MSSQL --> Microsoft SQL server 2

SSMS---> SQL Server Management Studio:

SQL---> structure query language which help us to interact with our database.

Database---> database is a kind of container which provide us space which we can use to store our data.

COMMAND CATEGORY:

a)-DDL COMMAND (Data Definitions Language)

b)-DQL COMMAND (Data Query Language)

c)-DML COMMAND (Data Manipulation Language)

d)-DCL COMMAND (Data Control Language)

a)-DDL COMMAND (Data definitions Language)

* CREATE
* ALTER
* DROP

1.CREATE : 🡪

CREATE DATABASE (DATABASE\_NAME)

CREATE TABLE (TABLE\_NAME)

Question:🡪

1. How we can create our database?

CREATE DATABASE SQL\_Class

1. How we can create Tables?

CREATE TABLE EmployeeDetails\_18thMay (EmployeeID INT, EmployeeName VARCHAR (50), Salary INTEGER)

1. ALTER :🡪

ALTER TABLE (TABLE NAME) ADD (TABLE NAME)

Question:🡪

1. Write a query to add city column in EmployeeDetails\_18thMay?

ALTER TABLE EmployeeDetails\_18thMay ADD City VARCHAR (50)

ALTER TABLE EmployeeDetails\_18thMay ADD Country VARCHAR (50),   
State VARCHAR (50)

1. DROP ---> this command we can use to remove any object from our database or database itself.

DROP DATABASE DATABASE\_NAME.

DROP TABLE TABLE\_NAME.

Question:🡪

How we can drop a database?

DROP DATABASE SQL\_Claass

Write a query to remove EmployeeName column from EmployeeDetails\_18thMay?

ALTER TABLE EmployeeDetails\_18thMay DROP COLUMN EmployeeName

b)-DQL COMMAND (Data Query Language)

a)-SELECT---> This command we can use to display the table structure along with data

SELECT \* FROM (TABEL\_NAME)

SELECT \* FROM EmployeeDetails\_18thMay

SELECT EmployeeID FROM EmployeeDetails\_18thMay

SELECT EmployeeID,EmployeeName FROM EmployeeDetails\_18thMay

c)-DML COMMAND (Data Manipulation Language)

a)-INSERT

b)-UPDATE

c)-DELETE

a)-INSERT

SELECT \* FROM (TABLENAME)

INSERT INTO TABLE NAME (TABLE NAME1, TABLE NAME2, TABLE NAME3)

VALUES (1,2,3)

SELECT \* FROM EmployeeDetails\_18thMay

INSERT INTO EmployeeDetails\_18thMay (Employee ID, Employee Name, Salary)

VALUES (1,'Sahil Jaiswal',70000)

If we don’t want to add values in specific column

INSERT INTO EmployeeDetails\_18thMay (Employee, Salary)

VALUES (2,10000)

Multiple values in single line to add “, (Commma)” for separator.

INSERT INTO EmployeeDetails\_18thMay (Employee ID, Employee Name, Salary)

VALUES(3,'Miraj',90000), (4,'Meenakshi',89000), (5,'Vishal',67000)

UPDATE---> this command we are going to use update the existing data.

SELECT \* FROM EmployeeDetails\_18thMay

UPDATE EmployeeDetails\_18thMay SET Salary=95000

Write a query to update salary=65000 for employeeID 3?

UPDATE EmployeeDetails\_18thMay SET Salary=65000 WHERE EmployeeID=3

UPDATE EmployeeDetails\_18thMay SET EmployeeName='abdul',Salary=70000 WHERE EmployeeID=1

DELETE-----> this command we can use to remove the data from our table.

SELECT \* FROM EmployeeDetails\_18thMay

DELETE FROM EmployeeDetails\_18thMay

Write a query to delete a data for employeeId 5?

SELECT \* FROM EmployeeDetails\_18thMay

DELETE FROM EmployeeDetails\_18thMay WHERE EmployeeID=5

d)-DCL COMMAND (Data Control Language)

a)-GRANT

b)-REVOKE

a)-GRANT

GRANT SELECT ON EmployeeDetails\_18thMay TO USER1

b)-REVOKE

REVOKE SELECT, INSERT, DELETE ON EmployeeDetails\_18thMay FROM User1

**DATATYPE**

1. NUMERIC
2. INTEGER
3. STRING
4. DATE

1)-NUMERIC

a)-TINYINT---------------->0 to 255

b)-SMALLINT---------------> -32768 to 32767

c)-INTEGER/INT

d)-BIGINT

e)-DECIMAL

CREATE TABLE TinyInt\_Test\_18thMay (ID TINYINT)

INSERT INTO TinyInt\_Test\_18thMay (ID) VALUES (0)

INSERT INTO TinyInt\_Test\_18thMay (ID)VALUES (-100) ---Error

INSERT INTO TinyInt\_Test\_18thMay (ID) VALUES (500) ----Error

INSERT INTO TinyInt\_Test\_18thMay (ID) VALUES (255)

INSERT INTO TinyInt\_Test\_18thMay (ID) VALUES (256)

CREATE TABLE SmallInt\_Test\_18thMay (ID SMALLINT)

INSERT INTO SmallInt\_Test\_18thMay (ID) VALUES (-40000) ---- Error

INSERT INTO SmallInt\_Test\_18thMay (ID) VALUES (40000) ---Error

INSERT INTO SmallInt\_Test\_18thMay (ID) VALUES (32767)

INSERT INTO SmallInt\_Test\_18thMay (ID) VALUES (32768) –Error

10.12

100.2375

DECIMAL(p,s)

p-->Precision---> total number of digit

s-->Scale--->Number of digit that can be stored after decimal

DECIMAL(4,2)

10.12

total no of digit--->4

After Decimal Place-->2

How many digit it will accept before decimal place-->4-2=2

CREATE TABLE Decimal Test(Price Decimal (5,2))

Total Digit-->5

After Decimal-->2

Before Decimal--->5-2=3

INSERT INTO Decimal Test (Price) VALUES (100.12)

INSERT INTO Decimal Test (Price) VALUES (1001.1) --->Error

INSERT INTO Decimal Test (Price) VALUES (1.1)

