# ORACLE

## Database :-

=> a database is a organized collection of interrelated data.Oracle is a relational database product which is used to store data permanently in secondary storage.

**Date**:- it is a collection of raw Facts.

**Information**:- it is a collection of meaningful data or processed data.

**Storage**:- It is a place where we can store data or information.

We have few types of Storages like

1.Books and papers 2. Files system 3.Database

Files:- This is traditional way to store data in individual files but there are lot of drawback in it like.

**Data Redundancy**:- sometimes we are maintaining multiple copies of same data in different locations then data duplication occurs. This duplication is called data redundancy.

->if we modify same data in one location then it will not modify in another location so data inconsistency may occurs.

->files system does not follow ACID property.

**Data Security**:- Data stored in files cannot be secured because files doesn't provide any security mechanism where as data base provide role based mechanism.

Like wise there are lot of drawback because of which file system is not used for data storage for applications.

**DBMS :- (Database Management System)**

=> DBMS is a software used to create and to manage database.

=> DBMS is an interface between user and database.

**Evolution of DBMS :-**

---

1960 FMS (File Management system)

1970 HDBMS (Hierarchical dbms)

NDBMS (Network dbms)

1980 RDBMS (Relational dbms)

1990 ORDBMS (Object relational dbms)

**RDBMS :-**

=> RDBMS concepts introduced by E.F.CODD

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

## RDBMS softwares :-

ORACLE from oracle corp

SQL SERVER from microsoft

DB2 from IBM

MYSQL from oracle corp

POSTGRESQL from postgresql forum dev

RDS from amazon

**ORDBMS :-**

=> ORDBMS is combination of RDBMS & OOPS

ORDBMS = RDBMS + OOPS (reusability)

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

ORACLE versions :-

---

| Version | Year | New Features |
|---|---|---|
| Oracle v2 | 1979 | First commercially SQL-based RDBMS |
| Oracle v3 | 1983 | Concurrency control, data distribution, scalability |
| Oracle v4 | 1984 | Multiversion read consistency |
| Oracle v5 | 1985 | Client/server computing Support & distributed database systems |
| Oracle v6 | 1988 | Row-level locking, scalability, online backup and recovery, PL/SQL, Oracle Parallel Server |
| Oracle 7 | 1992 | PL/SQL stored procedures, Triggers, Shared Cursors, Cost Based Optimizer, Transparent Application Failover |
| Oracle 8i | 1997 | Recovery Manager, Partitioning, Dataguard, Native internet protocols, Java, Virtual Private Database |
| Oracle 9i | 2001 | Oracle Real Application Clusters (RAC), Oracle XML DB, Data Mining, Streams, Logical Standby |

| | | |
|---|---|---|
| Oracle 10gR1 | 2003 | Grid infrastructure, Oracle ASM, Flashback Database, Automatic Database Diagnostic Monitor |
| Oracle 10gR2 | 2005 | Real Application Testing, Database Vault, Online Indexing, Advanced Compression, Transparent Data Encryption |
| Oracle 11gR1 | 2007 | Active Data Guard, Secure Files, Exadata |
| Oracle 11gR2 | 2009 | Data Redaction, Hybrid Columnar Compression, Cluster File System, Golden Gate Replication, Database Appliance |
| Oracle 12cR1 | 2013 | Multitenant architecture, In-Memory Column Store, Native JSON, SQL Pattern Matching, Database Cloud Service |
| Oracle 12cR2 | 2016 | Native Sharding, Zero Data Loss Recovery Appliance, Exadata Cloud Service, Cloud at Customer |
| Oracle 18c | 2018 | Autonomous Database, Data Guard Multi-Instance Redo Apply, Polymorphic Table Functions, Active Directory Integration |

**CLIENT/SERVER ARCHITECTURE :-**

1 SERVER

2 CLIENT

=> SERVER is a system where oracle software is installed and running

=> inside the server using oracle we can create database and we can manage database.

=> CLIENT is a system where users can

1 connects to server

2 submit requests to server

3 recieves reponse from server

**client tools :-**

CUI => character user interface

GUI => graphical user interface

SQLPLUS (CUI based)

SQL DEVELOPER (GUI based)

TOAD (GUI based)

## SQL

=> SQL stands for structured query language

=> Language used to communicate with oracle

=> user communicates with oracle by sending commands called queries

=> a query is a command/instruction submitted to oracle to perform some operation over db

=> SQL introduced by IBM and initial name of this langauge was SEQUEL and later it is renamed to SQL.

=> SQL is common to all rdbms.

| ORACLE | SQL SERVER | MYSQL | DB2 | POSTGRESQL |
|---|---|---|---|---|
| SQL | SQL | SQL | SQL | SQL |

=> based on operations over database SQL is categorized into following sublanguages

1 DDL (Data Definition Language)

2 DML (Data Manipulation Language)

3 DRL/DQL (Data Retrieval Language / Data Query Language)

4 TCL (Transaction Control Language)

5 DCL (Data Control Language)

## How to connect to oracle :-

=> to connect to oracle open sqlplus and enter username & password

**USERNAME** :- SYSTEM

**PASSWORD** :- manager

OR

USERNAME :- SYSTEM/manager

From current user we can switch to another user by

**>connect username/password**

**Tablespace:-** Oracle database is nothing but collection of datafiles physically or collection of tablespaces logically. Whenever we are installing Oracle then automatically 5 tablespaces are created if you want to view that then we can use “user_tablespaces”.

```
SQL> select tablespace_name from user_tablespaces;

TABLESPACE_NAME
______________________
SYSTEM
SYSAUX
UNDOTBS1
TEMP
USERS
```

**->In Oracle  all users information stored under dba_users datadictonary.**

```
SQL> select username,default_tablespace from dba_users;

USERNAME     DEFAULT_TABLESPACE
____________ ____________________
SYS          SYSTEM
SYSTEM       SYSTEM
XS$NULL      SYSTEM
OJVMSYS      SYSTEM
LBACSYS      SYSTEM
OUTLN        SYSTEM
SYS$UMF      SYSTEM
DBSNMP       SYSAUX
APPQOSSYS    SYSAUX
DBSFWUSER    SYSAUX
```

->in oracle data stored in different datafiles. And tablespace is nothing  but collection of datafiles .one datafiles belongs to one tablespace only.

**->if you want to create tablespace the we can use command :-**

Command:- create tablespace tablespacename datafile ‘path’ size  sizeNo

```
SQL> create tablespace mytab datafile 'myfiles.dbf' size 50m;

Tablespace MYTAB created.
```

```
SQL> select tablespace_name from user_tablespaces;

TABLESPACE_NAME
______________________
SYSTEM
SYSAUX
UNDOTBS1
TEMP
USERS
ABC
MYTAB
```

## Creating User in Oracle database syntax :-

CREATE USER <NAME> IDENTIFIED BY <PWD>

DEFAULT TABLESPACE USERS

QUOTA UNLIMITED ON USERS ;

```
SQL> create user c##myoffice identified by root
  2  default tablespace users
  3* quota unlimited on users;

User C##MYOFFICE created.

SQL>
```

=> a new user created in database but the user is dummy because user is not having permissions to connect to db and to create objects like tables.

## Grant permission

GRANT CONNECT,RESOURCE TO C##myoffice;

```
SQL> create user c##myoffice identified by root
  2  default tablespace users
  3* quota unlimited on users;

User C##MYOFFICE created.

SQL> grant connect,resource to c##myoffice;

Grant succeeded.
```

->now switch to new user

```
SQL> connect c##myoffice/root;
Connected.
SQL>
```

**To see current working user**

```
SQL> show user;
USER is "C##MYOFFICE"
SQL>
```

**Datatypes in ORACLE :-**

**DATATYPES**

| CHAR | | NUMERIC | DATE | BINARY |
|---|---|---|---|---|
| ASCII | UNICODE | number(p) | date | bfile |
| char | nchar | number(p,s) | timestamp | blob |
| varchar2 | nvarchar2 | | | |
| long | nclob | | | |
| clob | | | | |

## char(size) :-

=> allows character data upto 2000 chars.

=> recommended for fixed length character fields.

ex :- NAME CHAR(10)

=> in char datatype extra bytes are wasted so char is not recommended for variable length fields and char recommended for fixed length fields.

**VARCHAR2 :-**

=> allows character data upto 4000 characters.

=> recommended for variable length fields.

=> in varchar2 datatype extra bytes are released

**LONG :-**

=> allows character data upto 2GB.

=> using LONG datatype we can store large amount of text in db

ex :- history LONG

**CLOB :- (Character Large Object)**

=> allows character data upto 4GB.

=> using CLOB datatype also we can store large amount of text in db

ex :- TEXT CLOB

NOTE :- CHAR/VARCHAR2/LONG/CLOB allows ascii characters (256 chars) that includes

a-z,A-Z,0-9 and special characters, so all these datatype allows alphanumeric data.

**NCHAR/NVARCHAR2/NCLOB :- (N => National)**

=> these types allows unicode characters (65536 chars) that includes characters of different languages.

=> a unicode character occupies 2 bytes

**NUMBER(P) :-**

=> allows numeric data upto 38 digits => allows whole numbers i.e. numbers without decimal part (integers).

Ex: PHONE NUMBER(10)

**NUMBER(P,S) :-**

=> allows real numbers i.e. numbers contains decimal part (float)

P => precision => total no of digits allowed

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

ex :- SALARY NUMBER(7,2)

5000

5000.50

50000.50

500000.50 => NOT ACCEPTED

5000.507 => ACCEPTED => 5000.51

5000.503 => ACCEPTED => 5000.50

**DATE :-**

=> DATE datatype allows date & time

=> time is optional, if not entered oracle stores 00:00:00 (12AM).

=> default date format in oracle is dd-mon-yy / yyyy

ex :- DOB DATE

18-AUG-21 => 18 08 2021 00:00:00

10-OCT-95 => 10 10 2095 00:00:00

10-OCT-1995 => 10 10 1995 00:00:00

TIMESTAMP :-

=> timestamp allows date,time and also milliseconds

ex :- T TIMESTAMP

18-AUG-21 14:34:20.123

**=> diff b/w DATE & TIMESTAMP ?**

1 DATE allows only date,time but TIMESTAMP allows date,time and also milliseconds.

2 in DATE datatype we use TO_DATE function to insert date & time but in TIMESTAMP without function we can insert date & time.

**BINARY TYPES :-**

=> binary types allows binary data that includes audio,video,images i.e. multimedia objects

=> oracle supports 2 binary types

1 BFILE (Binary File Large Object)

2 BLOB (Binary Large Object)

**BFILE :-**

=> BFILE is called external lob becuase lob stored outside db but db stores path

**BLOB :-**

=> BLOB is called internal lob because lob stored inside db.

=> LOB can be upto 4GB.

**CREATING TABLES IN ORACLE DB :-**

CREATE TABLE <tabname>

(

COLNAME DATATYPE(size),

COLNAME DATATYPE(size),

----------);

```
SQL> create table hospital(drid number(4),drname varchar2(50),drjoindate date,drsal number(7,2));

Table HOSPITAL created.

SQL>
```

->to see details of table use **desc tablename**

```
SQL> desc hospital;

Name          Null?    Type
_____________ ________ _______________
DRID                   NUMBER(4)
DRNAME                 VARCHAR2(50)
DRJOINDATE             DATE
DRSAL                  NUMBER(7,2)
SQL>
```

**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 upto 1000 columns

5 table can have unlimited rows

emp123 valid

emp 123 invalid

emp*123 invalid

emp_123 valid

**INSERTING DATA INTO TABLE :-**

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

=> INSERT command creates a new row in table

=> INSERT command can be used insert

1  single row

2  multiple rows

**INSERTING SINGLE ROW :-**

syn :- INSERT INTO <TABNAME> VALUES(V1,V2,V3,-----);

```
SQL> insert into hospital (drid,drname,drjoindate,drsal) values (1111,'dr.rakesh','25-jan-2016',25647.89);

1 row inserted.

SQL> insert into hospital values (222,'dr.mukesh','25-jan-2016',25647.89);

1 row inserted.
```

**INSERTING MULTIPLE ROWS :-**

=> insert command can be executed multiple times with different values using  variables prefixed with "&" and use forward slash for new row.

```
SQL> insert into hospital values(&drid,&drname,&drjoindate,&drsal);
Enter value for drid: 333
Enter value for drname: 'dr.sukesh'
Enter value for drjoindate: '26-jan-2024'
Enter value for drsal: 47582.15
old:insert into hospital values(&drid,&drname,&drjoindate,&drsal)
new:insert into hospital values(333,'dr.sukesh','26-jan-2024',47582.15)

1 row inserted.

SQL> /
Enter value for drid:
```

**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 :- (explicit)

SQL>INSERT INTO hospital VALUES(104,'ravi','',NULL,sysdate,30);

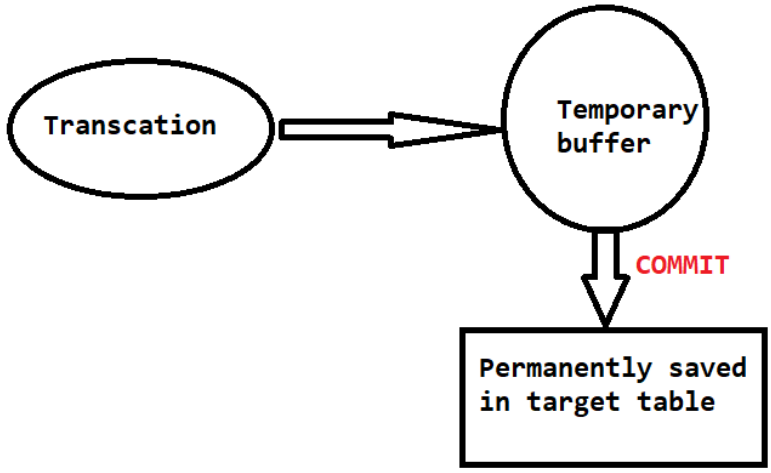
method 2 :- (implicit)

SQL>INSERT INTO hospital(drid,drname,drjoindate) VALUES(105,'abhi','10-OCT-19');

=> remaining fields filled with nulls.

**Commit:-**

->to permanently store data then use commit command after insert.




## Displaying Data /retrieving data

**->"select"** command is used to display data from table and by using select command we can display all the records(rows) or specific records , by using select command we can display all the columns or specific columns .

->in this select command we use some clauses(keywords)

FROM :- it is a clause which specify the table from which we are retrieving data.

SELECT:-it is a clause which used to specify about columns in that table like all columns or some particular columns.

```
1 => select colname,colname.. from tablename

to see all columns use (*)

=>select * from tablename
```

=>display drname ,drsal from hospital table ?

```
SQL> select drname,drsal from hospital;

DRNAME              DRSAL
_____________ _____________
dr.mukesh          25647.89
dr.rakesh          25647.89
```

=>display all data from hospital?

```
SQL> select * from hospital;

   DRID DRNAME       DRJOINDATE          DRSAL
_______ _____________ _____________ _____________
    222 dr.mukesh     25/JAN/16          25647.89
   1111 dr.rakesh     25/JAN/16          25647.89
```

# Operators

1.Arithmetic Operators : + , - , * , /

2.Relational Operators : = , > ,< , >= ,<= , <>

3.Logical Operator: AND ,OR ,NOT

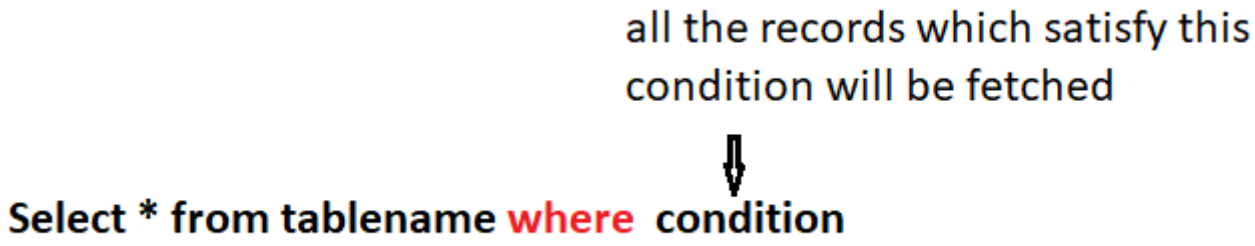
4.special Operator: Between , In ,Like,Is,All

5.Set Operator: Union , Union All ,Intersect,Minus

| | |
|---|---|
| **1.Arithmetic Operators :** | **+ , - , * , /** |
| **2.Relational Operators :** | **= , > ,< , >= ,<= , <>** |
| **3.Logical Operator:** | **AND ,OR ,NOT** |
| **4.special Operator:** | **Between , In ,Like,Is,All** |
| **5.Set Operator:** | **Union , Union All ,Intersect,Minus** |

## Where Clause

->where clause is used to get specific row/rows from table based on a condition.




