03-JUL-23

SQL SERVER

=> a database is a organized collection of interrelated data. For example
a univ db stores data related to students,courses,faculty etc.

Types of Databases :-

1 OLTP DB (online transaction processing)
2 OLAP DB (online analytical processing)

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

=> OLTP for running business and OLAP for analyzing business.

=> day-to-day operations on db includes

C	create
R	read
U	update
D	delete

DBMS :-

=> DBMS stands for Database Management System , It is a software
used to create and to manage database.

=> DBMS is an interface between user and database.

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

Data Models :-

=> based on the structure of the data data models are 3 types

1 Hierarchical

2 Network
3 Relational

Relational Model :-

=> Relational Model introduced by E.F.CODD.

=> according to E.F.CODD in relational model data must be organized in
tables i.e. rows and columns

=> a dbms that supports relational model is called rdbms

```
CUST
CID NAME  ADDR  =>  COLUMNS/FIELDS/ATTRIBUTES
10   A    HYD
11   B    BLR
12   C    DEL   =>  ROW/RECORD/TUPLE
```

```
DATABASE  =   COLLECTION OF TABLES
TABLE     =   COLLECTION OF ROWS & COLS
ROW       =   COLLECTION OF FIELD VALUES
COLUMN    =   COLLECTION OF VALUES ASSIGNED TO ONE FIELD
```

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

ex :- ACCNO,EMPID,AADHARNO,PANNO,VOTERID

RDBMS softwares :-

SQL SERVER	from microsoft
ORACLE	from oracle corp
DB2	from IBM
MYSQL	from oracle corp
POSTGRESQL	from postgresql forum
RDS	from amazon

ORDBMS :-

=> Object Relational Database Management System

ORDBMS = RDBMS + OOPS (reusability)

=> RDBMS doesn't support reusability but ORDBMS supports reusability

ORDBMS softwares :-

SQL SERVER
ORACLE
POSTGRESQL

what is SQL SERVER ?

SQL SERVER is basically a rdbms product from microsoft and also supports ordbms features and used to manage database.

=> SQL SERVER is used for DB Development & Administration

Development	Administration
creating tables	Installation of sql server
creating views	creating database
creating synonyms	creating logins
creating sequences	backup & restore
creating indexes	export & import
creating procedures	performance tuning
creating functions	
creating triggers	
writing queries	

CLIENT / SERVER Architecture :-
----

1 SERVER
2 CLIENT

=> server is a system where sql server software is installed and running.
=> inside server sql server manages database.
=> a client is a system from where users can

1 connects to server
2 submit requests to server

3 receives response from server

client tool :-

SSMS (SQL SERVER MANAGEMENT STUDIO)

How 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

creating database in server :-

=> in object explorer select Databases => New Database

Enter Database Name :- BATCH12

=> click OK

=> a new database is created with following two files

1 DATA FILE (.MDF)
2 LOG FILE (.LDF)

=> DATA FILE stores data and LOG FILE stores operations

```
    NAME                TYPE  INITIAL SIZE  AUTOGROWTH        PATH
    BATCH12   DATA   8               64              C:\
    BATCH12_LOG  LOG      8                  64               C:\
```

USER-----SSMS------------------------------------SQL SERVER ---------BATCH12(DB)

SQL :-

=> STRUCTURED QUERY LANGUAGE
=> 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 submitted to sql server to perform
some operation over db.
=> SQL is originally introduced by IBM   and initial name of this lang was SEQUEL
and later it is renamed to SQL.
=> SQL is common to all RDBMS

```
  sql server        oracle          mysql          postgresql
     SQL                  SQL        SQL                    SQL


 USER-----SSMS------------SQL---------------------SQL SERVER--------------DB
              tool                lang                        software                      storage


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


 USER----MYSQLWORKBENCH---------SQL------------MYSQL-------------DB
```

5-JUL-23

=> based on operations over db sql is divided 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
          TRUNCATEMERGE
```

DATA & DATA DEFINITION :-

```
EMPID   ENAME     SAL           DATA DEFINTION / METADATA
 100     A         5000         DATA
```

Datatypes in SQL SERVER :-

=> a datatype specifies

1 type of the data allowed in column
2 amount of memory allocated for column

character types :-

```
ASCII            UNICODE

char             nchar
   varchar             nvarchar
   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
```

NOTE :- in char datatype extra bytes are wasted , so char is not recommended for variable length columns and it is recommended for fixed length columns

```
STATE_CODE      CHAR(2)
```

```
                AP
                TS
                MH

                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
```

NOTE :-

char/varchar allows ascii characters (256 chars) that includes a-z,A-Z,0-9 and special chars. so char/varchar allows alphanumeric data.

```
  ex :-      PANNO     CHAR(10)
             VEHNO    CHAR(10)
             EMAILID  VARCHAR(30)
```

VARCHAR(MAX) :-

=> allows character data upto 2GB

```
 ex :-   FEEDBACK    VARCHAR(MAX)
```

NCHAR/NVARCHAR/NVARCHAR(MAX) :-

=> allows unicode chars (65536 chars) that includes all ascii chars and
     chars belongs to different languages.

Integer Types :-

=> allows numbers without decimal

TINYINT	1 BYTE	0 TO 255	
SMALLINT	2 BYTES	-32768 TO 32767	
INT	4 BYTES	-2^31 TO 2^31-1	(-2,147,483,647 to 2,147,483,646)
BIGINT	8 BYTES	-2^63 TO 2^63-1	(-9,223,372,036,854,775,807 to 9,223,372,036,854,775,806)

```
ex :-   AGE            TINYINT
        EMPID SMALLINT
        ACCNO          BIGINT
```

NUMERIC(P) :-

=> allows numbers upto 38 digits

```
 ex :-   EMPID   NUMERIC(4)

