20-feb-24

Database :-

---------------

=> a Database is a organized collection of interrelated data. For example a univ db

stores data related to students,courses and faculty etc and a bank db stores

data related to customers,accounts,transactions etc.

Types of Databases :-

------------------------------

1 OLTP DB (online transaction processing)

2 OLAP DB (online analytical processing)

=> organizations uses OLTP for storing day-to-day transactions and OLAP for analysis.

=> OLTP is for running business and OLAP is for to analyze business.

=> day-to-day opertions on db includes

C create

R read

U update

D delete

DBMS :-

-----------

=> DBMS stands for Database Management System and It is a software used to

create and to manage DB.

USER---------------------------DBMS------------DB

=> DBMS is an interface between user and db.

Evolution of DBMS :-

---------------------------

1960 FMS (File Mgmt System)

1970 HDBMS (Hierarchical DBMS)

NDBMS (Network DBMS)

1980 RDBMS (Relational DBMS)

1990 ORDBMS (Object Relational DBMS)

RDBMS :-

--------------

=> RDBMS concepts introduced by Edgar Frank CODD

=> E.F.CODD introduced 12 rules called CODD rules

=> a db that supports all 12 rules called perfect rdbms

information rule :-

------------------------

=> according to informtion rule data must be organized in tables i.e. rows and columns

CUST

CID CNAME CITY => columns / fields / attributes

10 SACHIN MUM

11 RAHUL DEL

12 VIJAY HYD => row / record / tuple

DATABASE = COLLECTION OF TABLES

TABLE = COLLECTION OF ROWS AND COLS

ROW = COLLECTION OF FIELD VALUES

COLUMNS = COLLECTION OF VALUES BELONGS TO ONE FIELD

=> every table must contain primary key to uniquely identify the records

ex :- accno,empid,aadharno,panno,voterid

Features :-

--------------

1 easy to access and manipulate data

2 less redundency (duplication of data)

3 more security

4 gurantees data quality or consistency

5 supports data sharing

6 supports transactions

RDBMS softwares :- (SQL Databases)

----------------------------

SQL SERVER microsoft

ORACLE oracle corp

MYSQL oracle corp

DB2 IBM

POSTGRESQL postgresql global development group

RDS amazon

NoSQL Databases :-

----------------------------

1 MongoDB

2 cassandra

ORDBMS :-

--------------

=> object relational dbms

=> it is the combination of rdbms & oops

ordbms = rdbms + oops (reusability)

=> rdbms doesn't support reusability but ordbms supports reusability

ORDBMS softwares :-

-----------------------------

1 sql server

2 oracle

3 postgresql

22-FEB-24

DB Development Life Cycle :-

---------------------------------------

Analysis

Design

Development

Testing

Implementation

Maintenance

Design :-

-----------

=> Designing DB means designing tables

=> DB is designed by DB Designers / Architects

=> DB is designed by using

1 ER Model (Entity Relationship Model)

2 Normalization

Development :-

--------------------

=> DB is developed by DB Developers & DBAs (DB Admins)

=> DB is developed by using any rdbms tools like sql server

Developers DBAs

creating tables installation of sql server

creating views creating database

creating synonyms creating logins

creating sequences db backup & restore

creating indexes db export & import

creating procedures db upgradation & migration

creating functions performance tuning

creating triggers

writing queries

sql server 2014 sql server 2022 upgradation

mysql sql server migration

Testing :-

------------

=> DB is tested by QA team (Quality Assurance)

=> DB is tested by using

1 manual

2 automation

Implementation :-

-----------------------

=> Implementation means moving db from dev environment to prod environment

summary :-

what is db ?

what is dbms ?

what is rdbms ?

what is ordbms ?

what db development ?

23-feb-24 SQL SERVER

===========

=> SQL Server is basically rdbms product from microsoft and also supports

ordbms features and used to create and to manage database.

=> SQL SERVER is used for DB Development and Administration

versions of sql server :-

--------------------------------

Version YEAR

SQL SERVER 1.1 1991

SQL SERVER 4.2 1993

SQL SERVER 6.0 1995

SQL SERVER 6.5 1996

SQL SERVER 7.0 1998

SQL SERVER 2000 2000

SQL SERVER 2005 2005

SQL SERVER 2008 2008

SQL SERVER 2012 2012

SQL SERVER 2014 2014

SQL SERVER 2016 2016

SQL SERVER 2017 2017

SQL SERVER 2019 2019

SQL SERVER 2022 2022

client/server architecture :-

-------------------------------------

1 SERVER

2 CLIENT

SERVER :-

---------------

=> server is a system where sql server is installed and running

=> inside the server sql server manages

1 DB

2 INSTANCE

=> DB is created in harddisk and acts as permanent storage

=> INSTANCE is created in ram and acts as temporary storage

CLIENT :-

-------------

=> client is also a system from where users can

1 connects to server

2 submit requests to server

3 receive response from server

client tool :-

--------------

SSMS (SQL SERVER MGMT STUDIO)

USER-----SSMS--------------------------------------------SQL SERVER-------DB

USER----SQLPLUS------------------------------------------ORACLE------------DB

USER------MYSQLWORKBENCH--------------------------MYSQL-----------DB

SQL :-

---------

=> SQL stands for structured query language.

=> It is a language used to communicate with sql server.

=> user communicates with sql server by sending commands called queries

=> a query is a command / instruction / question given to sql server to perform

some operation over db.

=> SQL is introduced by IBM and initial name of this language was "SEQUEL"

and later it is renamed to SQL.

=> SQL is common to all Relational Databases

SQL SERVER ORACLE MYSQL POSTGRESQL

SQL SQL SQL SQL

USER-----SSMS-----------------SQL---------------------SQL SERVER-------DB

USER-----SQLPLUS------------SQL--------------------ORACLE--------------DB

=> based on operations over db sql lang is categorized into 5 sublanguages

DDL (DATA DEFINITION LANG)

DML (DATA MANIPULATION LANG)

DQL (DATA QUERY LANG)

TCL (TRANSACTION CONTROL LANG)

DCL (DATA CONTROL LANG)

SQL

DDL DML DQL TCL DCL

create insert select commit grant

alter update rollback revoke

drop delete save transaction

truncate merge

DATA & DATA DEFINITION :-

---------------------------------------

EMPID ENAME SAL => DATA DEFINITION / METADATA

100 SACHIN 6000 => DATA

24-feb-24

How to connect to sql server :-

-----------------------------------------

=> to connect to sql server open SSMS and enter following details

SERVER TYPE :- DB ENGINE

SERVER NAME :- DESKTOP-G2DM7GI

AUTHENTICATION :- WINDOWS / SQL SERVER

LOGIN :- SA (SYSTEM ADMIN)

PASSWORD :- 123

=> click connect

=> a connection established between user and sql server through ssms

How to Create Database :-

------------------------------------

=> In object explorer select Databases => New Database

Enter Database Name ;- DB730

=> click OK

=> a DB is created with following two files

1 DATA FILE ( .MDF) (Master Data File)

2 LOG FILE (.LDF) (Log Data File)

=> Data File stores Data and Log File stores operations

NAME TYPE INITIAL SIZE AUTOGROWTH PATH

DB730 DATA 8 MB 64 MB C:\-------\DATA

DB730\_LOG LOG 8 MB 64 MB C:\-------\DATA

Command to Create Database :-

------------------------------------------

CREATE DATABASE <DBNAME>

EX :-

=> open master database and execute the following command

CREATE DATABASE NARESHIT

26-FEB-24

download & install :-

-------------------------

1 sql server

2 ssms (SQL SERVER MANAGEMENT STUDIO)

SQL SERVER :-

---------------------

Download :-

---------------

1 go to google type sql server 2022 download

2 https://www.microsoft.com/en-in/sql-server/sql-server-downloads

3 download developer edition

step by step installation :-

--------------------------------

1 go to google type step by step installation of sql server 2022 developere edition

2 https://www.mssqltips.com/sqlservertip/7313/install-sql-server-2022/

SSMS :-

-----------

Download :-

----------------

1 go to google type download sql server management studio

2 https://learn.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver16

Datatypes in SQL SERVER :-

--------------------------------------

=> a datatype specifies

1 type of the data allowed in column

2 amount of memory required for column

DATATYPES

CHAR INTEGER FLOAT CURRENCY DATE BINARY

tinyint decimal(p,s) smallmoney date binary

ASCII UNICODE smallint numeric(p,s) money time varbinary

int datetime varbinary(max)

char nchar bigint

varchar nvarchar numeric(p)

varchar(max) nvarchar(max)

CHAR(size) :-

-------------------

=> allows character data upto 8000 chars

=> recommended for fixed length char columns

ex :- NAME CHAR(10)

SACHIN----

wasted

RAVI------

wasted

=> in CHAR datatype extra bytes are wasted , so CHAR is not recommended for variable length

fields and char is recommended for fixed length fields

ex :- GENDER CHAR(1)

M

F

STATE\_CODE CHAR(2)

AP

TS

COUNTRY\_CODE CHAR(3)

IND

USA

VARCHAR(SIZE) :-

------------------------

=> allows character data upto 8000 chars

=> recommended for variable length fields

ex :- NAME VARCHAR(10)

SACHIN----

released

27-feb-24

VARCHAR(MAX) :-

--------------------------

=> allows character data upto 2GB

ex :- review VARCHAR(MAX)

=> CHAR/VARCHAR/VARCHAR(MAX) allows ascii chars (256) that includes A-Z,a-z,0-9 &

special chars.

ex :- PANNO CHAR(10)

IFSC CHAR(10)

EMAILID VARCHAR(30)

NCHAR/NVARCHAR/NVARCHAR(MAX) :- (N => National)

--------------------------------------------------------

=> allows unicode chars (65536 chars) that includes all ascii chars and also chars belongs to

different languages.

Integer Types :-

---------------------

=> allows numbers without decimal (integers)

TINYINT 1 BYTE 0 TO 255

SMALLINT 2 BYTES -32768 TO 32767

INT 4 BYTES -2^31 TO 2^31-1 (-2147483648 to 2147483647)

BIGINT 8 BYTES -2^63 TO 2^63-1

ex :- AGE TINYINT

EMPID SMALLINT

NUMERIC(P) :-

----------------------

=> allows numbers upto 38 digits

=> allows numbers without decimal

ex :- EMPID NUMERIC(4)

10

100

1000

10000 => NOT ALLOWED

AGE NUMERIC(2)

PHONE NUMERIC(10)

AADHARNO NUMERIC(12)

DECIMAL(P,S) / NUMERIC(P,S) :-

----------------------------------------------

=> allows numbers with decimal (float)

P => precision => total no of digits allowed

S => scale => no of digits allowed after decimal

ex :- SALARY DECIMAL(7,2)

5000

5000.55

50000.55

500000.55 => NOT ALLOWED

5000.5678 => ALLOWED => 5000.57

5000.5645 => ALLOWED => 5000.56

NOTE :- if before decimal exceeds number is not accepted

if after decimal exceeds number is rounded

SAVG DECIMAL(5,2)

BALANCE DECIMAL(13,4)

Currency Types :-

-----------------------

=> used for fields related to money

SMALLMONEY 4 BYTES -214748.3648 to 214748.3647

MONEY 8 BYTES -922337203685477.5808

to

922337203685477.5807

EX :- SALARY SMALLMONEY

BALANCE MONEY

DATE & TIME :

--------------------

DATE => allows only date

TIME => allows only time

DATETIME => allows both date & time

=> default date format in sql server is YYYY-MM-DD

=> default time format is HH:MI:SS

EX :- DOB DATE

2003-10-05

LOGIN TIME

9:00:00

TXN\_DT DATETIME

2024-02-27 8:00:00

Binary Types :-

---------------------

=> used for storing multimedia objects like audio,video,images

1 BINARY => allows binary data upto 8000 bytes (fixed length)

2 VARBINARY => allows binary data upto 8000 bytes (variable length)

3 VARBINARY(MAX) => allows binary data upto 2GB

ex :- PHOTO BINARY(5000)

PHOTO VARBINARY(5000)

MOVIE VARBINARY(MAX)

28-feb-24

CREATING TABLE IN SQL SERVER :-

---------------------------------------------------

CREATE TABLE <TABNAME>

(

COLNAME DATATYPE(SIZE),

COLNAME DATATYPE(SIZE),

------------------------------------ ,

--------------------------------------

)

Rules :-

----------

1 tabname should start with alphabet

2 name should not contain spaces & special chars but allows \_ , $ , #

3 name can be upto 128 chars

4 table can have 1024 cols

5 no of rows unlimited

ex :- 123emp INVALID

emp 123 INVALID

emp\*123 INVAILD

emp\_123 VALID

=> create table with following structure ?

EMP

EMPID ENAME JOB SAL HIREDATE AGE DNO

CREATE TABLE EMP

(

EMPID NUMERIC(4) ,

ENAME VARCHAR(10),

JOB VARCHAR(10),

SAL SMALLMONEY,

HIREDATE DATE ,

AGE TINYINT ,

DNO TINYINT

)

=> above command created table structure that includes columns,datatype and size

SP\_HELP :- (SP => stored procedure)

----------------

=> command to see the table structure

SP\_HELP <tabname>

ex :-

SP\_HELP EMP

COLUMN NAME DATATYPE LENGTH

EMPID numeric 5

ENAME varchar 10

JOB varchar 10

SAL smallmoney 4

HIREDATE date 3

AGE tinyint 1

DNO tinyint 1

INSERTING DATA INTO TABLE :-

---------------------------------------------

=> "INSERT" command is used to insert data into table.

=> we can insert

1 single row

2 multiple rows

Inserting single row :-

----------------------------

syn :-

INSERT INTO <tabname> VALUES(v1,v2,v3,--------)

ex :-

INSERT INTO emp VALUES(100,'SACHIN','CLERK',4000,'2024-02-28',25,10)

INSERT INTO emp VALUES(101,'DHONI','MANAGER',8000,GETDATE(),35,20)

inserting multiple rows :-

--------------------------------

INSERT INTO emp VALUES(102,'ARVIND','ANALYST',9000,'2020-03-15',40,30),

(103,'DAVID','MANAGER',10000,'2018-10-05',30,10)

inserting nulls :-

---------------------

=> a null means blank or empty

=> it is not equal to 0 or space

=> nulls can be inserted in two ways

method 1 :-

---------------

INSERT INTO emp VALUES(104,'KUMAR',NULL,5000,'2019-05-10',28,NULL)

method 2 :-

--------------

INSERT INTO emp(empid,ename,sal,hiredate,age) VALUES(105,'RAVI',6000,'2021-04-20',32)

remaining two fields job,dno are filled with NULLs

Operators in SQL SERVER :-

---------------------------------------

Arithmetic Operators => + - \* / %

Relational Operators => > >= < <= = <> !=

Logical Operators => AND OR NOT

Special Operators => BETWEEN

IN

LIKE

IS

ANY

ALL

EXISTS

PIVOT

Set Operators => UNION

UNION ALL

INTERSECT

EXCEPT

Displaying Data :-

-----------------------

=> "SELECT" command is used to display data from table.

=> we can display all rows or specific rows

=> we can display all cols or specific cols

syn :- SELECT columns / \* FROM tabname

SQL = ENGLISH

QUERIES = SENTENCES

CLAUSES = WORDS

\* => all columns

Examples :-

=> display employee names and salaries ?

SELECT ename,sal FROM emp

=> display all the data from emp table ?

SELECT \* FROM emp

29-FEB-24

WHERE clause :-

--------------------------

=> where clause is used to get specific row/rows from table

=> where clause is based on condition

SELECT columns / \*

FROM tabname

WHERE condition

condition :-

--------------

COLNAME OP VALUE

=> OP must be any relational operator like > >= < <= = <>

=> if cond = true row is selected , if false row is not selected

Ex :-

=> display employee details whose id = 103 ?

SELECT \* FROM EMP WHERE EMPID = 103

=> display employee details whose name = DHONI ?

SELECT \* FROM EMP WHERE ENAME = 'DHONI'

=> employees earning more than 5000 ?

SELECT \* FROM EMP WHERE SAL > 5000

=> employees joined after 2020 ?

SELECT \* FROM EMP WHERE HIREDATE > 2020 => ERROR

SELECT \* FROM EMP WHERE HIREDATE > '2020-12-31'

=> employees joined before 2020 ?

SELECT \* FROM EMP WHERE HIREDATE < '2020-01-01'

=> employees not working for dept 20 ?

SELECT \* FROM EMP WHERE DNO <> 20

=> employees earning multiples of 100 ?

SELECT \* FROM EMP WHERE SAL%100=0

compound condition :-

----------------------------

=> multiple conditions combined with AND / OR operators is called compound condition

WHERE COND1 AND COND2 RESULT

T T T

T F F

F T F

F F F

WHERE COND1 OR COND2 RESULT

T T T

T F T

F T T

F F F

=> employees working as clerk,manager ?

SELECT \* FROM EMP WHERE JOB='CLERK' ,'MANAGER' => ERROR

SELECT \* FROM EMP WHERE JOB='CLERK' OR JOB='MANAGER'

=> employees whose id = 100,103,105 ?

SELECT \* FROM EMP WHERE EMPID = 100 OR EMPID = 103 OR EMPID = 105

=> display 10th dept managers details ?

SELECT \* FROM EMP WHERE DNO=10 AND JOB='MANAGER'

=> display employees earning more than 5000 and less than 10000 ?

SELECT \* FROM EMP WHERE SAL > 5000 AND SAL < 10000

=> employees joined in 2020 year ?

SELECT \* FROM EMP WHERE HIREDATE >= '2020-01-01'

AND

HIREDATE <= '2020-12-31'

=> employees working as clerk,manager and earning more than 5000 ?

SELECT \*

FROM EMP

WHERE JOB='CLERK'

OR

JOB='MANAGER'

AND

SAL > 5000

=> above query returns clerks earning less than 5000 because operator SAL>5000

applied only to manager but not to clerk because operator AND has got more

priority than operator OR , to control this use ( )

SELECT \*

FROM EMP

WHERE (

JOB='CLERK'

OR

JOB='MANAGER'

)

AND

SAL > 5000

01-mar-24

IN opertator :-

-------------------

=> use IN operator for list comparision

=> use IN opertor for "=" comparision with multiple values

WHERE COLNAME = V1,V2,V3,------ => INVALID

WHERE COLNAME IN (V1,V2,V3,---) => VALID

=> employees working as clerk,manager ?

SELECT \* FROM EMP WHERE JOB = ' CLERK' OR JOB = 'MANAGER'

SELECT \* FROM EMP WHERE JOB IN ('CLERK','MANAGER')

=> employees whose id = 100,103,105 ?

SELECT \* FROM EMP WHERE EMPID IN (100,103,105)

=> employees not working for dept 10,20 ?

SELECT \* FROM EMP WHERE DNO NOT IN (10,20)

BETWEEN operator :-

------------------------------

=> use BETWEEN operator for comparision with range

WHERE COLNAME BETWEEN V1 AND V2 (COL >= V1 AND COL <= V2)

=> employees earning between 5000 and 10000 ?

SELECT \* FROM EMP WHERE SAL BETWEEN 5000 AND 10000

=> employees age between 30 and 40 ?

SELECT \* FROM EMP WHERE AGE BETWEEN 30 AND 40

=> employees not joined in 2020 year ?

SELECT \*

FROM EMP

WHERE HIREDATE NOT BETWEEN '2020-01-01' AND '2020-12-31'

Question :-

SELECT \*

FROM EMP

WHERE SAL BETWEEN 10000 AND 5000

A ERROR

B RETURNS ROWS

C RETURNS NO ROWS

D NONE

ANS :- C

WHERE SAL BETWEEN 5000 AND 10000 (SAL>=5000 AND SAL<=10000)

WHERE SAL BETWEEN 10000 AND 5000 (SAL>=10000 AND SAL<=5000)

NOTE :-

=> use BETWEEN operator with lower and upper but not with upper and lower

=> employees working as clerk,manager and earning between 5000 and 10000

and not joined in 2020 and working for dept 10,20 ?

SELECT \*

FROM EMP

WHERE JOB IN ('CLERK','MANAGER')

AND

SAL BETWEEN 5000 AND 10000

AND

HIREDATE NOT BETWEEN '2020-01-01' AND '2020-12-31'

AND

DNO IN (10,20)

=> list of samsung,redmi,realme mobile phones price between 10000 and 20000 ?

PRODUCTS

PRODID PNAME PRICE CATEGORY BRAND

SELECT \*

FROM PRODUCTS

WHERE BRAND IN ('SAMSUNG','REDMI','REALME')

AND

PRICE BETWEEN 10000 AND 20000

AND

CATEGORY='MOBILES'

=>

CREATE TABLE STUDENT

(

SNO INT,

SNAME VARCHAR(10),

S1 TINYINT,

S2 TINYINT,

S3 TINYINT

)

INSERT INTO STUDENT VALUES (1,'A',80,90,70),(2,'B',30,60,50)

STUDENT

SNO SNAME S1 S2 S3

1 A 80 90 70

2 B 30 60 50

=> list of students who are passed ?

SELECT \* FROM STUDENT WHERE S1>=35 AND S2>=35 AND S3>=35

=> list of students who are failed ?

SELECT \* FROM STUDENT WHERE S1<35 OR S2<35 OR S3<35

02-mar-24

LIKE operator :-

----------------------

=> use LIKE operator for pattern comparision

pattern => name starts with 'S'

name ends with 'D'

name contains 'A'

WHERE COLNAME LIKE 'PATTERN'

=> pattern contains alphabets,digits and wildcard chars

WILDCARD CHARS :-

-------------------------------

% => 0 or many chars

\_ => exactly 1 char

=> employees name starts with 'S' ?

SELECT \* FROM EMP WHERE ENAME LIKE 'S%'

=> name ends with 'D' ?

SELECT \* FROM EMP WHERE ENAME LIKE '%D'

=> name contains 'A' ?

SELECT \* FROM EMP WHERE ENAME LIKE '%A%'

=> where 'A' is the 2nd char in their name ?

SELECT \* FROM EMP WHERE ENAME LIKE '\_A%'

=> 'A' is the 3rd char from last ?

SELECT \* FROM EMP WHERE ENAME LIKE '%A\_\_'

=> name contains 4 chars ?