INSERT INTO Decimal Test (Price) VALUES (1.137)

INSERT INTO Decimal Test (Price) VALUES (1124.134)

SELECT \* FROM Decimal Test

STRING:

a)-CHAR

b)-VARCHAR

a)-CHAR

CREATE TABLE LearnerInfo (CandidateName CHAR(10))

INSERT INTO LearnerInfo(CandidateName)VALUES('Sahil')

INSERT INTO LearnerInfo(CandidateName)VALUES('Raj')

INSERT INTO LearnerInfo(CandidateName)VALUES('SahilJAISW')

INSERT INTO LearnerInfo(CandidateName)VALUES('SahilJAISWSDS')

SELECT \* FROM LearnerInfo

'Sahil '-->10-5=5

'Raj '-->10-3=7

'SahilJAISW'

--MAXIMUM WE CAN DEFINE 8000

b)-VARCHAR

CREATE TABLE Employee\_Data\_24thMay(EmployeeName VARCHAR(10))

INSERT INTO Employee\_Data\_24thMay(EmployeeName)VALUES('Sahil')

INSERT INTO Employee\_Data\_24thMay(EmployeeName)VALUES('Raj')

SELECT \* FROM Employee\_Data\_24thMay 75

'Sahil'

'Raj'

CREATE TABLE Employee\_Data\_24thMay(EmployeeName VARCHAR(10))

CREATE TABLE Employee\_Data\_24thMay(EmployeeName VARCHAR(MAX)---> this can accept upto 2 gb character)

a)-DATE

b)-TIME

c)-DATETIME

'yyyy-mm-dd hh:min:sec.ms'

CREATE TABLE SaleData\_24thMay(OrderDate DATE)

INSERT INTO SaleData\_24thMay(OrderDate)VALUES('2025-05-01')

INSERT INTO SaleData\_24thMay(OrderDate)VALUES('2025-05-02 09:10:40.123')

SELECT \* FROM SaleData\_24thMay

--2025-05-24 15:06:48.173

CREATE TABLE SalesTime\_24thMay(OrderTime TIME)

INSERT INTO SalesTime\_24thMay(OrderTime)VALUES('2025-05-02 09:10:40.1234')

INSERT INTO SalesTime\_24thMay(OrderTime)VALUES('12:11:51')

SELECT \* FROM SalesTime\_24thMay

Write a query to create a studentInfo table to store Student information, below is the data point

a)-StudentID

b)-StudentName

c)-Age

d)-DOB

and also insert 5 record in this table

CREATE TABLE StudentInfo(StudentID TINYINT,StudentName VARCHAR(50),Age TINYINT,DOB DATE)

INSERT INTO StudentInfo(StudentID,StudentName,Age,DOB)

VALUES(1,'Sahil Jaiswal',30,'1999-01-01')

SELECT \* FROM StudentInfo

CREATE TABLE Sales\_OrderData ( OrderDateTime DATETIME)

INSERT INTO Sales\_OrderData(OrderDateTime)VALUES('2025-05-02 09:10:40.123')

INSERT INTO Sales\_OrderData(OrderDateTime)VALUES('2025-05-03') 149

INSERT INTO Sales\_OrderData(OrderDateTime)VALUES('25-05-03') 151

INSERT INTO Sales\_OrderData(OrderDateTime)VALUES('09:10:40.123')

SELECT \* FROM Sales\_OrderData

Insert into Studentinfo(StudentID,StudentName,Age,DOB)

Values(1,'Ishaan',33,'1991-09-10')

CONSTRAINT:

TYPE OF CONSTRAINT:

a)-NOT NULL---> if we define any column as not null then that column will not accept any null value.

b)-UNIQUE-----> if we apply unique constraint then that column will not allow to store any duplicate value.

c)-CHECK

d)-DEFAULT

e)-PRIMARY

f)-FOREIGN

a)-NOT NULL

CREATE TABLE ManageInfo (ManagerID INT NOT NULL, ManagerName VARCHAR(50))

INSERT INTO ManageInfo(ManagerID,ManagerName)VALUES(1,'Paul') 178

INSERT INTO ManageInfo(ManagerID,ManagerName)VALUES(NULL,'Namrata') 180

INSERT INTO ManageInfo(ManagerName)VALUES('Paul') 182

SELECT \* FROM ManageInfo

b)-UNIQUE

CREATE TABLE OrderSale\_24thMay (OrderID INT UNIQUE, OrderAmount INT)

INSERT INTO OrderSale\_24thMay(OrderID,OrderAmount)VALUES(1,100) 195

INSERT INTO OrderSale\_24thMay(OrderID,OrderAmount)VALUES(2,50)

INSERT INTO OrderSale\_24thMay(OrderID,OrderAmount)VALUES(1,510) 198

SELECT \* FROM OrderSale\_24thMay

c)-CHECK

CREATE TABLE VoterListInfo (VoterId INT, VoterName VARCHAR(50), VoterAge TINYINT CHECK(VoterAge>=18))

INSERT INTO VoterListInfo(VoterId,VoterName,VoterAge) VALUES(1,'Sahil Jaiswal',17)

INSERT INTO VoterListInfo(VoterId,VoterName,VoterAge) VALUES(1,'Sahil Jaiswal',25)

SELECT \* FROM VoterListInfo

d)-DEFAULT

DEFAULT---> if user is not passing any value for country column then we want sql should automatically added India as a value.