          10
          100
          1000
          10000  => NOT ALLOWED

           ACCNO   NUMERIC(13)

          AADHARNO   NUMERIC(12)

          CARD_NO    NUMERIC(16)
```

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

=> allows numbers with decimal (float)

p => precision => total no of digits allowed
s => scale => no of digits allowed after decimal

ex :-   SAL   NUMERIC(7,2)

5000
5000.55
50000.55
500000.55 => NOT ALLOWED

CURRENCY TYPES :-

=> currency types are 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 :-

1	DATE	=> allows only date
2	TIME	=> allows only time
3	DATETIME	=> allows date & time

=> default date format in sql server YYYY-MM-DD
=> default time format is HH:MI:SS

EX :-

```
 DOB  DATE

 2003-04-20

 LOGIN      TIME

9:30:00

TXN_DT        DATETIME

2023-07-05 10:00:00
```

06-jul-23

CREATING TABLES IN DATABASE :-

```
CREATE TABLE <TABNAME>
(
  COLNAME  DATATYPE(SIZE),
  COLNAME  DATATYPE(SIZE),
  -------------------------
)
```

Rules :-
----

1 tabname should start with alphabet
2 tabname should not contain spaces & special chars but allows _,#,$
3 tabname can be upto 128 chars
4 table can have 1024 cols
5 no of rows unlimited

123cust	invalid
cust 123	invalid
cust*123	invalid
cust_123	valid

Example :-

=> create table with following structure

```
EMP
EMPID    ENAME    JOB    SAL    HIREDATE    DNAME
```

```
CREATE TABLE EMP
(
 EMPID       TINYINT ,
 ENAME       VARCHAR(10),
 JOB         VARCHAR(10),
 SAL         SMALLMONEY,
 HIREDATE            DATE,
 DNAME               VARCHAR(10)
)
```

=> above command created table structure (columns)

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 :-

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

Ex :-

```
 INSERT INTO EMP VALUES(100,'SACHIN','CLERK',4000,'2023-07-06','HR')
 INSERT INTO EMP VALUES(101,'ARVIND','MANAGER',8000,'2020-10-5','IT')
```

inserting multiple rows :-

```
INSERT INTO EMP VALUES(102,'VIJAY','CLERK',6000,'2019-05-10','HR') ,
                      (103,'RAVI','ANALYST',7000,'2018-02-15','SALES')
```

inserting nulls :-

=> a nulls 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,NULL,'2021-04-12','IT')
```

method 2 :-

```
 INSERT INTO EMP(EMPID,ENAME,HIREDATE,DNAME)
                  VALUES(105,'SATISH','2022-09-10','SALES')
```

remaining two fields job,sal filled with NULLs.

Operators in sql server :-

```
 1   Arithmetic Operators  =>    +   -    *     /       %

 2   Relational Operators  =>    >    >=     <     <=     =      <>  or   !=

 3   Logical Operartors     =>    AND   OR    NOT

 4   Special Operators      =>   BETWEEN
                                 IN
                                 LIKE
                                 IS
                                 ANY
                                 ALL
                                 EXISTS

  5   Set Operators              =>  UNION
                                     UNION ALL
                                     INTERSECT
                                     EXCEPT
```

Displaying Data :-

=> "SELECT" command is used to display data from table.
=> we can display all rows and all columns
=> we can display specific rows and specific columns

```
syn :- SELECT  COLUMNS /  *    FROM   TABNAME

          SQL      =      ENGLISH

         QUERIES =    SENTENCES

          CLAUSES =    WORDS

               *   => all columns
```

=> display all the data from emp table ?

```
   SELECT * FROM EMP
```

=> display employee names and salaries ?

```
   SELECT ENAME,SAL FROM EMP
```

=> display employee names and hiredates ?

```
   SELECT ENAME,HIREDATE FROM EMP
```

WHERE clause :-

=> used to get specific row/rows from table based on a 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 cond = false row is not selected

=> display employee details whose id = 103 ?

```
 SELECT * FROM EMP WHERE EMPID = 103
 SELECT * FROM EMP WHERE ENAME='KUMAR'
 SELECT * FROM EMP WHERE SAL>5000
 SELECT * FROM EMP WHERE HIREDATE > 2020  => ERROR
 SELECT * FROM EMP WHERE HIREDATE > '2020-12-31'
 SELECT * FROM EMP WHERE HIREDATE < '2020-01-01'
 SELECT * FROM EMP WHERE DNAME <> 'HR'
```

Compound condition :-

=> muliple 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
```

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

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

=> display employees working as CLERK,MANAGER ?

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

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

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

=> employees joined in 2020 ?

```
SELECT *
FROM  EMP
WHERE HIREDATE >= '2020-01-01' AND HIREDATE <= '2020-12-31'
```

=> employees working as CLERK and earning more than 5000 and working for HR dept ?

```
SELECT *
FROM EMP
WHERE  JOB='CLERK'  AND   SAL>5000 AND DNAME ='HR'
```

IN operator :-

=> use IN operator for list comparision
=> use IN operator for "=" comparision with multiple values

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

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

=> employees working for HR,IT depts ?

```
SELECT * FROM EMP WHERE DNAME='HR'   OR  DNAME='IT'

SELECT * FROM EMP WHERE DNAME IN ('HR','IT')
```

=> employees not working as CLERK,MANAGER ?

```
SELECT * FROM EMP WHERE JOB NOT IN ('CLERK','MANAGER')
```

BETWEEN operator :-

=> use BETWEEN operator for range comparision

```
WHERE  COLNAME  BETWEEN V1 AND V2
WHERE  COLNAME NOT BETWEEN V1 AND V2
```

=> display employees earning between 5000 and 10000 ?

```
SELECT *
FROM EMP
WHERE SAL BETWEEN 5000 AND 10000
```

=> employees joined in 2020 year ?