SELECT \* FROM EMP WHERE ENAME LIKE '\_\_\_\_'

=> employees joined in april month ? YYYY-MM-DD

SELECT \* FROM EMP WHERE HIREDATE LIKE '\_\_\_\_\_04\_\_\_'

=> employees joined in 2020 year ?

SELECT \* FROM EMP WHERE HIREDATE LIKE '2020%'

=> employees joined in 1st 9 days of any month any year ? YYYY-MM-DD

SELECT \* FROM EMP WHERE HIREDATE LIKE '%0\_'

Question :-

SELECT \* FROM EMP WHERE JOB IN ('CLERK','MAN%')

A ERROR

B RETURNS CLERK,MANAGER RECORDS

C RETURNS ONLY CLERK

D NONE

ANS :- C

SELECT \* FROM EMP WHERE JOB = 'CLERK' OR JOB LIKE 'MAN%'

ANS :- B

SCENARIO :-

------------------

CUST

CID CNAME

10 SACHIN\_TENDULKAR

11 VIRAT%KOHLI

12 MAHENDRA\_SINGH\_DHONI

=> display customer list name contains "\_" ?

SELECT \* FROM CUST WHERE CNAME LIKE '%\_%'

above query returns all cust records because "\_" is not treated as normal char and

it is treated as wildcard char , To overcome this problem use ESCAPE char.

SELECT \* FROM CUST WHERE CNAME LIKE '%\\_%' ESCAPE '\'

=> name contains "%" ?

SELECT \* FROM CUST WHERE CNAME LIKE '%\%%' ESCAPE '\'

=> name contains 2 "\_" ?

SELECT \* FROM CUST WHERE CNAME LIKE '%\\_%\\_%' ESCAPE '\'

IS operator :-

------------------

=> use IS opertaor for NULL comparision

WHERE COLNAME IS NULL

WHERE COLNAME IS NOT NULL

=> employees not assigned to any dept ?

SELECT \* FROM EMP WHERE DNO IS NULL

=> employees assigned to dept ?

SELECT \* FROM EMP WHERE DNO IS NOT NULL

summary :-

---------------

WHERE COLNAME IN (V1,V2,V3,---)

WHERE COLNAME BETWEEN V1 AND V2

WHERE COLNAME LIKE 'PATTERN'

WHERE COLNAME IS NULL

04-mar-24

ORDER BY clause :-

-----------------------------

=> order by clause is used to sort table data based on one or more columns

either in ascending or in descending order.

SELECT columns

FROM tabname

[WHERE cond]

ORDER BY colname ASC/DESC , -------

Examples :-

=> arrange employee list name wise ascending order ?

SELECT \*

FROM emp

ORDER BY ename ASC

=> arrange employyees list sal wise desc ?

SELECT \*

FROM emp

ORDER BY sal DESC

=> arrange employee list hiredate wise asc ?

SELECT \*

FROM emp

ORDER BY hiredate ASC

=> arrange employees list dept wise asc and with in dept sal wise desc ?

SELECT empno,ename,sal,deptno

FROM emp

ORDER BY deptno ASC , sal DESC

1 A 3000 20 3 C 4000 10

2 B 2000 30 6 F 3000 10

3 C 4000 10 ==============> 4 D 5000 20

4 D 5000 20 1 A 3000 20

5 E 6000 30 5 E 6000 30

6 F 3000 10 2 B 2000 30

=> arrange employee list dept wise asc and with in dept hiredate wise asc ?

SELECT empno,ename,hiredate,deptno

FROM emp

ORDER BY deptno ASC , hiredate ASC

=> arrange student list avg wise desc , m desc,p desc ?

STUDENT

SNO SNAME M P C

1 A 80 90 70

2 B 60 70 50

3 C 90 80 70

4 D 90 70 80

SELECT \*

FROM STUDENT

ORDER BY (M+P+C)/3 DESC ,M DESC,P DESC

3 C 90 80 70

4 D 90 70 80

1 A 80 90 70

2 B 60 70 50

=> to display avg in the output ?

SELECT \* , (M+P+C)/3 AS AVG

FROM STUDENT

ORDER BY (M+P+C)/3 DESC ,M DESC,P DESC

=> display ENAME ANNUAL SALARY

display only the employee joined in 1981 year

arrange output sal wise desc order ?

SELECT ENAME,HIREDATE,SAL\*12 AS [ANNUAL SALARY]

FROM EMP

WHERE HIREDATE LIKE '1981%'

ORDER BY SAL DESC

=> display ENAME SAL HRA DA TAX TOTSAL ?

HRA = house rent allowance = 20% on sal

DA = dearness allowance = 30% on sal

TAX = 10% on sal

TOTSAL = SAL + HRA + DA - TAX

SELECT ENAME,SAL,

SAL\*0.2 AS HRA,

SAL\*0.3 AS DA,

SAL\*0.1 AS TAX,

SAL+(SAL\*0.2)+(SAL\*0.3)-(SAL\*0.1) AS TOTSAL

FROM EMP

SMITH 800 160 240 80 1120

05-MAR-24

DISTINCT clause :-

------------------------

=> DISTINCT eliminates duplicates in select stmt output

SELECT DISTINCT COLNAME

SELECT DISTINCT COL1,COL2,---

SELECT DISTINCT \*

Example :-

1 SELECT DISTINCT JOB FROM EMP

ANALYST

CLERK

MANAGER

PRESIDENT

SALESMAN

2 SELECT DISTINCT DEPTNO FROM EMP

10

20

30

3 SELECT DISTINCT DEPTNO,JOB FROM EMP

--------------------

10 CLERK

20 ANALYST

20 CLERK

20 MANAGER

30 CLERK

30 MANAGER

30 SALESMAN

TOP clause :-

------------------

=> TOP clause used to display top n rows

SELECT TOP <n> \*

FROM TABNAME

[WHERE COND]

[ORDER BY ]

Ex :-

=> display first 5 rows from emp table ?

SELECT TOP 5 \* FROM EMP

=> display top 5 highest paid employees ?

SELECT TOP 5 \*

FROM EMP

ORDER BY SAL DESC

=> display top 3 max salaries ?

SELECT DISTINCT TOP 3 SAL

FROM EMP

ORDER BY SAL DESC

=> display top 3 employees based on experience ?

SELECT TOP 3 \*

FROM EMP

ORDER BY HIREDATE ASC

summary :-

WHERE => to filter data

ORDER BY => to sort data

DISTINCT => to eliminate duplicates

TOP => to select top n rows

DML commands :- (Data Manipulation Lang)

------------------------

INSERT

UPDATE

DELETE

MERGE

=> all DML commands acts on table data.

=> all DML commands are auto committed

=> to stop auto commit execute the following command

SET IMPLICIT\_TRANSACTIONS ON

=> to save the operation execute COMMIT command.

=> to cancel the operation execute ROLLBACK command.

UPDATE command :-

-------------------------------

=> command used to modify table data.

=> we can update all rows or specific rows.

=> we can update single column or multiple columns.

UPDATE <TABNAME>

SET COLNAME = VALUE , COLNAME = VALUE ,-------

[WHERE COND]

Ex :-

=> update all employees comm with 500 ?

UPDATE EMP SET COMM = 500

=> update employees comm with 500 whose comm = null ?

UPDATE EMP SET COMM = 500 WHERE COMM IS NULL

=> update employees comm with null whose comm <> null ?

UPDATE EMP SET COMM = NULL WHERE COMM IS NOT NULL

NULL assignment =

NULL comparision IS

=> increment sal by 20% and comm by 10% those working as salesman and joined in 1981 year ?

UPDATE EMP

SET SAL = SAL + (SAL\*0.2) , COMM = COMM + (COMM\*0.1)

WHERE JOB='SALESMAN'

AND

HIREDATE LIKE '1981%'

=> increase samsung,redmi,realme mobiles phones price by 10% ?

PRODUCTS

prodid pname price category brand

UPDATE PRODUCTS

SET PRICE = PRICE + (PRICE\*0.1)

WHERE CATEGORY = 'MOBILES'

AND

BRAND IN ('SAMSUNG','REDMI','REALME')

06-mar-24

DELETE command :-

------------------------

=> command used to delete row/rows from table

DELETE FROM <TABNAME> [WHERE COND]

=> delete employee whose id = 7844 ?

DELETE FROM EMP WHERE EMPNO = 7844

=> delete employees not working for 10,20 depts ?

DELETE FROM EMP WHERE DEPTNO NOT IN (10,20)

=> delete employees joined 1981 feb month ? (YYYY-MM-DD)

DELETE FROM EMP WHERE HIREDATE BETWEEN '1981-02-01' AND '1981-02-28'

DELETE FROM EMP WHERE HIREDATE LIKE '1981-02%'

=> delete all rows from emp table ?

DELETE FROM EMP

DDL commands :- (Data Definition Lang)

------------------------

CREATE

ALTER

DROP

TRUNCATE

=> all DDL commands acts on table structure

ALTER command :-

---------------------------

=> command used to modify table structure

=> using ALTER command we can

1 add columns

2 drop columns

3 modify column

changing datatype

changing size

Adding columns :-

---------------------------

ALTER TABLE <TABNAME>

ADD COLNAME DATATYPE(SIZE) , -----------------

Ex :-

=> add columns gender,dob to emp table ?

ALTER TABLE EMP

ADD GENDER CHAR(1) ,DOB DATE

after adding by default the new column is filled with NULLs , to insert data into the new

column use update command.

UPDATE EMP

SET GENDER = 'M' , DOB = '1960-05-10'

WHERE EMPNO = 7369

Droping columns :-

--------------------------

ALTER TABLE <TABNAME>

DROP COLUMN COL1,COL2,----------

Ex :-

=> drop columns gender,dob ?

ALTER TABLE EMP

DROP COLUMN GENDER,DOB

Modifying a column :-

-----------------------------

ALTER TABLE <TABNAME>

ALTER COLUMN COLNAME DATATYPE(SIZE)

Ex :-

=> change the datatype of sal to money ?

ALTER TABLE EMP

ALTER COLUMN SAL MONEY

=> change datatype of empno to smallint ?

ALTER TABLE EMP

ALTER COLUMN EMPNO SMALLINT

ALTER TABLE EMP

ALTER COLUMN EMPNO TINYINT => ERROR => empno's not in tinyint range

=> increase the size of ename to 20 ?

ALTER TABLE EMP

ALTER COLUMN ENAME VARCHAR(20)

ALTER TABLE EMP

ALTER COLUMN ENAME VARCHAR(5) => ERROR => some names contains

more than 5 chars

DROP command :-

---------------------------

=> command used to drop table from db

=> drops table structure along with data.

DROP TABLE <TABNAME>

EX :-

DROP TABLE EMP

07-mar-24

TRUNCATE :-

------------------

=> Deletes all the data from table but keeps structure

=> will empty the table

=> releases memory allocated for table

syn :- TRUNCATE TABLE <TABNAME>

Ex :- TRUNCATE TABLE EMP

DROP VS DELETE VS TRUNCATE :-

----------------------------------------------------

DROP DELETE/TRUNCATE

drops structure along with data deletes only data but not structure

DELETE TRUNCATE

1 DML command DDL command

2 can delete all rows can delete only all rows

and specific rows but can't delete specific rows

3 where cond can be where cond can't be used

used with delete with truncate

4 deletes row-by-row deletes all rows at a time

5 slower faster

6 will not release memory releases memory

7 will not reset identity will reset identity

SP\_RENAME (stored procedure) :-

-----------------------------------------------

=> command used to rename table or column

SP\_RENAME 'OLD NAME','NEW NAME'

Ex :-

=> rename table emp to employees ?

SP\_RENAME 'EMP' , 'EMPLOYEES'

=> rename column comm to bonus ?

SP\_RENAME 'EMPLOYEES.COMM','BONUS'

IDENTITY :-

----------------

=> used to generate sequence numbers

=> used to auto increment column values

IDENTITY(SEED,INCR)

SEED => start

optional

default 1

INCR => increment

optional

default 1

ex :-

CREATE TABLE STUDENT

(

SID INT IDENTITY(100,1) ,

SNAME VARCHAR(10)

)

INSERT INTO STUDENT(SNAME) VALUES('A')

INSERT INTO STUDENT(SNAME) VALUES('B')

INSERT INTO STUDENT(SNAME) VALUES('C')

INSERT INTO STUDENT(SNAME) VALUES('D')

SELECT \* FROM STUDENT

SID SNAME

100 A

101 B

102 C

103 D

DELETE VS TRUNCATE :-

-----------------------------------

DELETE TRUNCATE

SELECT \* FROM STUDENT SELECT \* FROM STUDENT

SID SNAME SID SNAME

100 A 104 K

101 B

102 C TRUNCATE TABLE STUDENT

103 D

INSERT INTO STUDENT(SNAME) VALUES('X')

DELETE FROM STUDENT

INSERT INTO STUDENT(SNAME) VALUES('K') SELECT \* FROM STUDENT

SELECT \* FROM STUDENT SID SNAME

100 X

SID SNAME

104 K

How to reset identity manually :-

-------------------------------------------

DBCC CHECKIDENT(table name,reseed,value)

DBCC => DB CONSOLE COMMAND

Ex :-

DBCC CHECKIDENT('STUDENT',RESEED,99)

How to insert explicit value into identity column :-

---------------------------------------------------------------

=> by default explicit value is not allowed into identity column

=> set identity\_insert to on to insert explicit value into identity column

SET IDENTITY\_INSERT <TABNAME> ON

Ex :-

INSERT INTO STUDENT VALUES(110,'P') => ERROR

SET IDENTITY\_INSERT STUDENT ON

INSERT INTO STUDENT(SID,SNAME) VALUES(110,'P') => 1 row affected

08-MAR-24

BUILT-IN FUNCTIONS IN SQL SERVER :-

---------------------------------------------------------

=> a function takes some input performs some calculation and returns one value

Types of functions :-

---------------------------

1 STRING

2 DATE

3 NUMERIC

4 CONVERSION

5 SPECIAL & SYSTEM

6 ANALYTICAL

7 AGGREGATE

STRING functions :-

--------------------------

UPPER() :-

---------------

=> converts string to uppercase

UPPER(arg)

string 'hello'

colname ename

ex :-

SELECT UPPER('hello') => HELLO

LOWER() :-

----------------

=> converts string to lowercase

LOWER(arg)

ex :-

SELECT LOWER('HELLO') => hello

=> display EMPNO ENAME SAL ?

display names in lowercase ?

SELECT EMPNO,LOWER(ENAME) AS ENAME,SAL FROM EMP

=> convert names to lowercase in table ?

UPDATE EMP SET ENAME = LOWER(ENAME)

LEN() :-

------------

=> returns string length i.e. no of chars

LEN(arg)

ex :-

SELECT LEN('HELLO WELCOME') => 13

=> employees name contains 4 chars ?

WHERE ENAME LIKE '\_\_\_\_'

SELECT \* FROM EMP WHERE LEN(ENAME)=4

=> arrange employee list length wise asc , if length is same then name wise asc ?

SELECT EMPNO,ENAME,SAL

FROM EMP

ORDER BY LEN(ENAME) ASC,ENAME ASC

LEFT() :-

------------

=> returns chars starting from left side

LEFT(string,length)

ex :-

SELECT LEFT('HELLO WELCOME',5) => HELLO

RIGHT() :-

--------------

=> returns chars starting from right side

RIGHT(string,length)

ex :-

SELECT RIGHT('HELLO WELCOME',4) => COME

=> employees name starts with 'S' ?

SELECT \* FROM EMP WHERE LEFT(ENAME,1) = 'S'

=> employees name ends with 'S' ?

SELECT \* FROM EMP WHERE RIGHT(ENAME,1) = 'S'

=> employees name starts and ends with same char ?

WHERE ENAME LIKE 'A%A'

OR

ENAME LIKE 'B%B'

SELECT \* FROM EMP WHERE LEFT(ENAME,1) = RIGHT(ENAME,1)

SUBSTRING() :-

-----------------------

=> returns characters starting from specific position

SUBSTRING(string,start,length)

ex :-

SELECT SUBSTRING('HELLO WELCOME',10,3) => COM

=> generate emailids for employees ?

EMPNO ENAME EMAILID

7369 smith smi736@tcs.com

7499 allen all749@tcs.com

'a' + 'b' => ab

SELECT EMPNO , ENAME,

LEFT(ENAME,3) + LEFT(EMPNO,3) + '@tcs.com' AS EMAILID

FROM EMP

=> store emailids in db ?

STEP 1 :- add emailid column to emp table

ALTER TABLE EMP

ADD EMAILID VARCHAR(30)

STEP 2 :- update the column with emailids

UPDATE EMP

SET EMAILID = LEFT(ENAME,3) + LEFT(EMPNO,3) + '@tcs.com'

09-mar-24

CHARINDEX() :-

----------------------

=> returns position of a char in a string

CHARINDEX(char,string,[start])

ex :-

SELECT CHARINDEX('O','HELLO WELCOME') => 5

H E L L O W E L C O M E

1 2 3 4 5 6 7 8 9 10 11 12 13

SELECT CHARINDEX('K','HELLO WELCOME') => 0

SELECT CHARINDEX('O','HELLO WELCOME',6) => 11

SELECT CHARINDEX('E','HELLO WELCOME',10) => 13

=> Display CID FNAME LNAME ?

CUST

CID CNAME

10 SACHIN TENDULKAR

11 VIRAT KOHLI

First Name = SUBSTRING(CNAME,1,CHARINDEX(' ',CNAME)-1)

Last Name = SUBSTR(CNAME,CHARINDEX(' ',CNAME)+1,LEN(CNAME))

SELECT CID,

SUBSTRING(CNAME,1,CHARINDEX(' ',CNAME)-1) AS FNAME,

SUBSTRING(CNAME,CHARINDEX(' ',CNAME)+1,LEN(CNAME)) AS LNAME

FROM CUST

REPLICATE() :-

---------------------

=> repeats character for given no of times

REPLICATE(char,length)

ex :-

SELECT REPLICATE('\*',5) => \*\*\*\*\*

=> display ENAME SAL ?

\*\*\*\*\*\*

\*\*\*\*\*\*\*

SELECT ENAME,REPLICATE('\*',LEN(SAL)) AS SAL FROM EMP

=>

ACCOUNTS

ACCNO BAL

1234567896 10000

your a/c no XXXX7896 debited ---- ?

REPLICATE('X',4) + RIGHT(ACCNO,4)

REPLACE() :-

-------------------

=> used to replace one string with another string

REPLACE(str1,str2,str3)

ex :-

SELECT REPLACE('HELLO','ELL','ABC') => HABCO

SELECT REPLACE('HELLO','L','ABC') => HEABCABCO

SELECT REPLACE('HELLO','ELO','ABC') => HELLO

SELECT REPLACE('@@HE@@LLO@@','@','') => HELLO

STUFF() :-

--------------

=> used to replace one string with another string based on start and length

STUFF(string1,start,no of chars,string2)

ex :-

SELECT STUFF('HELLO WELCOME',10,4,'GO') => HELLO WELGO

=>

ACCOUNTS

ACCNO BAL

1234567896 10000

your a/c no XXXX7896 debited ---- ?

STUFF(ACCNO,1,6,'XXXX')

TRANSLATE() :-

---------------------

=> used to translate one char to another char

TRANSLATE(str1,str2,str3)

EX :-

SELECT TRANSLATE('HELLO','ELO','ABC') => HABBC

E => A

L => B

O => C

SELECT TRANSLATE('HELLO','ELO','') => ERROR

=> translate function can be used to encrypt data i.e. converting plain text to cipher text

display ENAME SAL ?

SELECT ENAME,

TRANSLATE(SAL,'0123456789.','$tB\*q&@#^%!') AS SAL

FROM EMP

JONES 2975.00 B%#&!$$

=> remove all special chars from '@#HE$%LL^&O\*!' ?

output :- HELLO

11-mar-24

DATE functions :-

-----------------------

GETDATE() :-

-------------------

=> returns current date,time and also milliseconds

SELECT GETDATE() => 2024-03-11 07:39:34.180

--------------- ------------ -----

DATE TIME MS

DATEPART() :-

--------------------

=> returns part of the date

DATEPART(INTERVAL , DATE)

EX :-

SELECT DATEPART(YY,GETDATE()) => 2024

MM => 03

DD => 11 (1-31)

DW => 2 (1-7)

DY => 71 (1-366)

HH => hour part

MI => minutes

SS => seconds

QQ => quarter (1-4)

jan-mar 1

apr-jun 2

jul-sep 3

oct-dec 4

=> employees joined in 1980,1983,1985 ?

SELECT \* FROM EMP WHERE DATEPART(YY,HIREDATE) IN (1980,1983,1985)

=> employees joined in leap year ?

SELECT \* FROM EMP WHERE DATEPART(YY,HIREDATE) % 4 = 0

=> employees joined in jan,apr,dec months ?

SELECT \* FROM EMP WHERE DATEPART(MM,HIREDATE) IN (1,4,12)

=> employees joined in 2nd quarter of 1981 year ?

SELECT \* FROM EMP WHERE DATEPART(YY,HIREDATE) = 1981

AND

DATEPART(QQ,HIREDATE) = 2

DATENAME() :-

---------------------

=> similar to datepart used to get part of the date

DATENAME(INTERVAL,DATE)

MM DW

DATEPART 3 2

DATENAME March Monday

=> display ENAME DAY OF THE WEEK ?

SELECT ENAME , DATENAME(DW,HIREDATE) AS DAY

FROM EMP

=> display employee details joined on saturday,sunday ?

SELECT \*

FROM EMP

WHERE DATENAME(DW,HIREDATE) IN ('SATURDAY','SUNDAY')

=> write a query to display on which day india got independence ?

SELECT DATENAME(DW,'1947-08-15') => Friday

YEAR() :- returns year part

------------

SELECT YEAR(GETDATE()) => 2024

MONTH() :- returns month part

-------------

SELECT MONTH(GETDATE()) => 3

DAY() :- returns day of the month

---------

SELECT DAY(GETDATE()) => 11

DATEDIFF() :-

------------------

=> returns difference between two dates in given interval

DATEDIFF(INTERVAL,START DATE,END DATE)

Ex :-

SELECT DATEDIFF(YY,'2023-03-11',GETDATE()) => 1

SELECT DATEDIFF(MM,'2023-03-11,GETDATE()) => 12

SELECT DATEDIFF(DD,'2023-03-11,GETDATE()) => 366.