CREATE TABLE EmployeeData\_24thMay

(EmployeeID INT,EmployeeName VARCHAR(100),Country VARCHAR(100) DEFAULT('India'))

INSERT INTO EmployeeData\_24thMay(EmployeeID,EmployeeName,Country)

VALUES(1,'Sahil','India')

INSERT INTO EmployeeData\_24thMay(EmployeeID,EmployeeName,Country)

VALUES(2,'Anish','USA')

SELECT \* FROM EmployeeData\_24thMay

INSERT INTO EmployeeData\_24thMay(EmployeeID,EmployeeName)

VALUES(3,'Abdul')

INSERT INTO EmployeeData\_24thMay(EmployeeID,EmployeeName,Country)

VALUES(4,'Sonam',NULL)

SELECT \* FROM EmployeeData\_24thMay

CREATE TABLE NOT\_NULL\_CHECK (ID INT NOT NULL,FirstName VARCHAR(50) NOT NULL)

CREATE TABLE NOT\_NULL\_CHECK(ID INT NOT NULL UNIQUE,FirstName VARCHAR(50) NOT NULL)

PRIMARY KEY(UNIQUE + NOT NULL):

CREATE TABLE EmployeeDepartmentInfo (EmployeeID INT PRIMARY KEY,

EmployeeName VARCHAR(50), DepartmentID INT)

INSERT INTO EmployeeDepartmentInfo(EmployeeID, EmployeeName, DepartmentID)

VALUES(NULL,'Swapnil',1)

INSERT INTO EmployeeDepartmentInfo(EmployeeID, EmployeeName, DepartmentID)

VALUES(1,'Swapnil',1)

INSERT INTO EmployeeDepartmentInfo(EmployeeID,EmployeeName,DepartmentID)

VALUES(1,'Nimesh',1)

SELECT \* FROM EmployeeDepartmentInfo 290

Write a query to display employeeinfo for employeeid 2? 294

SELECT \* FROM EmployeeDetails WHERE EmployeeID=2 EmployeeID 2,3

Operator

IN-🡪when we have to compare more than one value for same column then we can use IN operator.

SELECT \* FROM EmployeeDetails WHERE EmployeeID IN (1,2)

NOT EQUAL(<>,!=)

Write a query to extract all employee info except employee ID 3?

SELECT \* FROM EmployeeDetails WHERE EmployeeID <>3

Write a query to display all employe except EmployeeID 1,5?

SELECT \* FROM EmployeeDetails WHERE EmployeeID NOT IN (1,5)

BETWEEN

10000 to 60000

SELECT \* FROM EmployeeDetails WHERE Salary BETWEEN 10000 AND 60000

--->=<=

SELECT \* FROM EmployeeDetails WHERE Salary>=10000 AND Salary<=600

LIKE :

SELECT \* FROM EmployeeDEtails WHERE FirstName='Vishal'

Write a query to display those employee whose firstname is start with 's'?

Sahil

Suman

Suraj

Syed

SELECT \* FROM EmployeeDetails WHERE FirstName LIKE 's%'

Write a query to display those employee whose firstname is end with 'l'?

SELECT \* FROM Employeedetails WHERE FirstName LIKE '%L'

Write a query to return those employee whose firstname is having 'a' character?

SELECT \* FROM EmployeeDetails WHERE FirstName LIKE '%a%'

Write a query to display those employees whose firstname is having 'a' character in second position?

\_-->

SELECT \* FROM EmployeeDetails WHERE FirstName LIKE '\_a%' 363

Write a query to displauy those employee whose firstname second last cjaracter is 'i'?

SELECT \* FROM EmployeeDetails WHERE FirstName LIKE '%i\_' 368

Write a query to display those employee whose firstname is having 's' as third letter?

SELECT \* FROM EmployeeDetails WHERE FirstName LIKE ' s%'

AND OPERATOR:

Whenever we want to apply a multiple condition and if all the condition is fulfilled on that scenario we

want to returned the data then we can use AND.

Write a query to display those employee who is working with departmentID 1 and Clientid 1?

SELECT \* FROM EmployeeDetails WHERE DepartmentID=1 AND ClientID=1

OR--->

if any one of the condition is true then it will return the data.

Write a query to display those employee who is working with departmentID 1 OR clientid 1?

SELECT \* FROM EmployeeDetails WHERE DepartmentID=1 OR ClientID=1

**JOIN:**

SELECT \* FROM EmployeeDetails

SELECT \* FROM Department

EmployeeID, FirstName, LastName, DepartmentName

TYPE OF JOIN:

a)-INNER JOIN

b)-LEFT JOIN

c)-RIGHT JOIN

d)-FULL OUTER JOIN

e)-CROSS JOIN

INNER JOIN---> this join will return those data which is matching between two table.

SELECT EmployeeID, FirstName, LastName, DepartmentName FROM EmployeeDetails INNER JOIN Department ON EmployeeDetails.DepartmentID=Department.DepartmentID

SELECT \* FROM Department

**Primary key** column only used to reference by another table

if we did not create a foreign on Employee table then we can face one problem, by mistake we can

insert a data in my employee details with department which is not present in my department table.

ex--> in my department table we have following record:

DepartmentID

1

2

3

4

5

and by mistake someone inserted a data in employee details table with departmentID 12 which is totally wrong, now if we want to apply such rule that before performing a insert statement in my employee details. SQL first validate the data for department and check that value is present in department table or not, if not present then it will throw error if value is present then it will successfully execute the statement and insert the data in our employee details table.

Note--> column which we are going to refer from another table that must be create as a primary key constraint.

CREATE TABLE DepartmentData\_25thMay

(DepartmentID INT PRIMARY KEy ,DepartmentName VARCHAR(50))

CREATE TABLE EmployeeData\_25thMay (EmployeeID INT, EmployeeName VARCHAR (50),

DepartmentID INT---->After creating a foreign column this column will allow only two type of value

1. -value which will be present in department table or it will allow null

CONSTRAINT FK\_DepartmentID FOREIGN KEY(DepartmentID) REFERENCES

DepartmentData\_25thMay(DepartmentID))

INSERT INTO EmployeeData\_25thMay(EmployeeID, EmployeeName, DepartmentID)

VALUES(1,'Sahil Jaiswal',1)

INSERT INTO EmployeeData\_25thMay(EmployeeID, EmployeeName, DepartmentID)

VALUES(2,'Raj',NULL)

INSERT INTO EmployeeData\_25thMay(EmployeeID, EmployeeName, DepartmentID)

VALUES(2,'Meenakshi',3)

SELECT \* FROM DepartmentData\_25thMay

INSERT INTO DepartmentData\_25thMay(DepartmentID,DepartmentName)

VALUES(1,'IT')

INSERT INTO DepartmentData\_25thMay(DepartmentID,DepartmentName)

VALUES(3,'Mechanical')

SELECT \* FROM DepartmentData\_25thMay

SELECT EmployeeID, EmployeeName, DepartmentName FROM EmployeeData\_25thMay

INNER JOIN DepartmentData\_25thMay ON EmployeeData\_25thMay.DepartmentID=DepartmentData\_25thMay.DepartmentID

SELECT EmployeeID, EmployeeName, DepartmentName FROM EmployeeData\_25thMay AS E

INNER JOIN DepartmentData\_25thMay AS D ON E.DepartmentID=D.DepartmentID

LEFT JOIN---> it will return all the data from left table and matching record from right table.