```
SELECT *
FROM EMP
WHERE HIREDATE BETWEEN '2020-01-01' AND '2020-12-31'
```

=> employees working as CLERK,MANAGER and earning between 5000 and 10000
and joined in 2020 year and not working for HR,SALES dept ?

```
SELECT *
FROM EMP
WHERE  JOB IN ('CLERK','MANAGER')
                AND
               SAL BETWEEN 5000 AND 10000
               AND
```

```
                    HIREDATE BETWEEN '2020-01-01' AND '2020-12-31'
                    AND
                    DNAME NOT IN ('HR','SALES')
```

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

```
PRODUCTS
prodid   pname    price    category    brand

SELECT *
FROM PRODUCTS
WHERE CATEGORY='MOBILES'
             AND
            BRAND IN ('SAMSUNG','REDMI','ONEPLUS')
            AND
            PRICE BETWEEN 10000 AND 20000
```

=> list of male customers age between 20 and 30 and staying hyd,mum,blr ?

```
CUST
CUSTID   NAME    AGE    CITY     GENDER

SELECT *
FROM CUST
WHERE  GENDER='M'
             AND
             AGE BETWEEN 20 AND 30
             AND
             CITY IN ('HYD','MUM','BLR')
```

08-JUL-23

LIKE operator :-

=> use LIKE operator for pattern comparision

ex :-   name starts with 'S'
        emailid ends with '.in'

```
WHERE   COLNAME   LIKE   'PATTERN'
WHERE   COLNAME  NOT 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 'S' ?

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

=> where 'A' is the 4th char in their name ?

```
SELECT * FROM EMP WHERE ENAME LIKE '___A%'
```

=> 'A' is the 2nd char from last ?

```
SELECT * FROM EMP WHERE ENAME LIKE  '%A_'
```

=> name contains 4 chars ?

```
SELECT * FROM EMP WHERE ENAME LIKE  '____'
```

=> list of employees joined in oct month ?

YYYY-MM-DD

```
SELECT * FROM EMP WHERE HIREDATE  LIKE  '_____10___'
```

=> employees joined in 2020 year ?

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

=> display employees name starts with 'A','K','R' ?

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

```
                                   OR
                                   ENAME LIKE 'K%'
                                   OR
                                   ENAME LIKE 'R%'
```

```
SELECT * FROM EMP WHERE  ENAME LIKE  '[AKR]%'
```

=> employees name starts between 'A' and 'P'   ?

```
SELECT * FROM EMP WHERE ENAME LIKE '[A-P]%'
```

IS operator :-

=> use IS operator for NULL comparision

```
WHERE COLNAME IS NULL
WHERE COLNAME IS NOT NULL
```

=>: employees not earning salary ?

```
SELECT * FROM EMP WHERE SAL IS  NULL
```

=> employees earning salary ?

```
SELECT * FROM EMP WHERE SAL IS NOT NULL
```

summary :-

```
WHERE COLNAME IN (V1,V2,V3,---)
WHERE COLNAME BETWEEN V1 AND V2
WHERE COLNAME LIKE 'PATTERN'
WHERE COLNAME IS NULL
```

Question :-

1

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

A ERROR

B RETURNS CLERK & MANAGER
C RETURNS ONLY CLERK
D NONE

ANS :- C

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

ANS :- B

3 SELECT * FROM EMP WHERE SAL BETWEEN 5000 AND 2000

A ERROR
B RETURNS ROWS
C RETURNS NO ROWS
D NONE

ANS :- C

WHERE SAL BETWEEN 2000 AND 5000   (SAL>=2000 AND SAL<=5000)

WHERE SAL BETWEEN 5000 AND 2000   (SAL>=5000 AND SAL<=2000)

ALIAS :-

=> alias means another name or alternative name
=> used to change column heading

syn :-   COLNAM / EXPR [AS]   ALIAS

=> display ENAME   ANNUAL SALARY ?

```
SELECT  ENAME,SAL*12 AS ANNSAL FROM EMP

SELECT  ENAME,SAL*12 AS [ANNUAL SAL] FROM EMP
```

=> 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
```

```
SACHIN   4000    800     1200    400     5600
```

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
```

=> default order is ASC

=> arrange employee list name wise asc order ?

```
SELECT *
FROM EMP
ORDER BY  ENAME  ASC
```

=> arrange sal wise desc order ?

```
SELECT *
FROM EMP
ORDER BY  SAL  DESC
```

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

```
SELECT ENAME,SAL,DNAME
FROM EMP
ORDER BY DNAME ASC,SAL DESC
```

```
1   A   3000  HR                     5  E   6000 HR
2  B   5000  SALES                        1  A    3000 HR
```

```
  3  C   4000  IT         ============>              6 F     5000 IT
  4  D   2000  SALES                                  3 C    4000 IT
  5  E   6000  HR                                    2  B    5000 SALES
  6  F   5000  IT                                    4  D    2000 SALES
```

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

```
  SELECT ENAME,SAL,HIREDATE,DNAME
  FROM EMP
  ORDER BY DNAME ASC ,HIREDATE ASC
```

scenario :-

```
 STUDENTS
 SNO  SNAME   M    P    C
 1        A           80   90  70
 2       B            60   50  70
 3       C            90   80  70
 4       D            90   70  80
```

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

```
SELECT  * , (M+P+C)/3  AS AVG
FROM STUDENTS
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   50  70
```

=> display students list along with avg who got distinction ?

```
SELECT * , (M+P+C)/3  AS AVG
FROM STUDENTS
WHERE (M+P+C)/3 >= 70
ORDER BY (M+P+C)/3 DESC,M DESC,P DESC
```

DISTINCT clause :-

=> eliminates duplicates from the select statement output.

```
SELECT   DISTINCT  colname
```