About condition:- to give condition we use operators and operators must be any relational operator like > ,< ,>= ,<= ,= ,<>(alternate of !=)

If condtion is true then row is selected.

If condition is false row is not selected.

Ex:

```
SQL> select * from hospital where drname='dr.mukesh';

   DRID DRNAME       DRJOINDATE          DRSAL
_______ ____________ _____________ ____________
    222 dr.mukesh    25/JAN/16        25647.89

SQL> select * from hospital where drid=1111;

   DRID DRNAME       DRJOINDATE          DRSAL
_______ ____________ _____________ ____________
   1111 dr.rakesh    25/JAN/16        25647.89
```

**Compound condition**

->to use multiple condition we use **AND/OR** operators

->in 'AND' both the condition needs to be satisfied.

->in 'OR' atleast one condition needs to be satisfied.

| condition1 | AND | condition2 | Result |
|---|---|---|---|
| true | | true | true |
| true | | false | false |
| false | | true | false |
| false | | false | false |

| condition1 | OR | condition2 | Result |
|---|---|---|---|
| true | | true | true |
| true | | false | true |
| false | | true | true |
| false | | false | false |

```
SQL> select * from hospital;

   DRID DRNAME       DRJOINDATE          DRSAL
_______ ___________ _____________ ____________
    222 dr.mukesh    25/JAN/16        25647.89
   1111 dr.rakesh    25/JAN/16        25647.89
    333 dr.ravi      21/OCT/20        78000.23
   4444 dr.deepak    11/FEB/11         54000.3

SQL> select * from hospital where drname='dr.ravi' and drsal>54000;

   DRID DRNAME     DRJOINDATE          DRSAL
_______ _________ _____________ ____________
    333 dr.ravi    21/OCT/20        78000.23

SQL> select * from hospital where drname='dr.ravi' or drsal>25000;

   DRID DRNAME       DRJOINDATE          DRSAL
_______ ___________ _____________ ____________
    222 dr.mukesh    25/JAN/16        25647.89
   1111 dr.rakesh    25/JAN/16        25647.89
    333 dr.ravi      21/OCT/20        78000.23
   4444 dr.deepak    11/FEB/11         54000.3
```

## In operator

->we use ‘IN’ operator for list comparasion i.e “=” comparision with multiple values.

=> **where colname in(value1,value2,value3…..)**

(Here internal meaning is **where colname=value1 or colname=value2 or colname=value3…..**)

->see if you want to compare name with 1 name then you can also use like select * from tablename where name=”value”;

But if you want to compare name with so many names then it is better to go with ‘in’ operator.

```
SQL> select * from hospital where drname in ('dr.mukesh','dr.rakesh','dr.ravi');

   DRID DRNAME       DRJOINDATE          DRSAL
_______ _____________ _____________ ____________
    222 dr.mukesh     25/JAN/16        25647.89
   1111 dr.rakesh     25/JAN/16        25647.89
    333 dr.ravi       21/OCT/20        78000.23
```

## Not in

->it is used to select that rows which is not satisfy then condition

Ex:- fetch the list of doctor whose name is not mukesh,rakesh ,ravi?

```
SQL> select * from hospital where drname not  in ('dr.mukesh','dr.rakesh','dr.ravi');

   DRID DRNAME       DRJOINDATE          DRSAL
_______ _____________ _____________ ____________
   4444 dr.deepak     11/FEB/11         54000.3
```

## Between Operator

=>use ‘between’  operator  when need to perform comparison with range

Ex:- where colname between v1 and v2 (means where col >=v1 and col<=v2)

Display list of doctors earning between 50,000 and 80,000?

```
SQL> select * from hospital where drsal between 50000 and 80000;

   DRID DRNAME       DRJOINDATE          DRSAL
_______ _____________ _____________ ____________
    333 dr.ravi       21/OCT/20        78000.23
   4444 dr.deepak     11/FEB/11         54000.3
```

Display list of doctor joined in 2020 ?

```
SQL> select * from hospital where drjoindate between '1-jan-2020' and '31-dec-2020';

   DRID DRNAME     DRJOINDATE          DRSAL
_______ _________ _____________ _____________
    333 dr.ravi    21/OCT/20         78000.23
```

## NOT between

->it is used when we want record which is not come in that range

```
SQL> select * from hospital where drjoindate not between '1-jan-2020' and '31-dec-2020';

   DRID DRNAME       DRJOINDATE          DRSAL
_______ ___________ _____________ _____________
    222 dr.mukesh    25/JAN/16         25647.89
   1111 dr.rakesh    25/JAN/16         25647.89
   4444 dr.deepak    11/FEB/11          54000.3
```

## Like Operator

->we use like operator when we need to compare with pattern.

Where colname like 'pattern'

Where colname not like 'pattern'

->PATTERN consists of alphabets , digits,wildcard characters

Wildcard characters

**%** => 0 or many character

**_(**underscore) => exactly 1 char

% =>if any symbol present before '%' then word should start with that symbol

Ex: '**s%**' word should start with '**s'**

'**ss%**' word should start with '**ss'**

'**sss%**' word should start with '**sss'**

Similarly if any symbol present after '%' then word should end with that symbol.

Ex:- '**%s'** word should ends with '**s'**

'**%ss'** word should ends with '**ss'**

'**%sss'** word should ends with '**sss'**

Similarly if any symbol present between this % then word should contains that symbol.

**Ex: - '%s%'** word should contain **'s'**

```
SQL> select * from hospital where drname like '%h';

   DRID DRNAME       DRJOINDATE          DRSAL
_______ ____________ _____________ ____________
    222 dr.mukesh    25/JAN/16        25647.89
   1111 dr.rakesh    25/JAN/16        25647.89

SQL> select * from hospital where drname like '%p%';

   DRID DRNAME       DRJOINDATE          DRSAL
_______ ____________ _____________ ____________
   4444 dr.deepak    11/FEB/11         54000.3

SQL> select * from hospital where drname like 'ra%';

no rows selected
```

Similarly

'_' -> this single underscore represent one symbol only.

'__'->this double underscore represent two symbol.

'___'->this triple underscore represent three symbol..etc.

Q)display list of doctor where 'a' is the 4$^{th}$ character in their name?

```
SQL> select * from hospital where drname like '____a%';

   DRID DRNAME          DRBIO           DRSAL DRJOINDATE
_______ _______________ _________ ____________ _____________
   1111 dr.rakesh       surgen       45678.96 14/FEB/20
   4444 dr.manoj        genral             475 14/NOV/96
   1111 dr.santosh      kid            78965.41 27/JAN/24
```

Q)display doctor list where ‘a’ is the 3rd character from last in their name?

```
SQL> select * from hospital where drname like '%a__';

   DRID DRNAME          DRBIO           DRSAL DRJOINDATE
_______ _______________ _________ ____________ _____________
    333 dr.prakash      heart        45678.93 01/FEB/12
```

Q)display doctor joined in oct month?

```
SQL> select * from hospital where drjoindate like '%OCT%';

   DRID DRNAME     DRBIO          DRSAL DRJOINDATE
_______ _________ _______ ____________ _____________
   2222 dr.abhi   skin        45687.92 12/OCT/16
```

Q)display doctor joined in 2020 year?

```
SQL> select * from hospital where drjoindate like '%20';

   DRID DRNAME       DRBIO           DRSAL DRJOINDATE
_______ _____________ _________ ____________ _____________
   1111 dr.rakesh    surgen       45678.96 14/FEB/20
```

## IS operator

->’is’ operator is used when we make comparision with NULL/NOT NULL.

Where colname is null

Where colname is not null

Q)display doctor who not get salary ?

```
SQL> select * from hospital where drsal is null;

   DRID DRNAME       DRBIO       DRSAL DRJOINDATE
_______ _____________ _________ _______ _____________
   6666 dr.pratek    eye
   6666 dr.pratek                      19/MAY/18
```

Q)display doctors who got salary ?

```
SQL> select * from hospital where drsal is not null;

   DRID DRNAME       DRBIO           DRSAL DRJOINDATE
_______ _____________ _____________ _______ _____________
   1111 dr.rakesh    surgen        45678.96 14/FEB/20
   2222 dr.abhi      skin          45687.92 12/OCT/16
    333 dr.prakash   heart         45678.93 01/FEB/12
   4444 dr.manoj     genral              475 14/NOV/96
   1111 dr.santosh   kid           78965.41 27/JAN/24
   5555 dr.abhi      dentist       78965.41 27/JAN/24
```

## ALIAS

->alias means another name or alternative name.

->alias are used to change column heading.

->we use ‘AS’ to use with alias.

Syn:- columnname as aliasname

```
SQL> select drname as doctorji from hospital ;

DOCTORJI
_____________
dr.rakesh
dr.abhi
dr.prakash
dr.manoj
dr.santosh
dr.abhi
dr.pratek
dr.pratek

8 rows selected.
```

Q)display drname and annual salary ?

```
SQL> select drname ,drsal*12  from hospital;

DRNAME          DRSAL*12
_____________ ____________
dr.rakesh       548147.52
dr.abhi         548255.04
dr.prakash      548147.16
dr.manoj             5700
dr.santosh      947584.92
dr.abhi         947584.92
dr.pratek
dr.pratek
```

Q)display doctor name and experience?

```
SQL> select drname ,(sysdate-drjoindate)/365 as exp from hospital;

DRNAME                                                        EXP
_____________ ____________________________________________________
dr.rakesh          3.95823449391171993911719939117199391172
dr.abhi            7.30070024733637747336377473363774733638
dr.prakash        11.99933038432267884322678843226788432268
dr.manoj          27.22398791856925418569254185692541856925
dr.santosh      0.00480983637747336377473363774733637747337
dr.abhi         0.00480983637747336377473363774733637747337
dr.pratek
dr.pratek          5.70070024733637747336377473363774733638

8 rows selected.
```

**‘AS’ command is use to create table from exisiting table**

->as we use create command to create table similarly AS statement is used to create a table from an exisiting table by copying the existing table’s columns.

Ex:- create table tablename  as select * from existingtable;

```
SQL> create table cityhospital as select * from hospital;

Table CITYHOSPITAL created.

SQL> select * from cityhospital;

   DRID DRNAME          DRBIO           DRSAL DRJOINDATE
_______ _______________ _________ _________ _____________
   1111 dr.rakesh       surgen      45678.96 14/FEB/20
   2222 dr.abhi         skin        45687.92 12/OCT/16
    333 dr.prakash      heart       45678.93 01/FEB/12
   4444 dr.manoj        genral            475 14/NOV/96
   1111 dr.santosh      kid         78965.41 27/JAN/24
   5555 dr.abhi         dentist     78965.41 27/JAN/24
   6666 dr.pratek       eye
   6666 dr.pratek                             19/MAY/18

8 rows selected.
```

## ORDER by clause

->to display information in a particular order(ascending or descending order)

Syn:-select <columnames> from table  where <condition> ORDER BY <column>  [ASC/DESC]

->default order is ascending but for descending order use DESC option.

Q)arrange doctor list name in ascending order ?

```
SQL> select * from cityhospital order by drname asc;

   DRID DRNAME          DRBIO           DRSAL DRJOINDATE
_______ _______________ _________ _________ _____________
   5555 dr.abhi         dentist     78965.41 27/JAN/24
   2222 dr.abhi         skin        45687.92 12/OCT/16
   4444 dr.manoj        genral            475 14/NOV/96
    333 dr.prakash      heart       45678.93 01/FEB/12
   6666 dr.pratek                             19/MAY/18
   6666 dr.pratek       eye
   1111 dr.rakesh       surgen      45678.96 14/FEB/20
   1111 dr.santosh      kid         78965.41 27/JAN/24

8 rows selected.
```

Q)arrange doctor list who get lowest salary ?

```
SQL> select * from cityhospital order by drsal asc;

   DRID DRNAME          DRBIO              DRSAL DRJOINDATE
_______ _______________ _____________ _________ _____________
   4444 dr.manoj        genral              475 14/NOV/96
    333 dr.prakash      heart          45678.93 01/FEB/12
   1111 dr.rakesh       surgen         45678.96 14/FEB/20
   2222 dr.abhi         skin           45687.92 12/OCT/16
   1111 dr.santosh      kid            78965.41 27/JAN/24
   5555 dr.abhi         dentist        78965.41 27/JAN/24
   6666 dr.pratek                              19/MAY/18
   6666 dr.pratek       eye

8 rows selected.
```

Q)arrange doctor list who get highest salary ?

```
SQL> select * from cityhospital order by drsal desc;

   DRID DRNAME          DRBIO              DRSAL DRJOINDATE
_______ _______________ _____________ _________ _____________
   6666 dr.pratek                              19/MAY/18
   6666 dr.pratek       eye
   1111 dr.santosh      kid            78965.41 27/JAN/24
   5555 dr.abhi         dentist        78965.41 27/JAN/24
   2222 dr.abhi         skin           45687.92 12/OCT/16
   1111 dr.rakesh       surgen         45678.96 14/FEB/20
    333 dr.prakash      heart          45678.93 01/FEB/12
   4444 dr.manoj        genral              475 14/NOV/96

8 rows selected.
```

Note:- in ORDER BY clause we can use either column name or column number.

->if we use order by clause using column number then column number is not according to table it should be according to selected column list.

Ex:-select drname,ename from hospital order by 4 asc;

=>error **(because we have selected 2 columns so we can give only 2 number)**

```
SQL> select drname ,drbio,drsal from hospital order by 4;

Error starting at line : 1 in command -
```

```
SQL> select drname ,drbio,drsal from hospital order by 2 desc;

DRNAME          DRBIO             DRSAL
_______________ _____________ _____________
dr.pratek
dr.rakesh       surgen          45678.96
dr.abhi         skin            45687.92
dr.santosh      kid             78965.41
dr.prakash      heart           45678.93
dr.manoj        genral               475
dr.pratek       eye
dr.abhi         dentist         78965.41
```

## DML (data manipulation language)

->data manipulation means performing operations on the data in the tables of the database .

->DML commands acts on table data.

->DML commands acts on instance (RAM)

->to save these operation permanently execute commit.

->to cancel these operations execute rollback

**UPDATE command:-**

->command used to modify the data in a table.

->using update command we can modify all rows or specific row

->using update command we can modify single column or multiple columns

Syn:- update tablename set colname=value ,colname=value….where condition.

