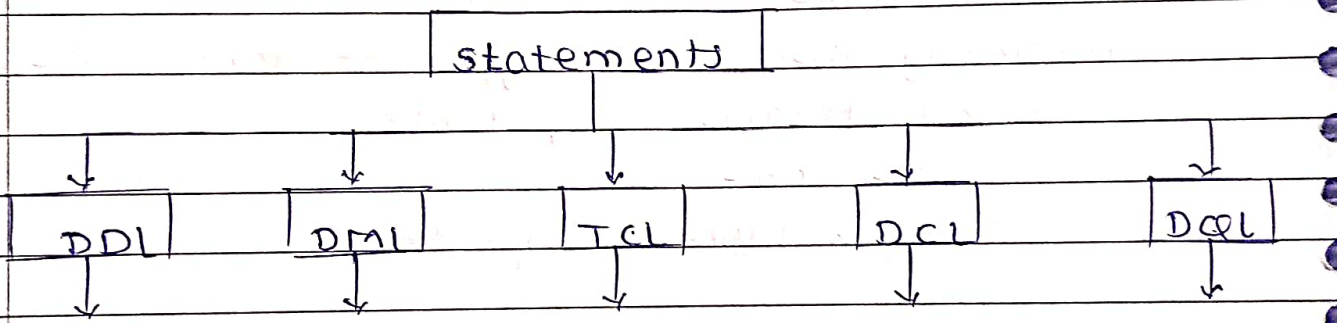
Date : 12-8-25

* Statements in SQL :-

- The statements are collection of all the multiple keywords which are used to perform CRUD operations.
- There are Five statements in SQL
  a) Data defination language
  b) Data manipulation language
  c) Transaction control language
  d) Data control language
  e) Data query language



a) Data defination language (DDL) :-

- It is used to CREATE, UPDATE or Destroyee database object
- The database objects are Procedure, views, Table, Triggers, Indexies, User define functions (UDF).

Date :

- There are five commands in DDL these are
  1) CREATE
  2) RENAME
  3) TRUNCATE
  4) DROP
  5) ALTER

1) CREATE :
- CREATE command is used to create database objects

• create table -
basically there are 3 types of tables these are
- Normal Table / Heap table
- Prime Table
- Complex Table / composite

(i) Normal Table - The table which does not consists of any primary key and foreign key are known as normal table.

• syntax = CREATE TABLE table_name
(
col_name1 Datatype [constraints],
col_name2 Datatype [constraints],
- ,
);

Note - The Table may have only one Primary key
- The Table name should not Repeats in same user
- The Table should not consists same columns
- The Table name / col_name shouldn't starts with any numbers or special characters

Date :

- If we are creating foreign key, we can have upto 255 F.K.
- The max number of columns 16,384 and number of Records 1,048,576

eg write a query to create the normal table table name = student, which consists of any 5 columns. Assign the proper datatype and constraints (No PK and Fk).

→

```
CREATE TABLE students
(
  SID NUMBER(3) NOT NULL,
  SName VARCHAR(30) NOT NULL,
  DOB DATE,
  Course VARCHAR(15) NOT NULL,
  MNO. NUMBER(10) CHECK(LENGTH(MNO.)=10)
);
```

Eg 2 CREATE cricket table without PK & FK which consists of 5 columns

→

```
CREATE TABLE CRICKET
(
  TEAM_NAME VARCHAR(20) NOT NULL,
  PLAYER_NAME VARCHAR(10),
  RUN_SCORED Number(5) Not NULL,
  No_OF_sixes Number(3),
  NO_OF_FOURS Number(3));
```

Rainbow

Date :

(ii) Prime table / parent table

The table which consists of primary key where the foreign key is optional is known as prime table or parent table

- Syntax -

```
CREATE TABLE TABLE_NAME
(
col_name1 datatype [constraints],
col_name2 datatype [constraints],
----------,
CONSTRAINT Refernce_Name PK (col_nam)
[constraint Reference Name FK (col_nam)
references Parent_table_Name (col_nam)]
);
```

eg write a query to create flight table with any 5 columns & use the proper datatype and make any one column as primary key [NO FK]

→

```
CREATE TABLE FLY
(
FNO Number,
FName VARCHAR (20) Not NULL,
DDate DATE Not null
ADate DATE Not null
NO_OF_Seats Number NOT NULL,
CONSTRAINT FNO_PK PRIMARY KEY (FNO)
);
```

Date:

Note:- To check All the constraints which used For table i.e SELECT *
FROM USER_CONSTRAINTS
WHERE TABLE_Name = 'table_name';

- P ⟶ Primary key
  C ⟶ Null / Not Null / CHECK
  R ⟶ Foreign key
  U ⟶ Unique.