Ex :-

```
SELECT DISTINCT DNAME FROM EMP

HR
IT
SALES
```

```
SELECT DISTINCT JOB FROM EMP

ANALYST
CLERK
MANAGER
```

TOP clause :-

=> used to find top n rows

```
syn :-    SELECT    TOP  <n>   COLNAMES / *
```

examples :-

=> display first 3 rows from emp table ?

```
SELECT TOP 3 *  FROM EMP
```

=> display top 3 highest paid employees ?

```
SELECT  TOP 3 *
FROM EMP
ORDER BY SAL DESC
```

=> display top 3 employees based on experience ?

```
SELECT TOP 3 *
FROM EMP
ORDER BY HIREDATE ASC
```

=> display top 3 max salaries ?

```
SELECT TOP 3 SAL
FROM EMP
ORDER BY SAL DESC
```

summary :-

WHERE	=> to select specific rows
ORDER BY	=> to sort rows
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.

11-jul-23

UPDATE :-

=> command used to modify table data.
=> we can update all rows or specific rows
=> we can update single column or multiple columns

syn :-

```
UPDATE   <TABNAME>
SET  COLNAME = VALUE ,  COLNAME = VALUE , ---------
[WHERE CONDITION]
```

Ex :-

=> update all employees comm with 500 ?

```
UPDATE EMP SET COMM = 500
```

NOTE :-

=> in SQL SERVER operations are auto committed (saved)
=> to stop auto commit execute the following command

```
SET IMPLICIT_TRANSACTIONS ON
```

=> after executing above command operations are not auotmatically committed
=> to save the operation execute commit.
=> to cancel the operation execute rollback.

=> update employees comm with 800 whose job is salesman and joined in 1981 year ?

```
UPDATE EMP
SET COMM = 800
WHERE JOB='SALESMAN'
           AND
          HIREDATE  LIKE  '1981%'
```

=> update sal with 1000 and comm with 800 whose empno = 7369 ?

```
UPDATE EMP
SET SAL = 1000 , COMM = 800
WHERE EMPNO = 7369
```

=> increment salaries by 20% and comm by 10% those working as CLERK,MANAGER ?

```
UPDATE EMP
SET SAL = SAL + (SAL*0.2) , COMM = COMM + (COMM*0.1)
WHERE JOB IN ('CLERK','MANAGER')
```

=> transfer employees from 10th dept to 30th dept ?

```
UPDATE EMP
SET DEPTNO = 30
WHERE DEPTNO = 10
```

scenario :-

PRODUCTS

prodid	pname	price	category	brand

=> increase samsung,oneplus,realme mobile phones price by 10% ?

```
 UPDATE PRODUCTS
 SET PRICE = PRICE + (PRICE*0.1)
 WHERE BRAND IN ('SAMSUNG','ONEPLUS','REALME')
              AND
              CATEGORY='MOBILES'
```

DELETE command :-

=> command used to delete row/rows from table.
=> we can delete all rows or specific rows

syn :- DELETE FROM <TABNAME> [WHERE COND]

ex :-

=> delete all rows from emp table ?

```
DELETE FROM EMP
```

=> delete employees whose id = 7369 , 7566,7844 ?

```
DELETE FROM EMP WHERE EMPNO IN (7369,7566,7844)
```

DDL commands :-   (Data Definition Lang)

CREATE
ALTER
DROP
TRUNCATE

=> all DDL commands acts on table structure (columns,datatype and size).

ALTER command :-

=> command used to modify table structure
=> using ALTER command we can

1 add columns

2 drop columns
3 modify a column
    changing datatype
    changing size

Adding column :-
----

ex :- add column gender to emp table ?

```
   ALTER TABLE EMP
       ADD GENDER  CHAR(1)
```

=> after adding by default the new column is filled with nulls
=> use update command to insert data into the new column

```
  UPDATE EMP SET GENDER='M'  WHERE EMPNO = 7369
```

Droping column :-

=> drop columns gender,comm from emp table ?

```
   ALTER TABLE EMP
           DROP COLUMN GENDER,COMM
```

Modifying a column :-

=> modify the empno column datatype to int ?

```
    ALTER TABLE EMP
         ALTER COLUMN EMPNO INT
```

=> increase 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

12-JUL-23

DROP command :-

=> command used to drop table from db
=> drops table structure along with data

syn :- DROP TABLE <tabname>

ex :- DROP TABLE STUDENTS

TRUNCATE command :-

=> deletes all 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 VS TRUNCATE :-

	DELETE	TRUNCATE
1	DML command	DDL command
2	can delete all rows and specific rows	can delete only all rows but cannot delete specific rows
3	where cond can	where cond cannot

	used with delete	be used 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 :-   (SP -> stored procedure)

=> used to change table name or column name

```
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'
```

Built-in Functions in SQL SERVER :-

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

Types of functions :-
----

1 DATE
2 STRING
3 NUMERIC
4 CONVERSION
5 SPECIAL
6 ANALYTICAL
7 AGGREGATE

DATE functions :-
----

1 GETDATE() :-

=> returns current date & time

```
SELECT GETDATE()   =>  2023-07-12 12:03:08.503
                       ---------------  ----------- ----
                         DATE        TIME      MS
```

2 DATEPART() :-

=> used to extract part of the date

```
DATEPART(interval,date)
```

ex :-

```
SELECT   DATEPART(YY,GETDATE())  =>  2023
                  MM                  07
                  DD                  12
                  DW                   4 (wed)
                  DY                   193 (day of year)
                  HH                    hour part
                  MI                    minutes
                  SS                    seconds
                  Q                     3
                                    jan-mar  1
                                   apr-jun   2
                                   jul-sep   3
                                              oct-de    4
```

=> display 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(Q,HIREDATE) = 2
```

DATENAME() :-

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

	MM	DW
DATEPART	7	4
DATENAME	JULY	WEDNESDAY

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

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

=> display SMITH joined on FRIDAY
ALLEN joined on WEDNESDAY ?

