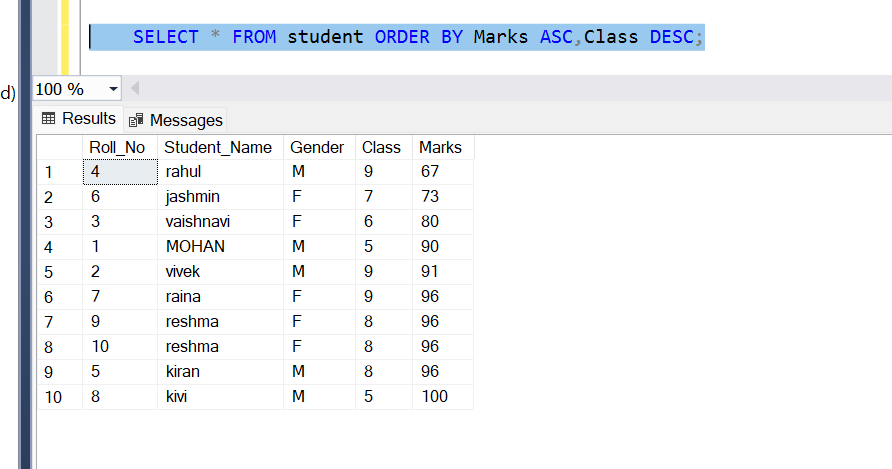
***Order by:***

* **The ORDER BY statement in SQL is used to sort the fetched data in either ascending or descending according to one or more columns. In this article, we will discuss different ways of using Order By in SQL.**
* **By default, ORDER BY sorts the data in ascending order.**
* **We can use the keyword DESC to sort the data in descending order and the keyword ASC to sort in ascending order.**
* **Syntax: SELECT \* FROM table\_name ORDER BY column\_name ASC | DESC**
* **To sort multiple columns: SELECT \* FROM table\_name ORDER BY column1 ASC|DESC , column2 ASC|DESC**

SELECT \* FROM student ORDER BY Class





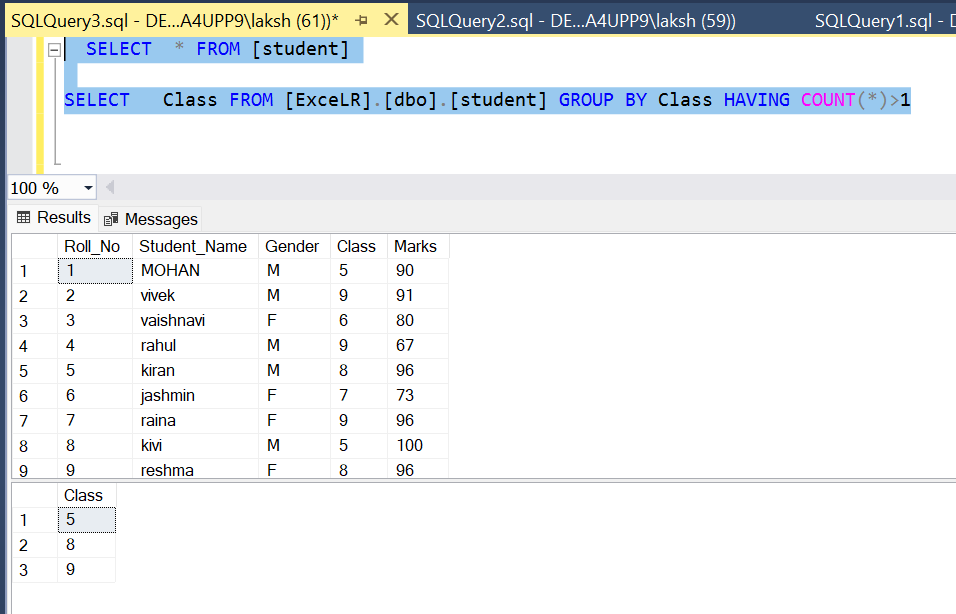


Having clause

* The Having Clause in MySQL is also used for filtering the data just like the where clause. Having Clause in MySQL filters the rows from the intermediate result set that is built by using the FROM, WHERE, or GROUP BY clauses in a SELECT statement.
* The Having Clause in MySQL is typically used with a GROUP BY clause. That means the Having Clause is used in combination with a GROUP BY clause to restrict the number of groups to be returned by satisfying the condition which is specified using the having clause.