- To display the structure of table
  DESCRIBE TABLE_NAME;

- To print the data from the table
  SELECT * FROM TABLE_NAME

- SET LINES Number PAGES number;
  no. should be more than 100 & less than 50000

Qn Write a query to create Hospital table by using any 6 columns assign proper datatypes and make HospitalID as primary key.

⟶

```
CREATE TABLE HOSPITAL
(
  HID Number,
  HName VARCHAR(20) NOT NULL,
  Location VARCHAR(20) NOT NULL,
  Bed_capacity Number,
  Doc_count Number Not null,
  Patients Number Not null
  Reference HID_PK Primary key (HID)
  CONSTRAINT
);
```

Rainbow

Date:

Q) write SQL query to create table named PIZZA ORDERS with columns
orderID, custName, Topping, Extracheese

→

```
CREATE TABLE PIZZAORDERS
(
  OrderID Number,
  CustName VARCHAR(50),
  Topping VARCHAR(50),
  Extracheese CHAR(1) check (Extracheese IN
                              ('Y', 'N')),
  Constraints OrderID_PK primary key (OrderID)
);
```

Date :

(iii) Complex table :

The table which consists of foreign key and prima key together is known as complex table

- Note - In order to create complex table the primary key is optional. Foreign key is mandetory
- whenever we wants to create the complex table it is mandetory we must have ano prime table
- To create complex table we have 2 method
  i) direct method - pk, Fk while creating table
  ii) indirect method - create both tables then add parent tbl col to child table then assign pk, Fk

① Direct method

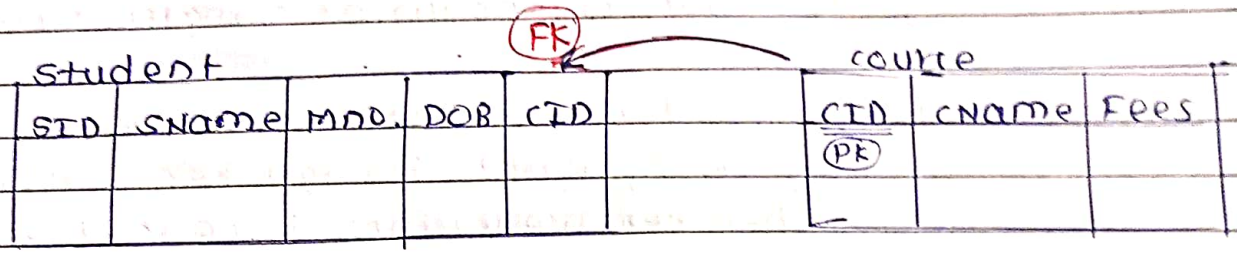
```
- CREATE TABLE TABLE_NAME
  (
  col_name1  Datatype [constraints],
  col_name2  Datatype [constraints],
  - - - - - - - - - - - - - - - -

  col_name (PF)  Datatype (PF) [constraints]
  (constraints Ref_name Primary key (colnam)),
  (constraints Ref_name Foreign key (colname)),
   References ParentTABLE_Name (colname])
  );
```

Date :

Q1 write a query to create the student table and a course table establish connection of both the tables (cource table is parent, student is child)

→



- parent table -

```
CREATE TABLE course
(
CID number NOT null,
CName varchar(20) null,
Fees number null check (Fees > 0),
constraint CID_PK primary key (CID)
);
```

- child table -

```
CREATE TABLE Student
(
SID number not null,
SName varchar(20) not null,
MNO number not null check (length(mno)=10),
DoB Date,
CID number not null
constraint SID_PK primary key (SID),
constraint CID_FK foreign key (CID)
references course (CID)
);
```

rainbow

Date :