```
  SELECT  ENAME + ' joined on ' + DATENAME(DW,HIREDATE)
  FROM EMP
```

13-JUL-23

DATEDIFF() :-

=> returns difference between two dates in given interval

```
    DATEDIFF(INTERVAL,START DATE,END DATE)
```

EX :-

```
  SELECT DATEDIFF(YY,'2022-07-13',GETDATE())   =>    1
                             MM                             =>   12
                             DD                             =>  365
```

=> display   ENAME   EXPERIENCE in years ?

```
   SELECT  ENAME,
                DATEDIFF(YY,HIREDATE,GETDATE()) AS EXPERIENCE
   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
```

FORMAT() :-

=> function used to display dates in different formats

```
        FORMAT(DATE,'format')
```

ex :-

```
  SELECT FORMAT(GETDATE(),'MM/dd/yy')                    => 07/13/23
  SELECT FORMAT(GETDATE(),'dd.MM.yyyy')                  => 13.07.2023
  SELECT FORMAT(GETDATE(),'dd.MM.yyyy hh:mm')       => 13.07.2023 11:46

 SELECT ENAME,FORMAT(HIREDATE,'MM/dd/yy') AS HIREDATE FROM EMP
```

scenario :-

```
INSERT INTO EMP(EMPNO,ENAME,JOB,SAL,HIREDATE)
            VALUES(999,'ABC','CLERK',5000,GETDATE())
```

=> list of employees joined today ?

```
 SELECT *
 FROM EMP
WHERE  HIREDATE =  GETDATE()   =>  NO ROWS

             2023-07-13  =  2023-07-13 11:58:20.123
```

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

```
SELECT *
 FROM EMP
WHERE  HIREDATE =  FORMAT(GETDATE(),'yyyy-MM-dd')

            2023-07-13 =  2023-07-13
```

DATEADD() :-

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

```
           DATEADD(INTERVAL,INT,DATE)

SELECT  DATEADD(DD,10,GETDATE())     =>   2023-07-23
SELECT  DATEADD(MM,2,GETDATE())      =>   2023-09-13
SELECT  DATEADD(MM,-2,GETDATE())     =>   2023-05-13
```

scenario :-

GOLD_RATES

```
 DATEID                    RATE
 2020-01-01   ?
 2020-01-02          ?

 2023-07-13          ?
```

1 display today's gold rate ?
2 display yesterday's gold rate ?
3 SELECT *

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

4 display last year same day gold rate ?

1

```
  SELECT  *
  FROM GOLD_RATES
 WHERE  DATEID =  FORMAT(GETDATE(),'yyyy-MM-dd')
```

2

```
  SELECT *
  FROM GOLD_RATES
  WHERE  DATEID =     FORMAT(DATEADD(DD,-1,GETDATE()),'yyyy-MM-dd')
```

3

```
  SELECT *
  FROM GOLD_RATES
  WHERE  DATEID =     FORMAT(DATEADD(MM,-1,GETDATE()),'yyyy-MM-dd')
```

4

```
 SELECT *
  FROM GOLD_RATES
  WHERE  DATEID =     FORMAT(DATEADD(YY,-1,GETDATE()),'yyyy-MM-dd')
```

5 display last 1 month gold rates ?

```
  2023-06-13   ?

  2023-07-13   ?
```

```
    SELECT *
    FROM GOLD_RATES
    WHERE DATEID BETWEEN
                          FORMAT(DATEADD(MM,-1,GETDATE()),'yyyy-MM-dd')
                          AND
                          FORMAT(GETDATE(),'yyyy-MM-dd')
```

EOMONTH() :-

=> returns last day of the month

```
                EOMONTH(DATE,INT)

  SELECT  EOMONTH(GETDATE(),0)   =>   2023-07-31
  SELECT  EOMONTH(GETDATE(),1)   =>   2023-08-31
  SELECT  EOMONTH(GETDATE(),-1)  =>  2023-06-30
```

=> display next month 1st day ?
=> display current month 1st day ?
=> display next year 1st day ?
=> display current year 1st day ?

STRING fuctions :-

UPPER() :-

=> converts string to uppercase

```
        UPPER(string)
```

ex :-

```
SELECT UPPER('hello')    =>   HELLO
```

LOWER() :-

=> converts string to lowercase

LOWER(string)

```
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)
```

14-jul-23

LEN() :-

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

```
LEN(string)
```

ex :-

```
SELECT LEN('hello welcome')    =>    13

SELECT EMPNO,ENAME,LEN(ENAME) AS LEN FROM EMP
```

=> display employees name contains 5 chars ?

```
SELECT *
FROM EMP
WHERE LEN(ENAME) = 5
```

LEFT() :-

=> returns character starting from left

```
LEFT(string,len)
```

```
SELECT  LEFT('hello welcome',5)     =>  hello
```

=> employees name starts with 's' ?

```
WHERE ENAME LIKE 's%'

SELECT * FROM EMP WHERE  LEFT(ENAME,1)  =  's'
```

=> generate emailids for employees ?