SELECT DATEDIFF(WW,'2023-03-11',GETDATE()) => 53

=> display ENAME EXPERIENCE in years ?

SELECT ENAME,

DATEDIFF(YY,HIREDATE,GETDATE()) AS EXPR

FROM EMP

=> display ENAME EXPERIENCE ?

M YEARS N MONTHS

experience = 40 months = 3 YEARS 4 MONTHS

years = months / 12 = 40/12 = 3

months = months % 12 = 40%12 = 4

SELECT ENAME,

DATEDIFF(MM,HIREDATE,GETDATE())/12 AS YEARS,

DATEDIFF(MM,HIREDATE,GETDATE())%12 AS MONTHS

FROM EMP

1980-12-17 2023-12-17 43

2024-01-17 1

2024-02-17 2

2024-03-17 3

FORMAT() :-

-----------------

=> used to display dates in different formats

FORMAT(DATE , 'FORMAT')

Ex :-

SELECT FORMAT(GETDATE(),'dd-MM-yyyy') => 11-03-2024

SELECT FORMAT(GETDATE(),'MM/dd/yyyy') => 03/11/12024

12-MAR-24

scenario :-

STEP 1 :-

INSERT INTO EMP(EMPNO,ENAME,SAL,HIREDATE)

VALUES(888,'ABC',4000,GETDATE())

STEP 2 :-

=> list of employees joined today ?

SELECT \*

FROM EMP

WHERE HIREDATE = GETDATE() => NO ROWS

2024-03-12 = 2024-03-12 7:45:20.123

"=" comparision with getdate() always fails , to overcome this problem use format function

SELECT \*

FROM EMP

WHERE HIREDATE = FORMAT(GETDATE() ,'yyyy-MM-dd')

2024-03-12 = 2024-03-12

DATEADD() :-

------------------

=> used to add/subtract years,days,months to/from a date

DATEADD(INTERVAL,INT,DATE)

Ex :-

SELECT DATEADD(DD,10,GETDATE()) => 2024-03-22

SELECT DATEADD(MM,1,GETDATE()) => 2024-04-12

=> display ENAME HIREDATE DOR ?

DOR = HIREDATE + 40 YEARS

SELECT ENAME,HIREDATE,

DATEADD(YY,40,HIREDATE) AS DOR

FROM EMP

scenario :-

--------------

GOLD\_RATES

DATEID RATE

2020-01-01 ?

2020-01-02 ?

2024-03-11 ?

=> display today's gold rate ?

SELECT \* FROM GOLD\_RATES WHERE DATEID = FORMAT(GETDATE(),'yyyy-MM-dd')

=> display yesterday's gold rate ?

SELECT \*

FROM GOLD\_RATES

WHERE DATEID = FORMAT(DATEADD(DD,-1,GETDATE()),'yyyy-MM-dd')

=> last month same day gold rate ?

SELECT \*

FROM GOLD\_RATES

WHERE DATEID = FORMAT(DATEADD(MM,-1,GETDATE()),'yyyy-MM-dd')

=> last year same day gold rate ?

SELECT \*

FROM GOLD\_RATES

WHERE DATEID = FORMAT(DATEADD(YY,-1,GETDATE()),'yyyy-MM-dd')

=> list of employees joined in last 5 years ?

SELECT \*

FROM EMP

WHERE HIREDATE >= FORMAT(DATEADD(YY,-5,GETDATE()),'yyyy-MM-dd')

EOMONTH():-

-------------------

=> returns last day of the month

EOMONTH(DATE,INT)

ex :-

SELECT EOMONTH(GETDATE(),0) => 2024-03-31

SELECT EOMONTH(GETDATE(),1) => 2024-04-30

SELECT EOMONTH(GETDATE(),-1) => 2024-02-29

=> display next month first day ?

=> display current month last day ?

=> display next year first day ?

=> display current year first day ?

=> display current year last day ?

Numeric functions :-

---------------------------

ABS() :-

----------

=> returns absolute value (positive value)

SELECT ABS(-10) => 10

POWER() :-

---------------

=> calculates power

SELECT POWER(3,2) => 9

SQRT() :-

-----------

=> retuns square root of a given number

SELECT SQRT(16) => 4

SQUARE() :-

-----------------

=> returns square of a number

SELECT SQUARE(5) => 25

Rounding numbers :-

----------------------------

ROUND

CEILING

FLOOR

ROUND() :-

----------------

=> rounds number to integer or to decimal places based on avg

ROUND(number,decimal places)

ex :-

SELECT ROUND(38.4567,0) => 38

38------------------------------38.5--------------------------------39

number >= avg => rounded to highest

number < avg => rounded to lowest

SELECT ROUND(38.5567,0) => 39

SELECT ROUND(38.4567,2) => 38.46

SELECT ROUND(38.4567,1) => 38.5

SELECT ROUND(386,-2) => 400

300-----------------------------350---------------------------400

SELECT ROUND(386,-1) => 390

380---------------------------385-------------------------------390

SELECT ROUND(386,-3) => 0

0----------------------------------500----------------------------------1000

SELECT ROUND(4567,-1),ROUND(4567,-2),ROUND(4567,-3)

o/p :- 4570 4600 5000

=> round sal to hundreds in table ?

UPDATE EMP SET SAL = ROUND(SAL,-2)

15-mar-24

CEILING() :-

----------------

=> rounds number always to highest

CEILING(number)

ex :-

SELECT CEILING(3.1) => 4

3----------------------------------4

SELECT CEILING(3.0) => 3

FLOOR() :-

---------------

=> rounds number always to lowest

FLOOR(number)

SELECT FLOOR(3.9) => 3

Conversion functions :-

------------------------------

=> these functions are used to convert one type to another type

1 CAST

2 CONVERT

CAST :-

---------

CAST(EXPR AS TARGET-TYPE)

Ex :-

1 SELECT CAST(10.5 AS INT)

=> display smith earns 800

allen earns 1600 ?

+ => string concatenation

1 SELECT ename + ' earns ' + sal FROM emp => error => for concatenation all must be

char type

2 SELECT ename + ' earns ' + CAST(sal AS VARCHAR) FROM emp

=> display smith joined on 1980-12-17 as clerk ?

SELECT ename + ' joined on ' + CAST(hiredate AS VARCHAR) + ' as ' + job

FROM emp

CONVERT() :-

-------------------

CONVERT(TARGET-TYPE , VALUE)

EX :-

SELECT CONVERT(INT,10.5) => 10

=> difference between CAST & CONVERT ?

1 using convert function we can display dates in different formats but not possible

by using cast function

2 using convert function we can display money in different styles but not possible

by using cast function

Displyaing dates in different formats :-

---------------------------------------------------

=> To display dates in different formats first convert date to char type

CONVERT(VARCHAR,DATE,STYLE-NUMBER)

ex :-

SELECT CONVERT(VARCHAR,GETDATE(),101) => 03/15/2024

105 => 15-03-2-2024

110 => 03-15-2024

114 => 07:17:12:443

=> display ENAME HIREDATE ? display hiredate in DD-MM-YYYY format ?

SELECT ENAME,CONVERT(VARCHAR,HIREDATE,105) AS HIREDATE FROM EMP

Displaying money in different styles :-

-------------------------------------------------

CONVERT(VARCHAR,MONEY,STYLE-NUMBER)

0 => displays two decimal places

1 => displays number with thousand seperator

2 => displays four decimal places

ex :-

=> display ENAME SAL ? display salaries with thousand seperator ?

SELECT ENAME,CONVERT(VARCHAR,SAL,1) AS SAL FROM EMP

=> SELECT CONVERT(VARCHAR,CAST(5000 AS MONEY),1) => 5,000.00

Special functions :-

-------------------------

ISNULL() :-

---------------

=> used to convert null values

ISNULL(arg1,arg2)

=> if arg1 = null returns arg2

if arg1 <> null returns arg1 only

1 SELECT ISNULL(100,200) => 100

2 SELECT ISNULL(NULL,200) => 200

=>display ENAME SAL COMM TOTSAL ?

TOTSAL = SAL + COMM

SELECT ENAME ,SAL,COMM , SAL+COMM AS TOTSAL FROM EMP

smith 800 null null

allen 1600 300 1900

SELECT ENAME,SAL,COMM,SAL+ISNULL(COMM,0) AS TOTSAL FROM EMP

smith 800 null 800

allen 1600 300 1900

=> display ENAME SAL COMM ?

if comm = NULL display NO COMM

SELECT ENAME ,SAL,ISNULL(CAST(COMM AS VARCHAR),'NO COMM') AS COMM

FROM EMP

16-mar-24

binary\_checksum :-

--------------------------

=> returns checksum value for given string

binary\_checksum(string)

ex :-

1 SELECT binary\_checksum('naresh') , binary\_checksum('NARESH')

o/p :- 109462104 77881464

=> In sql server string comparision is not case sensitive i.e. upper case & lower case strings

are same , to perform case sensitive comparision use binary\_checksum function.

display employee list whose name = SMITH ?

SELECT \*

FROM EMP

WHERE BINARY\_CHECKSUM(ENAME)= BINARY\_CHECKSUM('SMITH')

Analytical Functions :-

-----------------------------

RANK & DENSE\_RANK :-

------------------------------------

=> both functions are used to find ranks

=> ranking is based on some column

=> for ranking data must be sorted

RANK() OVER (ORDER BY COL ASC/DESC ,---------)

DENSE\_RANK() OVER (ORDER BY COL ASC/DESC,---------)

Ex :-

=> display ranks of the employees based on sal and highest paid employee should get 1st rank ?

SELECT ENAME,SAL,

RANK() OVER (ORDER BY SAL DESC) AS RNK

FROM EMP

SELECT ENAME,SAL,

DENSE\_RANK() OVER (ORDER BY SAL DESC) AS RNK

FROM EMP

difference between rank & dense\_rank ?

1 rank function generates gaps but dense\_rank will not generate gaps.

2 in rank function ranks are not in sequence but in dense\_rank ranks

are always in sequence.

SAL RNK DRNK

5000 1 1

4000 2 2

3000 3 3

3000 3 3

3000 3 3

2000 6 4

2000 6 4

1000 8 5

=> find ranks of the employees based on sal , if salaries are same then

ranking should be based on hiredate ?

SELECT ENAME,HIREDATE,SAL,

DENSE\_RANK() OVER (ORDER BY SAL DESC,HIREDATE ASC) AS RNK

FROM EMP

=> display ranks of the students based on avg ,m ,p ?

STUDENT

SNO SNAME M P C

1 A 80 90 70

2 B 50 70 60

3 C 90 80 70

4 D 90 70 80

SELECT \* , (M+P+C)/3 AS AVG,

DENSE\_RANK() OVER (ORDER BY (M+P+C)/3 DESC,M DESC,P DESC) AS RNK

FROM STUDENT

3 C 90 80 70 80 1

4 D 90 70 80 80 2

1 A 80 90 70 80 3

2 B 50 70 60 60 4

PARTITION BY clause :-

---------------------------------

=> partition by clause is used to find ranks with in group , for ex to find ranks with in dept

first we need to divide the table dept wise and apply dense\_rank function on each

dept.

SELECT DEPTNO,ENAME,SAL,

DENSE\_RANK() OVER (PARTITION BY DEPTNO ORDER BY SAL DESC) AS RNK

FROM EMP

18-MAR-24

ROW\_NUMBER() :-

---------------------------

=> returns record number

=> it is also based on some column

=> data must be sorted

row\_number() over (order by colname asc/desc)

ex :- row number based on empno ?

SELECT EMPNO,ENAME,SAL,

ROW\_NUMBER() OVER (ORDER BY EMPNO ASC) AS RNO

FROM EMP

ex :- row number with in dept based on empno

ROW\_NUMBER() OVER (PARTITION BYDEPTNO ORDER BY EMPNO ASC)

Aggregate functions :-

-----------------------------

=> These functions process multiple rows and returns one value.

=> Aggregate functions are also called multirow functions or group functions.

MAX

MIN

SUM

AVG

COUNT

COUNT(\*)

MAX() :-

----------

=> returns maximum value

MAX(arg)

SELECT MAX(SAL) FROM EMP => 5000

SELECT MAX(HIREDATE) FROM EMP => 1983-01-12

SELECT MAX(ENAME) FROM EMP => ward

MIN() :-

-----------

=> returns minimum value

MIN(arg)

SELECT MIN(SAL) FROM EMP => 800

SELECT MIN(HIREDATE) FROM EMP => 1980-12-17

SELECT MIN(ENAME) FROM EMP => adams

SUM() :-

-------------

=> returns total

SUM(arg)

SELECT SUM(SAL) FROM EMP => 29300.00

=> round total sal to thousands ?

SELECT ROUND(SUM(SAL),-3) FROM EMP => 29000

29000-------------------------29500-----------------------------------30000

=> after rounding display total sal with thousand seperator ?

SELECT CONVERT(VARCHAR,ROUND(SUM(SAL),-3) ,1) FROM EMP => 29,000

=> display total sal including comm ?

SELECT SUM(SAL+COMM) FROM EMP => 7900 => because some

employees comm is null

SELECT SUM(SAL+ISNULL(COMM,0)) FROM EMP => 31500

=> display total sal paid to managers ?

SELECT SUM(SAL)

FROM EMP

WHERE JOB='MANAGER'

AVG() :-

---------

=> returns average value

AVG(arg)

Ex :-

SELECT AVG(SAL) FROM EMP => 2092.8571

=> round avg(sal) to highest integer ?

SELECT CEILING(AVG(SAL)) FROM EMP => 2093.00

NOTE :-

=> SUM,AVG functions cannot be applied on char,date columns and can be applied

only on numeric columns

COUNT() :-

---------------

=> returs no of values present in a column

COUNT(arg)

SELECT COUNT(EMPNO) FROM EMP => 14

SELECT COUNT(COMM) FROM EMP => 4 => nulls are not counted

COUNT(\*) :-

------------------

=> returns no of rows in a table

COUNT(\*)

SELECT COUNT(\*) FROM EMP => 14

19-mar-24

difference between count & count(\*) ?

count function doesn't include nulls but count(\*) includes nulls

Ex :-

T1

F1

10

NULL

20

NULL

30

SELECT COUNT(F1) FROM T1 => 3

SELECT COUNT(\*) FROM T1 => 5

=> no of employees joined in 1981 year ?

SELECT COUNT(\*) FROM EMP WHERE HIREDATE LIKE '1981%'

=> no of employees joined on sunday ?

SELECT COUNT(\*) FROM EMP WHERE DATEPART(DW,HIREDATE) = 1

=> no of employees joined in 2nd quarter of 1981 ?

SELECT COUNT(\*)

FROM EMP

WHERE DATEPART(YY,HIREDATE) = 1981

AND

DATEPART(QQ,HIREDATE) = 2

NOTE :-

1 aggregate functions are not allowed in where clause and they are allowed only in

select,having clauses

SELECT ENAME

FROM EMP

WHERE SAL = MAX(SAL) => ERROR

SELECT DEPTNO

FROM EMP

WHERE COUNT(\*) > 3 => ERROR

to overcome this problem use sub-query

2 SELECT COL1,COL2 FROM TABNAME

no of values return by col1 = no of values return by col2

SELECT ENAME,MAX(SAL) FROM EMP => ERROR

14 1

SELECT ENAME,ROUND(SAL,-2) AS SAL FROM EMP

14 14

SELECT MIN(SAL),MAX(SAL) FROM EMP

1 1

SELECT UPPER(ENAME) , ROUND(SAL,-2) FROM EMP

14 14

string :- upper,lower,len,left,right,substring,charindex,replicate,replace,stuff,translate

numeric :- abs,power,sqrt,square,round,ceiling,floor

date :- getdate,datepart,datename,year,month,day,dateadd,datediff,eomonth,format

conversion :- cast,convert

special :- isnull

analytical :- rank,dense\_rank,row\_number

aggregate :- max,min,sum,avg,count,count(\*)

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

20-mar-24

GROUP BY clause :-

---------------------------

=> GROUP BY clause is used to group rows based on one or more columns

to calculate min,max,sum,avg,count for each group. For ex to find

dept wise no of employees first group the rows based on dept

and apply count(\*) function each group.

EMP

EMPNO ENAME SAL DEPTNO

1 A 5000 10 10 9000

2 B 4000 20 GROUP BY 20 10000

3 C 3000 30 =====================> 30 3000

4 D 6000 20

5 E 4000 10

detailed data summarized data

=> GROUP BY clause converts detailed data into summarized data which is useful for analysis.

syntax :-

SELECT columns

FROM tabname

[WHERE cond]

GROUP BY col1,col2,-------

[HAVING cond]

[ORDER BY col ASC/DESC,------]

execution :-

---------------

FROM

WHERE

GROUP BY

HAVING

SELECT

ORDER BY

Examples :-

=> display dept wise total sal ?

SELECT DEPTNO,SUM(SAL) AS TOTSAL

FROM EMP

GROUP BY DEPTNO

FROM EMP :-

------------------

EMP

EMPNO ENAME SAL DEPTNO

1 A 5000 10

2 B 4000 20

3 C 3000 30

4 D 6000 20

5 E 4000 10

GROUP BY DEPTNO :-

--------------------------------

10 1 A 5000

5 E 4000

20 2 B 4000

4 D 6000

30 3 C 3000

SELECT DEPTNO,SUM(SAL) AS TOTSAL :-

---------------------------------------------------------------

10 9000

20 10000

30 3000

=> display job wise summary (min sal,max sal,sum sal,avg sal,count) ?

SELECT JOB,MIN(SAL) AS MINSAL,

MAX(SAL) AS MAXSAL,

SUM(SAL) AS TOTSAL,

AVG(SAL) AS AVGSAL,

COUNT(\*) AS CNT

FROM EMP

GROUP BY JOB

=> display year wise no of employees joined ?

SELECT YEAR(HIREDATE) AS YEAR,COUNT(\*) AS CNT

FROM EMP

GROUP BY YEAR(HIREDATE)

=> display day wise no of employees joined ?

SELECT DATENAME(DW,HIREDATE) AS DAY,COUNT(\*) AS CNT

FROM EMP

GROUP BY DATENAME(DW,HIREDATE)

=> display no of employees joined in each month in the year 1981 ?

SELECT DATENAME(MM,HIREDATE) AS MONTH,COUNT(\*) AS CNT

FROM EMP

WHERE YEAR(HIREDATE)=1981

GROUP BY DATENAME(MM,HIREDATE)

=> display depts having more than 3 employees ?

SELECT DEPTNO,COUNT(\*)

FROM EMP

WHERE COUNT(\*) > 3

GROUP BY DEPTNO => ERROR

sql server can't calculate dept wise count before group by and it can calculate only

after group by , so apply the condition COUNT(\*) > 3 after group by using

having clause.

SELECT DEPTNO,COUNT(\*)

FROM EMP

GROUP BY DEPTNO

HAVING COUNT(\*) > 3 ;

WHERE VS HAVING :-

-------------------------------

WHERE HAVING

1 selects specific rows selects specific groups

2 conditions applied conditions applied after group by

before group by

3 where cond can be having cond cannot be

applied without group by applied without group by

4 use where clause use having clause

if cond not based if cond based on

on aggregate function aggregate function

=> display dep wise total sal where deptno = 10,20 and total sal > 10000 ?

SELECT DEPTNO,SUM(SAL) AS TOTSAL

FROM EMP

WHERE DEPTNO IN (10,20)

GROUP BY DEPTNO

HAVING SUM(SAL) > 10000

O/P :- 20 10900

21-mar-24

=> list of southern states having more than 5cr population ?

PERSONS

AADHARNO NAME GENDER AGE ADDR CITY

SELECT STATE,COUNT(\*)

FROM PERSONS

WHERE STATE IN ( 'AP','TS','KA','TN','KL')

GROUP BY STATE

HAVING COUNT(\*) > 50000000

Grouping based on multiple columns :-

-----------------------------------------------------

=> display dept wise and with in dept job wise total sal ?

SELECT DEPTNO,JOB,SUM(SAL)

FROM EMP

GROUP BY DEPTNO , JOB

ORDER BY DEPTNO ASC

10 CLERK 1300

MANAGER 2450

PRESIDENT 5000

20 ANALYST 6000

CLERK 1900

MANAGER 2975

30 CLERK 950

MANAGER 2850

SALESMAN 5600

=> display state wise and with in state gender wise population ?

AP MALE ?

FEMALE ?

SELECT STATE,GENDER,COUNT(\*)

FROM PERSONS

GROUP BY STATE,GENDER

ORDER BY STATE

=> display year wise and with in year quarter wise no of employees joined ?

1981 1 ?

2 ?

3 ?

4 ?

SELECT DATEPART(YY,HIREDATE) AS YEAR,

DATEPART(QQ,HIREDATE) AS QRT,

COUNT(\*) AS CNT

FROM EMP

GROUP BY DATEPART(YY,HIREDATE) , DATEPART(QQ,HIREDATE)

ORDER BY YEAR ASC

NOTE :-

----------

=> column alias are not allowed in group by clause because group by clause is executed

before select.

=> column alias can be used in order by clause because order by clause is executed

after select.

SELECT YEAR(HIREDATE) AS YEAR,COUNT(\*) AS CNT

FROM EMP

GROUP BY YEAR => INVALID

ORDER BY YEAR ASC => VALID

ORDER BY 1 ASC => sorting based on 1st column in the select

=> display duplicate records ?

EMP44

ENO ENAME SAL

1 A 5000

2 B 6000

3 C 7000

1 A 5000

2 B 6000

SELECT ENO,ENAME,SAL

FROM EMP44

GROUP BY ENO,ENAME,SAL

HAVING COUNT(\*) > 1

22-mar-24

ROLLUP & CUBE :-

----------------------------

=> both functions are used to display subtotals and grand total

GROUP BY ROLLUP(COL1,COL2,---)

GROUP BY CUBE(COL1,COL2,----)

ROLLUP :-

--------------

=> ROLLUP displays subtotals for each group and also displays grand total.