SELECT \* FROM [student];

SELECT Class FROM [ExceLR].[dbo].[student] GROUP BY Class HAVING COUNT(\*)>1;



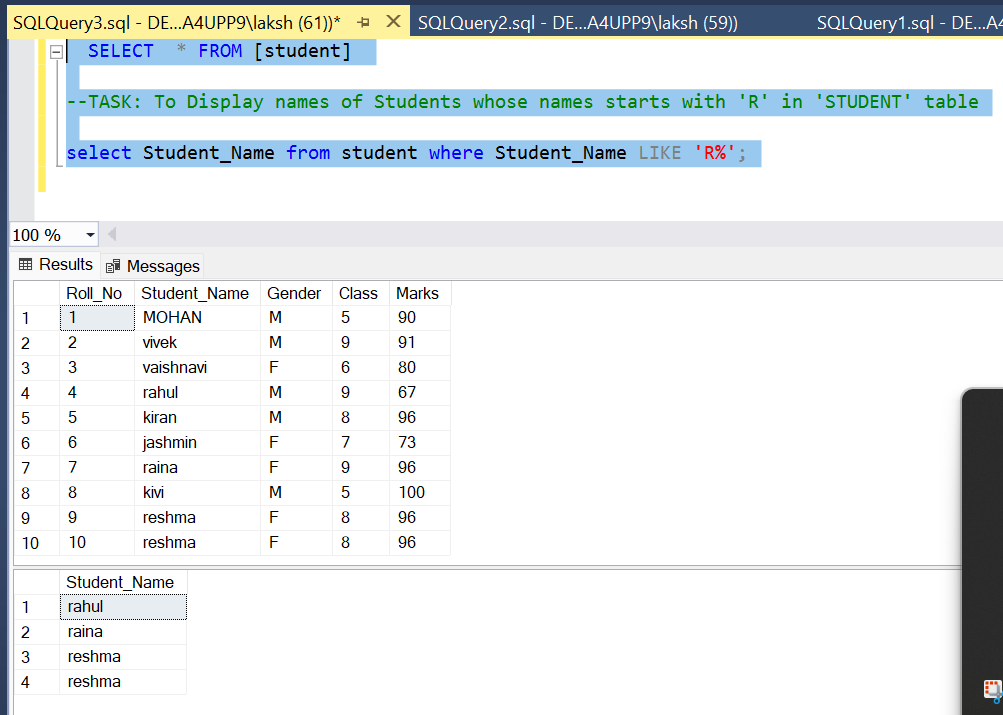
***Like condition(Condition based on pattern matches)***

* The LIKE operator is a logical operator that tests whether a string contains a specified pattern or not.
* In MySQL, LIKE condition is used to perform pattern matching to find the correct result. It is used in SELECT, INSERT, UPDATE and DELETE statement with the combination of WHERE clause.
* Syntax: expression LIKE pattern ESCAPE escape\_character
* MySQL provides two wildcard characters for constructing patterns: Percentage % and underscore.
  + The percentage ( % ) wildcard matches any string of zero or more characters.
  + The underscore ( \_ ) wildcard matches any single character.

SELECT \* FROM [student]

--TASK: To Display names of Students whose names starts with 'R' in 'STUDENT' table