Q)update doctor salary with 500 whose sal=null?

```
SQL> update cityhospital set drsal=52000 where drsal is null;

2 rows updated.

SQL> select * from cityhospital;

   DRID DRNAME          DRBIO           DRSAL DRJOINDATE
_______ _______________ _________ ___________ _____________
   1111 dr.rakesh       surgen       45678.96 14/FEB/20
   2222 dr.abhi         skin         45687.92 12/OCT/16
    333 dr.prakash      heart        45678.93 01/FEB/12
   4444 dr.manoj        genral             475 14/NOV/96
   1111 dr.santosh      kid          78965.41 27/JAN/24
   5555 dr.abhi         dentist      78965.41 27/JAN/24
   6666 dr.pratek       eye             52000
   6666 dr.pratek                       52000 19/MAY/18

8 rows selected.
```

Q)increase dr.rakesh salary by 10% ?

```
SQL> update cityhospital set drsal=drsal+(drsal*0.1) where drname='dr.rakesh';

1 row updated.

SQL> select * from cityhospital;

   DRID DRNAME          DRBIO           DRSAL DRJOINDATE
_______ _______________ _________ ___________ _____________
   1111 dr.rakesh       surgen       50246.86 14/FEB/20
   2222 dr.abhi         skin         45687.92 12/OCT/16
    333 dr.prakash      heart        45678.93 01/FEB/12
   4444 dr.manoj        genral             475 14/NOV/96
   1111 dr.santosh      kid          78965.41 27/JAN/24
   5555 dr.abhi         dentist      78965.41 27/JAN/24
   6666 dr.pratek       eye             52000
   6666 dr.pratek                       52000 19/MAY/18

8 rows selected.
```

**DELETE command**

->this command is used to delete row/rows from table based on condition

->using delete command we can delete all rows or specific rows

Syn:- DELETE from tablename where condition

Q)delete all rows from cityhospital table?

```
SQL> select * from cityhospital;

   DRID DRNAME          DRBIO           DRSAL DRJOINDATE
_______ _______________ _____________ _______ _____________
   1111 dr.rakesh       surgen        50246.86 14/FEB/20
   2222 dr.abhi         skin          45687.92 12/OCT/16
    333 dr.prakash      heart         45678.93 01/FEB/12
   4444 dr.manoj        genral              475 14/NOV/96
   1111 dr.santosh      kid           78965.41 27/JAN/24
   5555 dr.abhi         dentist       78965.41 27/JAN/24
   6666 dr.pratek       eye              52000
   6666 dr.pratek                        52000 19/MAY/18

8 rows selected.

SQL> delete from cityhospital;

8 rows deleted.
```

->you have to commit so that it will delete permanently

Q)delete doctor joined in year 96?

```
  DRID DRNAME          DRBIO           DRSAL DRJOINDATE
_______ _______________ _____________ _______ _____________
  1111 dr.rakesh       surgen        45678.96 14/FEB/20
  2222 dr.abhi         skin          45687.92 12/OCT/16
   333 dr.prakash      heart         45678.93 01/FEB/12
  4444 dr.manoj        genral              475 14/NOV/96
  1111 dr.santosh      kid           78965.41 27/JAN/24
  5555 dr.abhi         dentist       78965.41 27/JAN/24
  6666 dr.pratek       eye
  6666 dr.pratek                             19/MAY/18

 rows selected.

QL> delete from myhospital where drjoindate like '%96';

 row deleted.
```

**FLASHBACK**

->introduced in oracle 9i and extended in 10g version.

->using flashback we can see the data that exists some time back;

->a query that return past data i.e 5min back,10min back by using flashback concept.

->it is used to recover data after commit.

->we see flashback later as off now see this

Q)display 5 min back data from hospital table which is deleted 5min before?

->select * from hospital as of timestamp(sysdate – interval ‘5’ minute);

```
______ _____________ _________ ___________ _____________
  1111 dr.rakesh     surgen        45678.96 14/FEB/20
  2222 dr.abhi       skin          45687.92 12/OCT/16
   333 dr.prakash    heart         45678.93 01/FEB/12
  4444 dr.manoj      genral             475 14/NOV/96
  1111 dr.santosh    kid           78965.41 27/JAN/24
  5555 dr.abhi       dentist       78965.41 27/JAN/24
  6666 dr.pratek     eye
  6666 dr.pratek                             19/MAY/18

8 rows selected.

SQL> delete from hospital where drid=333;

1 row deleted.

SQL> commit;

Commit complete.

SQL> select * from hospital as of timestamp(sysdate - interval '5' minute);

   DRID DRNAME        DRBIO           DRSAL DRJOINDATE
_______ _____________ _________ ___________ _____________
   1111 dr.rakesh     surgen        45678.96 14/FEB/20
   2222 dr.abhi       skin          45687.92 12/OCT/16
    333 dr.prakash    heart         45678.93 01/FEB/12
   4444 dr.manoj      genral             475 14/NOV/96
   1111 dr.santosh    kid           78965.41 27/JAN/24
   5555 dr.abhi       dentist       78965.41 27/JAN/24
   6666 dr.pratek     eye
   6666 dr.pratek                             19/MAY/18
```

How to recover data commited  after delete operation  just few min before?

->to recover data after commit,get the data that exists and insert that data into current table.

=>insert into hospital select * from hospital as of timestamp(sysdate – internal  ‘5’ minute);

```
SQL> select * from hospital as of timestamp(sysdate - interval '5' minute);

   DRID DRNAME        DRBIO            DRSAL DRJOINDATE
_______ ____________ _________ ____________ _____________
   1111 dr.rakesh     surgen        45678.96 14/FEB/20
   2222 dr.abhi       skin          45687.92 12/OCT/16
   4444 dr.manoj      genral             475 14/NOV/96
   1111 dr.santosh    kid           78965.41 27/JAN/24
   5555 dr.abhi       dentist       78965.41 27/JAN/24
   6666 dr.pratek     eye
   6666 dr.pratek                             19/MAY/18

7 rows selected.

SQL> insert into hospital select * from hospital as of timestamp(sysdate - interval '5' minute);

7 rows inserted.

SQL> commit;

Commit complete.
```

Merge command

->the oracle merge statement select data from one or more source table and update  or insert it into a target table . merge statement allows you to specify a condition and according to that condition only determines whether to update or insert into target table.

Syn:

**merge into source table**
**using source table**
**on (select...)**
**when matched then**
**.....update.....**
**when not matched then**
**----------insert----------;**
**both table should identical then it is mergable**

```
SQL> select * from hospital1;

   DRID DRNAME          DRSAL DRJOINDATE
_______ _________ ____________ _____________
   1111 rakesh       78412.54 11/JAN/12
   2222 mohan        74125.84 19/OCT/13
    333 rohan        65445.65 01/SEP/09
```

```
SQL> select * from hospital5;

   DRID DRNAME          DRSAL DRJOINDATE
_______ _________ ____________ _____________
   1111 rakesh       78412.54 11/JAN/12
   2222 mohan        74125.84 19/OCT/13
    333 rohan        65445.65 01/SEP/09
```

```
SQL> merge into hospital5 x
  2  using (select * from hospital1) y
  3  on (x.drid=333)
  4  when matched then
  5  update set x.drname=y.drname where y.drid=1111
  6  when not matched then
  7* insert (x.drid,x.drname,x.drsal,x.drjoindate) values (y.drid,y.drname,y.drsal,y.drjoindate);

1 row merged.

SQL> select * from hospital5;

   DRID DRNAME          DRSAL DRJOINDATE
_______ _________ ____________ _____________
   1111 rakesh       78412.54 11/JAN/12
   2222 mohan        74125.84 19/OCT/13
    333 rakesh       65445.65 01/SEP/09
```

## Insert All

->in simple this command is used to insert multiple rows into a table.

->you can insert data into multiple table also

Syn:- insert all

Into table1 values (…….)

Into table2 values(…….)

Select * from dual;

```
SQL> insert all
  2  into hospital1 values(5555,'sharma',45632.51,'14-aug-2013')
  3  into hospital5 values(6666,'verma',45632.88,'21-dec-2023')
  4* select * from dual;

2 rows inserted.

SQL> select * from hospital1;

   DRID DRNAME          DRSAL DRJOINDATE
_______ _________ ___________ _____________
   1111 rakesh       78412.54 11/JAN/12
   2222 mohan        74125.84 19/OCT/13
    333 rohan        65445.65 01/SEP/09
   5555 sharma       45632.51 14/AUG/13

SQL> select * from hospital5;

   DRID DRNAME          DRSAL DRJOINDATE
_______ _________ ___________ _____________
   1111 rakesh       78412.54 11/JAN/12
   2222 mohan        74125.84 19/OCT/13
    333 rakesh       65445.65 01/SEP/09
   6666 verma        45632.88 21/DEC/23
```

**DDL (data definition language) commands**

->create ,alter,drop,truncate,rename,flashback,purge all these commands acts on table structure includes columns , datatypes and sizes.

->all DDL commands are auto committed (which means we no need to fire commit command after ddl commands).

## Alter command

->this command used to modify the table structure that includes columns ,datatype and size.

->using ALTER command we can

Add column , drop column,rename column,modify column like change datatype or resize its value..etc

->it is used to modify structure of the table

**Syntax**:- alter table <tablename> [add|modify|drop|rename..]

**ADD**:- for adding new columns into the table

**Modify**:- for modifying the structure of columns

**Drop**:- for removing a column in the table

**Rename**:- for renaming the column name.

```
SQL> select * from hospital1;

   DRID DRNAME          DRSAL DRJOINDATE
_______ _________ ____________ ___________
   1111 rakesh        78412.54 11/JAN/12
   2222 mohan         74125.84 19/OCT/13
    333 rohan         65445.65 01/SEP/09
```

Adding columns

Q)add column phno in above table?

=>alter table hospital1  add (phno number(10));

```
SQL> alter table hospital1 add (phno number(10));

Table HOSPITAL1 altered.

SQL> select * from hospital1;

   DRID DRNAME          DRSAL DRJOINDATE       PHNO
_______ _________ ____________ ___________ _______
   1111 rakesh        78412.54 11/JAN/12
   2222 mohan         74125.84 19/OCT/13
    333 rohan         65445.65 01/SEP/09
```

->after adding new column by default the new column is filled with nulls and to insert data to new column use update command.

```
SQL> update hospital1 set phno=7412589635 where drname='rakesh';

1 row updated.

SQL> commit;

Commit complete.

SQL> select * from hospital1;

   DRID DRNAME          DRSAL DRJOINDATE              PHNO
_______ _________ ___________ _____________ _____________
   1111 rakesh       78412.54 11/JAN/12       7412589635
   2222 mohan        74125.84 19/OCT/13
    333 rohan        65445.65 01/SEP/09
```

Drop column

Q)drop column phno from hospital1?

=>alter table hospital1 drop (phno);

```
SQL> select * from hospital1;

   DRID DRNAME          DRSAL DRJOINDATE              PHNO
_______ _________ ___________ _____________ _____________
   1111 rakesh       78412.54 11/JAN/12       7412589635
   2222 mohan        74125.84 19/OCT/13
    333 rohan        65445.65 01/SEP/09

SQL> alter table hospital1 drop (phno);

Table HOSPITAL1 altered.

SQL> select * from hospital1;

   DRID DRNAME          DRSAL DRJOINDATE
_______ _________ ___________ _____________
   1111 rakesh       78412.54 11/JAN/12
   2222 mohan        74125.84 19/OCT/13
    333 rohan        65445.65 01/SEP/09
```

**Renaming Columns**

**Syn:-** alter table tablename rename COLUMN  oldcolname to newcolname;

**Q)rename column drjoindate to hiredon ?**

```
   DRID DRNAME          DRSAL DRJOINDATE
_______ _________ ___________ _____________
   1111 rakesh       78412.54 11/JAN/12
   2222 mohan        74125.84 19/OCT/13
    333 rohan        65445.65 01/SEP/09

SQL> alter table hospital1 rename column drjoindate to hiredon;

Table HOSPITAL1 altered.

SQL> select * from hospital1;

   DRID DRNAME          DRSAL HIREDON
_______ _________ ___________ _____________
   1111 rakesh       78412.54 11/JAN/12
   2222 mohan        74125.84 19/OCT/13
    333 rohan        65445.65 01/SEP/09
```

**Difference between Rename and alias ?**

**Alias:-** this change is not permanent in table .it changes column name in select statement output only.

**Rename:-**this is used to change colum name permanently in table.

## Modifying column name

->modify means we can able to change column size as well as column datatype.

Syn:- alter table tablename MODIFY  (col….)

Q)increase the size of drname in hospital1?

```
SQL> alter table hospital1 modify (drname varchar2(500));

Table HOSPITAL1 altered.

SQL> desc hospital1;

Name       Null?    Type
_________ ________ _________________
DRID                NUMBER(4)
DRNAME              VARCHAR2(500)
DRSAL               NUMBER(7,2)
HIREDON             DATE
SQL>
```

```
Name       Null?    Type
_________ ________ _________________
DRID                NUMBER(4)
DRNAME              VARCHAR2(500)
DRSAL               NUMBER(7,2)
HIREDON             DATE
SQL> alter table hospital1 modify (drsal number(8,3));

Table HOSPITAL1 altered.

SQL> desc hospital1;

Name       Null?    Type
_________ ________ _________________
DRID                NUMBER(4)
DRNAME              VARCHAR2(500)
DRSAL               NUMBER(8,3)
HIREDON             DATE
```

Note:- numeric column must be empty if we **decrease** any precision or scale like we have taken salary number(7,2) then if we want to modify it like salary number(20) then salary column must be empty.

```
SQL> alter table hospital modify (drsal number(20));

Error starting at line : 1 in command -
alter table hospital modify (drsal number(20))
Error report -
ORA-01440: column to be modified must be empty to decrease precision or scale
01440. 00000 -  "column to be modified must be empty to decrease precision or scale"
```

## Drop command

->it is used to drop table from database.

->it drop table structure along with data.

->prior to 10g version when we drop table it is permanently removed and cannot be restored .we need to use backups for that.

But from 10g onwards when we drop table then it is moved to recyclebin.

```
SQL> select * from hospital5;

   DRID DRNAME          DRSAL DRJOINDATE
_______ _________ ____________ _____________
   1111 rakesh        78412.54 11/JAN/12
   2222 mohan         74125.84 19/OCT/13
    333 rohan         65445.65 01/SEP/09

SQL> drop table hospital5;

Table HOSPITAL5 dropped.

SQL> select * from hospital5;
```

->to see recyclebin use command **show recyclebin;**

```
SQL> show recyclebin;
ORIGINAL NAME RECYCLEBIN NAME                OBJECT TYPE DROP TIME
------------- ------------------------------ ----------- -------------------
HOSPITAL5     BIN$FOVtC+T1Sm6fHxIVndug2Q==$0 TABLE       2024-01-30:14:07:57
```

## FLASHBACK command

->it introduced in oracle 10g.

->it is used to restore table from recyclebin because we know that if we drop table then it goes to recyclebin .so if accidentally by mistake table droped then we can get back from recyclebin.

Syn:- flashback table tablename to before drop;

```
SQL> flashback table hospital5 to  before drop;

Flashback succeeded.

SQL> select * from hospital5;

   DRID DRNAME          DRSAL DRJOINDATE
_______ _________ ___________ _____________
   1111 rakesh       78412.54 11/JAN/12
   2222 mohan        74125.84 19/OCT/13
    333 rohan        65445.65 01/SEP/09
```

**PURGE command**

->introduced in oracle 10g

->it is used to delete the object from recyclebin

->once object is deleted from recyclebin we cannot use flashback.

Syn:- purge table tablename;

Q)drop table hospital5 permanently ?

```
SQL> select * from hospital5;

   DRID DRNAME          DRSAL DRJOINDATE
_______ _________ ___________ _____________
   1111 rakesh       78412.54 11/JAN/12
   2222 mohan        74125.84 19/OCT/13
    333 rohan        65445.65 01/SEP/09

SQL> drop table hospital5;

Table HOSPITAL5 dropped.

SQL> show recyclebin;
ORIGINAL NAME RECYCLEBIN NAME                OBJECT TYPE DROP TIME
------------- ------------------------------ ----------- -------------------
HOSPITAL5     BIN$uEYKTAv7S+eBP96dqkYnLQ==$0 TABLE       2024-01-30:14:15:56
SQL> purge table hospital5;

Table purged.

SQL> show recyclebin;
SQL>
```

To empty recyclebin execute the following command.

=>**purge recyclebin;**

## Truncate Command

->it delete all the data from table but keeps structure of tha table.

->truncate command releases all the blocks allocated for the table and when blocks are released then data stored in the block are also deleted.