SELECT DEPTNO,JOB,SUM(SAL)

FROM EMP

GROUP BY ROLLUP(DEPTNO,JOB)

ORDER BY DEPTNO ASC

NULL NULLL 29300 => GRAND TOTAL

10 CLERK 1300

MANAGER 2450

PRESIDENT 5000

NULL 8800 => SUBTOTAL

20 ANALYST 6000

CLERK 1900

MANAGER 2975

NULL 10900 => SUBTOTAL

30 CLERK 950

MANAGER 2850

SALESMAN 5600

NULL 9600 => SUBTOTAL

CUBE :-

-----------

=> display subtotal for each goup by column (deptno , job) and also displays grand total

SELECT DEPTNO,JOB,SUM(SAL) AS TOTSAL

FROM EMP

GROUP BY CUBE(DEPTNO,JOB)

ORDER BY DEPTNO ASC , JOB ASC

NULL NULL 29300.00 => grand total

NULL ANALYST 6000.00 => job subtotal

NULL CLERK 4200.00 => job subtotal

NULL MANAGER 8400.00

NULL PRESIDENT 5000.00

NULL SALESMAN 5700.00

10 NULL 8800.00 => dept subtotal

10 CLERK 1300.00

10 MANAGER 2500.00

10 PRESIDENT 5000.00

20 NULL 10900.00 => dept subtotal

20 ANALYST 6000.00

20 CLERK 1900.00

20 MANAGER 3000.00

30 NULL 9600.00 => dept subtotal

30 CLERK 1000.00

30 MANAGER 2900.00

30 SALESMAN 5700.00

Question :-

=> dislay state wise and with in state gender wise population and display state wise

and gender wise subtotals (use persons table)

=> display year wise and with in year quarter wise total amount and display year wise subtotals ?

SALES

DATEID PRODID CUSTID QTY AMT

21-MAR-24 100 10 1 2000

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

CASE statement :-

------------------------

=> case stmt is used to implement if-then-else.

=> case stmt is similiar to switch case.

=> used to return values based on conditions.

=> case statements are 2 types

1 simple case => if conditions based on "=" operator

2 searched case => if conditions not based on "=" operator

simple case :- searched case :-

------------------ ----------------------

CASE EXPR CASE

WHEN VALUE1 THEN RETURN EXPR1 WHEN COND1 THEN RETURN EXPR1

WHEN VALUE2 THEN RETURN EXPR2 WHEN COND2 THEN RETURN EXPR2

WHEN VALUE3 THEN RETURN EXPR3 -----------------

-------------

ELSE RETURN EXPR ELSE RETURN EXPR

END END

=> dipslay EMPNO ENAME SAL DNAME ?

IF DEPTNO=10 DISPLAY ACCOUNTS

20 HR

30 SALES

OTHERS UNKNOWN

SELECT EMPNO,ENAME,SAL,

CASE DEPTNO

WHEN 10 THEN 'ACCOUNTS'

WHEN 20 THEN 'HR'

WHEN 30 THEN 'SALES'

ELSE 'UNKNOWN'

END AS DNAME

FROM EMP

=> increment employee salaries as follows ?

if job=CLERK incr sal by 10%

SALESMAN 15%

MANAGER 20%

others 5%

UPDATE EMP

SET SAL = CASE JOB

WHEN 'CLERK' THEN SAL+(SAL\*0.1)

WHEN 'SALESMAN' THEN SAL + (SAL\*0.15)

WHEN 'MANAGER' THEN SAL + (SAL\*0.2)

ELSE SAL + (SAL\*0.05)

END

=> display ENAME SAL SALRANGE ?

IF SAL > 3000 DISPLAY HISAL

SAL < 3000 DISPLAY LOSAL

SAL = 3000 DISPLAY AVGSAL

SELECT ENAME,SAL,

CASE

WHEN SAL>3000 THEN 'HISAL'

WHEN SAL<3000 THEN 'LOSAL'

ELSE 'AVGSAL'

END AS SALRANGE

FROM EMP

=> display SNO TOTAL AVG RESULT ?

STUDENT

SNO SNAME S1 S2 S3

1 A 80 90 70

2 B 30 60 50

SELECT SNO,

S1+S2+S3 AS TOTAL,

(S1+S2+S3)/3 AS AVG,

CASE

WHEN S1>=35 AND S2>=35 AND S3>=35 THEN 'PASS'

ELSE 'FAIL'

END AS RESULT

FROM STUDENT

=> display no of employees based on sal range ?

0-2000 ?

2001-4000 ?

>4000 ?

SELECT CASE

WHEN SAL BETWEEN 0 AND 2000 THEN '0-2000'

WHEN SAL BETWEEN 2001 AND 4000 THEN '2001-4000'

ELSE 'ABOVE 4000'

END AS SALRANGE ,COUNT(\*) AS CNT

FROM EMP

GROUP BY CASE

WHEN SAL BETWEEN 0 AND 2000 THEN '0-2000'

WHEN SAL BETWEEN 2001 AND 4000 THEN '2001-4000'

ELSE 'ABOVE 4000'

END

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

23-MAR-24 integrity constraints

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

=> Integrity Constraints are rules to maintain Data Quality or Data Consistency.

=> used to prevent users from entering invalid data.

=> used to enforce rules like min bal must be 1000.

Types of constraints :-

----------------------------

1 NOT NULL

2 UNIQUE

3 PRIMARY KEY

4 CHECK

5 FOREIGN KEY

6 DEFAULT

=> above constraints can be declared in two ways

1 column level

2 table level

column level :-

--------------------

=> if constraints are declared immediately after declaring column then it is called column level.

CREATE TABLE <TABNAME>

(

COLNAME DATATYPE(SIZE) CONSTRAINT ,

-------------------------------------------------,

---------------------------------------

)

NOT NULL :-

------------------

=> NOT NULL constraint doesn't accept null values.

=> a field declared with NOT NULL is called mandatory field.

ex :-

CREATE TABLE EMP11

(

EMPID INT ,

ENAME VARCHAR(10) NOT NULL

)

Testing :-

INSERT INTO EMP11 VALUES(100, NULL) => ERROR

INSERT INTO EMP11 VALUES(101, 'A')

UNIQUE constraint :-

------------------------------

=> unique constraint doesn't accept duplicates

ex :-

CREATE TABLE EMP12

(

EMPID INT ,

ENAME VARCHAR(10) NOT NULL,

EMAILID VARCHAR(20) UNIQUE

)

Testing :-

------------

INSERT INTO EMP12 VALUES(101,'B','abc@gmail.com')

INSERT INTO EMP12 VALUES(102,'C','abc@gmail.com') => ERROR

INSERT INTO EMP12 VALUES(103,'D',NULL)

INSERT INTO EMP12 VALUES(104,'E',NULL) => ERROR

NOTE :- UNIQUE constraint allows only one null

PRIMARY KEY :-

-----------------------

=> primary key doesn't accept duplicates and nulls.

=> it is the combination of unique & not null.

=> In tables one column must be there to uniquely identify records

and into that column duplicates and nulls are not allowed , so declare that

column with primary key.

ex :-

CREATE TABLE EMP13

(

EMPID INT PRIMARY KEY,

ENAME VARCHAR(10) NOT NULL

)

Testing :-

------------

INSERT INTO EMP13 VALUES(100,'A')

INSERT INTO EMP13 VALUES(100,'B') => ERROR

INSERT INTO EMP13 VALUES(NULL,'C') => ERROR

Note :-

---------

=> because it is not allowing duplicates and nulls into empid so using

empid we can uniquely identify the records.

=> only one primary key is allowed per table , if we want multiple primary keys then

declare one column with primary key and other columns with unique not null.

ex :-

CREATE TABLE CUST

(

CUSTID INT PRIMARY KEY,

NAME VARCHAR(10) NOT NULL,

AADHARNO NUMERIC(12) UNIQUE NOT NULL,

PANNO CHAR(10) UNIQUE NOT NULL

)

Difference between unqiue & primary key ?

unique primary key

1 allows one null doesn't allow nulls

2 multiple columns can only one column

be declared with unique can be declared with primary key

3 sql server creates sql server creates

non clustered index clustered index on primary key column

on unique column

CHECK :-

------------

=> use check constraint when rule based on condition

CHECK(condition)

ex 1 :- SAL must be min 3000

CREATE TABLE EMP33

(

EMPID INT PRIMARY KEY,

ENAME VARCHAR(10) NOT NULL,

SAL MONEY CHECK(SAL>=3000)

)

INSERT INTO EMP33 VALUES(100,'A',1000) => ERROR

INSERT INTO EMP33 VALUES(101,'B',5000)

INSERT INTO EMP33 VALUES(102,'C',NULL)

NOTE :- check constraint allows nulls

2 gender must be 'M','F' ?

GENDER CHAR(1) CHECK(GENDER IN ('M','F'))

M

F

K => INVALID

3 amt must be multiple of 100 ?

AMT MONEY CHECK(AMT%100=0)

5000 => VALID

5255 => INVALID

4 pwd must be min 8 chars

PWD VARCHAR(10) CHECK(LEN(PWD)>=8)

5 emailid must contain '@'

must end with '.com' or '.co' or '.in' ?

EMAILID VARCHAR(20) CHECK(EMAILID LIKE '%@%'

AND

(

EMAILID LIKE '%.com'

OR

EMAILID LIKE '%.co'

OR

EMAILID LIKE '%.in'

))

FOREIGN KEY :-

-----------------------

=> foreign key is used to establish relationship between two tables

EMP

EMPID ENAME SAL DNO DNAME LOC

1 A 5000 10 HR HYD

2 B 500 10 HR HYD

=> above table contains redundency that leads to memory wastage and insert,

update,delete anamolies (errors). To reduce redundency store data in seperate tables

and establish relationship by adding foreign key.

=> To add foreign key take primary key of one table and add it to another table as

foreign key and declare with references constraint.

ex :-

DEPT

DNO DNAME LOC

10 HR HYD

20 IT BLR

30 SALES MUM

EMP

EMPID ENAME SAL DNO REFERENCES DEPT(DNO)

1 A 5000 10

2 B 6000 20

3 C 7000 90 => ERROR

4 D 4000 10

5 E 3000 NULL

26-MAR-24

=> values entered in foreign key column should match with values entered in primary key.

=> after declaring foreign key a relationship is established between two tables

called parent/child relationship.

=> primary key table is parent and foreign key table is child.

=> foreign key allows duplicates and nulls.

CREATE TABLE DEPT55

(

DNO INT PRIMARY KEY,

DNAME VARCHAR(10) UNIQUE NOT NULL,

LOC VARCHAR(10)

)

INSERT INTO DEPT55 VALUES(10,'HR','HYD'),(20,'IT','BLR'),(30,'SALES','MUM')

CREATE TABLE EMP55

(

EMPID INT PRIMARY KEY,

ENAME VARCHAR(10) NOT NULL,

SAL MONEY CHECK(SAL>=3000),

DNO INT REFERENCES DEPT55(DNO)

)

INSERT INTO EMP55 VALUES(1,'A',5000,10)

INSERT INTO EMP55 VALUES(2,'B',4000,90) => ERROR

INSERT INTO EMP55 VALUES(3,'C',6000,10)

INSERT INTO EMP55 VALUES(4,'D',3000,NULL)

Question :-

ACCOUNTS

ACCNO ACTYPE BAL

Rules :-

1 accno should not be duplicate & null

2 actype must be 'S' OR 'C'

3 bal must be min 1000

TRANSACTIONS

TRID TTYPE TDATE TAMT ACCNO

Rules :-

1 trid should not be duplicate & null

2 ttype must be 'W' OR 'D'

3 tamt must be multiple of 100

4 accno should match with accounts accno

DEFAULT :-

--------------

=> a column can be declared with default value as follows

ex :- HIREDATE DATE DEFAULT GETDATE()

=> while inserting if we skip hiredate then sql server inserts default value

CREATE TABLE EMP66

(

EMPNO INT PRIMARY KEY,

ENAME VARCHAR(10) NOT NULL,

HIREDATE DATE DEFAULT GETDATE()

)

INSERT INTO EMP66(EMPNO,ENAME) VALUES(100,'A')

INSERT INTO EMP66 VALUES(101,'B','2024-01-01')

INSERT INTO EMP66 VALUES(102,'C',NULL)

SELECT \* FROM EMP66

EMPNO ENAME HIREDATE

100 A 2024-03-26

101 B 2024-01-01

102 C NULL

TABLE LEVEL :-

-----------------------

=> if constraints are declared after declaring all columns then it is called table level.

=> use table level to declare constraint for combination or multiple columns

CREATE TABLE <TABNAME>

(

COLNAME DATATYPE(SIZE) ,

COLNAME DATATYPE(SIZE),

------------------------------ ,

CONSTRAINT (COL1,COL2,---)

)

Declaring check constraint at table level :-

----------------------------------------------------------

PRODUCTS

prodid pname price mfd\_dt exp\_dt

100 A 50 2024-03-01 2024-01-01 => INVALID

RULE :- exp\_dt > mfd\_dt

create table products

(

prodid int primary key,

pname varchar(10) not null,

price smallmoney,

mfd\_dt date,

exp\_dt date ,

check(exp\_dt > mfd\_dt)

)

INSERT INTO products VALUES(100,'A',50,GETDATE(),'2024-01-01') => ERROR

INSERT INTO products VALUES(100,'A',50, '2024-01-01',GETDATE())

27-mar-24

composite primary key :-

---------------------------------

=> if combination of columns uniquely identifies the records then that combination should

be declared primary key.

=> if primary key is declared for combination of columns then it is called composite primary key.

=> In composite primary key combination should not be duplicate.

=> composite primary key declared at table level.

Example :-

STUDENT COURSE

SID SNAME CID CNAME

1 A 10 .NET

2 B 11 SQL SERVER

REGISTRATIONS

SID CID DOR FEE

1 10 ? 5000

1 11 ? 2000

2 10 ? 5000

=> In the above example sid,cid combination uniquely identifies records , so declare this

combination as primary key at table level.

CREATE TABLE STUDENT

(

SID INT PRIMARY KEY,

SNAME VARCHAR(10) NOT NULL

)

INSERT INTO STUDENT VALUES(1,'A'),(2,'B')

CREATE TABLE COURSE

(

CID INT PRIMARY KEY,

CNAME VARCHAR(10) NOT NULL

)

INSERT INTO COURSE VALUES(10,'.NET'),(11,'SQL')

CREATE TABLE REGISTRATIONS

(

SID INT REFERENCES STUDENT(SID) ,

CID INT REFERENCES COURSE(CID) ,

DOR DATE,

FEE MONEY ,

PRIMARY KEY(SID,CID)

)

INSERT INTO REGISTRATIONS VALUES(1,10,GETDATE(),5000)

INSERT INTO REGISTRATIONS VALUES(1,11,GETDATE(),5000)

INSERT INTO REGISTRATIONS VALUES(2,10,GETDATE(),5000)

INSERT INTO REGISTRATIONS VALUES(1,10,GETDATE(),5000) => INVALID

Example 2 :-

SALES

DATEID PRODID CUSTID QTY AMT

2024-03-26 100 10 1 2000

2024-03-26 100 11 1 2000

2024-03-26 101 10 1 1000

2024-03-27 100 10 1 2000

=> identify primary key and write create table script ?

CREATE TABLE SALES

(

DATEID DATE ,

PRODID INT,

CUSTID INT,

QTY INT,

AMT MONEY,

PRIMARY KEY(DATEID,PRODID,CUSTID)

)

composite foreign key :-

--------------------------------

=> if foreign key declared for combination of columns then it is called composite foreign key.

REGISTRATIONS

SID CID DOR FEE

1 10 ? 5000

1 11 ? 2000

2 10 ? 5000

CERTIFICATES

CERTNO DOI SID CID

1000 ?? 1 10

1001 ?? 1 11

1002 ?? 2 11

=> In the above example SID,CID combination should match with registrations table SID,CID

combination , so declare this combination as foreign key at table level.

CREATE TABLE CERTIFICATES

(

CERTNO INT PRIMARY KEY,

DOI DATE ,

SID INT ,

CID INT ,

FOREIGN KEY(SID,CID) REFERENCES REGISTRATIONS (SID,CID)

)

which of the following constraint cannot be declared at table level ?

A CHECK

B UNIQUE

C NOT NULL

D PRIMARY KEY

E FOREIGN KEY

ANS :- C

28-MAR-24

Adding constraints to existing table :-

---------------------------------------------------

=> "ALTER" command is used to add constraints to existing table.

CREATE TABLE EMP88

(

EMPNO INT,

ENAME VARCHAR(10) ,

SAL MONEY,

DNO INT

)

Adding check constraint :-

------------------------------------

=> add check constraint with cond sal >= 3000 ?

ALTER TABLE EMP88

ADD CHECK(SAL>=3000)

ALTER TABLE EMP

ADD CHECK(SAL>=3000) => ERROR => some employee salaries are less than 3000

NOTE :- while adding constraint sql server also validates existing data.

WITH NOCHECK :-

--------------------------

=> if check constraint added with " WITH NOCHECK " then sql server will not validate

existing data and it validates only new data.

ALTER TABLE EMP

WITH NOCHECK ADD CHECK(SAL>=3000)

Adding primary key :-

-----------------------------

=> primary key cannot be added to nullable column , to add primary key

1 change the column to not null

2 add primary key

=> add primary key to empno ?

step 1 :-

ALTER TABLE EMP88

ALTER COLUMN EMPNO INT NOT NULL

step 2 :-

ALTER TABLE EMP88

ADD PRIMARY KEY(EMPNO)

Adding foreign key :-

--------------------------

=> add foreign key to dno that should refer dept55 table primary key i.e. dno ?

ALTER TABLE EMP88

ADD FOREIGN KEY(DNO) REFERENCES DEPT55(DNO)

changing from NULL to NOT NULL :-

------------------------------------------------

=> modify the column ename to not null ?

ALTER TABLE EMP88

ALTER COLUMN ENAME VARCHAR(10) NOT NULL

Droping constraints :-

----------------------------

ALTER TABLE <TABNAME>

DROP CONSTRAINT <NAME>

Ex :-

=> drop check constraint in emp88 table ?

ALTER TABLE EMP88

DROP CONSTRAINT CK\_\_EMP88\_\_SAL\_\_02084FDA

=> drop primary key in dept55 table ?

ALTER TABLE DEPT55

DROP CONSTRAINT PK\_\_DEPT55\_\_C035B8C24DDDCF50 => ERROR

DROP TABLE DEPT55 => ERROR

TRUNCATE TABLE DEPT55 => ERROR

NOTE :-

=> primary key constraint cannot be dropped if referenced by some fk

=> primary key table cannot be dropped if referenced by some fk

=> primary key table cannot be truncated if referenced by some fk

DELETE rules :-

-----------------------

1 ON DELETE NO ACTION (DEFAULT)

2 ON DELETE CASCADE

3 ON DELETE SET NULL

4 ON DELETE SET DEFAULT

=> these rules are declared with foreign key

=> delete rules specifies how child rows are affected if parent row is deleted

ON DELETE NO ACTION :-

-------------------------------------

=> parent row cannot be deleted if associated with child rows.

CREATE TABLE DEPT99

(

DNO INT PRIMARY KEY,

DNAME VARCHAR(10)

)

INSERT INTO DEPT99 VALUES(10,'HR'),(20,'IT')

CREATE TABLE EMP99

(

EMPNO INT PRIMARY KEY,

ENAME VARCHAR(10) NOT NULL,

DNO INT REFERENCES DEPT99(DNO)

)

INSERT INTO EMP99 VALUES(1,'A',10),(2,'B',10)

DELETE FROM DEPT99 WHERE DNO=10 => ERROR

scenario :-

---------------

ACCOUNTS

ACCNO ACTYPE BAL

100 S 10000

101 S 20000

LOANS

ID TYPE AMT ACCNO

1 H 30 100

2 C 10 100

Rule :- account closing is not allowed if associated with loans

ON DELETE CASCADE :-

-----------------------------------

=> if parent row is deleted then it is deleted along with child rows

CREATE TABLE DEPT99

(

DNO INT PRIMARY KEY,

DNAME VARCHAR(10)

)

INSERT INTO DEPT99 VALUES(10,'HR'),(20,'IT')

CREATE TABLE EMP99

(

EMPNO INT PRIMARY KEY,

ENAME VARCHAR(10) NOT NULL,

DNO INT REFERENCES DEPT99(DNO) ON DELETE CASCADE

)

INSERT INTO EMP99 VALUES(1,'A',10),(2,'B',10)

DELETE FROM DEPT99 WHERE DNO=10 => 1 ROW DELETED

SELECT \* FROM EMP99 => NO ROWS

scenario :-

---------------

ACCOUNTS

ACCNO ACTYPE BAL

100 S 10000

101 S 20000

TRANSACTIONS

TRID TTYPE TDATE TAMT ACCNO

1 W 2000 100

2 D 5000 100

Rule :- if account is closed (deleted) along with account delete transactions also

29-mar-24

ON DELETE SET NULL :-

------------------------------------

=> if parent row is deleted but child rows are not deleted but foreign key will be set to null

CREATE TABLE DEPT99

(

DNO INT PRIMARY KEY,

DNAME VARCHAR(10)

)

INSERT INTO DEPT99 VALUES(10,'HR'),(20,'IT')

CREATE TABLE EMP99

(

EMPNO INT PRIMARY KEY,

ENAME VARCHAR(10) NOT NULL,

DNO INT REFERENCES DEPT99(DNO) ON DELETE SET NULL

)

INSERT INTO EMP99 VALUES(1,'A',10),(2,'B',10)

DELETE FROM DEPT99 WHERE DNO=10 => 1 row affected

SELECT \* FROM EMP99

EMPNO ENAME DNO

1 A NULL

2 B NULL

scenario :-

---------------

projects

projid pname duration cost client

100 A 5 800 TATA MOTORS

101 B 4 500 L&T

emp

empid ename sal projid

1 K 5000 100

2 Q 4000 101

Rule :- if project is completed (deleted) then set the employee projid to null

ON DELETE SET DEFAULT :-

--------------------------------------

=> if parent row is deleted but child rows are not deleted but foreign key will be set to default.