SELECT EmployeeID, FirstName, LastName, DepartmentName FROM EmployeeDetails -🡪LEFT TABLE

LEFT JOIN Department ON Department.DepartmentID=EmployeeDetails.DepartmentID-->RIGHT TABLE

SELECT EmployeeID,FirstName AS FName,LastName AS LName,DepartmentName FROM

EmployeeDetails AS ED ------------------->LEFT TABLE

LEFT JOIN Department AS D ON ED.DepartmentID=D.DepartmentID--->RIGHT

RIGHT JOIN-----> it will return all the data from right table and matching record from left table

SELECT EmployeeID, FirstName AS FName, LastName AS LName, DepartmentName FROM

EmployeeDetails AS ED ------------------->LEFT TABLE

RIGHT JOIN Department AS D ON ED.DepartmentID=D.DepartmentID--->RIGHT

FULL OUTER JOIN (LEFT JOIN + RIGHT JOIN) --> it will return all the data from left table and all the data from right table will be consider as our output

SELECT EmployeeID, FirstName AS FName, LastName AS LName, DepartmentName FROM

EmployeeDetails AS ED ------------------->LEFT TABLE

FULL OUTER JOIN Department AS D ON ED.DepartmentID=D.DepartmentID--->RIGHT

CROSS JOIN:

SELECT EmployeeID, FirstName, LastName, DepartmentName FROM EmployeeDetails CROSS JOIN Department

SELECT EmployeeID, FirstName, LastName, Date FROM DimDate CROSS JOIN EmployeeDetails

WHERE Date>='2021-01-04' AND Date<='2021-01-08'

FUNCTION--->Function we can use to perform some specific task

Write a query to display current datetime?

SELECT GETDATE()

TWO TYPES OF FUNCTION:

1)-BUILT-IN FUNCTION--(SYSTEM DEFINED FUNCTION) ----> those function which is already available within our system that is called system, defined.

2)-USER DEFINED FUNCTION---> those function which is developed by the user and then we can use those function with in our code that is called user defined function.

CATEGORY WITH IN SYSTEM DEFINED FUNCTION:

a)-STRING CATEGORY

a)-UPPER ()

b)-LOWER ()

c)-LTRIM ()

d)-RTRIM ()

e)-TRIM ()

f)-LEN ()

g)-REPLACE ()

h)-REVERSE ()

i)-SUBSTRING ()

UPPER ()---> this function is used to transform the data into upper case.

SELECT UPPER('sahil')

SELECT FirstName, UPPER(FirstName) AS New\_FirstName FROM EmployeeDetails

SELECT \* FROM EmployeeDetails

LOWER()--------> this function is used to convert the value into lower case?

SELECT LOWER('SAHIL')

SELECT FirstName, LOWER(FirstName) FROM EmployeeDetails

LTRIM()---> this function is used to remove the blank space from left side/ Leading space.

SELECT LTRIM (' Bharat')

'Bharat'

SELECT FirstName, LTRIM(FirstName) FROM EmployeeDetails

UPDATE EmployeeDetails SET FirstName=LTRIM(FirstName)

' Vishal'

'Vishal'

RTRIM () -----> this function is used to remove the Space from Right side/ Trailing Space.

SELECT RTRIM ('Sahil ')

'Sahil'

SELECT FirstName, RTRIM(FirstName) FROM EmployeeDetails

'Sahil '

'Sahil'

UPDATE EmployeeDetails SET FirstName=' Sahil ' WHERE EmployeeID=1

NESTED FUNCTION----> Function with in a another function

SELECT FirstName, RTRIM(LTRIM(FirstName)), FROM EmployeeDetails

'Sahil '

' Sahil '

'Sahil'

SELECT FirstName, LTRIM(RTRIM(FirstName)), FROM EmployeeDetails

'Sahil '

' Sahil '

'Sahil'

SELECT 2+3

SELECT 3+2

TRIM()---> this function is used to remove the space from both side.

SELECT FirstName, TRIM(FirstName) FROM EmployeeDetails

'Sahil'

REVERSE () ----> this function will change the sequence of data.

SELECT REVERSE('Sahil’) --->'lihas'

'lihaS'

Write a query to display the FirstName value in reverse order?

Write a query to add a new column (RFirstName) in EmployeeDetails table and update the Reverse value from firstname column?

ALTER TABLE EmployeeDetails ADD RFirstName VARCHAR(50)

SELECT \* FROM EmployeeDetails UPDATE EmployeeDetails SET RFirstName=REVERSE(FirstName)

DESC

LEN()----> it will count the no of character is present in our data.

SELECT LEN (TRIM (' Sahil'))

SELECT ' Sahil'

SELECT 'Sahil '

SELECT LEN ('Sahil ')

REPLACE()---> this function is used to replace the exisiting character with new one.

This function will accept below parameter

a)-ColumnName/String Data

b)-Character/String Data which we want to change

c)-New Character/String

Write a query to replace 'a' with 'b' from 'Sahil'?

SELECT REPLACE ('Sahil', 'a', 'b')

-->Sbhil

Write a query to replace 's' with 'l' from firstname column?

SELECT REPLACE(FirstName,'s','l') FROM EmployeeDetails

SELECT \* FROM EmployeeDetails

SUBSTRING () ---> this function we can used to extract the sub part from the entire string value then we can use substring.

This function can accept three parameters

a)-String Data/ColumnName

b)-Starting Position No

c)-Len-->How many characters we want to extract

'Sahil'---

12345

Rajesh

Write a query a display first three letter from 'sahil' value?