Q1 write a query to create follow table

Product

PID	PName	Price	PDate	MDate	Gmail
should less than 4 digit num	varchar (20)	number > 0	date	date	FK varchar (30)

User (PK)

UName	Gmail	Addr	JDate
varchar (20)	varchar (30)	varchar (50)	Date

```
CREATE TABLE User
(
  UName varchar (20) not null,
  Gmail varchar (30),
  Address varchar (50) null,
  JDate Date not null,
  constraint GmailPK primary key (Gmail)
);

CREATE TABLE Product
(
  PID number check (length (PID < 4),
  PName varchar (20) not null,
  Price number check (price > 0),
  PDate Date,
  MDate Date,
  Gmail varchar (30),
  constraint PID_PK primary key (PID),
  constraint GmailFK Foreign key (Gmail)
  References user (Gmail)
);
```

Rainbow

Date :

Q) Write query to create following tables

user	Theater	movies
UID	TID PK	MID PK
Name	TName	MName
MNO PK	Taddr	Duration
Email	No_seats	Date
DOB	No_screen	s_time

→

```
CREATE TABLE Movies
(
MID number,
MName varchar(20) not null,
Duration varchar(10),
Date Date,
S_Time varchar(10) constraint MID_PK primary
key (MID)
);

CREATE TABLE Theater
(
TID number not null,
TName varchar(20),
Taddr varchar(50),
No_seats number not null,
No_screens number
MID number
CONSTRAINTS TID_PK primary key (TID),
CONSTRAINT MID_FK Foreign key (MID)
References movies (MID)
);
```

Date :

```
create table user
(
UID number,
UName varchar (20) not null,
MNO number check (length (MNO) = 10),
Email varchar (20),
DOB Date,
MID number,
CONSTRAINT MNO_PK primary key (MNO),
CONSTRAINT MID_FK Foreign key (MID)
References movies (MID)
);
```

Qn write query to create Follow table

players	Team	matches
PID	TID	MID
Name	TName	Date
Age	NO_Players	Time
MNO	NO_BATSMAN	stadius
Gender	NO_BOWLER	Addr

Date :

Q) write query to create follow table

players	Team	matches
PID	TID	MID
PName	TName	mDate
Age	Naplayer	mTimir
MNO	NO.bats	stadium
Gender	NO.bowlers	sAddr

→

```
CREATE table players (
    PID number,
    Pname varchar (50) not null,
    Age number,
    Mno number check (length (mno) = 10),
    Gender varchar (10),
    Constraint PID_PK primary key (PID)
    );

CREATE TABLE Team (
    TID number,
    TName varchar (20) not null,
    No_player number,
    No_bats number,
    No_bowlers number,
    PID number,
    CONSTRAINTS TID_PK primary key (TID),
    CONSTRAINT PID_FK Foreign key (PID)
    References players (PID)
    );
```

Date :

```
CREATE TABLE Matches (
    MID number,
    Mdate Date,
    Mtime varchar (10),
    stadium varchar (30),
    Addr varchar (30),
    TID number,
    CONSTRAINT MID_PK primary key (MID),
    CONSTRAINT TID_FK Foreign key (TID)
    Refrences Team (TID)
    );
```

ii) Rename :

- The second command in DDL is Rename
- This comment is use to change the name of existing Database Object
- Syntax

- RENAME OLD_Tab_Name to NEW_TAB_Name.

- eg RENAME Employees to Customers;

Date :

iii) TRUNCAIE :

- It is use to delect / Remove all the Records from the table
- Syntax

```
TRUNCAIE TABLE Table_Name ;
```