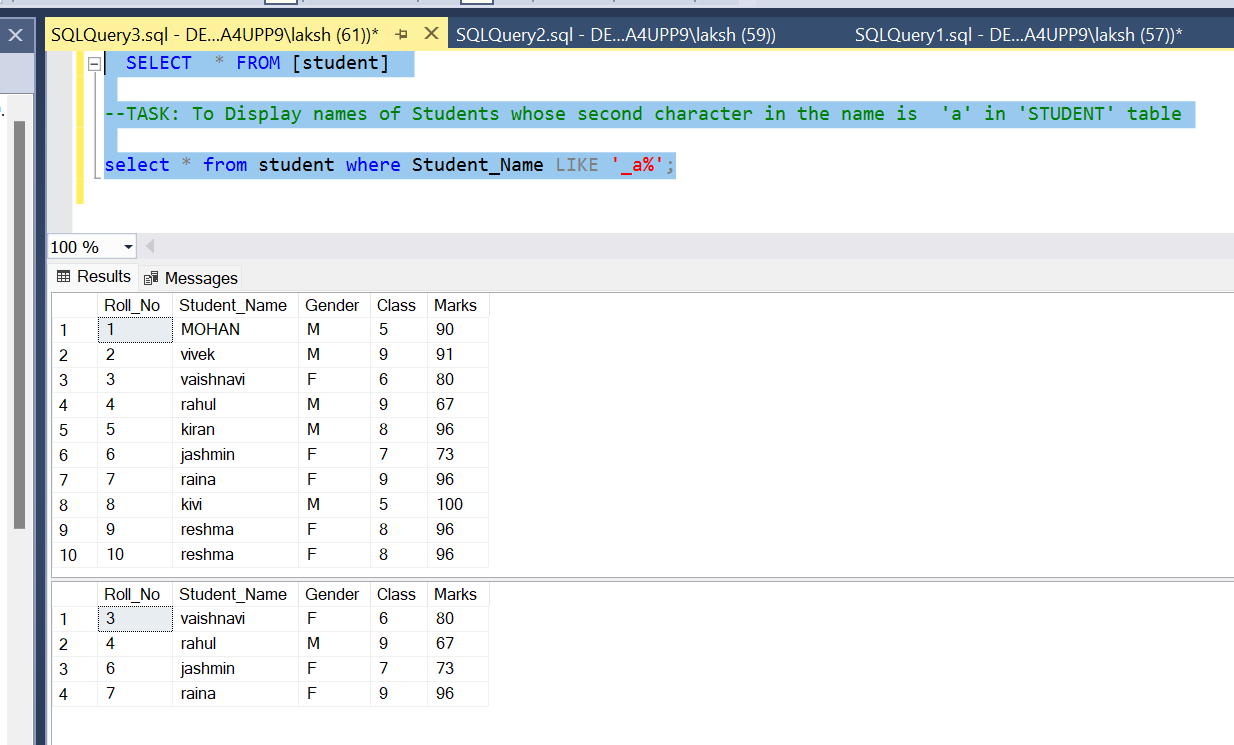
select Student\_Name from student where Student\_Name LIKE 'R%';

******

SELECT \* FROM [student]

--TASK: To Display names of Students whose second character in the name is 'a' in 'STUDENT' table

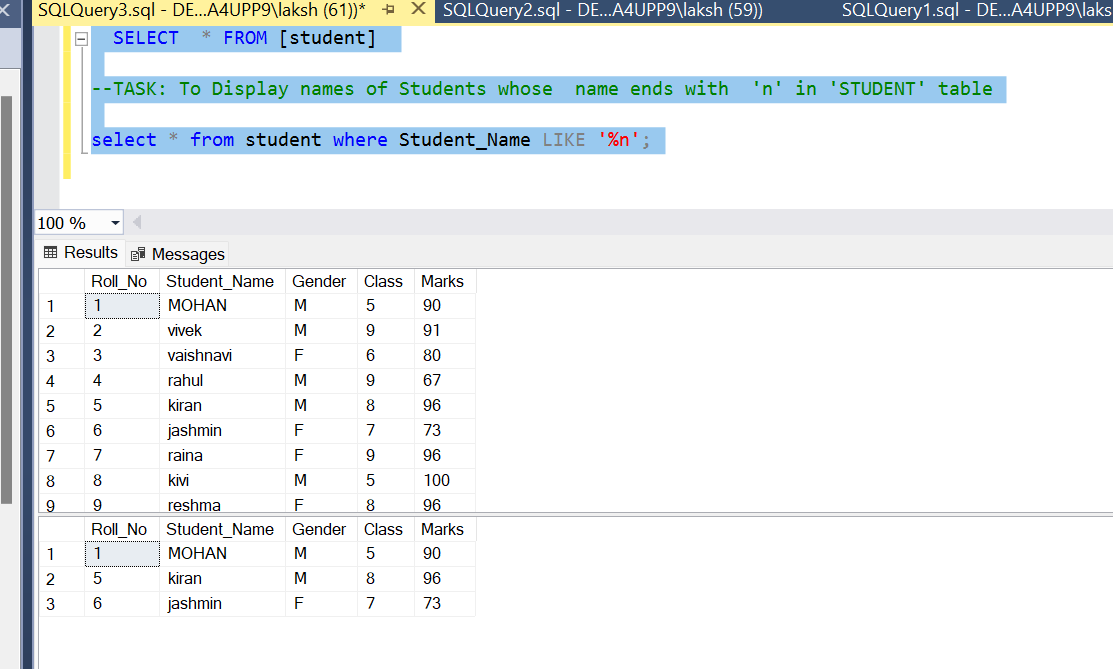
select \* from student where Student\_Name LIKE '\_a%';