SELECT SUBSTRING('sahil',1,3)

Write a query to display a third and forth letter from 'sahil'?

SELECT SUBSTRING('Sahil',3,2)

Write a query to display a third and fifth letter from 'sahil'?

SELECT SUBSTRING('Sahil',3,1),SUBSTRING('Sahil',5,1)

SELECT 'Sahil'+' '+'Jaiswal'

Write a query to Concat FirstName and LastName Column and display as a EmployeeName output?

SELECT TRIM(FirstName) +' ' + TRIM(LastName) As EmployeeName FROM EmployeeDetails

CONCAT ()

SELECT CONCAT (FirstName,' ', LastName) FROM EmployeeDetails

SELECT 'podila' + ' ' +'meghana'

Sahil Jaiswal

SELECT TRIM (FirstName +' ' + LastName) As EmployeeName FROM EmployeeDetails

SELECT TRIM(FirstName) +' ' + TRIM(LastName) As EmployeeName FROM EmployeeDetails

DATE CATEGORY: ---> those function which will work on date/datetime

a)-GETDATE ()

b)-DAY ()

c)-MONTH ()

d)-YEAR ()

e)-DATEADD ()

f)-DATEDIFF ()

g)-DATENAME ()

GETDATE () ---> This function will display current Datetime

'yyyy-mm-dd hh:mm:sec.ms'

SELECT GETDATE()

'2025-05-31 16:32:26.170'

DAY () ---> this function will return the Day no from Date value.

SELECT DAY ('2025-05-31')

SELECT DOB, DAY(DOB) FROM EmployeeDetails

MONTH () -----> this function will return the month no from Date/Datetime value.

SELECT

MONTH (GETDATE ())

YEAR () ----> this function will return the year no.

SELECT YEAR ('1999-01-01')

SELECT YEAR (GETDATE ())

'2025-05-31'

DATEADD () ---> this function is used to add no of days/no of month/no of year in any date value.

This function will accept below parameter:

a)-INTERVAL

a)-dd---->Day

b)-mm---->Month

c)-yy---->Year

b)-How many Days/Month/ year we want to add

c)-DATE VALUE/ColummName

'2025-05-01'

Write a query to add 5 Days in 1st may 2025?

SELECT DATEADD (dd,5,'2025-05-01')

Write a query to add 3 months in current date?

SELECT DATEADD (mm,3, GETDATE ())

Write a query to add 5 year in DOB column?

SELECT DOB, DATEADD (yy,5, DOB) FROM EmployeeDetails

DATEDIFF () ---> if we want to calculate the difference between two date then we can use DateDiff function?

Below is the parameter which we have to pass while using DateDiff function:

a)-INTERVAL

a)-dd-->Day

b)-mm-->month

c)-yy-->year

b)-Startdate

c)-EndDate

StartDate-->'2025-05-10'

EndDate---->'2025-05-17'

--Day difference

SELECT DATEDIFF (dd,'2025-05-10','2025-05-17')

SELECT DATEDIFF (mm,'2025-01-05','2026-02-01')

Write a query to Find out the Age in term of year for every employee which is present in my employeedetails?

SELECT \*, DATEDIFF (yy, DOB, GETDATE ()) FROM EmployeeDetails

DATENAME ()

SELECT DATENAME (mm, GETDATE ())

SELECT DATENAME (mm,'1999-10-01')

SELECT DATENAME (WeekDay, GETDATE ())

SELECT DATENAME(WeekDay,'2025-06-01')

SELECT DATENAME(WeekDay,'2025-06-01')

MATHMATICAL FUNCTION:

a)-POWER ()

b)-SQRT ()

c)-ABS ()

d)-FLOOR ()

e)-CEILING ()

2^3--->2\*2\*2

SELECT POWER (2,3)

25

ABS () -->it will convert all negative value into positive.

SELECT ABS (-100)

SELECT ABS (10)

FLOOR () ---> this function will work on the decimal value; it will return nearest smallest integer value.

SELECT FLOOR (10.45)

10 11

SELECT FLOOR (99.99)

99 100

CEILING () ----> this function will return nearest largest integer value.

SELECT CEILING (99.99)

99 100

SELECT CEILING (-75.32)

-75 -76

---sahil

--SELECT UPPER( SUBSTRING('sahil',1,1))+LOWER(SUBSTRING('sahil',2,4))

AGGREGATE FUNCTION:

SELECT UPPER(FirstName) FROM EmployeeDetails

Aggregate Function:

a)-SUM()

b)-COUNT()

c)-MAX()

d)-MIN()

e)-AVG()

Sum:-

SELECT SUM(Salary) FROM EmployeeDetails

COUNT () --> this function will used to count the no of record.

Write a query to display how many employee is working in our org?

SELECT COUNT (\*) FROM EmployeeDetails

--1+2=3

--1+2=12

Write a query to display maximum salary from employeetable?

SELECT MAX(Salary) FROM EmployeeDetails

SELECT \* FROM EmployeeDetails

Write a query to display minimum salary from employeedetails?

SELECT MIN(Salary) FROM EmployeeDetails

Write a query to find the average salary from employeedetais?

SELECT AVG(Salary) FROM EmployeeDetails

Write a query to display Maximum salary departmentwise?

SELECT DepartmentID,MAX(Salary) FROM EmployeeDetails

GROUP BY DepartmentID

Write a query to display departmentName and Employeecount?

SELECT D.DepartmentName, COUNT(ED.EmployeeID) AS EmployeeCount

FROM EmployeeDetails AS ED

JOIN Department AS D ON ED.DepartmentID=D.DepartmentID

GROUP BY D.DepartmentName

HAVING---> we can use to filter the apply, mostly we use aggregate function in this clause

Write a query to display those departmentID where minimum 3 employee is working?

SELECT DepartmentID,COUNT(\*) FROM EmployeeDetails

GROUP BY DepartmentID

HAVING COUNT(\*)>=3

ORDER BY---> this we can use to sort the data.