```
empno     ename           emailid
7369              smith                      smi736@tcs.com
7499              allen                      all749@tcs.com
```

```
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'
```

RIGHT() :-

=> returns character starting from right side

```
RIGHT(STRING,LEN)
```

```
SELECT RIGHT('hello welcome',7)    =>   welcome
```

=> employees name starts and ends with same char ?

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

SUBSTRING() :-

=> returns characters starting from specific position

```
      SUBSTRING(string,start,len)

   SELECT  SUBSTRING('hello welcome',7,4)     => welc
   SELECT  SUBSTRING('hello welcome',10,3)   =>  com
```

REPLICATE() :-

=> repeats character for given no of times

```
             REPLICATE(char,len)

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

 display   ENAME     SAL    ?
                              ****

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

        SMITH  ******
        ALLEN  *******
```

=>

```
    ACCOUNTS
    ACCNO                 PHONE
    123456789573          9876543292
```

1   your a/c no XXXX9573 debited -----

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

2   display phone as 98XXXXX892

```
    LEFT(PHONE,2) + REPLICATE('X',5) + RIGHT(PHONE,3)
```

REPLACE() :-

=> used to replace one string with another string.

```
          REPLACE(str1,str2,str3)
```

=> in str1 , str2 replaced with str3

```
   SELECT  REPLACE('hello','ell','abc')    =>     habco
   SELECT  REPLACE('hello','l','abc')       =>     heabcabco
   SELECT  REPLACE('hello','elo','abc')    =>    hello
   SELECT REPLACE('@@he@@ll@@o@@','@','')  => hello
```

TRANSLATE() :-

=> used to translate one char to another char

```
        TRANSLATE(str1,str2,str3)

 SELECT TRANSLATE('hello','elo','abc')   =>   habbc

                e  => a
                l   => b
                o  => c
```

NOTE :-

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

```
    SELECT  ENAME,
                TRANSLATE(SAL,'0123456789.' , '$kT*b^%&@#!') as SAL
     FROM EMP

      JONES  2975.00    T#&^!$$
```

15-jul-23

CHARINDEX() :-

=> returns position of a character in string.

```
          CHARINDEX(char , string,[start])

 ex :-

  SELECT CHARINDEX('O','HELLO WELCOME')              =>   5
  SELECT CHARINDEX('X','HELLO WELCOME')     =>   0
  SELECT CHARINDEX('O','HELLO WELCOME',6)              =>  11
  SELECT CHARINDEX('E','HELLO WELCOME',10)             =>  13
```

Assignment :-

```
 CUST
 CID    CNAME
 10      SACHIN TENDULKAR
 11      VIRAT KOHLI

=> display          CID  FNAME            LNAME          ?
              10    SACHIN           TENDULKAR

   using :- SUBSTRING , CHARINDEX
```

STUFF() :-

  => similar to replace used to replace a string based on start and length

```
                STUFF(string1,start,len,string2)

   SELECT STUFF('hello welcome',10,4,'abc')    =>   hello welabc
   SELECT STUFF('a,b,c,d,',8,1,'')                  =>   a,b,c,d
```

Numeric functions :-
----

rounding numbers :-

```
  ROUND
  FLOOR
  CEILING

   38.45678955   => 38
                       38.45
```

38.4567

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(38.4537,2)     => 38.45

   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
```

FLOOR() :-

=> always rounds number to lowest

```
         FLOOR(number)

 SELECT FLOOR(3.9)    =>   3
```

CEILING() :-

 => rounds number always to highest

```
        CEILING(number)

   SELECT CEILING(3.1)     =>  4
```

=> round employees salaries to hundreds ?

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

conversion :-

 => used to convert one datatype to another datatype.

   1 CAST
   2 CONVERT

CAST :-

```
       CAST(source-value  as target-type)
```

 EX :-

```
     SELECT   CAST(10.5 AS INT)               => 10
     SELECT   CAST(10  AS DECIMAL(5,3))           => 10.000
```

 => display smith earns 800
                  allen   earns 1600 ?

```
     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,SOURCE-VALUE)
```

```
SELECT CONVERT(INT,10.5)   =>  10
```

special functions :-

ISNULL() :-

=> used to convert null values

```
        ISNULL(arg1,arg2)
```

if arg1 = null returns arg2
if arg1 <> null returns arg1 only

```
 SELECT ISNULL(100,200)    =>  100
 SELECT ISNULL(NULL,200)   =>  200
```

```
display          ENAME          SAL      COMM        TOTSAL        ?
```

```
    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
```

17-JUL-23

Analytical Functions / Window Functions :-

RANK() & DENSE_RANK() :-

=> both functions are used to find ranks
=> ranks are based on some colum
=> for rank functions data must be sorted

```
RANK() OVER (ORDER BY COLNAME  ASC/DESC , ---------)
DENSE_RANK() OVER (ORDER BY COLNAME ASC/DESC,---)
```

Examples :-

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

```
 SELECT  empno,ename,sal,
              RANK() OVER (ORDER BY sal DESC) as rnk
 FROM emp

SELECT  empno,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 may no be 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  empno,ename,hiredate,sal,
        DENSE_RANK() OVER (ORDER BY sal DESC,hiredate ASC) as rnk
FROM emp

  king    1981-11-17    5000.00      1
  abc     2023-07-13    5000.00      2
  jones   1981-04-02    3000.00      3
  ford    1981-12-03    3000.00      4
  scott   1982-12-09    3000.00      5
  blake   1981-05-01    2900.00      6
```

=>

STUDENT

SNO	SNAME		M	P	C
1	A	80	90	70	
2	B	70	60	50	
3	C	90	70	80	
4	D	90	80	70	

=> find ranks of the students based on total desc, m desc,p desc ?