CREATE TABLE DEPT99

(

DNO INT PRIMARY KEY,

DNAME VARCHAR(10)

)

INSERT INTO DEPT99 VALUES(10,'HR'),(20,'IT')

CREATE TABLE EMP99

(

EMPNO INT PRIMARY KEY,

ENAME VARCHAR(10) NOT NULL,

DNO INT DEFAULT 20

REFERENCES DEPT99(DNO)

ON DELETE SET DEFAULT

)

INSERT INTO EMP99 VALUES(1,'A',10),(2,'B',10)

DELETE FROM DEPT99 WHERE DNO=10

SELECT \* FROM EMP99

EMPNO ENAME DNO

1 A 20

2 B 20

Summary :-

importance of constraints

types of constraints

declaring constraints

column level

table level

adding constraints

droping constraints

delete rules

Which statements are true regarding constraints ?

A a foreign key cannot contain NULL value F

B a column with UNIQUE constraint can contain NULL value T

C a constraint is enforced only for the INSERT operation on a table F

D all constraints can be defined at column level and table level. F

Which CREATE TABLE statement is valid?

A. CREATE TABLE ord\_details

(ord\_no INT PRIMARY KEY,

item\_no INT PRIMARY KEY,

ord\_date DATE NOT NULL);

B. CREATE TABLE ord\_details

(ord\_no INT UNIQUE , NOT NULL,

item\_no INT,

ord\_date DATE DEFAULT GETDATE() NOT NULL);

C. CREATE TABLE ord\_details

(ord\_no INT ,

item\_no INT,

ord\_date DATE DEFAULT NOT NULL,

UNIQUE (ord\_no),

PRIMARY KEY (ord\_no));

D. CREATE TABLE ord\_details

(ord\_no INT,

item\_no INT,

ord\_date DATE DEFAULT GETDATE() NOT NULL,

PRIMARY KEY (ord\_no, item\_no));

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

JOINS

----------

=> join is an opertion performed to display data from two or more tables.

=> to display data from two tables we need to join those tables.

=> In DB related data stored in multiple tables , to gather or to combine data

stored in multiple tables we need to join those tables.

Ex :-

ORDERS CUSTOMER

ORDID ORD\_DT DEL\_DT CID CID CNAME CADDR

1000 20- 30 10 10 A HYD

1001 21- 31 11 11 B HYD

1002 22- 01 12 12 C HYD

report :-

-------------

ORDID ORD\_DT DEL\_DT CNAME CADDR

1000 20- 30 A HYD

1001 21- 31 B HYD

Types of joins :-

---------------------

1 inner join or equi join

2 outer join

left join

right join

full join

3 non equi join

4 self join

5 cross join or cartesian join

30-MAR-24

Inner join :-

----------------

=> To perform inner join between the tables there must be a common field

=> Name of the common field need not to be same.

=> between the tables pk-fk relationship is not compulsory

=> Inner join is performed between the tables sharing common field with same datatype.

SELECT columns

FROM tab1 INNER JOIN tab2

ON join condition

join condition :-

--------------------

=> according to given join condition sql server joins the records of two tables.

table1.commonfield = table2.commonfield

=> this join is called equi join because here join condition based on "=" operator.

Ex :-

EMP DEPT

EMPNO ENAME SAL DEPTNO DEPTNO DNAME LOC

1 A 3000 10 10 ACCOUNTS NEW YORK

2 B 4000 20 20 RESEARCH

3 C 5000 30 30 SALES

4 D 3000 20 40 OPERATIONS

5 E 2000 NULL

=> display ENAME SAL DNAME LOC ?

------------------ ---------------------

EMP DEPT

SELECT ENAME,SAL,DNAME,LOC

FROM EMP INNER JOIN DEPT

ON EMP.DEPTNO = DEPT.DEPTNO

A 3000 ACCOUNTS NEW YORK

B 4000 RESEARCH ?

C 5000 SALES ?

D 3000 RESEARCH ?

NOTE :-

=> In join queries declare table alias and prefix column names with table alias for two reasons

1 to avoid ambiguity

2 for faster execution

SELECT E.ENAME,E.SAL,

D.DEPTNO,D.DNAME,D.LOC

FROM EMP AS E INNER JOIN DEPT AS D

ON E.DEPTNO = D.DEPTNO

=> display employee details with project details working at NEW YORK loc and earning more than 3000 ?

SELECT E.ENAME,D.DNAME,D.LOC

FROM EMP AS E INNER JOIN DEPT AS D

ON E.DEPTNO = D.DEPTNO /\* JOIN CONDITION \*/

WHERE D.LOC = 'NEW YORK' /\* FILTER CONDITION \*/

AND

E.SAL > 3000

=> no of employees working at NEW YORK loc ?

SELECT COUNT(\*)

FROM EMP AS E INNER JOIN DEPT AS D

ON E.DEPTNO = D.DEPTNO

WHERE D.LOC = 'NEW YORK'

=> display dept wise no of employees and display dept names ?

SELECT D.DNAME,COUNT(\*) AS CNT

FROM EMP AS E INNER JOIN DEPT AS D

ON E.DEPTNO = D.DEPTNO

GROUP BY D.DNAME

ON E.DEPTNO = D.DEPTNO :-

-------------------------------------------

EMP DEPT

EMPNO ENAME SAL DEPTNO DEPTNO DNAME LOC

1 A 3000 10 10 ACCOUNTS NEW YORK

2 B 4000 20 20 RESEARCH

3 C 5000 30 30 SALES

4 D 3000 20 40 OPERATIONS

5 E 2000 10

output :-

1 A 3000 ACCOUNTS

2 B 4000 RESEARCH

3 C 5000 SALES

4 D 3000 RESEARCH

5 E 2000 ACCOUNTS

GROUP BY D.DNAME :-

----------------------------------

ACCOUNTS 1 A 3000

5 E 2000

RESEARCH 2 B 4000

4 D 3000

SALES 3 C 5000

SELECT D.DNAME,COUNT(\*) AS CNT :-

---------------------------------------------------------

ACCOUNTS 2

RESEARCH 2

SALES 1

01-APR-24

joining more than 2 tables :-

--------------------------------------

=> if no of tables increases no of join conditions also increases

=> to join N tables N-1 join conditions required

SELECT columns

FROM tab1 INNER JOIN tab2

ON join condition

INNER JOIN tab3

ON join condition

INNER JOIN tab4

ON join condition ;

Example :-

EMP DEPT LOCATIONS COUNTRIES

empno deptno locid country\_id

ename dname city country\_name

sal locid state

deptno country\_id

=> display ENAME DNAME CITY STATE COUNTRY ?

---------- ----------- ---------------------- ------------------

EMP DEPT LOCATIONS COUNTRIES

SELECT E.ENAME,

D.DNAME,

L.CITY,L.STATE,

C.COUNTRY\_NAME

FROM EMP AS E INNE JOIN DEPT AS D

ON E.DEPTNO = D.DEPTNO

INNER JOIN LOCATIONS AS L

ON D.LOCID = L.LOCID

INNER JOIN COUNTRIES AS S

ON L.COUNTRY\_ID = C.COUNTRY\_ID

=> display ENAME COUNTRY\_NAME ?

=> display no of employees working in each country ?

=> display order details with cust details ?

=> display list of orders to be delivered today ?

=> display no of orders placed by each customer ?

OUTER JOIN :-

---------------------

=> Inner join returns only matching records but will not return unmatched records , To

display unmatched records perform outer join.

ex :-

EMP DEPT

EMPNO ENAME SAL DEPTNO DEPTNO DNAME LOC

1 A 3000 10 10 ACCOUNTS NEW YORK

2 B 4000 20 20 RESEARCH

3 C 5000 30 30 SALES

4 D 3000 20 40 OPERATIONS => unmatched row

5 E 2000 NULL => unmatched row

=> outer join is 3 types

1 left join

2 right join

3 full join

LEFT JOIN :-

-----------------

=> returns all rows ( matched + unmatched) from left side table and matching rows from

right side table.

SELECT E.ENAME,D.DNAME

FROM EMP AS E LEFT JOIN DEPT AS D

ON E.DEPTNO = D.DEPTNO

=> above query returns all rows from emp table and matching rows from dept table

A ACCOUNTS

B RESEARCH

C SALES

D RESEARCH

E NULL => unmatched row from emp

RIGHT JOIN :-

===========

=> returns all rows from right side table and matching rows from left side table.

SELECT E.ENAME,D.DNAME

FROM EMP AS E RIGHT JOIN DEPT AS D

ON E.DEPTNO = D.DEPTNO

=> above query returns unmatched records from right side table

A ACCOUNTS

B RESEARCH

C SALES

D RESEARCH

NULL OPERATIONS => unmatched from dept

02-APR-24

FULL JOIN :-

------------------

=> returns all rows from both tables

SELECT E.ENAME,D.DNAME

FROM EMP AS E FULL JOIN DEPT AS D

ON E.DEPTNO = D.DEPTNO

=> returns all rows from both emp & dept tables.

A ACCOUNTS

B RESEARCH

C SALES

D RESEARCH

E NULL => unmatched from emp

NULL OPERATIONS => unmatched from dept

Displaying only unmatched records :-

------------------------------------------------

left side table :-

----------------------

SELECT E.ENAME,D.DNAME

FROM EMP AS E LEFT JOIN DEPT AS D

ON E.DEPTNO = D.DEPTNO

WHERE D.DNAME IS NULL

E NULL

right side table :-

---------------------

SELECT E.ENAME,D.DNAME

FROM EMP AS E RIGHT JOIN DEPT AS D

ON E.DEPTNO = D.DEPTNO

WHERE E.ENAME IS NULL

NULL OPERATIONS

both tables :-

-----------------

SELECT E.ENAME,D.DNAME

FROM EMP AS E full JOIN DEPT AS D

ON E.DEPTNO = D.DEPTNO

WHERE D.DNAME IS NULL

OR

E.ENAME IS NULL

E NULL

NULL OPERATIONS

Questions :-

------------------

1

T1 T2

F1 C1

1 1

2 2

1 1

2 2

NULL NULL

NULL NULL

=> no of rows returned by the following operations ?

1 INNER JOIN => 8

2 LEFT JOIN => 10

3 RIGHT JOIN => 10

4 FULL JOIN => 12

2

EMP PROJECTS

EMPID ENAME SAL PROJID PROJID NAME DURATION

1 A 100 100

2 B 101 101

3 C NULL 102

=> display employee details with project details ?

=> display employee details with project details and also display employees not assigned to

any project ?

=> display only the projects where no employee assigned to it ?

NON EQUI JOIN :-

---------------------------

=> non equi join is performed between the tables not sharing a common field.

ex :-

EMP SALGRADE

EMPNO ENAME SAL GRADE LOSAL HISAL

1 A 5000 1 700 1000

2 B 4000 2 1001 2000

3 C 3000 3 2001 3000

4 D 2000 4 3001 4000

5 E 1500 5 4001 9999

=> This join is called non equi join because here join condition is not based on "="

operator and it is based on > < between operators.

=> display ENAME SAL GRADE ?

------------------ --------------

EMP SALGRADE

SELECT E.ENAME,E.SAL,S.GRADE

FROM EMP AS E JOIN SALGRADE AS S

ON E.SAL BETWEEN S.LOSAL AND S.HISAL

A 5000 5

B 4000 4

C 3000 3

D 2000 2

E 1500 2

=> display grade 4 employees list ?

SELECT E.ENAME,E.SAL,S.GRADE

FROM EMP AS E JOIN SALGRADE AS S

ON E.SAL BETWEEN S.LOSAL AND S.HISAL

WHERE S.GRADE = 4

Question 1 :-

=> display ENAME DNAME GRADE ?

----------- ---------- -----------

EMP DEPT SALGRADE

Question 2 :-

ORDERS PRODUCTS

ordid prodid qty prodid pname price

1000 100 3 100 A 2000

1000 101 2 101 B 1000

1000 102 1 102 C 3000

1001 100 2

DISCOUNTS

pct lamt hamt

5 1 1000

10 1001 5000

20 5001 99999

=> DISPLAY ORDID PNAME AMOUNT DIS AMOUNT\_AFTER\_DIS

SELECT O.ORDID,P.PNAME,O.QTY\*P.PRICE AS AMOUNT,D.PCT,

(O.QTY\*P.PRICE) - ((O.QTY\*P.PRICE)\*D.PCT/100) AS AMT\_AFT\_DIS

FROM ORDERS AS O INNER JOIN PRODUCTS AS P

ON O.PRODID = P.PRODID

JOIN DISCOUNTS AS D

ON (O.QTY\*P.PRICE) BETWEEN D.LAMT AND D.HAMT

03-APR-24

self join :-

------------

=> joining a table to itself is called self join

=> In self join a record in one table joined with another record of same table

=> to perform self join the same table must be declared two times with different alias

FROM EMP AS X JOIN EMP AS Y

EMP X EMP Y

EMPNO ENAME MGR EMPNO ENAME MGR

7369 smith 7902 7369 smith 7902

7499 allen 7698 7499 allen 7698

7698 blake 7839 7698 blake 7839

7839 king null 7839 king null

7902 ford 7566 7902 ford 7566

ex :-

=> display ENAME MGRNAME ?

SELECT X.ENAME,Y.ENAME AS MANAGER

FROM EMP AS X JOIN EMP AS Y

ON X.MGR = Y.EMPNO

smith ford

allen blake

blake king

=> display employee names reporting to blake ?

SELECT X.ENAME,Y.ENAME

FROM EMP AS X JOIN EMP AS Y

ON X.MGR = Y.EMPNO

WHERE Y.ENAME='blake'

=> blake's manager name ?

SELECT X.ENAME,Y.ENAME

FROM EMP AS X JOIN EMP AS Y

ON X.MGR = Y.EMPNO

WHERE X.ENAME='blake'

=> employees earning more than their manager ?

SELECT X.ENAME,Y.ENAME

FROM EMP AS X JOIN EMP AS Y

ON X.MGR = Y.EMPNO

WHERE X.SAL > Y.SAL

=> employees joined befrore their manager ?

SELECT X.ENAME,X.HIREDATE,

Y.ENAME AS MGR,Y.HIREDATE AS MGRHIRE

FROM EMP AS X JOIN EMP AS Y

ON X.MGR = Y.EMPNO

WHERE X.HIREDATE < Y.HIREDATE

=>

TEAMS

ID COUNTRY

1 IND

2 AUS

3 ENG

04-apr-24

write a query to display following output ?

IND VS AUS

IND VS ENG

AUS VS ENG

TEAMS A TEAMS B

ID COUNTRY ID COUNTRY

1 IND 1 IND

2 AUS 2 AUS

3 ENG 3 ENG

SELECT A.COUNTRY + ' VS ' + B.COUNTRY

FROM TEAMS AS A JOIN TEAMS AS B

ON A.ID < B.ID

=> display ENAME DNAME GRADE MNAME ?

SELECT E.ENAME,D.DNAME,S.GRADE,M.ENAME AS MGR

FROM EMP AS E INNER JOIN DEPT AS D

ON E.DEPTNO = D.DEPTNO

JOIN SALGRADE AS S

ON E.SAL BETWEEN S.LOSAL AND S.HISAL

JOIN EMP AS M

ON E.MGR = M.EMPNO

CROSS JOIN / CARTESIAN JOIN :-

-------------------------------------------------

=> cross join returns cross product or cartesian product of two tables

A=1,2

B=3,4

AXB = (1,3) (1,4) (2,3) (2,4)

=> In cross join sql server joins all the records of 1st table with all the records of 2nd table.

=> To perform cross join submit the join query without join condition.

Example :-

SELECT E.ENAME,D.DNAME

FROM EMP AS E CROSS JOIN DEPT AS D

=> above query join all records of emp table with all records of dept table

=> above query returns 60 rows (15 rows in emp & 4 rows in dept)

GROUP BY & JOIN :-

------------------------------

SELECT columns

FROM tab1 JOIN tab2

ON condition

GROUP BY colname

=> display dept wise no of employees but display dept names ?

SELECT D.DNAME,COUNT(\*) AS CNT

FROM EMP AS E INNER JOIN DEPT AS D

ON E.DEPTNO = D.DEPTNO

GROUP BY D.DNAME

=> display grade wise no of employees ?

=> display no of employees reporting to each manager ?

=>

SALES

DATEID PRODID CUSTID QTY AMT

2024-04-03 100 10 1 2000

PRODUCTS

PRODID PNAME PRICE CATEGORY BRAND

100 ABC 2000 ELECTRONICS SAMSUNG

CUSTOMERS

CUSTID NAME ADDR COUNTRY

10 K HYD IND

=> display year wise total amount ?

=> display category wise total amount ?

=> display country wise total amount ?

=> display year wise ,country wise,category wise total amount ?

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

SET OPERATORS :-

--------------------------

UNION

UNION ALL

INTERSECT

EXCEPT

A = 1,2,3,4

B = 1,2,5,6

A UNION B = 1,2,3,4,5,6

A UNION ALL B = 1,2,3,4,1,2,5,6

A INTERSECT B = 1,2

A EXCEPT B = 3,4

B EXCEPT A = 5,6

=> in sql server set operations performed between records return by two queries

SELECT STATEMENT 1

UNION / UNION ALL / INTERSECT / EXCEPT

SELECT STATEMENT 2

Rules :-

-----------

1 no of columns return by both queries must be same

2 corresponding columns datatype must be same

UNION :-

------------

=> combines rows return by two queries

=> duplicates are eliminated

=> result is sorted

SELECT JOB FROM EMP WHERE DEPTNO = 20

CLERK

MANAGER

ANALYST

CLERK

ANALYST

SELECT JOB FROM EMP WHERE DEPTNO = 30

SALESMAN

SALESMAN

SALESMAN

MANAGER

SALESMAN

CLERK

Ex 1 :-

SELECT JOB FROM EMP WHERE DEPTNO = 20

UNION

SELECT JOB FROM EMP WHERE DEPTNO = 30

ANALYST

CLERK

MANAGER

SALESMAN

Ex 2 :-

SELECT JOB,SAL FROM EMP WHERE DEPTNO = 20

UNION

SELECT JOB,SAL FROM EMP WHERE DEPTNO = 30

ANALYST 3150.00

CLERK 1100.00

CLERK 1210.00

CLERK 3000.00

MANAGER 3480.00

MANAGER 3600.00

SALESMAN 1495.00

SALESMAN 1725.00

SALESMAN 1840.00

=> difference between union & join ?

UNION JOIN

1 combines rows combines columns

2 horizontal merge vertical merge

3 performed between performed between two tables

two queries

4 performed between two performed between two

similar structures dissimilar structures

ex :-

T1 T2

F1 C1

1 10

2 20

3 30

UNION :- JOIN :-

------------- ----------

1 1 10

2 2 20

3 3 30

10

20

30

scenario :-

-------------

EMP\_US

ENO ENAME SAL DNO

DEPT

EMP\_IND DNO DNAME LOC

ENO ENAME SAL DNO

=> total employee list ?

SELECT \* FROM EMP\_US

UNION

SELECT \* FROM EMP\_IND

=> employees working at US loc with dept details ?

SELECT E.\*,D.\*

FROM EMP\_US AS E INNER JOIN DEPT AS D

ON E.DNO = D.DNO

=> total employees with dept details ?

SELECT E.\*,D.\*

FROM EMP\_US AS E INNER JOIN DEPT AS D

ON E.DNO = D.DNO

UNION

SELECT E.\*,D.\*

FROM EMP\_IND AS E INNER JOIN DEPT AS D

ON E.DNO = D.DNO

UNION ALL :-

--------------------

=> combines rows return by two queries

=> duplicates are not eliminated

=> result is not sorted

Ex :-

SELECT JOB FROM EMP WHERE DEPTNO = 20

UNION ALL

SELECT JOB FROM EMP WHERE DEPTNO = 30

CLERK

MANAGER

ANALYST

CLERK

ANALYST

SALESMAN

SALESMAN

SALESMAN

MANAGER

SALESMAN

CLERK

=> difference between UNION & UNION ALL ?

UNION UNION ALL

1 eliminates duplicates duplicates are not eliminated

2 sorts result result is not sorted

2 slower faster

INTERSECT :-

-------------------

=> returns common values from the output of two select statements

SELECT JOB FROM EMP WHERE DEPTNO = 20

INTERSECT

SELECT JOB FROM EMP WHERE DEPTNO = 30

CLERK

MANAGER

EXCEPT :-

--------------

=> returns values present in 1st query output and not present in 2nd query output

SELECT JOB FROM EMP WHERE DEPTNO = 20

EXCEPT

SELECT JOB FROM EMP WHERE DEPTNO = 30

ANALYST

SELECT JOB FROM EMP WHERE DEPTNO = 30

EXCEPT

SELECT JOB FROM EMP WHERE DEPTNO = 20

SALESMAN

Question :-

========

T1 T2

F1 C1

1 1

2 2

3 3

10 40

20 50

30 60

=>write outputs for the following operations ?

1 inner join

2 left join

3 right join

4 full join

5 union

6 union all

7 intersect

8 except

06-apr-24

SUB-QUERIES / NESTED QUERIES :-

----------------------------------------------------

=> a query in another query is called sub-query or nested query

=> one query is called inner/child/sub query

=> other query is called outer/parent/main query

=> first sql server executes inner query and then it executes outer query

=> output of inner query is input to outer query

=> use sub-queries when where cond based on unknown value

Types of sub-queries :-

--------------------------------

1 single row sub-queries

2 multi row sub-queries

3 co-related sub-queries

4 derived tables / CTEs

5 scalar sub-queries

single row sub-queries :-

-----------------------------------

=> if sub-query returns one value then it is called single row sub-query

syntax :-

------------

SELECT columns

FROM tabname

WHERE colname OP (SELECT STATEMENT)

=> OP must be any relational operator like > >= < <= = <>

Examples :-

=> display employees earning more than blake ?

SELECT \*

FROM EMP

WHERE SAL > (SELECT sal FROM emp WHERE ename='blake' ) ;

=> display employees who are senior to king ?

SELECT \*

FROM EMP

WHERE HIREDATE < (SELECT HIREDATE FROM EMP WHERE ENAME='KING')

=> name of the employee earning max sal ?