SELECT \* FROM EmployeeDetails

ORDER BY Salary ASC

SELECT \* FROM EmployeeDetails

ORDER BY Salary DESC

WINDOW FUNCTION:

a)-ROW\_NUMBER ()

b)-RANK ()

c)-DENSE\_RANK ()

d)-NTILE ()

SELECT ROW\_NUMBER()OVER(ORDER BY Salary ASC) AS RowID, \* FROM EmployeeDetails

SELECT RANK () OVER (ORDER BY Salary ASC) AS RankID, \* FROM EmployeeDetails

SELECT DENSE\_RANK () OVER (ORDER BY Salary ASC) AS DenseRankID, \* FROM EmployeeDetails

NTILE ()

NTILE (3) ---> this will create 3 group

In My table 9 Record:

9/3--->3

G1 --->3----> All Record will be assigned as RowNo 1

G2 --->30------>2

G3---->3--->3

SELECT Ntile (3) OVER (ORDER BY Salary ASC) AS NtileID, \* FROM EmployeeDetails

SELECT Ntile (5) OVER (ORDER BY Salary ASC) AS NtileID, \* FROM EmployeeDetails

9/2=4.5

G1 ----> 5

G2----> 4

Total record we have 17

NTILE (5)

17/5---> 3--> 2 record Extra

G1 -->3+1=4

G2--->3+1=4

G3-->3

G4-->3

G5-->3

18/5

G1 -->3+1=4

G2--->3+1=4

G3-->3+1=4

G4-->3

G5-->3

USERDEFINED FUNCTION:

a)-SCALAR FUNCTION---> this function will only return a single value as a output

b)-Table Valued Function----> those function which is going to return more than one record or more than one column.

CREATE FUNCTION function\_name

(

@Parametername DataType

)

RETURNS DATATYPE

AS

BEGIN

RETURN

(

----CODE

)

END

Write a function which will accept a DepartmentID and Return a EmployeeCount ?

udf---->user Defined function

CREATE FUNCTION udf\_get\_employeecount\_bydepartment

(

@DeptID INT

)

RETURNS INT

AS

BEGIN ---Starting

RETURN (

SELECT COUNT(\*) FROM EmployeeDetails WHERE DepartmentID=@DeptID

)

END---Ending

How we call our userdefined function?

SELECT dbo.udf\_get\_employeecount\_bydepartment (2)

SELECT dbo.udf\_get\_employeecount\_bydepartment (3)

SELECT dbo.udf\_get\_employeecount\_bydepartment (1)

SELECT COUNT(\*) FROM EmployeeDetails

SELECT COUNT(\*) FROM EmployeeDetails WHERE DepartmentID=1 AND Salary>50000

SELECT COUNT(\*) FROM EmployeeDetails WHERE DepartmentID=2 AND Salary>50000

ALTER FUNCTION udf\_get\_employeecount\_bydepartment

(

@DeptID INT

)

RETURNS INT

AS

BEGIN ---Starting

RETURN

(

SELECT COUNT(\*) FROM EmployeeDetails WHERE DepartmentID=@DeptID AND Salary>50000

)

END---Ending

If we want to remove any function from our database then we can use Drop?

DROP FUNCTION udf\_get\_employeecount\_bydepartment?

Write a function which will accept DepartmentName from the user and return maximum salary?

CREATE FUNCTION udf\_get\_max\_salary\_By\_Department\_1stJune

(

@DepartmentName VARCHAR(50)

)

RETURNS INT

AS

BEGIN

RETURN

(

SELECT

MAX(Salary)

FROM

EmployeeDetails AS ED

JOIN Department AS D ON ED.DepartmentID=D.DepartmentID

WHERE D.DepartmentName=@DepartmentName

)

END

SELECT dbo.udf\_get\_max\_salary\_By\_Department\_1stJune ('IT')

Write a query which will accept Departmentname and return all the employee who is working under this department?

CREATE FUNCTION function\_name

(

@parametername Datatype

)

RETURNS TABLE

RETURN

(

)

CREATE FUNCTION udf\_get\_employee\_InfoByDepartment

(

@DepartmentName VARCHAR(50)

)

RETURNS TABLE

RETURN

(

SELECT

EmployeeID,FirstName,LastName,Salary FROM EmployeeDetails AS ED

JOIN Department AS D ON ED.DepartmentID=D.DepartmentID

WHERE D.DepartmentName=@DepartmentName

)

SELECT \* FROM udf\_get\_employee\_InfoByDepartment ('IT')

SELECT \* FROM udf\_get\_employee\_InfoByDepartment ('Civil')

Write a function which will accept DepartmentID and ClientID from the user and return employeeinfo?

CREATE FUNCTION udf\_get\_employeeData\_by\_dept\_Client

(

@DepartmentID INT,

@ClientID INT

)

RETURNS TABLE

RETURN

(

SELECT

EmployeeID,FirstName,LastName,Salary,DepartmentID,ClientID

FROM

EmployeeDetails

WHERE DepartmentID=@DepartmentID AND ClientID=@ClientID

)

SELECT \* FROM udf\_get\_employeeData\_by\_dept\_Client (1,2)

DROP FUNCTION udf\_get\_employeeData\_by\_dept\_Client

STORE PROCEDURE---> this is a another sql server object, with the help of this we can perform any kind of operation. eg. -->DML/DQWL/DDL

CREATE PROCEDURE procedure\_name

(

@ParameterName DataType,

@ParameterName2 DataType

)

AS

BEGIN ----Starting Point

END------Ending Point

Write a procedure which will accept a below datapoint from the user and insert a data into our employeedetails table?

a)-EmployeeID

b)-FirstName

c)-LastName

d)-Salary

e)-DepartmentID

USP---> User Store Procedure

CREATE PROCEDURE usp\_Insert\_EmployeeDetails\_1stJune

(

@EmployeeID INT,

@FirstName VARCHAR(50),

@LastName VARCHAR(50),

@Salary INT,

@DepartmentID INT

)

AS

BEGIN

INSERT INTO EmployeeDetails

(EmployeeID,FirstName,LastName,Salary,DepartmentID)

VALUES(@EmployeeID,@FirstName,@LastName,@Salary,@DepartmentID)

END

How we can executed the Store procedure?