Syn:- truncate table <tablename>

```
   DRID DRNAME          DRSAL DRJOINDATE
_______ _________ ___________ _____________
        rakesh       78412.54 11/JAN/12
        mohan        74125.84 19/OCT/13
        rohan        65445.65 01/SEP/09

SQL> truncate table hospital6;

Table HOSPITAL6 truncated.

SQL> select * from hospital6;

no rows selected
SQL>
```

| DROP | DELETE | TRUNCATE |
|---|---|---|
| DDL command | DML command | DDL command |
| it drop structure with data | deletes only data but not structure | delete only data but not structure |

**What is difference between DELETE and TRUNCATE**

| DELETE | TRUNCATE |
| --- | --- |
| dml command | ddl command |
| can used to delete specific records | deletes all rows not specific rows |
| 'where' can be used to specify condition | 'where' cannot be used with truncate to specify condition. |
| deletes row by row | deletes all rows at one shot |
| slower | faster |
| not release memory | release memory |

## Rename Table

->rename command is used to change table name

Syn:- rename oldname to newname;

```
SQL> select * from hospital6;

no rows selected
SQL> rename hospital6 to hospital7;

Table renamed.
```

## String functions

->we have various String function using which we can transform our data few important function are listed below with examples.

**UPPER()** :- this function convert string into upper case. Just we need to pass string value in single quote or pass column name .

Syn:- select upper(colname) from tablename;

```
SQL> select * from hospital1;

   DRID DRNAME          DRSAL HIREDON
_______ _________ ____________ _____________
   1111 rakesh        78412.54 11/JAN/12
   2222 mohan         74125.84 19/OCT/13
    333 rohan         65445.65 01/SEP/09

SQL> select upper(drname) from hospital;

UPPER(DRNAME)
_____________________
DR.RAKESH
DR.ABHI
DR.MANOJ
DR.SANTOSH
DR.ABHI
DR.RAKESH
DR.ABHI
DR.MANOJ
DR.SANTOSH
DR.ABHI
DR.PRATEK
DR.PRATEK
```

```
SQL> select upper('welcome') from dual;

UPPER('WELCOME')
_____________________
WELCOME
```

**Note:- dual is dummy table provided by oracle used to select non db values.so we can check our queries directly on inbuilt DUAL table.**

**LOWER():-** it is used to convert string into lower case.

```
SQL> select lower(drname) from hospital;

LOWER(DRNAME)
_____________
dr.rakesh
dr.abhi
```

Q)convert all doctor name in uppercase?

```
   DRID DRNAME          DRBIO           DRSAL DRJOINDATE
_______ _______________ _____________ _______ _____________
   1111 dr.rakesh       surgen        45678.96 14/FEB/20
   2222 dr.abhi         skin          45687.92 12/OCT/16
   4444 dr.manoj        genral             475 14/NOV/96
   1111 dr.santosh      kid           78965.41 27/JAN/24
   5555 dr.abhi         dentist       78965.41 27/JAN/24
   1111 dr.rakesh       surgen        45678.96 14/FEB/20
   2222 dr.abhi         skin          45687.92 12/OCT/16
   4444 dr.manoj        genral             475 14/NOV/96
   1111 dr.santosh      kid           78965.41 27/JAN/24
   5555 dr.abhi         dentist       78965.41 27/JAN/24
   6666 dr.pratek       eye
   6666 dr.pratek                             19/MAY/18

12 rows selected.

SQL> update hospital set drname=upper(drname);

12 rows updated.

SQL> select * from hospital;

   DRID DRNAME          DRBIO           DRSAL DRJOINDATE
_______ _______________ _____________ _______ _____________
   1111 DR.RAKESH       surgen        45678.96 14/FEB/20
   2222 DR.ABHI         skin          45687.92 12/OCT/16
   4444 DR.MANOJ        genral             475 14/NOV/96
   1111 DR.SANTOSH      kid           78965.41 27/JAN/24
   5555 DR.ABHI         dentist       78965.41 27/JAN/24
   1111 DR.RAKESH       surgen        45678.96 14/FEB/20
   2222 DR.ABHI         skin          45687.92 12/OCT/16
```

**INITCAP():-** it is used to convert initial letter into capital.

```
SQL> select * from hospitalplus;

   DRID DRNAME    DRBIO          DRSAL DRJOINDATE
_______ ________ _______ ____________ _____________
   1001 darsh    ortho      74125.892 01/JAN/22
   1002 mahi     skin       74125.836 23/MAY/11
   1003 suraj    heart        741.254 14/OCT/23

SQL> select drid ,initcap(drname) as drname ,drbio ,drsal,drjoindate from hospitalplus;

   DRID DRNAME    DRBIO          DRSAL DRJOINDATE
_______ ________ _______ ____________ _____________
   1001 Darsh    ortho      74125.892 01/JAN/22
   1002 Mahi     skin       74125.836 23/MAY/11
   1003 Suraj    heart        741.254 14/OCT/23
```

```
SQL> select initcap('hello how are you') from dual;

INITCAP('HELLOHOWAREYOU')
_________________________
Hello How Are You
```

**LENGTH():-**it is used to return string length i.e number of characters

```
SQL> select * from hospitalplus;

   DRID DRNAME    DRBIO           DRSAL DRJOINDATE
_______ _________ _______ _____________ _____________
   1001 darsh     ortho       74125.892 01/JAN/22
   1002 mahi      skin        74125.836 23/MAY/11
   1003 suraj     heart         741.254 14/OCT/23

SQL> select length(drname) from hospitalplus;

   LENGTH(DRNAME)
_________________
                5
                4
                5
```

Q)display all doctorname whose name contains more then 4 character?

```
SQL> select * from hospitalplus where length(drname)>4;

   DRID DRNAME    DRBIO           DRSAL DRJOINDATE
_______ _________ _______ _____________ _____________
   1001 darsh     ortho       74125.892 01/JAN/22
   1003 suraj     heart         741.254 14/OCT/23
```

**SUBSTR():-**it is used to extract part of the String

Syn:- substr(string ,start,length)

Note:- if you not specify length then it goes till last.

Ex:- substr('hello welcome',1,5)== > hello

substr('hello welcome',10,3)== >com

substr('hello welcome',10)== > come

Note:- if you provide negative start point then it starts from right side and goes right side.

substr('hello welcome',-5,3) == >lco

substr('hello welcome',-10,4)== >lo w

```
SQL> select substr('hello welcome',10,3) from dual;

SUBSTR('HELLOWELCOME',10,3)
___________________________
com

SQL> select substr('hello welcome',-10,3) from dual;

SUBSTR('HELLOWELCOME',-10,3)
____________________________
lo
```

Q)display doctors names starts with 's'?

->we can do this by two ways . by using like operator and by using substr method.

```
SQL> select * from hospitalplus ;

   DRID DRNAME    DRBIO           DRSAL DRJOINDATE
_______ _________ ________ _____________ _____________
   1001 darsh     ortho       74125.892 01/JAN/22
   1002 mahi      skin        74125.836 23/MAY/11
   1003 suraj     heart         741.254 14/OCT/23

SQL> select * from hospitalplus where drname like 's%';

   DRID DRNAME    DRBIO         DRSAL DRJOINDATE
_______ _________ ________ ___________ _____________
   1003 suraj     heart       741.254 14/OCT/23

SQL> select * from hospitalplus where substr(drname,1,1)='s';

   DRID DRNAME    DRBIO         DRSAL DRJOINDATE
_______ _________ ________ ___________ _____________
   1003 suraj     heart       741.254 14/OCT/23
```

**|| (pipe operator)** :- it is used for string concatenation.

```
SQL> select * from hospitalplus ;

   DRID DRNAME    DRBIO           DRSAL DRJOINDATE
_______ _________ _______ _____________ _____________
   1001 darsh     ortho       74125.892 01/JAN/22
   1002 mahi      skin        74125.836 23/MAY/11
   1003 suraj     heart         741.254 14/OCT/23

SQL> select drname||' '||drbio from hospitalplus;

DRNAME||''||DRBIO
_____________________
darsh ortho
mahi skin
suraj heart
```

**INSTR():-**it is used to return position of a character in a string, if character found then it return position of that character and if character not found then return 0.

Syn:- instr(string , char ,startindex,occurance)

Note:- occurance is optional if you provide occurance then it will return position of that character occurred at that number.

Instr('hello welcome','o')== > 5

Instr('hello welcome','x')== > 0 (because there is no x)

Instr('hello welcome','o',1,2)== > 11 (because 2nd 'o' lies at 11th position)

```
SQL> select instr('hello welcome','w') from dual;

   INSTR('HELLOWELCOME','W')
____________________________
                           7

SQL> select instr('hello welcome','o',1,2) from dual;

   INSTR('HELLOWELCOME','O',1,2)
_________________________________
                             11
```

LPAD,RPAD:- both the function are used to fill string with a character

LPAD(string,length,char):- fill on left side.

RPAD(string,length,char):- fill on right side.

Note:- here length includes total length including given string.

Ex:- LPAD('hello',10,'*') == > *****hello (total length is 10)

RPAD('hello',8,'*')== >hello*** (total length is 8)

```
SQL> select lpad(drname,10,'*') from hospital;

LPAD(DRNAME,10,'*')
--------------------
*dr.rakesh
***dr.abhi
**dr.manoj
dr.santosh
***dr.abhi
*dr.rakesh
***dr.abhi
**dr.manoj
dr.santosh
***dr.abhi
*dr.pratek
*dr.pratek

12 rows selected.

SQL> select lpad(drname,10,'$') from hospital;

LPAD(DRNAME,10,'$')
--------------------
$dr.rakesh
$$$dr.abhi
$$dr.manoj
dr.santosh
$$$dr.abhi
```

```
SQL> select * from hospitalplus ;

   DRID DRNAME    DRBIO          DRSAL DRJOINDATE
_______ _________ ________ ____________ ____________
   1001 darsh     ortho       74125.892 01/JAN/22
   1002 mahi      skin        74125.836 23/MAY/11
   1003 suraj     heart         741.254 14/OCT/23

SQL> select rpad(drname,10,'$') from hospital;

RPAD(DRNAME,10,'$')
____________________
dr.rakesh$
dr.abhi$$$
dr.manoj$$
dr.santosh
dr.abhi$$$
dr.rakesh$
dr.abhi$$$
dr.manoj$$
```

**LTRIM(),RTRIM(),TRIM():-**

->it is used to remove unwanted space and characters.

LTRIM(string1,string2)== >removes from left side

RTRIM(string1,string2)== >removes from right side

Note:-2nd parameter we will give when we want to remove any character

Ex:-LTRIM(‘ hello ‘) == > ‘hello ‘

RTRIM(‘ hello ‘)== > ‘ hello’

LTRIM(‘@@@hello@@’,’@’)== >hello@@ (removes @ from left)

RTRIM(‘@@@hello@@’,’@’)== >@@@hello (removes @ from right)

**TRIM**:- it is used to remove unwanted space and character from start,end both sides.

TRIM(‘ hello ‘)== >hello

TRIM (leading ‘@’ from ‘@@@hello@’) == > hello@

TRIM(trailing ‘@’ from ‘@@hello@@’) == > @@hello

TRIM(both ‘@’ from ‘@@hello@@’)== > hello

```
SQL> select ltrim('    hello') from dual;

LTRIM('HELLO')
______________
hello

SQL> select rtrim(' hello   ') from dual;

RTRIM('HELLO')
______________
 hello

SQL> select trim('   hello   ') from dual;

TRIM('HELLO')
_____________
hello

SQL> select ltrim('@@@hello@@@','@') from dual;

LTRIM('@@@HELLO@@@','@')
_________________________
hello@@@

SQL> select rtrim('@@@hello@@@','@') from dual;

RTRIM('@@@HELLO@@@','@')
_________________________
@@@hello
```

```
TRIM(LEADING'@'FROM'@@@HELLO@@@')
_________________________________
hello@@@

SQL> select trim(trailing  '@' from '@@@hello@@@') from dual;

TRIM(TRAILING'@'FROM'@@@HELLO@@@')
__________________________________
@@@hello

SQL> select trim(both  '@' from '@@@hello@@@') from dual;

TRIM(BOTH'@'FROM'@@@HELLO@@@')
______________________________
hello
```

**REPLACE():-** it is used to replace one string with another string.

Syn:- replace(oldstring,removableString,replacingString)

replace('hello','ell','abc') == > habco

replace('hello','l','abc')== > heabcabco

note:-in replace() method string portion spelling should exact match then only it will replace means that portion must exact present in that string .

```
SQL> select replace('hello','ell','abc') from dual;

REPLACE('HELLO','ELL','ABC')
--------------------------------
habco

SQL> select replace('hello','l','abc') from dual;

REPLACE('HELLO','L','ABC')
-------------------------------
heabcabco

SQL> select replace('hello','o','abc') from dual;

REPLACE('HELLO','O','ABC')
-------------------------------
hellabc
```

```
SQL> select replace('hello','oo','abc') from dual;

REPLACE('HELLO','OO','ABC')
--------------------------------
hello
```

->no replacement happen because here there is only one 'o' but we matching with 'oo' so no replacement.

**TRANSLATE():-**it is used to translate one character to another character.and it replace character by character so if it found matched character it replace it with new character.

->it found character frm any position and replace it.

TRANSLATE(string1,string2,string3)

Ex:- translate('hello','elo','abc')== > habbc

e --> a

l --- > b

o-- > c

```
SQL> select  translate('hello','elo','abc') from dual;

TRANSLATE('HELLO','ELO','ABC')
------------------------------
habbc
```

By using translate function we can encrypt data.

Q)show all doctor records but salary should in encrypt form?

```
SQL> select * from hospitalplus;

   DRID DRNAME    DRBIO           DRSAL DRJOINDATE
------- --------- -------- ----------- -----------
   1001 darsh     ortho       74125.892 01/JAN/22
   1002 mahi      skin        74125.836 23/MAY/11
   1003 suraj     heart         741.254 14/OCT/23

SQL> select drid,drname,drbio,drjoindate , translate(drsal,'0123456789','$#@%^&*$%#') from hospitalplus;

   DRID DRNAME    DRBIO    DRJOINDATE    TRANSLATE(DRSAL,'0123456789','$#@%^&*$%#')
------- --------- -------- ----------- ------------------------------------------
   1001 darsh     ortho    01/JAN/22     $^#@&.%#@
   1002 mahi      skin     23/MAY/11     $^#@&.%%*
   1003 suraj     heart    14/OCT/23     $^#.@&^
```

Q)show all doctor records in normal format?

```
SQL> select drid,drname,drbio,drjoindate , translate(drsal,'0123456789','$#@%^&*$%#') from hospitalplus;

   DRID DRNAME    DRBIO    DRJOINDATE    TRANSLATE(DRSAL,'0123456789','$#@%^&*$%#')
------- --------- -------- ----------- ------------------------------------------
   1001 darsh     ortho    01/JAN/22     $^#@&.%#@
   1002 mahi      skin     23/MAY/11     $^#@&.%%*
   1003 suraj     heart    14/OCT/23     $^#.@&^

SQL> select drid,drname,drbio,drjoindate , translate(drsal,'$#@%^&*$%#','123456789') from hospitalplus;

   DRID DRNAME    DRBIO    DRJOINDATE    TRANSLATE(DRSAL,'$#@%^&*$%#','123456789')
------- --------- -------- ----------- ------------------------------------------
   1001 darsh     ortho    01/JAN/22     74125.892
   1002 mahi      skin     23/MAY/11     74125.836
   1003 suraj     heart    14/OCT/23     741.254
```

## Mathematical function

->many times we need some mathematical calculations ,we have many SQL provided arithmetic  functions to achieve this.