SELECT ENAME

FROM EMP

WHERE SAL = MAX(SAL) => ERROR => aggregates are not allowed in where clause and they are

allowed only in select,having clauses

SELECT ENAME

FROM EMP

WHERE SAL = (SELECT MAX(SAL) FROM EMP)

-----------------------------------------------

5000

=> name of the employee having max experience ?

SELECT ENAME

FROM EMP

WHERE HIREDATE = (SELECT MIN(HIREDATE) FROM EMP)

----------------------------------------------------

1980-12-17

=> names of the employees having min,max experience ?

SELECT ENAME

FROM EMP

WHERE HIREDATE = (SELECT MAX(HIREDATE) FROM EMP)

OR

HIREDATE = (SELECT MIN(HIREDATE) FROM EMP)

SELECT ENAME,HIREDATE

FROM EMP

WHERE HIREDATE = (SELECT MAX(HIREDATE) FROM EMP)

UNION

SELECT ENAME,HIREDATE

FROM EMP

WHERE HIREDATE = (SELECT MIN(HIREDATE) FROM EMP)

=> display 2nd max salary ?

SELECT MAX(SAL)

FROM EMP

WHERE SAL <> (SELECT MAX(SAL) FROM EMP)

=> name of the employee earning 2nd max sal ?

SELECT ENAME

FROM EMP

WHERE SAL = ( SELECT MAX(SAL)

FROM EMP

WHERE SAL <> (SELECT MAX(SAL) FROM EMP))

=> names of the employees working at NEW YORK loc ?

using sub-query :-

--------------------------

SELECT ENAME

FROM EMP

WHERE DEPTNO = ( SELECT DEPTNO FROM DEPT WHERE LOC='NEW YORK')

using join :-

-----------------

SELECT E.ENAME

FROM EMP AS E INNER JOIN DEPT AS D

ON E.DEPTNO = D.DEPTNO

WHERE D.LOC ='NEW YORK'

=> display ENAME DNAME working at NEW YORK loc ?

JOIN :-

---------

SELECT E.ENAME,D.DNAME

FROM EMP AS E INNER JOIN DEPT AS D

ON E.DEPTNO = D.DEPTNO

WHERE D.LOC ='NEW YORK'

SUB-QUERY :-

----------------------

not possible

=> difference between sub-query and join ?

1 to display data from one table and condition based on another table then we can use join or sub-query

2 to display data from two tables then compulsory use join

7-apr-24

=> delete the employee having max experience ?

DELETE

FROM EMP

WHERE HIREDATE = (SELECT MIN(HIREDATE) FROM EMP)

=> increment sal by 10% employee having max experience ?

UPDATE EMP

SET SAL = SAL + (SAL\*0.1)

WHERE HIREDATE = (SELECT MIN(HIREDATE) FROM EMP)

Multi row sub-queries :-

-----------------------------

=> if sub-query returns more than one value then it is called multi-row sub-query

SELECT columns

FROM tabname

WHERE colname OP (SELECT STATEMENT)

=> OP must be IN ,NOT IN,ANY,ALL

SINGLE MULTI

= IN

<> NOT IN

> >ANY >ALL

< <ANY <ALL

=> employees working at NEW YORK,CHICAGO locations ?

SELECT ENAME

FROM EMP

WHERE DEPTNO IN (SELECT DEPTNO

FROM DEPT

WHERE LOC IN ('NEW YORK','CHICAGO'))

ANY,ALL operators ;-

----------------------------

=> use ANY,ALL operators for > < comparision with multiple values.

WHERE X > ANY (1000,2000,3000) WHERE X < ANY(1000,2000,3000)

IF X = 800 FALSE IF X = 800 TRUE

1500 TRUE 1500 TRUE

4500 TRUE 4500 FALSE

WHERE X > ALL(1000,2000,3000) WHERE X < ALL(1000,2000,3000)

IF X=800 FALSE IF X=800 TRUE

1500 FALSE 1500 FALSE

4500 TRUE 4500 FALSE

=> employees earning more than all managers ?

SELECT \*

FROM EMP

WHERE SAL > ALL (SELECT SAL FROM EMP WHERE JOB='MANAGER')

=> employees earning more than atleast one manager ?

SELECT \*

FROM EMP

WHERE SAL > ANY (SELECT SAL FROM EMP WHERE JOB='MANAGER')

CO-RELATED SUB-QUERIES :-

------------------------------------------

=> if inner query references values of outer query then it is called co-related sub-query.

=> execution starts from outer query and inner query is executed no of times depends on

no of rows return by outer query.

=> use co-related sub-query to execute sub-query for each row return by outer query.

Example 1 :-

EMP

EMPNO ENAME SAL DEPTNO

1 A 5000 10

2 B 3000 20

3 C 4000 30

4 D 6000 20

5 E 3000 10

=> employees earning more than avg(sal) of the organization ? (non co-related )

SELECT \*

FROM EMP

WHERE SAL > (SELECT AVG(SAL) FROM EMP)

=> employees earning more than avg sal of their dept ?

SELECT \*

FROM EMP AS X

WHERE SAL > (SELECT AVG(SAL) FROM EMP WHERE DEPTNO = X.DEPTNO)

EMP

EMPNO ENAME SAL DEPTNO

1 A 5000 10 5000 > (4000) TRUE

2 B 3000 20 3000 > (4500) FALSE

3 C 4000 30 4000 > (4000) FALSE

4 D 6000 20 6000 > (4500) TRUE

5 E 3000 10 3000 > (4000) FALSE

08-apr-24

=> display employees earning max sal in their dept ?

SELECT \*

FROM EMP AS X

WHERE SAL = (SELECT MAX(SAL) FROM EMP WHERE DEPTNO = X.DEPTNO)

EMP

EMPNO ENAME SAL DEPTNO

1 A 5000 10 5000 = (5000) TRUE

2 B 3000 20 3000 = (6000) FALSE

3 C 4000 30 4000 = (4000) TRUE

4 D 6000 20 6000 = (6000) TRUE

5 E 3000 10 3000 = (5000) FALSE

=> Display top 3 max salaries ?

SELECT DISTINCT A.SAL

FROM EMP AS A

WHERE 3 > (SELECT COUNT(DISTINCT B.SAL)

FROM EMP AS B

WHERE A.SAL < B.SAL)

ORDER BY SAL DESC

EMP A EMP B

SAL SAL

5000 5000 3 > (0) TRUE

1000 1000 3 > (4) FALSE

2000 2000 3 > (3) FALSE

3000 3000 3 > (2) TRUE

4000 4000 3 > (1) TRUE

=> display 5th max salary ?

SELECT DISTINCT A.SAL

FROM EMP AS A

WHERE (5-1) = (SELECT COUNT(DISTINCT B.SAL)

FROM EMP AS B

WHERE A.SAL < B.SAL)

ORDER BY SAL DESC

DERIVED TABLES :-

-----------------------------

=> sub-queries in FROM clause are called derived tables

SELECT columns

FROM (SELECT STATEMENT) AS <ALIAS>

WHERE COND

=> sub-query output acts like a table for outer query.

=> derived tables are used in following scenarios

1 to control order of execution of clauses

2 to use result of one operation in another operation

3 to join two query outputs

controlling order of execution of clauses :-

------------------------------------------------------

default order :-

--------------------

FROM

WHERE

GROUP BY

HAVING

SELECT

ORDER BY

=> use derived tables to control this order of execution

Example 1 :-

=> display ranks of the employees based on sal and highest paid should get 1st rank ?

SELECT EMPNO,ENAME,SAL,

DENSE\_RANK() OVER (ORDER BY SAL DESC) AS RNK

FROM EMP

=> above query returns ranks of all the employees but to display top 5 employees

SELECT EMPNO,ENAME,SAL,

DENSE\_RANK() OVER (ORDER BY SAL DESC) AS RNK

FROM EMP

WHERE RNK <=5 => ERROR

above query returns error because column alias cannot be used in where clause

because where clause is executed before select ,to overcome this problem use

derived table

SELECT \*

FROM (SELECT EMPNO,ENAME,SAL,

DENSE\_RANK() OVER (ORDER BY SAL DESC) AS RNK

FROM EMP) AS E

WHERE RNK <= 5

=> display top 5 max salaries ?

SELECT DISTINCT SAL

FROM (SELECT SAL,

DENSE\_RANK() OVER (ORDER BY SAL DESC) AS RNK

FROM EMP) AS E

WHERE RNK <= 5

ORDER BY SAL DESC

=> display 5th max salary

WHERE RNK = 5

10-apr-24

=> display first 5 rows from emp table ?

STEP 1 :- generate row number according to table

SELECT EMPNO,ENAME,SAL,

ROW\_NUMBER() OVER (ORDER BY EMPNO ASC) AS RNO

FROM EMP

STEP 3 :- select records with row number upto 5

SELECT \*

FROM ( SELECT EMPNO,ENAME,SAL,

ROW\_NUMBER() OVER (ORDER BY EMPNO ASC) AS RNO

FROM EMP) AS E

WHERE RNO<=5

=> display 5th,7th,10th records ?

WHERE RNO IN (5,7,10)

=> display 5th to 10th records ?

WHERE RNO BETWEEN 5 AND 10

=> display even no rows ?

WHERE RNO%2=0

=> display last 3 rows ?

SELECT \*

FROM ( SELECT EMPNO,ENAME,SAL,

ROW\_NUMBER() OVER (ORDER BY EMPNO ASC) AS RNO

FROM EMP) AS E

WHERE RNO <= (SELECT COUNT(\*)-2 FROM EMP)

=> delete first 5 rows from emp ?

DELETE

FROM ( SELECT EMPNO,ENAME,SAL,

ROW\_NUMBER() OVER (ORDER BY EMPNO ASC) AS RNO

FROM EMP) AS E

WHERE RNO <= 5 => ERROR

NOTE :-

=> In derived tables outer query cannot be DML and it must be always SELECT.

To overcome this problem use CTEs.

CTE :-

--------

=> CTE stands for common table expression.

=> using CTE we can give name to the query output and we can use that name

in another queries like SELECT/INSERT/UPDATE/DELETE.

=> using CTE we can simplify complex operations.

syn :-

--------

WITH <CTE-NAME>

AS

(SELECT STATEMENT)

SELECT / INSERT / UPDATE / DELETE

Example 1 :- delete first 5 rows from emp table ?

WITH E

AS

(SELECT EMPNO,ENAME,SAL,

ROW\_NUMBER() OVER (ORDER BY EMPNO ASC) AS RNO

FROM EMP)

DELETE FROM E WHERE RNO <= 5

Example 2 :- delete duplicate records ?

EMP44

ENO ENAME SAL

1 A 5000

2 B 6000

3 C 7000

1 A 5000 => duplicate row

2 B 6000 => duplicate row

STEP 1 :- group the records with same ENO,ENAME,SAL and

with in group generate row\_numbers

SELECT ENO,ENAME,SAL,

ROW\_NUMBER() OVER (PARTITION BY ENO,ENAME,SAL

ORDER BY ENO ASC) AS RNO

FROM EMP44

1 A 5000 1

1 A 5000 2

2 B 6000 1

2 B 6000 2

3 C 7000 1

STEP 2 :- delete the records with rno > 1

WITH E

AS

(

SELECT ENO,ENAME,SAL,

ROW\_NUMBER() OVER (PARTITION BY ENO,ENAME,SAL

ORDER BY ENO ASC) AS RNO

FROM EMP44

)

DELETE FROM E WHERE RNO > 1

12-apr-24

SCALAR SUB-QUERIES :-

-------------------------------------

=> sub-queries in SELECT clause are called scalar-subqueries

SELECT (subquery1),(subquery2),------------

FROM tabname

WHERE cond

=> subquery output acts like a column for outer query.

=> use scalar subquery to show the query output in seperate column.

Example 1 :-

SELECT (SELECT COUNT(\*) FROM EMP) AS EMP,

(SELECT COUNT(\*) FROM DEPT) AS DEPT

EMP DEPT

9 4

Example 2 :-

=> display dept wise total salary ?

SELECT DEPTNO,SUM(SAL) AS DEPT\_TOTSAL

FROM EMP

GROUP BY DEPTNO

10 8750.00

20 10875.00

30 9400.00

=> display DEPTNO DEPT\_TOTSAL TOTSAL ?

SELECT DEPTNO,SUM(SAL) AS DEPT\_TOTSAL,

(SELECT SUM(SAL) FROM EMP) AS TOTSAL

FROM EMP

GROUP BY DEPTNO

10 8750.00 29025.00

20 10875.00 29025.00

30 9400.00 29025.00

=> display DEPTNO DEPT\_TOTSAL TOTSAL PCT ?

PCT = (DEPT\_TOTSAL/TOTSAL)\*100

SELECT DEPTNO,SUM(SAL) AS DEPT\_TOTSAL,

(SELECT SUM(SAL) FROM EMP) AS TOTSAL,

(SUM(SAL) / (SELECT SUM(SAL) FROM EMP))\*100 AS PCT

FROM EMP

GROUP BY DEPTNO

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

PIVOT operator :-

------------------------

=> operator used to convert rows into columns

=> used to display data in matrix form

SELECT \*

FROM (SELECT required data) AS <ALIAS>

PIVOT

(

AGGR-EXPR FOR COLNAME IN (V1,V2,V3,---)

) AS <NAME>

ORDER BY COLNAME ASC/DESC

Example 1 :-

10 20 30

analyst ? ? ?

clerk ? ? ?

manager ? ? ?

salesman ? ? ?

SELECT \*

FROM (SELECT DEPTNO,JOB,SAL FROM EMP) AS E

PIVOT

(

SUM(SAL) FOR DEPTNO IN ([10],[20],[30])

) AS PIVOT\_TBL

ORDER BY JOB ASC

Example 2 :-

1 2 3 4

1980 ? ? ? ?

1981 ? ? ? ?

1982 ? ? ? ?

1983 ? ? ? ?

SELECT \*

FROM (SELECT DATEPART(YY,HIREDATE) AS YEAR,

DATEPART(QQ,HIREDATE) AS QRT,

EMPNO

FROM EMP) AS E

PIVOT

(

COUNT(EMPNO) FOR QRT IN ([1],[2],[3],[4])

) AS PIVOT\_TBL

ORDER BY YEAR ASC

Example 3 :-

STUDENT

SNO SNAME SUBJECT MARKS

1 A MAT 80

1 A PHY 60

1 A CHE 70

2 B MAT 60

2 B PHY 50

2 B CHE 70

OUTPUT :-

SNO SNAME MAT PHY CHE

1 A 80 60 70

2 B 60 50 70

Writing queries ?

simple select

where

order by

distinct

top

functions

group by

joins

set opertaors

sub-queries

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

13-apr-24

creating new table from existing table :- (creating replica)

----------------------------------------------------

SELECT columns INTO <NEW-TABNAME>

FROM <OLD-TABNAME>

[WHERE COND]

Example 1 :- copying complete table

SELECT \* INTO EMP10

FROM EMP

=> structure and data of emp table is copied to emp10

Example 2 :- copying specific rows and cols

SELECT EMPNO,ENAME,JOB,SAL INTO EMP11

FROM EMP

WHERE JOB IN ('clerk','manager')

Example 3 :- copy only structure (cols) but not data (rows)

SELECT \* INTO EMP12

FROM EMP

WHERE 1=2

Example 4 :- copy table from one db to another db

copy emp table from db730 to temp ?

SELECT \* INTO TEMP.DBO.EMP

FROM EMP

copying data from one table to another table :-

--------------------------------------------------------------

INSERT INTO <TARGET-TABLE>

SELECT COLUMNS FROM <SOURCE-TABLE> [WHERE COND]

Example 1 :- copy data from emp to emp12

INSERT INTO EMP12

SELECT \* FROM EMP

MERGE command :-

----------------------------

=> merge command used to merge data into a table.

=> it is the combination of insert , update and delete.

=> used to manage replicas.

=> using merge command we can apply changes made to source table to target table (replica).

syn :-

MERGE INTO <TARGET-TABLE> AS <ALIAS>

USING <SOURCE-TABLE> AS <ALIAS>

ON (CONDITION)

WHEN MATCHED THEN

UPDATE

WHEN NOT MATCHED THEN

INSERT

WHEN NOT MATCHED BY SOURCE THEN

DELETE ;

Example :-

STEP 1 :- create source table

CREATE TABLE CUSTS

(

CID INT ,

CNAME VARCHAR(10),

CADDR VARCHAR(10)

)

INSERT INTO CUSTS VALUES(1,'A','HYD'),(2,'B','BLR')

STEP 2 :- create target table (replica)

SELECT \* INTO CUSTT FROM CUSTS

STEP 3 :- modify the source table

INSERT INTO CUSTS VALUES(3,'C','DEL')

UPDATE CUSTS SET CADDR = 'MUM' WHERE CID=1

STEP 4 :- apply changes made to custs to custt

MERGE INTO CUSTT AS T

USING CUSTS AS S

ON (S.CID = T.CID)

WHEN MATCHED THEN

UPDATE SET T.CADDR = S.CADDR

WHEN NOT MATCHED THEN

INSERT VALUES(S.CID,S.CNAME,S.CADDR) ;

custs

cid cname caddr

1 A MUM => updated

2 B BLR

3 C DEL => inserted

custt

cid cname caddr

1 A HYD => MUM

2 B BLR => BLR

3 C DEL => inserted

15-apr-24

Database Transactions :-

---------------------------------

=> a transaction is a unit of work that contains one or more dmls and it must be saved

as a whole or must be cancelled as a whole.

example :- money transfer

acct1------------------------1000-------------------------------acct2

update1 update2

(bal=bal-1000) (bal=bal+1000)

successful failed INVALID

failed successful INVALID

successful successful VALID

failed failed VALID

=> every transaction must gurantee a property called atomocity i.e. all or none , if

transaction contains multiple operations , if all are successful then it must be saved,

if one of the operation fails then entire transaction must be cancelled.

=> the following commands provided by sql server to handle transactions called TCL commands

COMMIT => to save txn

ROLLBACK => to cancel txn

SAVE TRANSACTION => to cancel part of the txn

=> every txn has a begin point and an end point

=> in sql server a txn begins implicitly with DML commands and ends implicitly with COMMIT.

=> a user can also start txn by executing "BEGIN TRANSACTION" command and

ends with COMMIT / ROLLBACK command.

Example 1 :-

create table a(a int) /\* implicitly committed \*/

begin transaction /\* txn begins t1 \*/

insert into a values(10)

insert into a values(20)

insert into a values(30)

insert into a values(40)

commit /\* txn ends \*/

=> if txn ends with commit then it is called successful txn and operations are saved.

Example 2 :-

create table a(a int) /\* implicitly committed \*/

begin transaction /\* txn begins t1 \*/

insert into a values(10)

insert into a values(20)

insert into a values(30)

insert into a values(40)

rollback /\* txn ends \*/

=> if txn ends with rollback then it is called aborted txn and operations are cancelled.

Example 3 :-

create table a(a int) /\* implicitly committed \*/

begin transaction /\* txn begins t1 \*/

insert into a values(10)

insert into a values(20)

commit /\* txn ends \*/

insert into a values(30) /\* implicitly committed \*/

insert into a values(40) /\* implicitly committed \*/

rollback /\* trying to end without starting => error \*/

Example 4 :-

create table a(a int) /\* implicitly committed \*/

begin transaction /\* txn begins t1 \*/

insert into a values(10)

insert into a values(20)

commit /\* txn ends \*/

begin transaction /\* txn begins t2 \*/

insert into a values(30)

insert into a values(40)

rollback /\* txn ends \*/

16-apr-24

Database security :-

--------------------------

1 LOGINS => provides security at server level

2 USERS => provides security at db level

3 PRIVILEGES => provides security at table level

4 VIEWS => provides security at row & col level

creating logins :-

---------------------

DB Objects :-

-----------------

TABLES

VIEWS

SYNONYMS

SEQUENCES

INDEXES

VIEWS :-

------------

=> a view is a subset of a table i.e. part of the table.

=> a view is a virtual table because it doesn't store data and doesn't occupy memory

and it always derives data from base table.

=> a view is a representation of a query

=> views are created

1 to provide security

2 to reduce complexity

=> with the help of views we can grant specific rows and columns to users.

=> views are 2 types

1 simple views

2 complex views

simple views :-

-------------------

=> a view said to be simple if based on single table

CREATE VIEW <NAME>

AS

SELECT STATEMENT

Ex :-

CREATE VIEW V1

AS

SELECT EMPNO,ENAME,JOB,DEPTNO FROM EMP

=> sql server creates view V1 and stores query but not query output (data)

SELECT \* FROM V1

=> when above query submitted to sql server it executes the query as follows

SELECT \* FROM (SELECT EMPNO,ENAME,JOB,DEPTNO FROM EMP)

Granting permissions on view to user :-

-----------------------------------------------------

DBO :-

---------

GRANT SELECT,INSERT,UPDATE,DELETE ON V1 TO VIJAY

VIJAY :-

-----------

1 SELECT \* FROM V1

2 UPDATE V1 SET JOB='MANAGER' WHERE EMPNO = 7369

3 UPDATE V1 SET SAL = 3000 WHERE EMPNO = 7369 => ERROR

4 INSERT INTO V1 VALUES(100,'ABC','CLERK',30)

ROW LEVEL SECURITY :-

-------------------------------------

CREATE VIEW V2

AS

SELECT EMPNO,ENAME,JOB,DEPTNO

FROM EMP

WHERE DEPTNO = 20

GRANT SELECT,INSERT,UPDATE,DELETE ON V2 TO VIJAY

VIJAY :-

----------

1 SELECT \* FROM V2

2 INSERT INTO V2 VALUES(222,'KLM','CLERK',30)

above insert command executed successfully even though it is violating where cond

17-apr-24

WITH CHECK option :-

-------------------------------

=> if view created with " WITH CHECK " option then any DML command through view

violates where cond that DML is not accepted

CREATE VIEW V3

AS

SELECT EMPNO,ENAME,JOB,DEPTNO

FROM EMP

WHERE DEPTNO = 20

WITH CHECK OPTION

GRANT SELECT,INSERT,UPDATE,DELETE ON V3 TO VIJAY

VIJAY :-

-----------

INSERT INTO V3 VALUES(888,'KKK','CLERK',30) => ERROR

complex views :-

----------------------

=> a view said to be complex view

1 if based on multiple tables

2 if query contains group by clause

distinct clause

aggregate functions

set operators

sub-queries

=> with the help of views complex queries can be converted into simple queries

Ex 1 :-

CREATE VIEW CV1

AS

SELECT E.EMPNO,E.ENAME,E.SAL,

D.DEPTNO,D.DNAME,D.LOC

FROM EMP AS E INNER JOIN DEPT AS D

ON E.DEPTNO = D.DEPTNO

=> after creating view whenever we want employee details with dept details execute following

simple query.