PARTITION BY clause :-

=> used to find ranks with in group , for ex to find ranks with in dept first divide the table dept wise and apply rank functions on each dept instead of applying it on whole table

```
SELECT  empno,ename,sal,deptno,
        dense_rank() over (partition by deptno
                           order by sal desc) as rnk
FROM emp

10
        5000        1
        2450        2
```

```
            1300            3

    20
             3000           1
            3000            1
            2975            2
            1100            3
            800             4
```

ROW_NUMBER() :-

=> returns record numbers based on some column
=> data must be sorted

```
    SELECT  empno,ename,sal,
                row_number() over (order by sal desc) as rnk
    FROM emp
```

```
    SAL     RNK    DRNK  RNO
    5000    1       1       1
    4000    2       2       2
    3000    3       3       3
    3000    3       3       4
    3000    3       3       5
    2000    6       4       6
    2000    6       4       7
    1000    8       5       8
```

Aggregate Functions / Multi-row functions :-

 => these functions process multiple rows and returns one value

MAX() :-

=> returns maximum value

```
   MAX(arg)
```

```
SELECT MAX(SAL) FROM EMP                  => 5000.00
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
```

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 )  AS TOTSAL
FROM EMP
```

O/P :- 29,000.00

=> calculate total sal including comm ?

```
SELECT  SUM(SAL+COMM) AS TOTSAL  FROM EMP  => 7900
```

```
     SAL                COMM          SAL+COMM
     5000               NULL          NULL
     4000               500           4500
     3000               NULL          NULL
```

```
   SUM(SAL)             =  12000
   SUM(SAL+COMM)              =   4500
```

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

```
     SAL                COMM          SAL+ISNULL(COMM,0)
     5000               NULL          5000
     4000               500           4500
     3000               NULL          3000
```

```
  SUM(SAL) = 12000
  SUM(SAL+ISNULL(COMM,0))  = 12500
```

AVG() :-

=> returns average value

```
        AVG(arg)
```

```
 SELECT  AVG(SAL)   FROM  EMP   => 2092.8571
```

=> round avg sal to highest integer

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

18-JUL-23

COUNT(*) :-

=> returns no of rows in a table.

```
   SELECT COUNT(*) FROM EMP
```

=> no of employees joined in 1981 year ?

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

=> no of employees joined on sunday ?

```
   SELECT COUNT(*)
  FROM EMP
  WHERE DATENAME(dw,HIREDATE) = 'SUNDAY'
```

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

```
  SELECT COUNT(*)
  FROM EMP
  WHERE DATEPART(YY,HIREDATE)=1981
              AND
             DATEPART(Q,HIREDATE) = 2
```

NOTE :-

=> 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
```

summary :-

DATE :-   datepart,datename,datediff,dateadd,format,eomonth

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

NUMERIC :- round,floor,ceiling

CONVERSION :- cast,convert

SPECIAL   :-   isnull

ANALYTICAL :-   rank,dense_rank,row_number

AGGREGATE :- max,min,sum,avg,count(*)

===

CASE statement :-

=> case statement is similar to switch case.
=> used to implement if-else in sql.
=> using case statement we can return values based on condition.
=> case statements are 2 types

1 simple case
2 searched case

simple case :-

```
 CASE COLNAME
 WHEN VALUE1 THEN RETURN EXPR1
 WHEN VALUE2 THEN RETURN EXPR2
 ---------------------
 ELSE RETURN EXPR
 END
```

=> DISPLAY   ENAME   JOB   ?

```
         IF  JOB=CLERK   DISPLAY  WORKER
                  MANAGER            BOSS
                  PRESIDENT           BIG BOSS
                  OTHERS              EXECUTIVE
```

```
   SELECT  ENAME,
              CASE JOB
              WHEN 'CLERK' THEN   'WORKER'
              WHEN 'MANAGER' THEN 'BOSS'
              WHEN 'PRESIDENT' THEN 'BIG BOSS'
              ELSE 'EXECUTIVE'
              END  AS JOB
  FROM EMP
```

=> increment employee salaries as follows ?

```
    IF  deptno = 10   incr  sal  by   10%
```

20	15%
30	20%
others	5%

```
UPDATE EMP
SET SAL =  CASE DEPTNO
             WHEN 10 THEN    SAL + (SAL*0.1)
             WHEN 20 THEN    SAL + (SAL*0.15)
             WHEN 30 THEN    SAL + (SAL*0.2)
             ELSE  SAL + (SAL*0.05)
             END
```

searched case :-

=> use searched case when conditions not based on "=" i.e. based on > < between operators

```
CASE
WHEN COND1 THEN RETURN EXPR1
WHEN COND2 THEN RETURN EXPR2
---------------------------
ELSE RETURN EXPR
END
```

=> display ENAME   SAL   SALRANGE   ?

```
IF  SAL > 3000  DISPLAY    HISAL
    SAL < 3000   DISPLAY   LOSAL
    SAL=3000                 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       50       60

  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
```

19-JUL-23

GROUP BY clause :-

=> GROUP BY clause groups rows based on one or more columns to calculate min,max,sum,avg,count for each group. For ex to calculate total sal paid to each dept first we need to group rows based on dept and apply sum(sal) function on each dept instead of applying on whole table.

```
EMP
EMPNO   ENAME  SAL   DEPTNO
 1          A        5000  10
 2          B       4000   20     GROUP BY     10       7000
 3          C       3000  30 ==================> 20       8000
 4          D       2000  10                     30       3000
 5          E       4000  20

detailed data                                   summarized data
```

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

syn :-

```
SELECT columns
FROM tabname
[WHERE cond]
GROUP BY col1,col2,---
[HAVING cond]
[ORDER BY colname ASC/DESC]
```

Execution :-

```
 FROM
 WHERE
 GROUP BY
 HAVING
 SELECT
 ORDER BY
```

=> display dept wise total salary ?

```
SELECT DEPTNO,SUM(SAL) AS TOTSAL
FROM EMP
GROUP BY DEPTNO

       10       8800.00
       20       10900.00
       30       9600.00
```

FROM EMP :-

```
EMP
EMPNO   ENAME  SAL   DEPTNO
1              A            5000  10
2              B           4000   20
3             C            3000  30
4             D            2000  10
5             E            4000  20
```

GROUP BY DEPTNO :-

```
10    1         A         5000
      4         D         2000

20    2         B         4000
      5         E         4000

30    3         C         3000
```

SELECT DEPTNO,SUM(SAL) AS TOTSAL :-

```
-------------------------------------------------------

10      7000
20      8000
30      3000
```

=> display job wise no of employees ?

```
SELECT  JOB,COUNT(*) AS CNT
FROM EMP
GROUP BY JOB
```

=> display year wise no of employees joined ?

```
SELECT DATEPART(YY,HIREDATE) AS YEAR,COUNT(*) AS CNT
FROM EMP
GROUP BY DATEPART(YY,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 month wise no of employees joined in 1981 year ?