**ABS():-**this function is used to return absolute value of any number ,ABS changes all the negative numbers to positive and leaves positive number.

```
SQL> select abs(-10) from dual;

   ABS(-10)
___________
         10
```

```
SQL> select abs(-10.30) from dual;

   ABS(-10.30)
______________
          10.3
```

**POWER():-** to calculate any number with certain power we use this function.in this function we need to pass two arguments first argument is raised to the power of the second.

```
SQL> select power(5,2) from dual;

   POWER(5,2)
_____________
           25

SQL> select power(5,3) from dual;

   POWER(5,3)
_____________
          125
```

**SQRT():-** the function sqrt return square root of an argument .you cannot use sqrt on negative number because square root of a negative number is undefined.

```
SQL> select sqrt(25) from dual;

   SQRT(25)
___________
          5

SQL> select sqrt(625) from dual;

   SQRT(625)
____________
         25
```

**SIGN():-** it return 3 values ,if function argument is less then 0 then it returns -1,if argument is equal to 0 then return 0,if argument is greater then 0 then return 1.

```
SQL> select sign(-10), sign(10) ,sign(0) from dual;

   SIGN(-10)    SIGN(10)    SIGN(0)
____________ ____________ __________
          -1            1          0

SQL>
```

**MOD():-**it return reminder when we divide one value with another value.

```
SQL> select mod(5,2) from dual;

   MOD(5,2)
___________
          1

SQL> select mod(8,2) from dual;

   MOD(8,2)
___________
          0

SQL> select mod(15,2) from dual;

   MOD(15,2)
____________
           1
```

Q)display doctor list earning in multiples of 100?

```
SQL> select * from hospitalplus where mod(drsal,100)=0;

no rows selected
SQL>
```

**ROUND():-**used to round numbers to integer or to decimal places based on average .

round(number,decimal place)

round(37.4567) == > 37

37----------37.5----------38

if number <avg == > rounded to lowest value

if number >=avg === > rounded to highest value

round(37.5) === >38

round(37.4567,2) == > 37.46

round(37.4537,2) == >37.45

round(37.4537,3)== >37.454

round(383,-2) == > 400 (-2 means take difference of 100)

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

round(383,-1)== >380 (-1 means take difference of 10)

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

round(383,-3) === > 0 (-3 means take difference of 1000)

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

```
SQL> select round(37.4567) from dual;

   ROUND(37.4567)
_________________
               37

SQL> select round(37.5) from dual;

   ROUND(37.5)
______________
           38

SQL> select round(37.4567,2) from dual;

   ROUND(37.4567,2)
___________________
             37.46

SQL> select round(37.4537,3) from dual;

   ROUND(37.4537,3)
___________________
            37.454

SQL> select round(383,-2) from dual;

   ROUND(383,-2)
_________________
            400
```

TRUNC():-it rounds the number always to lowest

trunc(number,decimalplace)

trunc(37.7574) == > 37

trunc(37.7574,2) == >37.75

trunc(387,-1) == >380

trunc(387,-2)== >300

trunc (999,-3)== >0

```
SQL> select trunc(37.7574) from dual;

   TRUNC(37.7574)
_________________
               37

SQL> select trunc(37.7574,2) from dual;

   TRUNC(37.7574,2)
___________________
             37.75

SQL> select trunc(387,-1) from dual;

   TRUNC(387,-1)
_________________
             380

SQL> select trunc(387,-2) from dual;

   TRUNC(387,-2)
_________________
             300

SQL> select trunc(999,-3) from dual;

   TRUNC(999,-3)
_________________
               0
```

## Date and Time functions

->in real time many times we need to deal with date and times and we need to take help of date and time functions to transform it as per our requirements.

**EXTRACT():-**it is used to extract part of date,using this we can extract year/month/day.

extract(year from datecol/sysdate) -- >2024

extract(month from datecol/sysdate) --- >08

extract(day from datecol/sysdate) ---- >31

```
SQL> select * from hospitalplus;

   DRID DRNAME    DRBIO           DRSAL DRJOINDATE
_______ _________ ________ _____________ _____________
   1001 darsh     ortho       74125.892 01/JAN/22
   1002 mahi      skin        74125.836 23/MAY/11
   1003 suraj     heart         741.254 14/OCT/23

SQL> select extract(year from drjoindate) from hospitalplus;

   EXTRACT(YEARFROMDRJOINDATE)
______________________________
                          2022
                          2011
                          2023

SQL> select extract(month from drjoindate) from hospitalplus;

   EXTRACT(MONTHFROMDRJOINDATE)
_______________________________
                             1
                             5
                            10

SQL> select extract(day from drjoindate) from hospitalplus;

   EXTRACT(DAYFROMDRJOINDATE)
_____________________________
                           1
                          23
                          14
```

**MONTHS_BETWEEN()** :-it is used to return number of months between two dates.

```
SQL> select months_between('1-jan-2024','1-jan-2025') from dual;

   MONTHS_BETWEEN('1-JAN-2024','1-JAN-2025')
____________________________________________
                                          -12
```

**ADD_MONTHS() :-**it is used to add,substract  months from a date.

Add_months(date,number)

```
SQL> select add_months('2-jan-2024',2) from dual;

ADD_MONTHS('2-JAN-2024',2)
__________________________
02/MAR/24

SQL> select add_months('2-jan-2024',-2) from dual;

ADD_MONTHS('2-JAN-2024',-2)
___________________________
02/NOV/23
```

**Conversion**

->conversion means converting one datatype to another datatype

->conversion is 2 types.

1.implicit conversion

2.explicit conversion

**Implicit conversion:-**if conversion performed by oracle then it is called implicit conversion.

Ex1: select 1000+'1000' from dual ; == >2000

Here string '1000' automatically converted to number by oracle.

```
SQL> select 1000 + '1000' from dual;

   1000+'1000'
______________
          2000
```

```
SQL> create table accountant(empdate date);

Table ACCOUNTANT created.

SQL> insert into accountant values ('2-jan-2024');

1 row inserted.

SQL> select * from accountant;

EMPDATE
_________
02/JAN/24
```

Here also we are passing date into string format but internally it is converting into date.

**Explicit conversion:-**if conversion performed by user then it is called explicit conversion, to perform explicit conversion various function provided by oracle.

**Converting date to char type**

->dates converted to char type to display dates in different formats

TO_CHAR(date,format)

**formats :- (date = SYSDATE)**

| | |
|---|---|
| **yyyy** | **2021** |
| **yy** | **21** |
| **year** | **twenty twenty one** |
| **mm** | **09** |
| **mon** | **sep** |
| **month** | **september** |
| **ddd** | **(day of the year)** |
| **dd** | **01 (day of the month)** |
| **d** | **04 (day of the week)** |
| **dy** | **wed** |
| **day** | **wednesday** |
| **hh** | **hour part** |
| **hh24** | **hour part in 24 hrs format** |

**mi** minutes

**ss** seconds

**AM/PM** AM time OR PM time

**Q** 3 (quarter)

1 jan-mar

2 apr-jun

3 jul-sep

4 oct-dec

**w** week of the month

**ww** week of the year

```
SQL>  SELECT TO_CHAR(SYSDATE,'yyyy yy  year') FROM DUAL ;

TO_CHAR(SYSDATE,'YYYYYYYEAR')
_____________________________
2024 24  twenty twenty-four
```

```
SQL>  SELECT TO_CHAR(SYSDATE,'ddd dd  d  dy day') FROM DUAL ;

TO_CHAR(SYSDATE,'DDDDDDDYDAY')
______________________________
033 02  5  fri friday
```

**Q)display all doctor who joined on monday?**

```
SQL> select * from hospitalplus where to_char(drjoindate,'dy')='mon';

   DRID DRNAME    DRBIO           DRSAL DRJOINDATE
_______ _________ _______ _____________ ___________
   1002 mahi      skin        74125.836 23/MAY/11
```

## converting date strings to date :-

TO_DATE(datestring,format)

```
SQL> select to_date('2-feb-2024')+20 from dual;

TO_DATE('2-FEB-2024')+20
_____________________________
22/FEB/24

SQL> select to_date('2/10/2024','MM/DD/YY') from dual;

TO_DATE('2/10/2024','MM/DD/YY')
_____________________________________
10/FEB/24
```

## converting number to char type :-

=> numbers are converted to char type to display numbers in different formats.

TO_CHAR(number,[format])

formats

| | |
|---|---|
| 9 | represents a digit |
| 0 | represents a digit |
| , | thousand seperator |
| . | decimal seperator |
| L | currency symbol |
| C | currency |

TO_CHAR(1234,'99999') => 1234

TO_CHAR(1234,'00000') => 01234

TO_CHAR(1234,'000000') => 001234

TO_CHAR(1234,'9,999') => 1,234

TO_CHAR(1234,'9,999.99) => 1,234.00

TO_CHAR(1234,'L9,999') => $1,234

TO_CHAR(1234,'C9,999')  =>  USD1,234

**How to change currency ?**

ALTER SESSION SET NLS_TERRITORY='INDIA' ;

```
SQL> select to_char(1234,'99999') from dual;

TO_CHAR(1234,'99999')
_____________________
  1234

SQL> select to_char(1234,'00000') from dual;

TO_CHAR(1234,'00000')
_____________________
 01234

SQL> select to_char(1234,'9,999') from dual;

TO_CHAR(1234,'9,999')
_____________________
 1,234

SQL> select to_char(1234,'9,999.99') from dual;

TO_CHAR(1234,'9,999.99')
________________________
 1,234.00

SQL> select to_char(1234,'c9,999.99') from dual;

TO_CHAR(1234,'C9,999.99')
_________________________
     AUD1,234.00
```

## converting numeric string to number :-

->to_number(numericstring,format);

Ex:- to_number(‘1234’,’9,999’)

```
SQL> select to_number('1,234','9,999') from dual;

   TO_NUMBER('1,234','9,999')
______________________________
                          1234

SQL> select to_number('1234','9999') from dual;

   TO_NUMBER('1234','9999')
___________________________
                       1234
```

**Aggregate Functions / GROUP functions :-**

**->these functions process group  of rows and return one values.**

**MAX():-** return maximum value

**MIN():-** return minimum value

Q)show doctor who is getting maximum salary?

```
SQL> select max(drsal) from hospitalplus;

   MAX(DRSAL)
______________
    74125.892

SQL> select min(drsal) from hospitalplus;

   MIN(DRSAL)
______________
     741.254
```

**SUM():-** return total value

```
SQL> select sum(drsal) from hospitalplus;

   SUM(DRSAL)
______________
   148992.982
```

**AVG():-** return average value

```
SQL> select avg(drsal) from hospitalplus;

                                AVG(DRSAL)
__________________________________________
  49664.327333333333333333333333333333333
```

**COUNT() :-** returns no of values present in a column

Q)count no of records in hospitalplus table?

```
SQL> select count(*) from hospitalplus;

   COUNT(*)
___________
          3
```

```
SQL> select count(*) from hospitalplus;

   COUNT(*)
___________
          3

SQL> select count(drsal) from hospitalplus;

   COUNT(DRSAL)
_______________
              3

SQL> select count(drname) from hospitalplus;

   COUNT(DRNAME)
_______________
              3
```

**diff b/w COUNT & COUNT(*) ?**

=> COUNT function ignores nulls but COUNT(*) includes nulls

**NOTE** :- aggregate functions are not allowed in where clause they are allowed only in SELECT,HAVING clauses .

SELECT drname FROM hospitalplus

WHERE drsal = MAX(drsal) ; => ERROR

## Integrity Constraints

->Integrity constraints are rules to maintain data integrity ,it is also used to prevent users from entering invalid data, we have different integrity constraints in oracle.

1 NOT NULL

2 UNIQUE

3 PRIMARY KEY

4 CHECK

5 FOREIGN KEY

6 DEFAULT

**=> above constraints are declared in two ways**

1 column level

2 tablel evel

**COLUMN LEVEL :-**

=> if constraint is declared immediately after declaring column then it is called column level

syn :- CREATE TABLE <tabname>

```
(
 COLNAME  DATATYPE(SIZE) CONSTRAINT,
 COLNAME  DATATYPE(SIZE) CONSTRAINT,
 COLNAME  DATATYPE(SIZE) CONSTRAINT,
 ------------
);
```

**NOT NULL :-**

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

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

```
SQL> create table details(id number(4),addr varchar2(200) not null);

Table DETAILS created.
```

```
SQL> insert into details values(1111,'mp');

1 row inserted.

SQL> insert into details values(1111,null);

Error starting at line : 1 in command -
insert into details values(1111,null)
Error at Command Line : 1 Column : 33
Error report -
SQL Error: ORA-01400: cannot insert NULL into ("C##CODEMINES"."DETAILS"."ADDR")
01400. 00000 -  "cannot insert NULL into (%s)"
*Cause:    An attempt was made to insert NULL into previously listed objects.
*Action:   These objects cannot accept NULL values.
```

## Unique

->unique constraints  doesn’t  accept duplicates

```
SQL> create table pract1(id number(4),addr varchar2(200) unique);

Table PRACT1 created.

SQL> insert into pract1 values(1111,'mp');

1 row inserted.

SQL> insert into pract1 values(1111,'mp');

Error starting at line : 1 in command -
insert into pract1 values(1111,'mp')
Error report -
ORA-00001: unique constraint (C##CODEMINES.SYS_C0059384) violated
```

**PRIMARY KEY :-** primary key doesn’t accept duplicates and nulls

->it is combinationof unique and not null.

Primary key = unique + not null

->in table it is recommended to use one unique column so that you can identify records uniquely for that we can use primary key.

```
SQL> create table pract2(id number(4) primary key,addr varchar2(200));

Table PRACT2 created.
```

=> only one primary key is allowed per table , if we want two primary keys then declare one column with primary key and another column with unique & not null.

**candidate key :-** a field which is eligible for primary key is called candidate key.

Ex:-id, modeno,..etc.

**CHECK constraint :-**use check constraint when rule based on condition

Syn:- check(condition)

->check constraints is always declared with condition.

->if condition=true then value is accepted, if condtion=false then value is not accepted.

```
SQL> create table empdemo (empid number(4),ename varchar2(200),empsal number(7) check(empsal>8000));

Table EMPDEMO created.

SQL> insert into empdemo values (1111,'ravi',9000);

1 row inserted.

SQL> insert into empdemo values (1111,'ravi',10000);

1 row inserted.

SQL> insert into empdemo values (1111,'ravi',7999);

Error starting at line : 1 in command -
insert into empdemo values (1111,'ravi',7999)
Error report -
ORA-02290: check constraint (C##CODEMINES.SYS_C0059386) violated
```

**FOREIGN KEY :-**

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

=> to establish relationship between two tables take pk of one table and add it to another table as fk and declare with refernces constraint.

```
SQL> create table projectdetails (pid number(4) primary key,name varchar2(20),duration varchar2(30),cost number(10));

Table PROJECTDETAILS created.

SQL> create table emp(empid number(4) primary key, name varchar2(30) ,sal number(10),projid number(4) references projectdetails(pid));

Table EMP created.

SQL> desc projectdetails

Name        Null?       Type
_____________ _________ _____________
PID         NOT NULL    NUMBER(4)
NAME                    VARCHAR2(20)
DURATION                VARCHAR2(30)
COST                    NUMBER(10)
SQL> desc emp;

Name      Null?       Type
_________ _________ _____________
EMPID     NOT NULL    NUMBER(4)
NAME                  VARCHAR2(30)
SAL                   NUMBER(10)
PROJID                NUMBER(4)
SQL>
```

```
SQL> select * from projectdetails;

    PID NAME       DURATION             COST
_______ __________ _____________ _____________
   1111 hari       2 years       7500000000
   2222 health     2 months      7600000000
   3333 rail       8 months       700000000

SQL> select * from emp;

   EMPID NAME          SAL    PROJID
_______ _________ _________ _________
    1001 ravi        25000      2222
    1002 selv        36000      1111
    1003 teja        38000      3333
```

=> values entered in fk column should match with values entered in pk column

=> fk allows duplicates and nulls.

=> after declaring fk a relationship is established between two tables called parent/child relationship.

=> by default sql server creates one to many (1:m) relationship between two tables , to  establish one to one (1:1) relationship declare foreign key with unique constraint.