SELECT \* FROM CV1

Ex 2 :-

CREATE VIEW CV2

AS

SELECT D.DNAME , MIN(E.SAL) AS MINSAL,

MAX(E.SAL) AS MAXSAL,

SUM(E.SAL) AS TOTSAL,

COUNT(\*) AS CNT

FROM EMP AS E INNER JOIN DEPT AS D

ON E.DEPTNO = D.DEPTNO

GROUP BY D.DNAME

=> after creating view , if we want dept wise summary then execute the following query

SELECT \* FROM CV2

difference between simple and complex views ?

simple complex

1 based on single table based on multiple tables

2 query performs simple operations query performs complex operations

3 always updatable (allows dmls) not updatable (doesn't allow dmls)

=> list of views ?

SELECT \* FROM INFORMATION\_SCHEMA.VIEWS

Droping views :-

---------------------

DROP VIEW V1

if we drop table what about views created on table ?

ans :- views are not dropped but views cannot be queried

SCHEMABINDING :-

-----------------------------

=> if view created with schemabinding then sql server will not allow user to drop tables

if view exists on the table.

Rules :-

1 "\*" is not allowed in schemabinding

2 table name should prefix with schema name.

CREATE VIEW V10

WITH SCHEMABINDING

AS

SELECT DEPTNO,DNAME,LOC FROM DBO.DEPT

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

SYNONYMS :-

-------------------

=> a synonym is another name or alternative name for a table or view

=> if tablename is lengthy then we can give a simple and short name to the table

called synonym , instead of using tablename we can use synonym name

in SELECT / INSERT / UPDATE / DELETE queries.

syn :- CREATE SYNONYM <NAME> FOR <TABNAME>

ex :-

CREATE SYNONYM E FOR EMP

=> after creating synonym instead of using table name use synonym name in

SELECT/INSERT/UPDATE/DELETE queries.

1 SELECT \* FROM E

2 UPDATE E SET COMM = 500 WHERE EMPNO = 7369

Question :-

---------------

1 CREATE SYNONYM E FOR EMP

2 SELECT \* FROM EMP AS E

3 SP\_RENAME 'EMP','E'

=> difference between synonym and alias ?

synonym alias

1 permanent not permanent

2 stored in db not stored in db

3 scope of the synonym scope of the alias is

is upto the schema upto the query

=> list of synonyms ?

SELECT name,base\_object\_name FROM SYS.synonyms

18-APR-24

SEQUENCES :-

---------------------

=> sequence is also a db object created to generate sequence numbers.

=> sequences are created to auto increment column values.

syn :-

CREATE SEQUENCE <NAME>

[START WITH <VALUE>]

[INCREMENT BY <VALUE>]

[MAXVALUE <VALUE>]

[MINVALUE <VALUE>]

[CYCLE/NOCYCLE]

Example 1 :-

CREATE SEQUENCE S1

START WITH 1

INCREMENT BY 1

MAXVALUE 5

using sequence :-

------------------------

CREATE TABLE STUDENT

(

SID INT ,

SNAME VARCHAR(10)

)

use above sequence to generate values for sid ?

INSERT INTO STUDENT VALUES(NEXT VALUE FOR S1 ,'A')

INSERT INTO STUDENT VALUES(NEXT VALUE FOR S1 ,'B')

INSERT INTO STUDENT VALUES(NEXT VALUE FOR S1 ,'C')

INSERT INTO STUDENT VALUES(NEXT VALUE FOR S1 ,'D')

INSERT INTO STUDENT VALUES(NEXT VALUE FOR S1 ,'E')

INSERT INTO STUDENT VALUES(NEXT VALUE FOR S1 ,'F') => ERROR

SELECT \* FROM STUDENT

SID SNAME

1 A

2 B

3 C

4 D

5 E

Example 2 :- calling sequence in update command

CREATE SEQUENCE S2

START WITH 100

INCREMENT BY 1

MAXVALUE 999

use above sequence to generate empno in emp table ?

UPDATE EMP SET EMPNO = NEXT VALUE FOR S2

How to restart sequence ?

1 manually

2 cycle option

restarting sequence manually :-

------------------------------------------

ALTER SEQUENCE S2 RESTART WITH 100

cycle option :-

--------------------

=> by default sequence created with nocycle.

=> if sequence created with NOCYCLE then it starts from start with and generates upto max

and after reaching max then it stops.

=> if sequence created with CYCLE then it starts from start with and generates upto max

and after reaching max then it will reset to min.

CREATE SEQUENCE S3

START WITH 1

INCREMENT BY 1

MAXVALUE 5

MINVALUE 1

CYCLE

How to generate values for primary key columns ?

1 IDENTITY

2 SEQUENCE

difference between identity & sequence ?

IDENTITY SEQUENCE

1 identity always bind to specific column not bind to any column

in specific table

2 identity is generate whenever a new sequence next value is generated

is created in table. by calling NEXT VALUE FOR SEQUENCE

3 identity is not declared with max value can be declared with max value

=> list of sequences ?

SELECT \* FROM INFORMATION\_SCHEMA.SEQUENCES

Droping sequence :-

-------------------------

DROP SEQUENCE S1

19-apr-24

INDEXES :-

---------------

=> index is also a db object created to improve performance to data accessing.

=> index in db is similar to index in textbook , In textbook using index a particular topic

can be located fastly and in db using index a particular record can be located fastly.

=> indexes are created on columns and that column is called index key.

=> indexes are created on columns

1 frequently used in where clause

2 frequently used in join operation

Types of Indexes :-

-------------------------

1 Non Clustered Indexes

simple

composite

unique

2 Clustered Indexes

Non Clustered simple index :-

--------------------------------------

=> if index created on single column then it is called simple index.

syn :- CREATE INDEX <NAME> ON <TABNAME>(COLNAME)

ex :- CREATE INDEX I1 ON EMP(SAL)

=> after creating index sql server creates a structure called BTREE (balanced binary tree)

EMP 3000

SAL

3000

1000 2000 4000

4000

5000 1000 \* 2500 \* 4000 \* 5000 \*

2500 1500 \* 3000 \*,\*

1500 2000 \*

3000

2000

=> when we submit a query to sql server it uses following methods to find the row in a table

1 TABLE SCAN

2 INDEX SCAN

=> in TABLE SCAN sql server scans complete table i.e. each and every row.

=> in INDEX SCAN on avg sql server scans half of the table , so index scan is faster than

table scan.

SELECT \* FROM EMP WHERE SAL = 3000 ; (INDEX SCAN)

SELECT \* FROM EMP WHERE SAL >= 3000 ; (INDEX SCAN)

SELECT \* FROM EMP WHERE SAL<=3000; (INDEX SCAN)

SELECT \* FROM EMP ; (TABLE SCAN)

SELECT \* FROM EMP WHERE ENAME='BLAKE' (TABLE SCAN)

composite index :-

------------------------

=> if index created on multiple columns then it is called composite index.

ex :- CREATE INDEX I2 ON EMP(DEPTNO,JOB)

=> sql server uses above index when where cond based on leading column of the index i.e. deptno

SELECT \* FROM EMP WHERE DEPTNO = 20 (INDEX SCAN)

SELECT \* FROM EMP WHERE DEPTNO = 20 AND JOB='CLERK' (INDEX SCAN)

SELECT \* FROM EMP WHERE JOB='CLERK' (TABLE SCAN)

unique index :-

--------------------

=> unique index doesn't allow duplicate values into the column on which index is created.

ex :- CREATE UNIQUE INDEX I3 ON EMP(ENAME)

K

G Q

ADAMS \* JAMES \* MARTIN \* SCOTT \*

ALLEN \* JONES \* MILLER \* SMITH \*

BLAKE \* WARD \*

1 SELECT \* FROM EMP WHERE ENAME='BLAKE'

2 INSERT INTO EMP(EMPNO,ENAME,SAL) VALUES(100,'BLAKE',4000) +> ERROR

=> how many methods are there to enforce uniqueness ?

1 declare primary key / unique constraint

2 create unique index

=> primary key / unique columns are automatically indexed by sql server and

sql server creates unique index on primary key/unique columns and unique

index doesn't allow duplicates , so primary key also doesn't allow duplicates

PRIMARY KEY = UNIQUE INDEX + NOT NULL

20-apr-24

clustered index :-

-----------------------

=> a non clustered index stores pointers to actual records where as clustered index stores

actual record.

=> In non clustered index table and index are seperate objects where as in clustered index

table and index are one.

ex :-

CREATE TABLE CUST

(

CID INT,

CNAME VARCHAR(10)

)

CREATE CLUSTERED INDEX I10 ON CUST(CID)

INSERT INTO CUST VALUES(10,'A')

INSERT INTO CUST VALUES(80,'B')

INSERT INTO CUST VALUES(40,'C')

INSERT INTO CUST VALUES(60,'D')

50

30 70

10 A 40 C 60 D 80 B

1 SELECT \* FROM CUST => sql server goes to clustered index and access all the

leaf nodes from left to right

10 A

40 C

60 D

80 B

2 SELECT \* FROM CUST WHERE CID=60 => sql server goes to clustered index

and finds cid = 60 and returns row

from clustered index.

NOTE :-

=> only one clustered index allowed per table.

=> sql server implicitly creates a clustered index on primary key columns

=> difference between non clustered and clustered indexes ?

non clustered clustered

1 stores pointers to actual records stores actual records

2 table and index are seperate objects table and index are not seperate objects

3 needs extra storage doesn't need extra storage

4 needs two lookups to access needs one lookup to access specific row

specific row

5 sql server allows 999 non clustered sql server allows only one clustered

indexes per table index per table

6 by default non clustered indexes by default clustered index is

are created on unique columns created on primary key column

list of indexes ?

SP\_HELPINDEX EMP

Droping :-

-------------

DROP INDEX EMP.I1

if we drop table what about indexes created on table ?

ANS :- indexes are also dropped

Object hierarchy :-

------------------------

SERVER

DATABASE

TABLE

ROWS & COLS

CONSTRAINTS

INDEXES

TRIGGERS

VIEWS

SYNONYMS

SEQUENCES

PROCEDURES

FUNCTIONS

SQL

commands clauses opertors operations objects

DDL WHERE IN data filtering tables

DML ORDER BY BETWEEN data sorting views

DQL DISTINCT LIKE eliminating duplicates synonyms

TCL TOP IS find top n rows sequences

DCL GROUP BY ANY / ALL data grouping indexes

HAVING PIVOT joins

set operators

sub-queries

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

23-apr-24 T-SQL programming (Transact-SQL)

--------------------------------------------------

Features :-

-----------------

1 improves performance :-

--------------------------------

=> In T-SQL we can group sql commands into one block and we submit that block to sql server

so T-SQL no of requests and response between user and sql server are reduced and

performance is improved.

2 supports conditional statements :-

-----------------------------------------------

=> t-sql supports conditional statements like IF-ELSE , so in t-sql we can execute

sql commands based on conditions.

3 supports loops :-

-----------------------

=> t-sql supports looping statements like while , so in t-sql we can execute sql commands

repeatedly multiple times.

4 supports reusability :-

-------------------------------

=> t-sql programs can be stored in db , so applications which are connected to db can reuse

t-sql programs.

5 supports error handling :-

---------------------------------

=> in t-sql if any statement causes error then we can handle that error and we can

replace system generated message with our own simple and user friendly message.

=> T-SQL blocks are 2 types

1 Anonymous

2 Named

procedures

functions

triggers

Anonymous Blocks :-

----------------------------

=> a t-sql block without name is called anonymous block.

=> the following statements are used in t-sql programs.

1 DECLARE

2 SET

3 PRINT

DECLARE stmt :-

------------------------

=> used to declare variables & cursors

DECLARE @varname datatype(size)

ex :-

DECLARE @x INT

DECLARE @s VARCHAR(10)

DECLARE @d DATE

OR

DECLARE @x INT,@s VARCHAR(10),@d DATE

SET stmt :-

----------------

=> stmt used to assign value to variable

SET @varname = value

ex :-

SET @x = 100

SET @s = 'abc'

SET @d = getdate()

PRINT :-

--------------

=> statement used to print values or messages

PRINT 'message'

PRINT @varname

ex :-

PRINT 'HELLO'

PRINT @x

Example 1 :-

DECLARE @a INT,@b INT,@c INT

SET @a=100

SET @b=200

SET @c = @a + @b

PRINT @c

Example 2 :-

=> write a prog to input date and print day of the week ?

DECLARE @d DATE

SET @d = GETDATE()

PRINT DATENAME(DW,@d)

DB programming with T-SQL :-

-----------------------------------------

=> from t-sql to work with db execute sql commands from t-sql program and the

following commands are executed from t-sql program.

1 DML (insert,update,delete,merge)

2 DQL (select)

3 TCL (commit,rollback,save transaction)

SELECT stmt syntax :-

-------------------------------

SELECT @var1 = col1 ,

@var2 = col2,--------

FROM tabname

[WHERE cond]

ex :-

DECLARE @n VARCHAR(10),@s MONEY

SELECT @n = ename ,

@s = SAL

FROM emp

WHERE empno = 7369

PRINT @n + ' ' + CAST(@s AS VARCHAR)

24-apr-24

=> write a prog to input empno and calculate experience in years ?

DECLARE @eno INT,@hire DATE,@expr TINYINT

SET @eno = 7844

SELECT @hire=hiredate FROM emp WHERE empno = @eno

SET @expr = DATEDIFF(YY,@hire,GETDATE())

PRINT 'EXPERIENCE = ' + CAST(@expr AS VARCHAR) + ' YEARS '

=> write a prog to input empno and calculate total sal ?

DECLARE @eno INT,@sal MONEY,@comm MONEY,@totsal MONEY

SET @eno = 7566

SELECT @sal=sal,@comm=comm FROM emp WHERE empno = @eno

SET @totsal = @sal + ISNULL(@comm,0)

PRINT ' Total Sal = ' + CAST(@totsal as varchar)

conditional statements :-

-------------------------------

1 if-else

2 multi if

3 nested if

if-else :-

-----------

if cond

begin

statements

end

else

begin

statements

end

multi if :-

-------------

if cond1

begin

statements

end

else if cond2

begin

statements

end

else if cond3

begin

statements

end

else

begin

statements

end

nested if :-

---------------

if cond

begin

if cond

begin

statements

end

else

begin

statements

end

end

else

begin

statements

end

=> write a prog to input empno and increment sal by specific amount

and after increment if sal exceeds 5000 then cancel that increment ?

DECLARE @eno INT,@amt MONEY,@sal MONEY

SET @eno=7788

SET @amt = 2500

BEGIN TRANSACTION

UPDATE EMP SET SAL = SAL + @amt WHERE EMPNO = @eno

SELECT @sal = SAL FROM EMP WHERE EMPNO = @eno

IF @sal > 5000

ROLLBACK

ELSE

COMMIT

=> write a prog to input empno and calculate experience , if expr > 40

then delete row otherwise incr sal by 10% ?

DECLARE @eno INT,@hire DATE,@expr TINYINT

SET @eno = 7844

SELECT @hire=hiredate FROM emp WHERE empno = @eno

SET @expr = DATEDIFF(YY,@hire,GETDATE())

IF @expr > 40

DELETE FROM EMP WHERE EMPNO = @eno

ELSE

UPDATE EMP SET SAL = SAL + (SAL\*0.1) WHERE EMPNO = @eno

25-apr-24

=> write a prog to input empno and increment salary as follows

if job=clerk incr sal by 10%

salesman 15%

manager 20%

others 5%

DECLARE @eno INT,@job VARCHAR(10),@pct TINYINT

SET @eno = 7788

SELECT @job = job FROM emp WHERE empno = @eno

IF @job='clerk'

SET @pct = 10

ELSE IF @job='salesman'

SET @pct = 15

ELSE IF @job='manager'

SET @pct = 20

ELSE

SET @pct = 5

UPDATE EMP SET SAL = SAL + (SAL\*@pct/100) WHERE EMPNO = @eno

=> Write a prog to process bank transaction (w/d) ?

ACCOUNTS

ACCNO ACTYPE BAL

100 S 10000

101 S 20000

DECLARE @acno INT,@type CHAR(1),@amt MONEY,@bal MONEY

SET @acno = 100

SET @type='W'

SET @amt=1000

IF @type='w'

BEGIN

SELECT @bal=bal FROM accounts WHERE accno = @acno

IF @amt > @bal

PRINT 'insufficient balance'

ELSE

UPDATE accounts SET bal = bal - @amt WHERE accno = @acno

END

ELSE IF @type='d'

UPDATE accounts SET bal = bal + @amt WHERE accno = @acno

ELSE

PRINT 'invalid transaction type'

=> write a prog for money transfer ?

DECLARE @sacno INT,@tacno INT,@amt MONEY,@bal MONEY

SET @sacno=100

SET @tacno=101

SET @amt = 1000

SELECT @bal = bal FROM accounts WHERE accno = @sacno

IF @amt > @bal

PRINT 'insufficient balance'

ELSE

BEGIN

BEGIN TRANSACTION

UPDATE accounts SET bal = bal - @amt WHERE accno = @sacno

UPDATE accounts SET bal = bal + @amt WHERE accno = @tacno

COMMIT

END

WHILE loop :-

-------------------

=> loops are used to execute statements repeatedly multiple times

WHILE(cond)

BEGIN

statements

END

=> if cond = true loop continues

if cond = false loop terminates

=> prog to print nos from 1 to 20 ?

declare @x int=1

while(@x<=20)

begin

print @x

set @x = @x + 1

end

26-APR-24

=> write a prog to print 2024 calendar ?

date day

2024-01-01 ?

2024-12-31 ?

declare @d1 date,@d2 date

set @d1 = '2024-01-01'

set @d2 = '2024-12-31'

while(@d1<=@d2)

begin

print CAST(@d1 AS VARCHAR) + ' ' + DATENAME(DW,@d1)

set @d1 = DATEADD(DD,1,@d1)

end

=> write a prog to print sundays between two given dates ?

=> write a prog to input string and print following pattern ?

input :- NARESH

output :-

N

A

R

E

S

H

DECLARE @s VARCHAR(20),@x TINYINT = 1

SET @s = 'NARESH'

WHILE(@x <= LEN(@s))

BEGIN

PRINT SUBSTRING(@s,@x,1)

SET @x = @x+1

END

=> write a prog to input string and print following pattern ?

input :- NARESH

output :-

N

NA

NAR

NARE

NARES

NARESH

DECLARE @s VARCHAR(20),@x TINYINT = 1

SET @s = 'NARESH'

WHILE(@x <= LEN(@s))

BEGIN

PRINT SUBSTRING(@s,1,@x)

SET @x = @x+1

END

=> write a prog to input string and print reverse of that string ?

input :- NARESH

output :- HSERAN

CURSORS :-

------------------

=> cursors are used to access row-by-row in t-sql program.

=> from t-sql program if we submit a query to sql server , it goes to db

and copies the data from db to temporary memory called cursor

In t-sql program we can give name to cursor and access row-by-row

from cursor and process the row.

=> follow below steps to use cursor in t-sql program

1 declare cursor

2 open cursor

3 fetch record from cursor

4 close cursor

5 deallocate cursor

Declaring cursor :-

--------------------------

syn :- DECLARE <name> CURSOR FOR SELECT STATEMENT

ex :- DECLARE C1 CURSOR FOR SELECT ENAME,SAL FROM EMP

Opening cursor :-

-----------------------

OPEN <cursor-name>

ex :- OPEN C1 ;

1 select stmt submitted to sql server

2 sql server executes the query and data returned by query is copied to cursor

3 c1 points to cursor

Fetching records from cursor :-

------------------------------------------

=> "fetch" stmt is used to fetch record from cursor

FETCH NEXT FROM <CURSOR-NAME> INTO VARIABLES ;

Ex :-

FETCH NEXT FROM C1 INTO @name,@sal

=> a fetch stmt fetches one row at a time but to process multiple rows

fetch stmt should be executed multiple times , so fetch stmt should

be in a loop.

closing cursor :-

------------------------

CLOSE C1

Deallocate cursor :-

------------------------

DEALLOCATE C1 ;

@@fetch\_status :-

----------------------------

=> It is a system variable that returns fetch status i.e. fetch is successful or not

0 => fetch is successful

-1 => fetch is unsuccessful

27-APR-24

=> write a prog to print all employee names and salaries ?

DECLARE C1 CURSOR FOR SELECT ENAME,SAL FROM EMP

DECLARE @NAME VARCHAR(10),@SAL MONEY

OPEN C1

FETCH NEXT FROM C1 INTO @NAME,@SAL

WHILE(@@FETCH\_STATUS=0)

BEGIN

PRINT @NAME + ' ' + CAST(@SAL AS VARCHAR)

FETCH NEXT FROM C1 INTO @NAME,@SAL

END

CLOSE C1

DEALLOCATE C1

=> write a prog to calculate total sal without using sum function ?

DECLARE C1 CURSOR FOR SELECT SAL FROM EMP

DECLARE @SAL MONEY,@TOTSAL MONEY=0

OPEN C1

FETCH NEXT FROM C1 INTO @SAL

WHILE(@@FETCH\_STATUS=0)

BEGIN

SET @TOTSAL = @TOTSAL + @SAL

FETCH NEXT FROM C1 INTO @SAL

END

PRINT @TOTSAL

CLOSE C1

DEALLOCATE C1

=> write a prog to calculate max sal without using max function ?

=> write a prog to calculate min sal without using min function ?

=> write a prog to calculate all the students total,avg,result and insert into result table ?