SELECT \* FROM [student]

--TASK: To Display names of Students whose name ends with 'n' in 'STUDENT' table

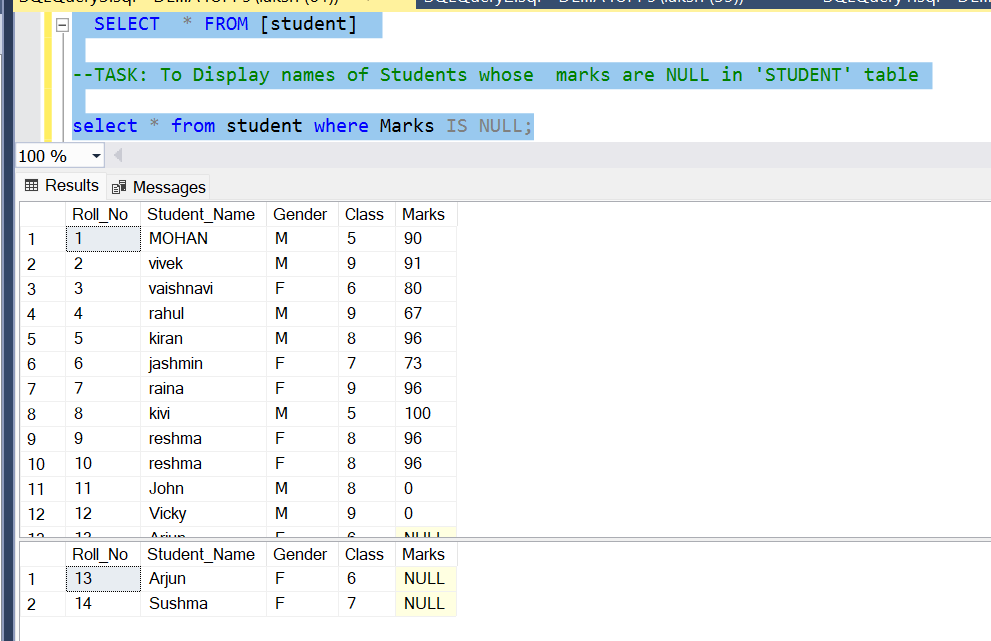
select \* from student where Student\_Name LIKE '%n';



SELECT \* FROM [student]

--TASK: To Display names of Students whose marks are NULL in 'STUDENT' table

select \* from student where Marks IS NULL;



SELECT \* FROM [student]

--TASK: To Display names of Students whose marks are NULL in 'STUDENT' table

select \* from student where Marks IS NULL;

/\*SELECT column\_name(s)

FROM table\_name

WHERE column\_name IN (value1, value2, ...); \*/

CREATE TABLE CUSTOMER (

ID INT NOT NULL,

NAME VARCHAR(20) NOT NULL,

AGE INT NOT NULL,

ADDRESS CHAR (25),

SALARY DECIMAL (18,2),

PRIMARY KEY (ID)

);

INSERT INTO CUSTOMER VALUES (1,'RAMESH',32,'AAHMEDABAD',2000.00);

INSERT INTO CUSTOMER VALUES (2,'KHILAN',25,'DELHI',1500.00);

INSERT INTO CUSTOMER VALUES (3,'KAUSHIK',23,'KOTA',2000.00);

INSERT INTO CUSTOMER VALUES (4,'CHAITALI',25,'MUMBAI',6500.00);

INSERT INTO CUSTOMER VALUES (5,'HARDHIK',27,'BHOPAL',8500.00);

INSERT INTO CUSTOMER VALUES (6,'KOMAL',22,'HYDERABAD',4500.00);

INSERT INTO CUSTOMER VALUES (7,'MUFFY',24,'INDORE',10000.00);

SELECT \* FROM CUSTOMER;

SELECT \*

FROM CUSTOMER

WHERE NAME IN ('KHILAN','HARDHIK','MUFFY');