**DEFAULT :-** if you want to make some column value default then you can use this which means while inserting if we skip that column  then oracle insert default value **.**

```
SQL> create table book12 (bid number(4), pubdate date default sysdate,bookname varchar2(20) default 'fullstack');

Table BOOK12 created.
```

```
SQL> insert into book12 (bid) values (1111);

1 row inserted.

SQL> insert into book12 (bid) values (2222);

1 row inserted.

SQL> insert into book12 (bid) values (3333);

1 row inserted.

SQL> commit;

Commit complete.

SQL> select * from book12;

    BID PUBDATE      BOOKNAME
_______ ____________ ____________
   1111 02/FEB/24    fullstack
   2222 02/FEB/24    fullstack
   3333 02/FEB/24    fullstack
```

**TABLE LEVEL :-**

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

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

```
SQL>  CREATE TABLE product24
  2   (
  3     prodid   NUMBER(3),
  4     name     VARCHAR2(10),
  5     price    NUMBER(5),
  6     mfd_dt   DATE,
  7     exp_dt   DATE,
  8          CHECK(exp_dt > mfd_dt)
  9*  );

Table PRODUCT24 created.

SQL> CREATE TABLE products25
  2   (
  3     prodid   NUMBER(3),
  4     name     VARCHAR2(10),
  5     price    NUMBER(5),
  6     mfd_dt   DATE,
  7     exp_dt   DATE,
  8          CHECK(exp_dt > mfd_dt),primary key(prodid)
  9*  );

Table PRODUCTS25 created.
```

**composite primary key :-**

=> in some tables we can't uniquely identify records using single column and we need combination of columns to uniquely identify , if combination of columns declare primary key then it is called composite primary key.

```
SQL> CREATE TABLE studentplus
  2   (
  3     sid  NUMBER(2) PRIMARY KEY,
  4     sname VARCHAR2(10)
  5*  );
SQL> CREATE TABLE course
  2  (
  3    cid   NUMBER(2) PRIMARY KEY,
  4    cname VARCHAR2(10)
  5* );
SQL> CREATE TABLE registrations
  2  (
  3    sid  NUMBER(2) REFERENCES STUDENTPLUS(sid),
  4    cid  NUMBER(2) REFERENCES COURSE(cid),
  5    do   DATE,
  6       PRIMARY KEY(sid,cid)
  7*  );

Table REGISTRATIONS created.
```

```
SQL> insert into registrations values(1,10,sysdate);

Error starting at line : 1 in command -
insert into registrations values(1,10,sysdate)
Error report -
ORA-00001: unique constraint (C##CODEMINES.SYS_C0059396) violated
```

Q)which of the following constraint cannot be declared at the table?

->not null

**Adding constraints to existing table**

**->ALTER** command is used to add constraints to existing table.

```
SQL> create table emp55 (
  2  empno number(4),
  3  ename varchar2(10),
  4  sal number(7,2)
  5* );

Table EMP55 created.
```

**Adding primary Key:-**

add primary key to column empno?

```
SQL> create table emp55 (
  2  empno number(4),
  3  ename varchar2(10),
  4  sal number(7,2)
  5* );

Table EMP55 created.

SQL> alter table emp55 add primary key (empno);

Table EMP55 altered.

SQL> desc emp55;

Name      Null?       Type
_________ ___________ _____________
EMPNO     NOT NULL    NUMBER(4)
ENAME                 VARCHAR2(10)
SAL                   NUMBER(7,2)
SQL>
```

**Note:-** while adding constraint oracle also validates existing data,if existing data doesn't satisfy as per as constraint then that given constraint will not be added.

Note:- if check constraint added with novalidate then oracle will not validate existing data and oracle validates only from new data.

```
SQL> create table emp66
  2  (empid number(4),
  3  ename varchar2(20),
  4  esal number(10)
  5* );
SQL> alter table emp66 add check(esal>8000);

Error starting at line : 1 in command -
alter table emp66 add check(esal>8000)
Error report -
ORA-02293: cannot validate (C##CODEMINES.) - check constraint violated
02293. 00000 - "cannot validate (%s.%s) - check constraint violated"
*Cause:    an alter table operation tried to validate a check constraint to
           populated table that had nocomplying values.
*Action:   Obvious
SQL> alter table emp66 add check(esal>8000) novalidate;

Table EMP66 altered.
```

## Drop constraints:-

->we can also drop constraints if we don't want that.

Syn:- alter table tablename drop constrainName;

```
SQL> desc emp55;

Name      Null?       Type
_________ ___________ _______________
EMPNO     NOT NULL    NUMBER(4)
ENAME                 VARCHAR2(10)
SAL                   NUMBER(7,2)
SQL> alter table emp55 drop  PRIMARY KEY;

Table EMP55 altered.

SQL> desc emp55;

Name      Null?    Type
_________ ________ _______________
EMPNO              NUMBER(4)
ENAME              VARCHAR2(10)
SAL                NUMBER(7,2)
SQL>
```

Note:-primary key constraint cannot be dropped if referenced by foreign key.

**With CASCADE**

->if want to drop primary key along with foreign key use cascade property.

Syn:- alter table tableName drop primary key cascade;

**DELETE RULES**

->we have certain rules and these rules are declared with foreign key.

->these rules specifies how child rows are affected if we delete parent row.

1.ON DELETE NO ACTION ( it is default)

2.ON DELETE CASCADE

3.ON DELETE SET NULL

**1.ON DELETE NO ACTION**

->parent row cannot be deleted if associated with child rows

Ex:

```
SQL> create table emp99
  2  (eno number(4) primary key,
  3* ename varchar2(10) );
```

```
SQL> insert into emp99 values (10,'HR');

1 row inserted.

SQL> insert into emp99 values (20,'dev');

1 row inserted.
```

```
SQL> create table proj99
  2  (pno number(4) primary key,
  3  pname varchar2(10),
  4  empno number(4) references emp99(eno)
  5*  );

Table PROJ99 created.
```

```
SQL> insert into proj99 values(101,'health',10);

1 row inserted.

SQL> insert into proj99 values(102,'banking',20);

1 row inserted.
```

```
SQL> delete from emp99 where eno=10;

Error starting at line : 1 in command -
delete from emp99 where eno=10
Error report -
ORA-02292: integrity constraint (C##CODEMINES.SYS_C0059492) violated - child
record found
```

**ON DELETE CASCADE**

->if parent row is deleted then it is deleted along with child row.

```
SQL>   CREATE TABLE dept99
  2  (
  3    dno  NUMBER(2) PRIMARY KEY,
  4    dname VARCHAR2(10)
  5*  );

Table DEPT99 created.

SQL> INSERT INTO dept99 VALUES(10,'HR');

1 row inserted.

SQL> INSERT INTO dept99 VALUES(20,'IT');

1 row inserted.
```

```
SQL> CREATE TABLE emp100
  2  (
  3    empno NUMBER(4) PRIMARY KEY,
  4    ename VARCHAR2(10),
  5    dno   NUMBER(2) REFERENCES dept99(dno)
  6                    ON DELETE CASCADE
  7*  );

Table EMP100 created.

 INSERT INTO emp100 VALUES(1,'A',10);

1 row inserted.

SQL> INSERT INTO emp100 VALUES(2,'B',10);

1 row inserted.
```

```
SQL> delete from dept99 where dno=10;

1 row deleted.
```

**ON DELETE SET NULL**

->if we delete parent row ,child rows will not deleted but foregn key will be set to null.

```
SQL>   CREATE TABLE dept99
  2  (
  3    dno  NUMBER(2) PRIMARY KEY,
  4    dname VARCHAR2(10)
  5*  );
SQL>
SQL> CREATE TABLE emp99
  2  (
  3    empno NUMBER(4) PRIMARY KEY,
  4    ename VARCHAR2(10),
  5    dno   NUMBER(2) REFERENCES dept99(dno)
  6                    ON DELETE SET NULL
  7*  );
```

## JOINS

->joins is an operation performed to fetch data from two or more table, to fetch data from two tables we need to join those table.

->join will enable you to gather and manipulate data across several tables.

->this is one of the most powerful feature of sql .without this you need to store all data in single table which is not that much efficient.

**Types of joins :-**

1 Equi Join / Inner Join

2 Outer Join

left outer

right outer

full outer

3 Non -Equi Join

4 Self Join

**=> we can write join queries in 2 styles**

1 Native style (oracle style)

2 ANSI style

**Equi Join / Inner Join:-**

->to perform equi join between two tables there must be a common field and name of the common field need not to be same and primary key-foreign key relationship is also not compulsory. Equi join is performed on common field.

->it is also called simple join or inner join.

```
SQL> select * from empmgmt;

     EMPNO ENAME            ESAL     DEPTNO
_________ _________ _____________ __________
      1111 ravi         45632.12       1001
      2222 akash        40632.12       1002
      3333 rahul        40032.12       1003
      4444 varsha       40011.12       1004

SQL> select * from empdept;

       DNO DNAME     ADDR
__________ ________ _________
      1002 it       pune
      1001 test     hyd
      1004 hr       mumbai
```

How to write query:- we need to write the join condition in where clause. Prefix the column name with table name when the same column name appears in more then one table.

Q) display empno,ename,esal,dname,addr ?(native style)

```
SQL> select empno,ename,esal,dname from empdept,empmgmt where empmgmt.deptno=empdept.dno;

   EMPNO ENAME            ESAL DNAME
________ _________ ____________ ________
    1111 ravi         45632.12 test
    2222 akash        40632.12 it
    4444 varsha       40011.12 hr
```

->if you take common column then use table name as prefix..see below

```
SQL> select empmgmt.empno,empmgmt.ename,empmgmt.esal,empmgmt.deptno,empdept.dno,empdept.dname from empmgmt,empdept where empmgmt.deptno=empdept.dno;

   EMPNO ENAME           ESAL    DEPTNO     DNO DNAME
________ _________ ___________ _________ _______ ________
    1111 ravi         45632.12      1001    1001 test
    2222 akash        40632.12      1002    1002 it
    4444 varsha       40011.12      1004    1004 hr
```

Note:- you can use column name with table prefix with any column but it is mandatory with common table otherwise how oracle will know from which table column belongs

Ex:-

```
SQL> select * from empmgmt1;

   EMPNO ENAME            ESAL    DEPTNO
________ _________ ____________ _________
    1111 akash        40632.12      1002
    2222 akash        40632.12      1003

SQL> select * from empdept1;

   DEPTNO DNAME    ADDR
_________ ________ _________
     1003 hr       mumbai
     1002 test     pune
```

```
SQL> select empno,ename,esal,deptno from empmgmt1,empdept1 where empmgmt1.deptno=empdept1.deptno;

Error starting at line : 1 in command -
select empno,ename,esal,deptno from empmgmt1,empdept1 where empmgmt1.deptno=empdept1.deptno
Error at Command Line : 1 Column : 25
Error report -
SQL Error: ORA-00918: column ambiguously defined
00918. 00000 -  "column ambiguously defined"
*Cause:
*Action:
```

->you can also create statements with temporary name  ex:

```
SQL> select e.empno,e.ename,e.esal,e.deptno,d.dno,d.dname from empmgmt e,empdept d where e.deptno=d.dno;

    EMPNO ENAME           ESAL     DEPTNO      DNO DNAME
_________ _________ ___________ _________ ________ _________
     2222 akash       40632.12       1002     1002 it
     1111 ravi        45632.12       1001     1001 test
     4444 varsha      40011.12       1004     1004 hr
     3333 rahul       40032.12       1003     1003 hr
```

=> when no of tables increases no of join conditions also increases to join N tables  N-1 join conditions required.

**ANSI style :-**

=> introduced in oracle 9i

=> Adv of ANSI style is portability

=> ANSI style queries can be migrated/used  from one db to another db

=> in ANSI style tablenames are seperated by keywords

=> use **ON** clause for join conditions instead of WHERE clause

```
SQL> select e.empno,e.ename,d.dno,d.dname from  empmgmt e inner join empdept d on  e.deptno=d.dno;

    EMPNO ENAME          DNO DNAME
_________ _________ ________ _________
     2222 akash       1002 it
     1111 ravi        1001 test
     4444 varsha      1004 hr
     3333 rahul       1003 hr
```

Now use ON clause for join and use where condition for filter..

```
SQL> select e.empno,e.ename,d.dno,d.dname from  empmgmt e inner join empdept d on  e.deptno=d.dno where d.dno=1004;

    EMPNO ENAME          DNO DNAME
_________ _________ ________ _________
     4444 varsha      1004 hr
```

**OUTER JOIN**

->as we seen above equi join  .equi join returns only matching records but cannot return unmatched records. But to get unmatched records also perfrom outer join.

->you can also say it is a join which forcibly joins multiple tables even without having  then common data.

->it is represented by +.(in native style)

```
   EMPNO ENAME           ESAL    DEPTNO
________ _________ ____________ _________
    1111 akash        40632.12      1002
    2222 akash        40632.12      1003
     333 raaj         41257.85      1004

SQL> select * from empdept1;

   DEPTNO DNAME     ADDR
_________ _________ _________
     1003 hr        mumbai
     1002 test      pune
     1005 devops    blr
```

Here above we have one unmatched record ..

**=> outer join is 3 types**

1 left outer join

2 right outer join

3 full outer join

**Left Outer Join:-** it return all rows (matched  + unmatched ) from left  side and matching  rows from  right side table.

```
SQL> select e.empno,e.ename,e.esal,e.deptno,d.deptno,d.dname,d.addr from empmgmt1 e,empdept1 d where e.deptno=d.deptno(+);

   EMPNO ENAME          ESAL    DEPTNO    DEPTNO DNAME    ADDR
________ _________ _________ _________ _________ _______ _______
    2222 akash       40632.12      1003      1003 hr      mumbai
    1111 akash       40632.12      1002      1002 test    pune
     333 raaj        41257.85      1004
```

**Right Outer Join:-** return all rows  from right side table and matching row from left side table.

```
SQL> select e.empno,e.ename,e.esal,e.deptno,d.deptno,d.dname,d.addr from empmgmt1 e,empdept1 d where e.deptno(+)=d.deptno;

   EMPNO ENAME          ESAL     DEPTNO    DEPTNO DNAME     ADDR
________ _______ _____________ _________ _________ _________ _________
    1111 akash       40632.12       1002      1002 test      pune
    2222 akash       40632.12       1003      1003 hr        mumbai
                                           1005 devops    blr
```

## Full Outer Join

->return all rows from both tables.