EXECUTE usp\_Insert\_EmployeeDetails\_1stJune 20 ,'Srikanth','Kumar',60000,1

SELECT \* FROM EmployeeDetails

In this existing procedure (usp\_Insert\_EmployeeDetails\_1stJune) if user is passing salary less than 10k so instead of inserting a data we have to display a message please pass salary value more than 10k, if salary is more than 10k then we have to simply insert a data.

ALTER PROCEDURE usp\_Insert\_EmployeeDetails\_1stJune

(

@EmployeeID INT,

@FirstName VARCHAR(50),

@LastName VARCHAR(50),

@Salary INT,

@DepartmentID INT

)

AS

BEGIN

IF(@Salary>10000) ---> if this condition will be true only on that time sql can go inside the IF part

BEGIN

INSERT INTO EmployeeDetails (EmployeeID, FirstName, LastName, Salary, DepartmentID)

VALUES (@EmployeeID, @FirstName, @LastName, @Salary, @DepartmentID)

END

ELSE

BEGIN

SELECT ' please pass salary value more than 10k'

END

END

EXECUTE usp\_Insert\_EmployeeDetails\_1stJune 21 ,'Souvik','Chatterjee',9000,1

SELECT \* FROm EmployeeDetails

EXECUTE usp\_Insert\_EmployeeDetails\_1stJune 21 ,'Souvik','Chatterjee',19000,1

Write a procedure which will accept a below datapoint from the user and insert a data into our employeedetails table?

a)-EmployeeID

b)-FirstName

c)-LastName

d)-Salary

e)-DepartmentID

note\*\*\*---> before inserting a data in employeedetails we have to check the employeeid which is passed by the user is already present in Employeedetails table or not, if it's not present then we can insert a data otherwise we have to display a msg 'EmployeeID is already exists'

EXECUTE usp\_Insert\_EmployeeDetails\_1stJune\_v2 11 ,'Sahil','Jaiswal',12000,1

SELECT \* FROM EmployeeDetails

CREATE PROCEDURE usp\_Insert\_EmployeeDetails\_1stJune\_v2

(

@EmployeeID INT,

@FirstName VARCHAR(50),

@LastName VARCHAR(50),

@Salary INT,

@DepartmentID INT

)

AS

BEGIN

IF EXISTS (SELECT EmployeeID FROM EmployeeDetails WHERE

EmployeeID=@EmployeeID)---> if this query will return any data then sql allow to go inside the if part

BEGIN

SELECT 'EmployeeID already exists'

END

ELSE

BEGIN

INSERT INTO EmployeeDetails (EmployeeID, FirstName, LastName, Salary, DepartmentID)

VALUES (@EmployeeID, @FirstName, @LastName, @Salary, @DepartmentID)

END

END

EXCEPTION HANDLING: --> mechanism which we can use to handle the exception is called Exception handling

TYPE OF ERROR:

a)-SYNTAX ERROR

b)-EXCEPTION(Run Time Error)

SELECT FROM EmployeeDetails

SELECT 1+1

SELECT 'A'+'B'

SELECT 1+'A'

BEGIN TRY

SELECT 1+'A'

SELECT \* FROM EmployeeDEtails

END TRY

BEGIN CATCH

SELECT 'something went wrong'

END CATCH

Write a store proc which will accept employeeid and return employeeinfo?

EXECUTE usp\_Get\_EmployeeInfoByEmpID 1

ALTER PROCEDURE usp\_Get\_EmployeeInfoByEmpID

(

@EmployeeID INT

)

AS

BEGIN

BEGIN TRY

--SELECT 1/0

SELECT

\*

FROM

EmployeeDetails

WHERE EmployeeID=@EmployeeID

END TRY

BEGIN CATCH

SELECT 'Something Went wrong'

END CATCH

END

TRANSACTION:

SELECT \* FROM CustomerAccountDetails

Transfer 200 rs from CustomerID 2 to CustomerID-->1

UPDATE CustomerAccountDetails SET CurrentBalance=CurrentBalance-200 WHERE CustomerID=2

UPDATE CustomerAccountDetails SET CurrentBalance=CurrentBalance+200 WHERE CustomerID=1

BEFORE UPDATE:

CustomerID CustomerName CurrentBalance

1 Sahil Jaiswal 700

2 Ankit Gupta 1300

UPDATE CustomerAccountDetails SET CurrentBalance=CurrentBalance-300 WHERE

CustomerID=2

SELECT 1+'A'

UPDATE CustomerAccountDetails SET CurrentBalance=CurrentBalance+300 WHERE

CustomerID=1

CustomerID CustomerName CurrentBalance

1 Sahil Jaiswal 700

2 Ankit Gupta 1000

BEGIN TRANSACTION --------->Whenever we execute this command internally sql

started a Transaction block,

--now after that whatever changes we are doing that is not

permanently saved into our database.

SELECT \* FROM EmployeeDEtails

DELETE FROM EmployeeDetails WHERE EmployeeID=21

SELECT @@TRANCOUNT

DELETE FROM EmployeeDetails

COMMIT TRANSACTION-----> Whenever we execute this command it will save all the changes permanently whatever we made after begin transaction, it will close the open transaction block

ROLLBACK TRANSACTION-----> whenever we execute this command it will revert all the changes whatever we made after the Begin transaction and it also close the transaction block.