STUDENT

SNO SNAME S1 S2 S3

1 A 80 90 70

2 B 30 60 50

RESULT

SNO TOTAL AVG RESULT

DECLARE C1 CURSOR FOR SELECT SNO,S1,S2,S3 FROM STUDENT

DECLARE @SNO INT,@S1 INT,@S2 INT,@S3 INT

DECLARE @TOTAL INT,@AVG DECIMAL(5,2),@RES CHAR(4)

OPEN C1

FETCH NEXT FROM C1 INTO @SNO,@S1,@S2,@S3

WHILE(@@FETCH\_STATUS=0)

BEGIN

SET @TOTAL = @S1 + @S2 + @S3

SET @AVG = @TOTAL/3.0

IF @S1>=35 AND @S2>=35 AND @S3>=35

SET @RES = 'PASS'

ELSE

SET @RES = 'FAIL'

INSERT INTO RESULT VALUES(@SNO,@TOTAL,@AVG,@RES)

FETCH NEXT FROM C1 INTO @SNO,@S1,@S2,@S3

END

CLOSE C1

DEALLOCATE C1

=> write a prog to increment employee salaries based on the pct in emp\_hike table ?

EMP\_HIKE

EMPNO PCT

7369 20

7499 10

7521 15

7566 20

DECLARE C1 CURSOR FOR SELECT EMPNO,PCT FROM EMP\_HIKE

DECLARE @ENO INT,@PCT INT

OPEN C1

FETCH NEXT FROM C1 INTO @ENO,@PCT

WHILE(@@FETCH\_STATUS-0)

BEGIN

UPDATE EMP SET SAL = SAL + (SAL \* @PCT/100) WHERE EMPNO = @ENO

FETCH NEXT FROM C1 INTO @ENO,@PCT

END

CLOSE C1

DEALLOCATE C1

29-apr-24

SCROLLABLE CURSOR :-

-----------------------------------

=> by default cursor is forward only cursor because it supports forward

navigation and doesn't support bakcward navigation.

=> if cursor declared with SCROLL then it is called scrollable cursor

and it supports both forward and backward navigation.

=> a forward only cursor supports only fetch next statement but scrollable

cursor supports the following fetch statements.

FETCH FIRST => fetches first record

FETCH NEXT => fetches next record

FETCH PRIOR => fetches previous record

FETCH LAST => fetches last record

FETCH ABSOLUTE N => fetches Nth record from first record

FETCH RELATIVE N => fetches Nth record from current record

example 1 :-

DECLARE C1 CURSOR SCROLL FOR SELECT ENAME FROM EMP

DECLARE @NAME VARCHAR(10)

OPEN C1

FETCH FIRST FROM C1 INTO @NAME

PRINT @NAME

FETCH ABSOLUTE 5 FROM C1 INTO @NAME

PRINT @NAME

FETCH RELATIVE 5 FROM C1 INTO @NAME

PRINT @NAME

FETCH LAST FROM C1 INTO @NAME

PRINT @NAME

FETCH PRIOR FROM C1 INTO @NAME

PRINT @NAME

CLOSE C1

DEALLOCATE C1

Example 2 :-

=> write a prog to print names from last to first ?

DECLARE C1 CURSOR SCROLL FOR SELECT ENAME FROM EMP

DECLARE @NAME VARCHAR(10)

OPEN C1

FETCH LAST FROM C1 INTO @NAME

WHILE(@@FETCH\_STATUS=0)

BEGIN

PRINT @NAME

FETCH PRIOR FROM C1 INTO @NAME

END

CLOSE C1

DEALLOCATE C1

=> write a prog to print every 5th row ?

DECLARE C1 CURSOR SCROLL FOR SELECT ENAME FROM EMP

DECLARE @NAME VARCHAR(10)

OPEN C1

FETCH RELATIVE 5 FRO C1 INTO @NAME

WHILE(@@FETCH\_STATUS=0)

BEGIN

PRINT @NAME

FETCH RELATIVE 5 FROM C1 INTO @NAME

END

CLOSE C1

DEALLOCATE C1

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

ERROR HANDLING / EXCEPTION HANDLING :-

------------------------------------------------------------------

1 syntax errors

2 logical errors

3 runtime errors

=> errors that are raised during program execution are called runtime errors

ex :- declare @x tinyint

set @x = 1000 => runtime error

print @x

=> if any statement causes runtime error then sql server displays error message and

continues program execution. To replace system generated message with our own

simple and user friendly message then we need to handle that runtime error.

=> To handle runtime error include a block called TRY-----CATCH block

BEGIN TRY

statements => causes exception

END TRY

BEGIN CATCH

statements => handles exception

END CATCH

=> if any statement is try block causes exception control is transferred to catch block

and executes the statements in catch block.

Example 1 :-

declare @a tinyint ,@b tinyint,@c tinyint

begin try

set @a=100

set @b=500

set @c=@a/@b

print @c

end try

begin catch

print 'runtime error----try again'

end catch

error handling functions :-

----------------------------------

1 ERROR\_NUMBER() => error code

2 ERROR\_MESSAGE() => error message

3 ERROR\_SEVERITY() => error severity level

4 ERROR\_STATE() => error state

5 ERROR\_LINE() => line number

30-apr-24

example 1 :-

declare @a tinyint ,@b tinyint,@c tinyint

begin try

set @a=100

set @b=0

set @c=@a/@b

print @c

end try

begin catch

if error\_number()=220

print 'value exceeding limit'

else if error\_number()=8134

print 'divisor cannot zero'

else

print 'unknown error'

end catch

example 2 :-

CREATE TABLE EMP88

(

EMPID INT PRIMARY KEY,

ENAME VARCHAR(10) NOT NULL,

SAL MONEY CHECK(SAL>=3000)

)

=> write a prog to insert data into emp88 ?

DECLARE @eno INT,@name VARCHAR(10),@sal MONEY

BEGIN TRY

SET @eno=101

SET @name='B'

SET @sal=5000

INSERT INTO EMP88 VALUES(@eno,@name,@sal)

END TRY

BEGIN CATCH

IF ERROR\_NUMBER()=2627

PRINT 'empno should not be duplicate'

ELSE IF ERROR\_NUMBER() = 515

PRINT 'name should not be null'

ELSE IF ERROR\_NUMBER() = 547

PRINT 'sal >= 3000'

ELSE

PRINT ERROR\_MESSAGE()

END CATCH

USER DEFINED ERRORS :-

--------------------------------------

=> errors raised by user are called user defined errors.

=> user raises error to stop program execution based some condition.

=> a user can raise error by using raiserror procedure

RAISERROR(error msg,severity level,state)

severity level => 0 to 25

0 to 10 => messages

11 to 18 => errors

19 to 25 => fatal errors

state => 0 to 255

example 1 :-

=> write a prog to increment specific employee sal by specific amount but

sunday updates are not allowed ?

DECLARE @ENO INT,@AMT MONEY

SET @ENO = 7844

SET @AMT = 1000

IF DATENAME(DW,GETDATE())='SUNDAY'

RAISERROR('sunday not allowed',16,1)

ELSE

UPDATE EMP SET SAL = SAL + @AMT WHERE EMPNO = @ENO

list of errors in sql server ?

SELECT \* FROM SYS.MESSAGES

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

NAMED T-SQL BLOCKS :-

----------------------------------

1 STORED PROCEDURES

2 FUNCTIONS

3 TRIGGERS

SUB-PROGRAMS :-

--------------------------

1 STORED PROCEDURES

2 FUNCTIONS

Advantages :-

-------------------

1 modular programming :-

---------------------------------

=> with the help of procedures and functions a big t-sql program can be divided into small modules

2 reusability :-

-----------------

=> proc & func are stored in db and applications which are connected to db can reuse proc & func.

3 security :-

---------------

=> because these programs are stored in db so they are secured and only authorized

users can execute these programs.

4 invoked from front-end :-

----------------------------------

=> proc/func can be invoked from front-end applications like java / .net / python etc

5 improves performance :-

---------------------------------

=> proc/func improves performance because they are precompiled i.e. when we create a

procedure program is compiled and stored in db and whenever we call procedure

only execution is repeated but not compilation , so this improves performance.

STORED PROCEDURES :-

-------------------------------------

=> a stored procedure is a named t-sql block that accepts some input performs some action

on db and may or may not returns a value.

=> procedures are created to perform one or more actions like insert,update,delete

Syntax :-

CREATE OR ALTER PROCEDURE <NAME>

parameters if any

AS

statements

parameters :-

-----------------

=> with the help of parameters procedures can receive and sends value to main program.

=> parameters are 2 types

1 INPUT (DEFAULT)

2 OUTPUT

=> input parameter always receives value

=> output parameter always sends value

Example 1 :- without parameters

=> create procedure to incerment all the employee salaries by 1000

CREATE OR ALTER PROCEDURE raise\_salary

AS

U

00

procedure created (compiled + stored in db)

execution :-

---------------

EXECUTE proc-name

ex :-

EXECUTE raise\_salary

Example 2 :-

=> create procedure to increment specific employee sal by specific amount ?

CREATE OR ALTER PROCEDURE raise\_salary

@eno INT ,

@amt MONEY

AS

UPDATE EMP SET SAL = SAL + @amt WHERE EMPNO = @eno

execution :-

----------------

EXECUTE raise\_salary 7369,1000

Example 3 :-

=> create a procedure to increment specific employee sal by specific amount

and after increment send the updated sal to calling program ?

CREATE OR ALTER PROCEDURE raise\_salary

@eno INT,

@amt MONEY,

@newsal MONEY OUTPUT

AS

UPDATE EMP SET SAL = SAL + @amt WHERE EMPNO = @eno

SELECT @newsal = sal FROM EMP WHERE EMPNO = @eno

Execution :-

postional association :-

-------------------------------

declare @sal money

execute raise\_salary 7369,1000,@sal output

print @sal

named association :-

---------------------------

declare @sal money

execute raise\_salary @eno=7369,@amt=1000,@newsal = @sal output

print @sal

Declaring parameters with default value :-

---------------------------------------------------------

=> a parameter can be declared with default value as follows

@amt money = 500

=> while calling procedure if we don't pass value to @amt then sql server assigns default value

Example 4 :-

CREATE OR ALTER PROCEDURE raise\_salary

@eno INT,

@amt MONEY = 500,

@newsal MONEY OUTPUT

AS

UPDATE EMP SET SAL = SAL + @amt WHERE EMPNO = @eno

SELECT @newsal = sal FROM EMP WHERE EMPNO = @eno

execution :-

declare @s money

execute raise\_salary 7369,default,@s output

print @s

declare @s money

execute raise\_salary @eno=7369,@newsal=@s output

print @s

2-may-24

example 5 :-

ACCOUNTS

ACCNO ACTYPE BAL

100 S 10000

101 S 20000

=> create a procedure for money withdrawl ?

CREATE OR ALTER PROCEDURE DEBIT

@acno INT ,

@amt MONEY,

@newbal MONEY OUTPUT

AS

DECLARE @BAL MONEY

SELECT @BAL = BAL FROM ACCOUNTS WHERE ACCNO = @acno

IF @amt > @BAL

RAISERROR('insufficient funds',16,1)

ELSE

BEGIN

UPDATE ACCOUNTS SET BAL = BAL - @amt WHERE ACCNO = @acno

SELECT @newbal = BAL FROM ACCOUNTS WHERE ACCNO = @acno

END

execution :-

----------------

declare @b money

execute debit 100,1000,@b output

print @b

=> create a procedure for money deposit ?

=> create a procedure for money transfer ?

USER DEFINE FUNCTIONS :-

-----------------------------------------

=> when predefine functions not meeting our requirements then we create our own functions

called user define functions.

=> a function is also a named T-SQL block that accepts some input performs some calculation

and must return a value.

=> functions are created

1 for calculations

2 to fetch data from table

=> functions are 2 types

1 scalar valued functions (SVF)

2 table valued functions (TVF)

SCALAR VALUED FUNCTIONS :-

---------------------------------------------

=> these functions returns one value

=> return type must be scalar types like int,varchar,money etc

=> return expression must be scalar variable

syn :-

CREATE OR ALTER

FUNCTION <NAME> (parameters) RETURNS <type>

AS

BEGIN

statements

RETURN <expr>

END

example 1 :-

CREATE OR ALTER

FUNCTION CALC(@a INT,@b INT,@op CHAR(1)) RETURNS INT

AS

BEGIN

DECLARE @c INT

IF @op='+'

SET @c = @a+@b

ELSE IF @op='-'

SET @c = @a-@b

ELSE IF @op='\*'

SET @c = @a\*@b

ELSE

SET @c = @a/@b

RETURN @c

END

Execution :-

-------------------

SELECT DBO.CALC(10,20,'\*') => 200

declare @res int

set @res = dbo.calc(10,20,'\*')

print @res

example 2 :-

=> create a function to calculate experience of the employee ?

CREATE OR ALTER

FUNCTION expr(@eno INT) RETURNS INT

AS

BEGIN

DECLARE @hire DATE,@expr INT

SELECT @hire = HIREDATE FROM EMP WHERE EMPNO = @eno

SET @expr = DATEDIFF(YY,@hire,getdate())

RETURN @expr

END

EXECUTION :-

SELECT DBO.EXPR(7369) => 44

03-may-24

=> create a function that accepts deptno and returns names of the employees working for

that dept ?

input deptno = 20

output :- smith,jones,scott,adams,ford

CREATE OR ALTER

FUNCTION getNames(@dno INT) RETURNS VARCHAR(1000)

AS

BEGIN

DECLARE C1 CURSOR FOR SELECT ENAME FROM EMP WHERE DEPTNO = @dno

DECLARE @name VARCHAR(10),@s VARCHAR(1000)=' '

OPEN C1

FETCH NEXT FROM C1 INTO @name

WHILE(@@FETCH\_STATUS=0)

BEGIN

SET @s = @s + @name + ','

FETCH NEXT FROM C1 INTO @name

END

CLOSE C1

DEALLOCATE C1

RETURN @s

END

EXECUTION :-

SELECT DBO.GETNAMES(20) => SMITH,JONES,SCOTT,ADAMS,FORD,

TABLE VALUED FUNCTIONS :-

---------------------------------------------

=> these functions returns rows (records)

=> return type must be TABLE

=> return expression must be select stmt

=> TVF allows only one stmt and it must be return stmt

syn :-

CREATE OR ALTER

FUNCTION <NAME>(parameters) RETURNS TABLE

AS

RETURN (SELECT STATEMENT)

Ex 1 :-

=> create function that accepts deptno and returns employees working for that dept ?

CREATE OR ALTER

FUNCTION GETEMPLIST(@dno INT) RETURNS TABLE

AS

RETURN (SELECT \* FROM EMP WHERE DEPTNO = @dno)

EXECUTION :-

=> TVF are invoked in FROM clause

SELECT \* FROM DBO.GETEMPLIST(20)

Ex 2 :-

=> create a function that returns top n employee list based on sal ?

CREATE OR ALTER

FUNCTION GETTOPNEMPLIST(@n INT) RETURNS TABLE

AS

RETURN ( SELECT \*

FROM (SELECT EMPNO,ENAME,SAL,

DENSE\_RANK() OVER (ORDER BY SAL DESC) AS RNK

FROM EMP) AS E

WHERE RNK <= @n ) )

EXECUTION :-

select \* from dbo.gettopnemplist(10)

=> difference between scalar and table valued functions ?

SCALAR TABLE

1 returns one value returns records

2 return type must be return type must table

scalar types like int

3 return expr is a return expr is select stmt

scalar variable

4 invoked in select clause invoked in from clause

=> difference between procedures and functions ?

PROCEDURES FUNCTIONS

1 may or may not returns a value must return a value

2 can return multiple always returns one

3 returns values using OUTPUT returns value using return stmt

parameter

4 we can execute dml commands dmls commands are not allowed in

from procedure functions

5 cannot be executed from select can be executed from select command

command

6 created to perform one or more dmls created for calculations

7 create procedure to update balance create function to get balance

=> list of procedures & functions created by user ?

SELECT \* FROM INFORMATION\_SCHEMA.ROUTINES

04-may-24

TRIGGERS :-

------------------

=> a trigger is also a named T-SQL block like procedure but executed implicitly by sql server

whenever user submits dml commands to sql server.

=> triggers are created

1 to control dmls

2 to enforce complex rules and validations

3 to audit day-to-day operations on tables

Syntax :-

CREATE OR ALTER TRIGGER <NAME>

ON <TABNAME>

AFTER / INSTEAD OF INSERT,UPDATE,DELETE

AS

STATEMENTS

AFTER triggers :-

-----------------------

=> if trigger is after then sql server executes the trigger after executing dml

INSTEAD OF triggers :-

---------------------------------

=> if trigger is instead of then sql server executes the trigger instead of executing dml

example 1 :-

=> create trigger to not to allow dmls on emp table on sunday ?

CREATE OR ALTER TRIGGER T1

ON EMP

AFTER INSERT,UPDATE,DELETE

AS

IF DATENAME(DW,GETDATE())='SUNDAY'

BEGIN

ROLLBACK

RAISERROR('sunday not allowed',16,1)

END

Testing :-

UPDATE EMP SET SAL = 2000 WHERE EMPNO = 7369 => error

Example 2 :-

=> create trigger to not to allow dmls on emp table as follows ?

mon - fri <10am and >4pm

sat <10am and >2pm

sun -----------------------

CREATE OR ALTER TRIGGER T2

ON EMP

AFTER INSERT,UPDATE,DELETE

AS

IF DATEPART(DW,GETDATE()) BETWEEN 2 AND 6

AND

(

DATEPART(HH,GETDATE()) < 10

OR

DATEPART(HH,GETDATE()) >= 16

)

BEGIN

ROLLBACK

RAISERROR('only between 10am and 4pm',16,1)

END

ELSE IF DATEPART(DW,GETDATE())=7

AND

(

DATEPART(HH,GETDATE()) < 10

OR

DATEPART(HH,GETDATE()) >= 14

)

BEGIN

ROLLBACK

RAISERROR('only between 10am and 2pm',16,1)

END

ELSE IF DATEPART(DW,GETDATE())=1

BEGIN

ROLLBACK

RAISERROR('sunday not allowed',16,1)

END

Example 3 :-

=> create trigger to not to allow to update empno ?

CREATE OR ALTER TRIGGER T3

ON EMP

AFTER UPDATE

AS

IF UPDATE(EMPNO)

BEGIN

ROLLBACK

RAISERROR('empno cannot be updated',16,1)

END

Testing :-

UPDATE EMP SET EMPNO = 9999 WHERE EMPNO = 7844 => ERROR

06-may-24

Magic tables :-

-------------------

INSERTED

DELETED

=> with the help of magic tables we can access data affected by dmls

=> these tables are called magic tables because these tables are created and destroyed automatically

=> these tables can be access with in triggers

ex :-

INSERT INTO EMP VALUES(100,'A','CLERK',4000,----) => INSERTED

EMPNO ENAME JOB SAL

` 100 A CLERK 4000

DELETE FROM EMP WHERE EMPNO = 7369 => DELETED

EMPNO ENAME JOB SAL

7369 smith clerk 800

UPDATE EMP SET SAL =5000 WHERE EMPNO = 100 => INSERTED

EMPNO SAL

100 5000

DELETED

EMPNO SAL

100 4000

Example 4 :-

=> create trigger to not to allow user to decrement salary ?

CREATE OR ALTER TRIGGER T4

ON EMP

AFTER UPDATE

AS

DECLARE @OLDSAL MONEY,@NEWSAL MONEY

SELECT @OLDSAL = SAL FROM DELETED

SELECT @NEWSAL = SAL FROM INSERTED

IF @NEWSAL < @OLDSAL

BEGIN

ROLLBACK

RAISERROR('sal cannot be decremented',16,1)

END

testing :-

UPDATE EMP SET SAL = SAL-100 WHERE EMPNO = 7369 => ERROR

Example 5 :-

-----------------

=> create trigger to insert details into emp\_resign table whenever employee resigns ?

EMP\_RESIGN

EMPNO ENAME JOB SAL HIREDATE DOR

CREATE TABLE EMP\_RESIGN

(

EMPNO INT,

ENAME VARCHAR(10),

JOB VARCHAR(10),

SAL MONEY,

HIREDATE DATE,

DOR DATE

)

CREATE OR ALTER TRIGGER T5

ON EMP

AFTER DELETE

AS

INSERT INTO EMP\_RESIGN

SELECT EMPNO,ENAME,JOB,SAL,HIREDATE,GETDATE() FROM DELETED

Testing :-

DELETE FROM EMP WHERE EMPNO = 7369

SELECT \* FROM EMP\_RESIGN

7369 SMITH CLERK 800 1980-12-17 2024-05-06

Droping triggers :-

-----------------------

DROP TRIGGER T1

ENABLE & DISABLE TRIGGERS :-

-----------------------------------------------

DISABLE TRIGGER T1 ON EMP

ENABLE TRIGGER T1 ON EMP

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

Dynamic SQL :-

---------------------

=> SQL commands generated at runtime are called dynamic sql commands

ex :- DROP TABLE EMP (static sql)

DECLARE @TNAME VARCHAR(20)

SET @TNAME = 'EMP'

DROP TABLE @TNAME (dynamic sql)

=> dynamic sql commands are executed by using EXEC

EXEC ('dynami sql command')

=> dynamic sql command that you want to execute should be passed as a string to EXEC

example 1 :-

DECLARE @TNAME VARCHAR(20)

SET @TNAME = 'EMP'

EXCEC (' DROP TABLE ' + @TNAME)

example 2 :-

=> create a procedre to drop table ?

CREATE OR ALTER PROCEDURE DROP\_TABLE

@tname VARCHAR(20)

AS

EXEC (' DROP TABLE ' + @tname)

execution :-

EXEC DROP\_TABLE 'STUDENT'

example 3 :-

=> create a procedre to drop all tables ?

CREATE OR ALTER PROCEDURE DROP\_ALL\_TABLES

AS

DECLARE C1 CURSOR FOR SELECT TABLE\_NAME

FROM INFORMATION\_SCHEMA.TABLES

WHERE TABLE\_TYPE='BASE TABLE'

DECLARE @TNAME VARCHAR(20)

OPEN C1

FETCH NEXT FROM C1 INTO @TNAME

WHILE(@@FETCH\_STATUS=0)

BEGIN

EXEC (' DROP TABLE ' + @TNAME)

FETCH NEXT FROM C1 INTO @TNAME

END

CLOSE C1

DEALLOCATE C1