->note:- native style  doesn't  support full outer join ,only ansi style supports.

```
SQL> select e.empno,e.ename,e.esal,e.deptno,d.deptno,d.dname,d.addr from empmgmt1 e,empdept1 d where e.deptno(+)=d.deptno(+);

Error starting at line : 1 in command -
select e.empno,e.ename,e.esal,e.deptno,d.deptno,d.dname,d.addr from empmgmt1 e,empdept1 d where e.deptno(+)=d.deptno(+)
Error at Command Line : 1 Column : 108
Error report -
```

=>  to perform full outer join in native tyle combine the outputs of left outer and righ outer by using UNION operator.

```
SQL> select e.empno,e.ename,e.esal,e.deptno,d.deptno,d.dname,d.addr from empmgmt1 e,empdept1 d where e.deptno(+)=d.deptno union
  2* select e.empno,e.ename,e.esal,e.deptno,d.deptno,d.dname,d.addr from empmgmt1 e,empdept1 d where e.deptno=d.deptno(+);

   EMPNO ENAME          ESAL     DEPTNO    DEPTNO DNAME     ADDR
________ _______ _____________ _________ _________ _________ _________
     333 raaj       41257.85       1004
    1111 akash      40632.12       1002      1002 test      pune
    2222 akash      40632.12       1003      1003 hr        mumbai
                                           1005 devops    blr
```

**ANSI STYLE :-** for all left ,right,full join

## Left Outer Join in Ansi style:

```
SQL> select e.ename,e.esal,e.deptno,d.deptno,d.dname,d.addr
  2* from empmgmt1 e LEFT OUTER JOIN empdept1 d on e.deptno=d.deptno;

ENAME          ESAL     DEPTNO    DEPTNO DNAME     ADDR
_______ _____________ _________ _________ _________ _________
akash      40632.12       1003      1003 hr        mumbai
akash      40632.12       1002      1002 test      pune
raaj       41257.85       1004
```

## Right Outer Join in Ansi style

```
SQL> select e.ename,e.esal,e.deptno,d.deptno,d.dname,d.addr
  2* from empmgmt1 e RIGHT OUTER JOIN empdept1 d on e.deptno=d.deptno;

ENAME           ESAL     DEPTNO     DEPTNO DNAME      ADDR
_______ _____________ __________ __________ _________ _________
akash       40632.12       1002       1002 test      pune
akash       40632.12       1003       1003 hr        mumbai
                                      1005 devops    blr
```

```
SQL> select e.ename,e.esal,e.deptno,d.deptno,d.dname,d.addr
  2* from empmgmt1 e FULL OUTER JOIN empdept1 d on e.deptno=d.deptno;

ENAME           ESAL     DEPTNO     DEPTNO DNAME      ADDR
_______ _____________ __________ __________ _________ _________
akash       40632.12       1003       1003 hr        mumbai
akash       40632.12       1002       1002 test      pune
                                      1005 devops    blr
raaj        41257.85       1004
```

## Non Equi Join :-

**->this is used to perform join between tables not sharing a common fields.**

**->this join is called non equi join because here join condition is not based on “=” operator.**

```
SQL> select * from empsal;

   EMPNO ENAME       ESAL
________ _______ _______
       1 Abhi         600
       2 raj         2500
       3 surah       1700
       4 manoj       1650

SQL> select * from salgrade;

   GRADEID    LOWGRADE    HIGRADE
_________ ___________ __________
        1         700       1000
        2        1200       2000
        3        1800       2100

SQL> select e.ename,e.esal from empsal e,salgrade g  where e.esal between g.lowgrade and g.higrade;

ENAME       ESAL
_______ _______
surah       1700
manoj       1650
```

**In ansi style**

```
SQL> select e.ename,e.esal from empsal e join salgrade g  on e.esal between g.lowgrade and g.higrade;

ENAME       ESAL
_______ _______
surah       1700
manoj       1650
```

## 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.

```
SQL> select * from managerdata;

    EMPNO ENAME         MGRNO
_________ _________ _________
     7369 john           7902
     7499 akshya         7698
     7521 hary           7698
     7566 rohan          7839
     7654 mahi           7566
```

->above table contain manager no but to display  manager name we need to perform  self join because in same table employee and manager stored and normal and manager both are employee only.

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

Q)display ename, managername ?

```
SQL> select * from managerdata;

    EMPNO ENAME         MGRNO
_________ _________ _________
     7369 john           7902
     7499 akshya         7698
     7521 hary           7698
     7566 rohan          7839
     7654 mahi           7566

SQL> select x.ename,y.ename from managerdata x,managerdata y where x.mgrno=y.empno;

ENAME    ENAME
_______ _______
mahi     rohan
```

In ansi style

```
SQL> select x.ename,y.ename from managerdata x join managerdata y on x.mgrno=y.empno;

ENAME    ENAME
_______ ________
mahi     rohan
```

## subqueries / nested queries :-

->a query in another query is called subquery or nested query.

->one query is called outer/parent/main-query while another query is called inner/child/sub-query.

->first oracle executes inner query then it executes outer query which means result of inner query  is input to outer query.

->we use subquery when  WHERE condition  based on unkown value.

**Syn**:- select columns from tablename where colname (inner query);

```
SQL> select * from officework;

   EMPID ENAME      JOB              SAL HIREDATE        AGE
_______ _________ _________ _____________ _____________ _____
     100 imran     dev             92400 28/OCT/23        21
     101 sumit     dev             99000 29/OCT/23        21
     103 ganesh    tester          88000 27/NOV/24        21
     104 akshay    dba             11000 01/JAN/24        22
     105 nikita
     106 rocky     devops           5500
                   tester
         ganesh                    88000
    1995 santosh   mba              1650 04/DEC/24        31
     118 mahi      devops       81538.46 03/FEB/24        25

10 rows selected.
```

Q)display employee earning more then mahi?

-> select * from officework where sal > (select sal from officework where ename='mahi');

```
SQL> select * from officework where sal > (select sal from officework where ename='mahi');

   EMPID ENAME      JOB            SAL HIREDATE        AGE
________ _________ _________ _______ _____________ _____
     100 imran     dev          92400 28/OCT/23        21
     101 sumit     dev          99000 29/OCT/23        21
     103 ganesh    tester       88000 27/NOV/24        21
         ganesh                 88000
```

Q)display employe list senior to santosh ?

```
SQL> select * from officework where hiredate < (select hiredate from officework where ename='santosh');

   EMPID ENAME      JOB            SAL HIREDATE        AGE
________ _________ _________ _______ _____________ _____
     100 imran     dev          92400 28/OCT/23        21
     101 sumit     dev          99000 29/OCT/23        21
     103 ganesh    tester       88000 27/NOV/24        21
     104 akshay    dba          11000 01/JAN/24        22
     118 mahi      devops     81538.46 03/FEB/24        25
```

**FETCH clause :-**

->it is introduced in oracle 12c version. It is used to limit no of rows return by select statement.

Q) display first 5 rows from officework table?

```
SQL> select * from officework ;

   EMPID ENAME      JOB            SAL HIREDATE        AGE
________ _________ _________ _______ _____________ _____
     100 imran     dev          92400 28/OCT/23        21
     101 sumit     dev          99000 29/OCT/23        21
     103 ganesh    tester       88000 27/NOV/24        21
     104 akshay    dba          11000 01/JAN/24        22
     105 nikita
     106 rocky     devops        5500
                   tester
         ganesh                 88000
    1995 santosh   mba           1650 04/DEC/24        31
     118 mahi      devops     81538.46 03/FEB/24        25

10 rows selected.

SQL> select * from officework FETCH FIRST 5 ROWS ONLY;

   EMPID ENAME      JOB            SAL HIREDATE        AGE
________ _________ _________ _______ _____________ _____
     100 imran     dev          92400 28/OCT/23        21
     101 sumit     dev          99000 29/OCT/23        21
     103 ganesh    tester       88000 27/NOV/24        21
     104 akshay    dba          11000 01/JAN/24        22
     105 nikita
```

```
SQL> select * from officework FETCH FIRST 20 PERCENT ROWS ONLY;

   EMPID ENAME    JOB         SAL HIREDATE        AGE
________ ________ ______ _______ _____________ _____
     100 imran    dev      92400 28/OCT/23        21
     101 sumit    dev      99000 29/OCT/23        21
```

**Database Transactions**

-> a transaction is a unit of work that contains one or more dmls and that must be saved as a whole or must be cancelled as a whole.

->every database system must gurantee a property called atomicity i.e all or none . if any db transactions consists of multiple operations if all are successful then it must be saved , if one of operation fails then entire transactions must cancelled.

->the following commands provided by oracle to control transcation called TCL(Transcation Control Language).

**COMMIT:-** to save transcation

**ROLLBACK:-**to cancel transcation

**SAVEPOINT:-** to cancel transcation upto savepoint

Note:- DDL/DCL command ends with commit.

If any transcation(insert,update,delete..) ends with commit then it is called successful transcation and operations are committed.

If any transcation ends with rollback then it is called aborted transcation and operation are cancelled.

see below

```
SQL> create table empbio (eid number(2));

Table EMPBIO created.

SQL> insert into empbio values (01);

1 row inserted.

SQL> insert into empbio values (02);

1 row inserted.

SQL> commit;

Commit complete.

SQL> update empbio set eid=03 where eid=02;

1 row updated.

SQL> commit;

Commit complete.
```

```
SQL> insert into emppf values (01,752145,'rocky');

1 row inserted.

SQL> insert into emppf values (02,75212,'jeet');

1 row inserted.

SQL> rollback;

Rollback complete.
```

**SAVEPOINT:-** we can declare savepoint and we can rollback upto the savepoint, using savepoint we can cancel part of the transcation.

insert into emppf values (01,752145,'rocky');

insert into emppf values (02,752145,'dars');

savepoint point1;

insert into emppf values (03,22145,'kansh');

insert into emppf values (04,8895,'praveen');

rollback to point1;

```
SQL> insert into emppf values (01,752145,'rocky');

1 row inserted.

SQL> insert into emppf values (02,752145,'dars');

1 row inserted.

SQL> savepoint point1;

Savepoint created.

SQL> insert into emppf values (03,22145,'kansh');

1 row inserted.

SQL> insert into emppf values (04,8895,'praveen');

1 row inserted.

SQL> rollback to point1;

Rollback complete.
```

**DB objects/SCHEMA objects :-**

TABLES (already seen)

VIEWS

INDEXES

**VIEWS :-**

->view is a subset of a table because it includes specific cols and rows. Generally VIEW are created from tables those tables are also called as base table.

->In real world generally from data security point of view “Database Administrator “ creates Views from base table and then those view given to the number of users.

->View doesn’t store data that’s why VIEW is also called as VIRTUAL TABLE .

->Views are created to provide security and to reduce complexity.

**View are 2 types**

**1.Simple View**

**2.Complex View**

**->**In oracle to create view we need to take permission from sysdba means from system user.

**Step1-connect system/manager**

**Step2 grant create view to c##username**

**Step3-connect c##user/password**

```
SQL> grant create view to c##codemines
  2* ;

Grant succeeded.
```

**SIMPLE VIEW**

->Simple View is a View which is created from only One Base Tables Where as “Complex View “ is a View which is created from Multiple base table.

Syn:- CREATE VIEW <NAME> AS SELECT STATEMENT ;

```
SQL> connect c##codemines/root
Connected.
SQL> create view myview
  2  as
  3* select * from officework;

View MYVIEW created.
```

```
SQL> select * from myview;

   EMPID ENAME        JOB              SAL HIREDATE        AGE
________ ___________ _________ _________ _____________ _____
     100 imran        dev           92400 28/OCT/23        21
     101 sumit        dev           99000 29/OCT/23        21
     103 ganesh       tester        88000 27/NOV/24        21
     104 akshay       dba           11000 01/JAN/24        22
     105 nikita
     106 rocky        devops         5500
                      tester
         ganesh                     88000
    1995 santosh      mba            1650 04/DEC/24        31
    4545 ravi         testerr    74125.87 05/FEB/24        26
    9999 nikhillll    devops     74125.87 05/FEB/24        24
     118 mahi         devops     81538.46 03/FEB/24        25

12 rows selected.
```

Note:- if a simple view having group function, group by clause ,rownum,distinct ,joins etc then we can't perform DML operation through simple view base table.

```
SQL> insert into myview values (6666,'rohan','recept',823.20,'05-feb-24',26);

1 row inserted.
```

We can also create VIEW using syn:

Create or replace view viewname as select statement.

```
SQL> create or replace view officeview
  2  as
  3* select * from officework;

View OFFICEVIEW created.

SQL>
```

**Complex view:-** a view said to be complex ,if based on multiple tables , if query contains group by clause ,having clause ,distinct clause ,aggregate clause,aggregate function,subqueries.

```
SQL> create view empview as
  2  select e.empno,e.ename,e.esal,e.deptno,d.dname,d.addr
  3* from empmgmt1 e INNER JOIN empdept1 d on e.deptno=d.deptno;

View EMPVIEW created.
```

```
SQL> select * from empview;

    EMPNO ENAME           ESAL     DEPTNO DNAME    ADDR
_________ ________ ____________ __________ ________ _________
     2222 akash       40632.12       1003 hr       mumbai
     1111 akash       40632.12       1002 test     pune
```

->Generally in oracle we cannot perform DML operation through Complex View to base table.

**INDEX :-**

->index is also a database object created to improve performance of data accessing .it is used to retrieve data very fast from database ,that's why indexes are used to improvement of query.

->index in database is similar to index in textbook, in textbook using index a particular topic can be located fastly ,in database using index a particular record can be located fast.

->indexes are created on columns and that column is called index key.

->indexes are created on columns.

**Indexes are created in 2 ways**

1.Automatically

2.Manually

Automatically:- whenever we create primary key or unique key then oracle server automatically creates "B-tree" indexes on those columns.

Manually:- by using create index command.

**Syn:** CREATE INDEX <NAME> ON <TABNAME>(COLNAME) ;

```
SQL> select * from officework;

   EMPID ENAME        JOB                SAL HIREDATE        AGE
________ ___________ ___________ ___________ _____________ _____
     101 sumit        dev               99000 29/OCT/23        21
     103 ganesh       tester            88000 27/NOV/24        21
     104 akshay       dba               11000 01/JAN/24        22
     105 nikita
     106 rocky        devops             5500
                      tester
         ganesh                         88000
    1995 santosh      mba                1650 04/DEC/24        31
    4145 raaj         operator       45632.14 06/FEB/24        26
    4545 ravi         testerr        74125.87 05/FEB/24        26
    9999 nikhillll    devops         74125.87 05/FEB/24        24
    6666 rohan        recept           823.2 05/FEB/24        26
    9996 mansi
     118 mahi         devops         81538.46 03/FEB/24        25
```

```
SQL> create index i1 on officework(sal);

Index I1 created.
```

```
SQL> select * from officework where sal>5000;
SQL> select * from officework where sal<80000;
```

Internally whenever we request data by using 'where' clause or 'order by' clause then Oracle internally searching index . if oracle found indexed column then mechanism for retrieving data become very fast from the database.

If column's doesn't have any indexes then Oracle server internally uses full table scan to retrieve data from database.

```
SQL> select index_name,column_name from user_ind_columns where table_name='OFFICEWORK';

INDEX_NAME    COLUMN_NAME
_____________ _____________
I1            SAL
I2            HIREDATE
```

Above command is used to see which columns are indexed in table .

**SET Operators**

->Set operator are used to retrieve data from single or multiple tables.These operators are also called “Vertical joins”.

1.**Union**:-it return unique values and also automatically sorting data.

2.**Union All**:-it return unique + duplicate data (no automatic sorting)

3.**intersects:-** it return common values

4.**Minus**:- Values are in first query those values are not in second query

Ex

| EMPID | ENAME | JOB | SAL | HIREDATE | AGE |
|---|---|---|---|---|---|
| 101 | sumit | dev | 99000 | 29/OCT/23 | 21 |
| 103 | ganesh | tester | 88000 | 27/NOV/24 | 21 |
| 104 | akshay | dba | 11000 | 01/JAN/24 | 22 |
| 105 | nikita | | | | |
| 106 | rocky | devops | 5500 | | |
| | | tester | | | |
| | ganesh | | 88000 | | |
| 1995 | santosh | mba | 1650 | 04/DEC/24 | 31 |
| 4145 | raaj | operator | 45632.14 | 06/FEB/24 | 26 |
| 4545 | ravi | testerr | 74125.87 | 05/FEB/24 | 26 |
| 9999 | nikhillll | devops | 74125.87 | 05/FEB/24 | 24 |
| 6666 | rohan | recept | 823.2 | 05/FEB/24 | 26 |

```
SQL> select job from officework where empid=101
  2  union
  3* select job from officework where empid=106;

JOB
---------
dev
devops
```

Using union for multiple table

Note:-whenever we use set operators always corresponding expression must belong to same data type .

```
SQL> select * from empmgmt1;

   EMPNO ENAME           ESAL    DEPTNO
________ ________ ___________ _________
    1111 akash       40632.12      1002
    2222 akash       40632.12      1003
     333 raaj        41257.85      1004

SQL> select * from empdept1;

   DEPTNO DNAME     ADDR
_________ _________ _________
     1003 hr        mumbai
     1002 test      pune
     1005 devops    blr

SQL> select ename from empmgmt1 where ename='akash'
  2  union
  3* select dname from empdept1 where dname='hr';

ENAME
_______
akash
hr
```

```
SQL> select ename from empmgmt1 where ename='akash'
  2  union all
  3* select dname from empdept1 where dname='hr';

ENAME
________
akash
akash
hr
```

```
SQL> select ename from empmgmt1 where ename='akash'
  2  intersect
  3* select dname from empdept1 where dname='hr';

no rows selected
```

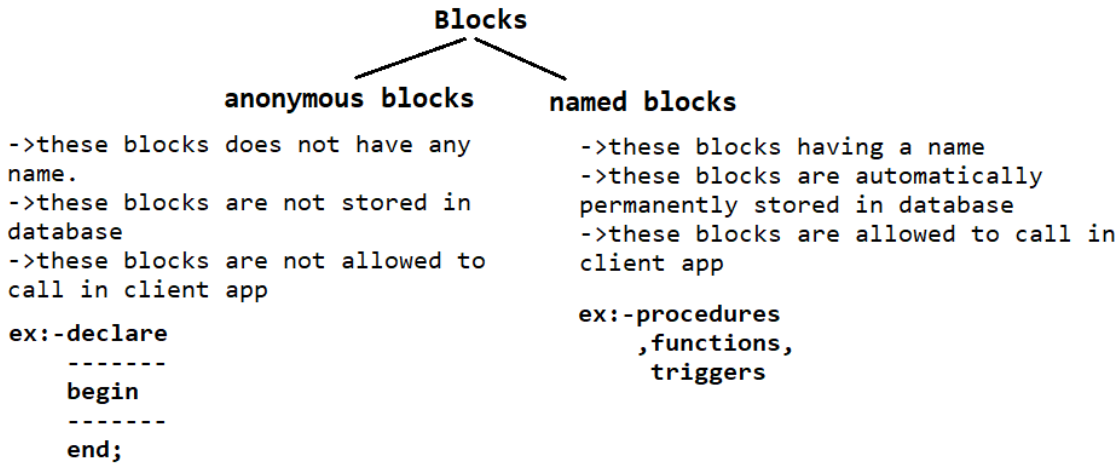
```
SQL> select ename from empmgmt1 where ename='akash'
  2  minus
  3* select dname from empdept1 where dname='hr';

ENAME
________
akash
```

## PL/SQL (Procedural Language/SQL)

->in sql at a time we submit only one command to oracle but in PL/SQL we submit group of commands to oracle. So in PL/SQL number of request and response between user and oracle are reduced and performance improved.