SELECT \* FROM CustomerAccountDetails

CustomerID CustomerName CurrentBalance

1 Sahil Jaiswal 700

2 Ankit Gupta 1000

BEGIN TRY

BEGIN TRANSACTION

UPDATE CustomerAccountDetails SET CurrentBalance=CurrentBalance-300 WHERE

CustomerID=2

SELECT 1+'A'

UPDATE CustomerAccountDetails SET CurrentBalance=CurrentBalance+300 WHERE

CustomerID=1

COMMIT TRANSACTION

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION

SELECT 'something went wrong'

END CATCH

PIVOT: ---> this concept we can use to transform the row data into columnar format

SELECT \* FROM SalesOrder

CREATE TABLE SalesOrder

(

OrderID INT,

ProductName VARCHAR(50),

OrderYear INT,

OrderAmount INT

)

INSERT INTO SalesOrder(OrderID,ProductName,OrderYear,OrderAmount) SELECT

1,'Fan',2023,4000

INSERT INTO SalesOrder(OrderID,ProductName,OrderYear,OrderAmount) SELECT

2,'Fan',2023,3000

INSERT INTO SalesOrder(OrderID,ProductName,OrderYear,OrderAmount) SELECT

3,'Car',2023,500000

INSERT INTO SalesOrder(OrderID,ProductName,OrderYear,OrderAmount) SELECT

4,'Car',2023,700000

INSERT INTO SalesOrder(OrderID,ProductName,OrderYear,OrderAmount) SELECT

5,'MotorBike',2023,50000

INSERT INTO SalesOrder(OrderID,ProductName,OrderYear,OrderAmount) SELECT

6,'MotorBike',2023,45000

INSERT INTO SalesOrder(OrderID,ProductName,OrderYear,OrderAmount) SELECT

7,'Fan',2024,1200

INSERT INTO SalesOrder(OrderID,ProductName,OrderYear,OrderAmount) SELECT

8,'Fan',2024,3000

INSERT INTO SalesOrder(OrderID,ProductName,OrderYear,OrderAmount) SELECT

9,'Car',2024,1000000

INSERT INTO SalesOrder(OrderID,ProductName,OrderYear,OrderAmount) SELECT

10,'Car',2024,1200000

INSERT INTO SalesOrder(OrderID,ProductName,OrderYear,OrderAmount) SELECT

11,'MotorBike',2024,70000

INSERT INTO SalesOrder(OrderID,ProductName,OrderYear,OrderAmount) SELECT

12,'MotorBike',2024,80000

---3rd Step

SELECT

ProductName, [2023],[2024]

FROM

---1st Step

(

SELECT

ProductName, OrderYear, OrderAmount

FROM

SalesOrder

) AS W

--2nd Step

PIVOT

(

SUM(OrderAmount) FOR OrderYear IN ([2023], [2024])

) AS P1

UNPIVOT---> transform the Columnar data to Row format

CREATE TABLE StudentSubjectMark

(

StudentID INT,

English INT,

Math INT,

Science INT

)

INSERT INTO StudentSubjectMark SELECT 1,55,98,95

INSERT INTO StudentSubjectMark SELECT 2,75,67,76

INSERT INTO StudentSubjectMark SELECT 3,87,74,36

SELECT

StudentID,

SubjectName,Marks

FROM

--1st step

(

SELECT

StudentID,English,Math,Science

FROM

StudentSubjectMark

)AS P1

--2nd Step

UNPIVOT

(

Marks FOR SubjectName IN ([English],[Math],[Science])

)AS P2

SELECT EmployeeID FROM

(

SELECT

\*

FROM

EmployeeDetails

) AS W

VIEW: ---> view is like a virtual table whenever we are executing the query on the view it is look like we are accessing a data from the table, but every time SQL run the query which we have under the view definition.

CREATE VIEW vw\_EmployeeDetails\_8thJune

AS

SELECT

EmployeeID,

FirstName,

LastName,

DepartmentID

FROM

EmployeeDetails

SELECT \* FROM vw\_EmployeeDetails\_8thJune

ALTER VIEW Vw\_EmployeeFullInfo

AS

SELECT

EmployeeID,

FirstName,

LastName,

ED.DepartmentID,

D.DepartmentName,

ED.DOB

FROM

EmployeeDetails AS ED

LEFT JOIN Department AS D ON ED.DepartmentID=D.DepartmentID

SELECT \* FROM Vw\_EmployeeFullInfo

---> 10 different places---Have to DOB column in every place

SELECT

EmployeeID,

FirstName,

LastName,

ED.DepartmentID,

D.DepartmentName

FROM

EmployeeDetails AS ED

LEFT JOIN Department AS D ON ED.DepartmentID=D.DepartmentID

DROP VIEW Vw\_EmployeeFullInfo

Write a procedure which will accept below parameter and transfer the fund

from one customer to another customer:

a)-FromCustomerID

b)-ToCustomerID

c)-TransferAmount

Note:

a)-Please handle run time error also if any run time error got occurred then we have to display

'Network down please try after few min.'

b)-before doing the transfer please check the transfer amount should always greater or equal to current balance.

SELECT \* FROM CustomerAccountDetails

--EXEC usp\_customer\_fund\_transfer 1,2,100

CREATE PROCEDURE usp\_customer\_fund\_transfer

(

@FromCustomerID INT,

@ToCustomerID INT,

@TransferAmount INT

)

AS

BEGIN

BEGIN TRY

BEGIN TRANSACTION

IF EXISTS(SELECT CurrentBalance FROM CustomerAccountDetails WHERe

CustomerID=@FromCustomerID

AND CurrentBalance>=@TransferAmount)

BEGIN

UPDATE CustomerAccountDetails SET CurrentBalance=CurrentBalance-

@TransferAmount WHERe CustomerID=@FromCustomerID

UPDATE CustomerAccountDetails SET CurrentBalance=CurrentBalance

+@TransferAmount WHERe CustomerID=@ToCustomerID

END

ELSE

BEGIN

SELECT 'Insufficent Funds'

END

COMMIT TRANSACTION

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION

SELECT 'Network down please try after few min.'

END CATCH

END

SUBQUERY-----> Query with in another query is called subquery.

Write a query to get the those employeeinfo who is getting maximum salary?

SELECT MAX(Salary) FROM EmployeeDetails

SELECT \* FROM EmployeeDetails WHERE Salary=100000

SELECT

\*

FROM

EmployeeDetails

WHERE Salary=(SELECT MAX(Salary) FROM EmployeeDetails)

SELECT

\*

FROM

EmployeeDetails WHERE DepartmentID IN(SELECT DepartmentID FROM

EmployeeDetails)

Write a query to get the information for those employee who is getting maximum salary within the department?

co-releated subquery---> this is the extended of the subquer now here one query is depenedt on each other

SELECT

\*

FROM

EmployeeDetails AS O

WHERE Salary=(SELECT MAX(Salary) FROM Employeedetails AS I WHERE

I.DepartmentID=O.DepartmentID)

ORDER BY EmployeeID ASC