UPDATE CUSTOMER SET AGE = 30 WHERE AGE IN(25,27);

SELECT \* FROM CUSTOMER WHERE AGE BETWEEN 20 AND 25; --20>=AGE<=25

select \* from student where Marks IS NULL;

select \* from student where Marks IS NOT NULL;

IFNULL (EXP1,EXP2); --IF THE EXP1 IS NOT NULL THEN IT RETURNS FIRST ARGUMENT

--IF THE EXP1 IS NULL THEN IT RETURNS SECOND ARGUMENT

SELECT \*, ISNULL(Marks,1234) FROM student;

Select \*, coalesce(Marks,Roll\_No,Class) from student;

select top 4 \* from student;

SELECT column\_list FROM table\_name LIMIT offset, count;

***Mysql Stored Procedure And Exceptions***

SELECT \* FROM [ExceLR].[dbo].[student] WHERE Student\_Name='MOHAN'

SELECT \* FROM [ExceLR].[dbo].[student] WHERE Student\_Name='kivi'

SELECT \* FROM [ExceLR].[dbo].[student] WHERE Student\_Name='Sushma'

SELECT \* FROM [ExceLR].[dbo].[student] WHERE Student\_Name='vaishnavi'

--creating a simple stored procedure

CREATE PROCEDURE ToGetSelectedListOfStudents

as

begin

SELECT \* FROM [ExceLR].[dbo].[student] WHERE Student\_Name='MOHAN'

SELECT \* FROM [ExceLR].[dbo].[student] WHERE Student\_Name='kivi'

SELECT \* FROM [ExceLR].[dbo].[student] WHERE Student\_Name='Sushma'

SELECT \* FROM [ExceLR].[dbo].[student] WHERE Student\_Name='vaishnavi';

End

--executing a stored procedure

exec ToGetSelectedListOfStudents;

--creating a simple stored procedure BY PASSING PARAMETER

Create Procedure ToGetSelectedListOfStudent

(@amount bigint=800)

AS

BEGIN

SET NOCOUNT ON

select o\_cust\_id,customer\_id,amount,order\_id,first\_name from orders o

left outer join customers c

on c.customer\_id = o.o\_cust\_id;

END

--WHILE EXECUTING THE PROCEDURE WE NEED TO PASS THE INPUT PARAMETER

--executing a stored procedure

exec ToGetSelectedListOfStudent 800;

Create Procedure ToGetSelectedStudent

(@amount bigint)

AS

BEGIN

SET NOCOUNT ON

select o\_cust\_id,customer\_id,amount,order\_id,first\_name from orders o

left outer join customers c

on c.customer\_id = o.o\_cust\_id;

END

exec ToGetSelectedStudent 800;

Create Procedure ToGetStd

(@OID INT)

AS

BEGIN

SET NOCOUNT ON

select O.order\_id,O.amount,C.first\_name from orders O

inner join customers C

on C.customer\_id =@OID;

END

exec ToGetStd 5;

CREATE TABLE EMP (EmpID int identity(1,1),EmpName varchar(50))

create procedure ins\_NewEmp\_with\_outputparameters

(@Ename varchar(50),

@EId int output)

AS

BEGIN

SET NOCOUNT ON

INSERT INTO ExceLR.dbo.EMP (EmpName) VALUES (@Ename)

SELECT @EId=SCOPE\_IDENTITY()

END

declare @EmpID INT

EXEC ins\_NewEmp\_with\_outputparameters 'vinny', @EmpID OUTPUT

select @EmpID as EmployeeID

declare @EmpID2 INT

EXEC ins\_NewEmp\_with\_outputparameters 'mohan', @EmpID2 OUTPUT

select @EmpID2 as EmployeeID

select \* from emp;

pre-requisites:

1. JAVA JDK(for java programming)
2. VS CODE EDITER(for code editing purpose and importing the jar files)
3. SSMS(whole community server)

import java.sql.\*;

public class App {

    private Connection con;

    private Statement st;

    ResultSet rs;

    public App()

    {

        try {

            Class .forName("com.mysql.jdbc.Driver") ;

            con = DriverManager.getConnection("jdbc:mysql://mysql2.000webhost.com:3306/" +

            "MySQL\_Database","MySQL\_Usernme","MySQL\_Pass");

        } catch (Exception ex) {

            ex.printStackTrace();

        }

    }

}