->basically ,PL/SQL is a block Structured Programming language.

->PL/SQL has two types of block.




## Datatypes in PL/SQL :-

1 NUMBER(P)/NUMBER(P,S)

2 CHAR/VARCHAR2/LONG/CLOB

3 NCHAR/NVARCHAR2/NCLOB

4 DATE/TIMESTAMP

5 BFILE/BLOB

6 BINARY_FLOAT/BINARY_DOUBLE

7 BINARY_INTEGER

8 BOOLEAN

1 TO 6 => allowed in **SQL**,**PL/SQL**

7,8 => allowed only in **PL/SQL** but not allowed in **SQL**

**Declaring variable**

**->syntax:-** variablename datatype(size);

**Assigning value to variable :-**

->using assignment operator **:=** we can store value in variable

**Display/print message or variable value.**

**Syntax:-** dbms_output.put_line('hello');

**Syntax:-**dbms_output.put_line(variable);

To enable output statement first on server output.

**Command**:-set serveroutput on

```
SQL> set serveroutput on;
SQL> declare
  2  name varchar2(50);
  3  age number(10);
  4  addr varchar2(50);
  5  begin
  6  name:='codemines';
  7  age:=200;
  8  addr:='india';
  9  dbms_output.put_line(name);
 10  dbms_output.put_line(age);
 11  dbms_output.put_line(addr);
 12  end;
 13* /
codemines
200
india
```

**Q)write a program to add two numbers?**

```
SQL> declare
  2  a number(10);
  3  b number(10);
  4  c number(10);
  5  begin
  6  a:=100;
  7  b:=200;
  8  c:=a+b;
  9  dbms_output.put_line(c);
 10  end;
 11* /
300
```

```
SQL> declare
  2  a number(10):= 50;
  3  b number(10):=20;
  4  total number(10);
  5  begin
  6  total:=a+b;
  7  dbms_output.put_line(total);
  8  end;
  9* /
70
```

## How to input values at runtime :-

->to input values at runtime use variables prefixed with "&".

```
SQL> declare
  2  a number(10);
  3  b number(10);
  4  c number(10);
  5  begin
  6  a:=&a;
  7  b:=&b;
  8  c:=a+b;
  9  dbms_output.put_line(c);
 10  end;
 11* /
Enter value for a: 300
Enter value for b: 400
```

Q)write a prog to input  date and print day of the week?

```
SQL> declare
  2  mydate date;
  3  begin
  4  mydate:='&mydate';
  5  dbms_output.put_line(to_char(mydate,'day'));
  6  end;
  7* /
Enter value for mydate: 07-feb-24
```

Q)write a program to display salary from empmgmt table?

```
   EMPNO ENAME           ESAL    DEPTNO
________ __________ ___________ _________
    1111 ravi         45632.12      1001
    2222 akash        40632.12      1002
    3333 rahul        40032.12      1003
    4444 varsha       40011.12      1004

SQL> declare
  2  name varchar2(50);
  3  id number(4);
  4  salary number(7,2);
  5  begin
  6  name:='&name';
  7  id:=&id;
  8  select esal  into salary from empmgmt where empno=id and ename=name;
  9  dbms_output.put_line(salary);
 10  end;
 11* /
Enter value for name: ravi
Enter value for id: 1111
```

## conditional statements :-

->if-then-else

->multiple  if

->nested if

| if | multi if | nested if |
|---|---|---|
| if cond then<br>statements;<br>else<br>statements;<br>end if; | if cond1 then<br>statements;<br>elsif cond2 then<br>statements;<br>elsif cond3 then<br>statements;<br>else<br>statements;<br>end if; | if cond then<br>if cond then<br>statements<br>else<br>statements<br>end if;<br>else<br>statements<br>end if ; |

```
   EMPNO ENAME            ESAL    DEPTNO
_______ _________ _____________ _________
    1111 ravi         45632.12       1001
    2222 akash        40632.12       1002
    3333 rahul        40032.12       1003
    4444 varsha       40011.12       1004

SQL> declare
  2  eno number(4);
  3  name varchar2(50);
  4  begin
  5  eno:=&eno;
  6  select ename into name from empmgmt where empno=eno;
  7  if name='akash' then
  8  dbms_output.put_line('welcome akash');
  9  else
 10  dbms_output.put_line('other...');
 11  end if;
 12  end;
 13* /
Enter value for eno: 2222
```

Ans : welcome akash

## loops in pl/sql :-

1.simple loop

2.while loop

3.for loop

**simple loop**

```
loop
   statements;
   exit when cond;
end loop;
```

**while loop**

```
while(cond)
loop
   statements;
end loop;
```

**for loop**

```
for <var> in low..upp
loop
  statements;
end loop;
```

if cond=true loop continues
if cond=false loop terminates

```
SQL>  DECLARE
  2    x NUMBER(2) := 1;
  3   BEGIN
  4    loop
  5       dbms_output.put_line(x);
  6       x := x+1;
  7       exit when x>8;
  8    end loop;
  9   END;
 10*   /
1
2
3
4
5
6
7
8
```

```
SQL>  DECLARE
  2    x NUMBER(3) := 1;
  3   BEGIN
  4    WHILE(x<=8)
  5   loop
  6      dbms_output.put_line(x);
  7      x := x+1;
  8   end loop;
  9   END;
 10*   /
1
2
3
4
5
6
7
8
```

```
SQL> BEGIN
  2     FOR x in 1..6
  3  loop
  4     dbms_output.put_line(x);
  5  end loop;
  6  END;
  7* /
1
2
3
4
5
6
```

**PROCEDURES :-**

->a procedure is a named PL/SQL block that accepts some input performs some action on database and may or may not return a value.

```
syntax:-
CREATE OR REPLACE PROCEDURE <NAME>
 (
  parameters if any
 )
 IS
   <declaration-part>;
 BEGIN
   statements;
 END;
  /
```

->In oracle ,whenever we are using create or replace keyword in front of procedure then those procedures are internally automatically,permanently stored in database that's why these procedures are also called 'stored procedures'.

**parameters :-**

=> we can declare parameters and we can pass values to parameters

=> parameters are 3 types

1 IN

2 OUT

3 IN OUT

**IN :-**

=> always recieves value

=> default

=> read only

**OUT :-**

-=> always sends value

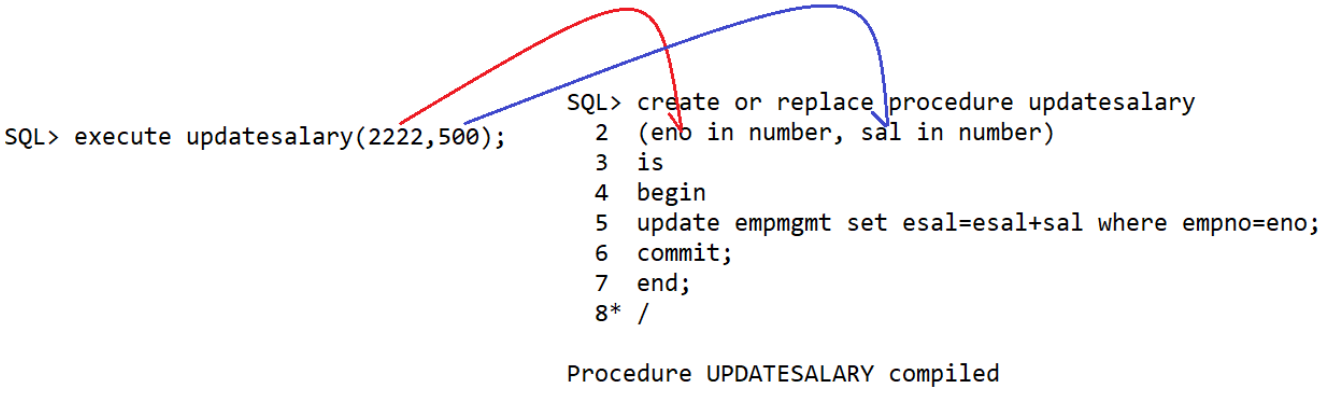
=> write only

**IN OUT :-**

=> recieves and sends

=> read & write



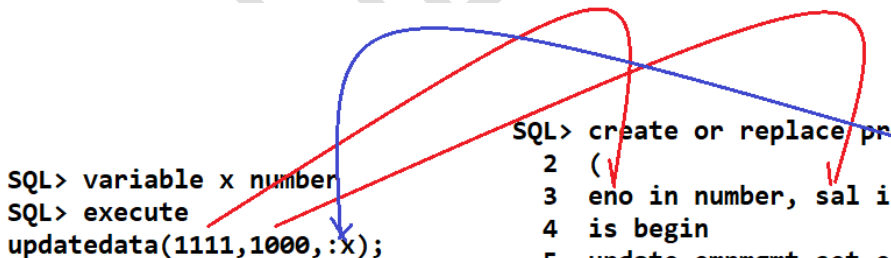


```
SQL> execute updatesalary(2222,500);
```

```
SQL> create or replace procedure updatesalary
  2  (eno in number, sal in number)
  3  is
  4  begin
  5  update empmgmt set esal=esal+sal where empno=eno;
  6  commit;
  7  end;
  8* /

Procedure UPDATESALARY compiled
```

```
   EMPNO ENAME            ESAL    DEPTNO
_______ _________ _____________ _________
    1111 ravi         45632.12       1001
    2222 akash        40632.12       1002
    3333 rahul        40032.12       1003
    4444 varsha       40011.12       1004

SQL> create or replace procedure updatesalary
  2  (eno in number, sal in number)
  3  is
  4  begin
  5  update empmgmt set esal=esal+sal where empno=eno;
  6  commit;
  7  end;
  8* /

Procedure UPDATESALARY compiled

SQL> execute updatesalary(2222,500);
```

```
SQL> select * from empmgmt;

   EMPNO ENAME            ESAL    DEPTNO
_______ _________ _____________ _________
    1111 ravi         45632.12       1001
    2222 akash        41132.12       1002
    3333 rahul        40032.12       1003
    4444 varsha       40011.12       1004
```



```
SQL> variable x number
SQL> execute
updatedata(1111,1000,:x);
```

```
SQL> create or replace procedure updatedata
  2  (
  3  eno in number, sal in number, x out number)
  4  is begin
  5  update empmgmt set esal=esal+sal where empno=eno;
  6  commit;
  7  select esal into x from empmgmt where empno=eno;
  8  end;
  9* /

Procedure UPDATEDATA compiled
```

```
   EMPNO ENAME            ESAL    DEPTNO
________ __________ _____________ __________
    1111 ravi          45632.12       1001
    2222 akash         41132.12       1002
    3333 rahul         40032.12       1003
    4444 varsha        40011.12       1004

SQL> create or replace procedure updatedata
  2  (
  3  eno in number, sal in number, x out number)
  4  is begin
  5  update empmgmt set esal=esal+sal where empno=eno;
  6  commit;
  7  select esal into x from empmgmt where empno=eno;
  8  end;
  9* /

Procedure UPDATEDATA compiled

SQL> variable x number
SQL> execute updatedata(1111,1000,:x);

PL/SQL procedure successfully completed.

SQL> print :x

         X
----------
  46632.12
```

create a procedure to format phone number ?

```
SQL>  CREATE OR REPLACE PROCEDURE format_phone
  2   (
  3     p IN OUT VARCHAR2
  4   )
  5   IS
  6   BEGIN
  7     p := '+1 ('||SUBSTR(p,1,3)||') '||SUBSTR(p,4,3)||'-'||SUBSTR(p,-4,4);
  8   END;
  9*   /

Procedure FORMAT_PHONE compiled

SQL>  DECLARE
  2     X  VARCHAR2(100) := '1234561234';
  3   BEGIN
  4     format_phone(X);
  5     dbms_output.put_line(X);
  6   END;
  7*   /
+1 (123) 456-1234


PL/SQL procedure successfully completed.
```

**USER DEFINE FUNCTIONS :-**

=> functions created by user are called 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 PL/SQL block that accepts some input performs some calculation and must return a value.

=> functions are created

1 for calculation

2 to fetch value from db

**syn :-**

```
CREATE OR REPLACE FUNCTION <NAME>(parameters if any) RETURN <type>
IS
 <declaration-part>;
BEGIN
  statements;
  RETURN <expr>;
END;
 /
```

```
SQL>  CREATE OR REPLACE FUNCTION CALC(a  NUMBER,b NUMBER,op CHAR) RETURN NUMBER
  2   IS
  3   BEGIN
  4      IF op='+' THEN
  5       RETURN (a+b);
  6      ELSIF op='-' THEN
  7       RETURN(a-b);
  8      ELSIF op='*' THEN
  9       RETURN(a*b);
 10      ELSE
 11       RETURN(a/b);
 12      END IF;
 13   END;
 14*  /

Function CALC compiled

SQL> select calc(10,20,'*') from dual;

   CALC(10,20,'*')
__________________
               200
```

**TRIGGERS :-**

=> a trigger is also a named PL/SQL block like procedure but executed implicitly by oracle whenever user submits DML/DDL commands.

=> triggers are created

1 to control dml/ddl

2 to enforce complex rules & validations

3 to audit tables

4 to manager replicas (duplicate copy)

5 to generate values primary key columns

**syntax :-**

```
CREATE OR REPLACE TRIGGER <NAME>
BEFORE/AFTER INSERT OR UPDATE OR DELETE
ON <TABNAME>
[FOR EACH ROW]
BEGIN
  STATEMENTS;
END;
/
```

create trigger to not to allow dmls on empmgmt table on wednesday ?

```
SQL>  CREATE OR REPLACE TRIGGER T1
  2   BEFORE INSERT OR UPDATE OR DELETE
  3   ON empmgmt
  4   BEGIN
  5     IF TO_CHAR(sysdate,'dy')='wed' THEN
  6         RAISE_APPLICATION_ERROR(-20001,'wednesday not allowed');
  7     END IF;
  8   END;
  9*   /

Trigger T1 compiled
```

```
SQL> update empmgmt set esal=200 where empno=1111;

Error starting at line : 1 in command -
update empmgmt set esal=200 where empno=1111
Error at Command Line : 1 Column : 8
Error report -
SQL Error: ORA-20001: wednesday not allowed
ORA-06512: at "SYSTEM.T1", line 3
ORA-04088: error during execution of trigger 'SYSTEM.T1'
```