```
SELECT DATENAME(MM,HIREDATE) AS MONTH,COUNT(*) AS CNT
FROM EMP
WHERE DATEPART(YY,HIREDATE)=1981
GROUP BY DATENAME(MM,HIREDATE)
```

=> find the departments having more than 3 employees ?

```
SELECT DEPTNO,COUNT(*) AS CNT
FROM EMP
WHERE COUNT(*) > 3
GROUP BY DEPTNO             => ERROR
```

sql server cannot 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(*) AS CNT
FROM EMP
```

```
    GROUP BY DEPTNO
    HAVING COUNT(*) > 3
```

WHERE VS HAVING :-

	WHERE	HAVING
1	selects specific rows	selects specific groups
2	conditions executed before group by	conditions executed after group by
3	use where clause if cond doesn't contain aggregate function	use having clause if cond contains aggregate function

=> find southern states having more than 5CR population ?

```
  PERSONS
  AADHARNO    NAME    GENDER    AGE   ADDR   CITY   STATE

  SELECT  STATE,COUNT(*)
  FROM PERSONS
  WHERE  STATE IN ( 'AP','TS','KA','KL','TN')
  GROUP BY STATE
  HAVING  COUNT(*) > 50000000
```

20-jul-23

=> display dept wise total salaries where deptno = 10,20 and sum(sal) > 10000 ?

```
        select deptno,sum(sal)
        from emp
        where deptno in (10,20)
        group by deptno
        having sum(sal) > 10000
```

Grouping based on multiple columns :-

=> display dept wise and with in dept job wise no of employees ?

```
         SELECT deptno,job,COUNT(*) as cnt
         FROM emp
```

```
        GROUP BY deptno,job
        ORDER BY deptno ASC

        10      CLERK                1
                MANAGER     1
                PRESIDENT   1

        20      ANALYST              2
                CLERK                2
        `       MANAGER     1

        30      CLERK                1
                MANAGER     2
                SALESMAN    4
```

=>

```
  PERSONS
  AADHARNO    NAME    GENDER    AGE   ADDR   CITY   STATE
```

display state wise and with in state gender wise population ?

```
  SELECT  STATE,GENDER,COUNT(*) AS CNT
  FROM EMP
 GROUP BY STATE,GENDER
 ORDER BY STATE ASC

     AP      MALE         ?
            FEMALE      ?

    AR      MALE        ?
            FEMALE     ?
```

=> display duplicate records ?

```
  EMP11
  ENO          ENAME        SAL
  1    A       5000
  2    B       6000
  1    A       5000
  2    B       6000
  3    C       4000
```

```
SELECT  ENO,ENAME,SAL
FROM  EMP11
GROUP BY ENO,ENAME,SAL
HAVING COUNT(*) > 1

      1         A         5000
      2         B         6000
```

==

INTEGRITY CONSTRAINTS

=> Integrity Constraints are rules to maintain Data Quality.
=> used to prevent users from entering invalid data.
=> used to enforce rules like min bal must be 1000.
=> different integrity constraints in sql server

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

NOT NULL :-

=> NOT NULL constraint doesn't accept null values.
=> a column declared with NOT NULL is called mandatory column.

ex :-

```
 CREATE TABLE EMP15
 (
     ENO  INT,
     ENAME   VARCHAR(10)  NOT NULL
 )

INSERT INTO EMP15 VALUES(1,NULL)  => ERROR
INSERT INTO EMP15 VALUES(2,'B')
```

UNIQUE :-

=> unique constraint doesn't accept duplicates

ex :-

```
 CREATE TABLE CUST
 (
  CID   INT ,
  CNAME  VARCHAR(10),
  EMAILID    VARCHAR(20)  UNIQUE
)


INSERT INTO CUST VALUES(10,'A','abc@gmail.com')
INSERT INTO CUST VALUES(11,'B','abc@gmail.com')   => ERROR
INSERT INTO CUST VALUES(12,'C',NULL)
INSERT INTO CUST VALUES(13,'D',NULL)                 => ERROR
```

PRIMARY KEY :-

=> primary key doesn't accept duplicates and nulls.
=> it is combination of unique & not null.
=> in tables one column must be there to uniquely identify the records and that column must be declared with primary key.

ex :-

```
     CREATE TABLE EMP16
     (
        EMPID   INT  PRIMARY KEY,
        ENAME  VARCHAR(10)  NOT NULL
     )
```

```
   INSERT INTO EMP16 VALUES(100,'A')
   INSERT INTO EMP16 VALUES(100,'B')     => ERROR
   INSERT INTO EMP16 VALUES(NULL,'A')  => ERROR
```

=> 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.

```
 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 UNIQUE & PRIMARY KEY ?

	UNIQUE	PRIMARY KEY
1	allows one null	doesn't allow null
2	multiple columns can be declared with unique	only one column can be declared with primary key

candidate key :-

=> a field eligible for primary key is called candidate key

ex :-

```
      VEHICLE
      VEHNO    VNAME     MODEL     COST     CHASSISNO

       candidate keys :-    VEHNO,CHASSISNO
       primary key      :-    VEHNO
       secondary key  :-    CHASSISNO
       or
       alternate key
```

=> while creating table secondary keys are declared with UNIQUE NOT NULL.