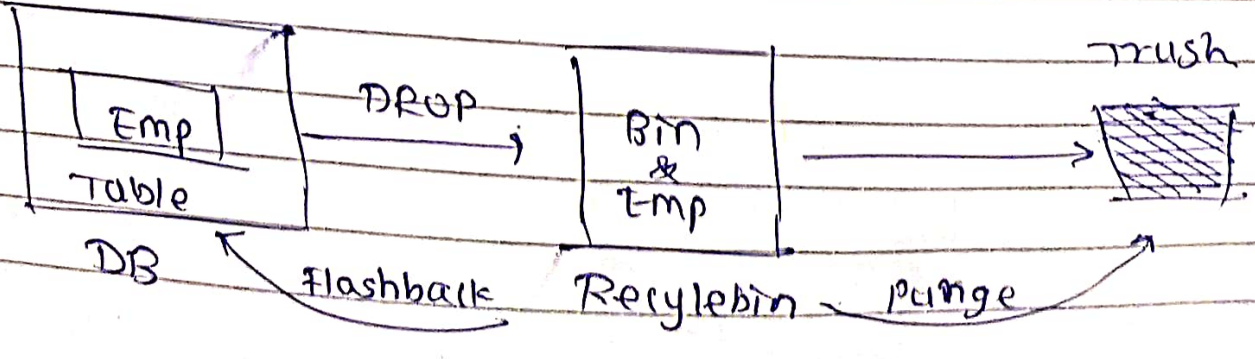
- Note - i) IF we use the truncate it will Remove, all the record from table permanently
  ii) Table structure will be remains same
  iii) while truncating parent table First we should remove the connection From parent table to child table

iv) DROP :

- This command is use to delete the table From the database
- syntax

```
DROP TABLE Table_Name ;
```

- IF we use drop command the perticular table will goes to recycle been
- form the re-cylle been if we wanted to brkined back to the database then we have to perform the operation called as Flashback
- IF we wanted to delete the respective table from the recylle been also we can perform the purge operations



syntax - Flashback

Date :

1) Flashback table table_Name
- Before Drop

2) syntax for purge
purge table table_name

3) To check the table from recyclebin
SELECT * FROM Recyclebin;

4) If we wanted drop the parent table first we need to resume the foreign key connection from then

DROP TABLE emp;

DROP TABLE bonus;

purge Table bonus;

Flashback Table emp to Before Drop

SELECT * FROM recyclebin;

v) ALTER:
- It is use to modify the structure of table
- There are 4 operations in ALTER

① Add column -
Syntax
ALTER TABLE table_name
Add col_name datatname (constraints);

Date :

- If the table is empty then we can add NULL or NOT NULL column

- Drop column
  Syntax : ALTER TABLE table_name ;
  DROP column column_name ;

Note - 1) If we drop column we don't have Flashback option

- Resume column
  Syntax : ALTER TABLE table_name
  RENAME column old_name to new_col_na

/*

ALTER TABLE Dept
Add Adhar_no. not null ;

ALTER TABLE Dept
Add Adhar_number Null ;

ALTER TABLE Dept
drop column MNO ;

ALTER TABLE Dept
Rename column DName to SQL.

*/

Date : 21-8-25

4) Change Datatype of column.

- Syntax

ALTER TABLE TABLE_NAME
MODIFY COL_NAME NEW DATATYPE

- SQL will not support for type casting
- If one column consists of one type of datatype to change the datatype of that column is not possible
- we can increase the size of Datatype of coln.
- eg ALTER TABLE Dept
MODIFY MNO. DATE;

5) Make column as NULL to Not NULL / not NULL to NULL

- Syntax

ALTER TABLE TABLE_NAME
MODIFY COL_NAME Datatype Not NULL / NULL;
(Write which we req.)

- eg ALTER TABLE Bonus
MODIFY EName VARCHAR(10) Not NULL;

ALTER TABLE Bonus
MODIFY EName VARCHAR(10) NULL;

* Note :) converting from null to not null / not null to null if it is PK column it is very difficult.

Rainbow

Date :

6) DROP constraints :

- syntax

- ALTER TABLE TABLE_NAME
  DROP constraint REF_NAME ;

eg ALTER TABLE EMP
DROP constraint FK_DEPTNO ;

7) To make any column as PK and FK

- syntax

For PK
ALTER TABLE TABLE_NAME
ADD constraint REF_Name Primary key (COL_Name);

For FK
ALTER TABLE TABLE_NAME
ADD constraint REF_Name Foreign key (col_name)
References Parent_TAB_Name (col_name);

Note : IF I want to add any constraint to the column
ALTER TABLE TABLE_NAME
ADD constraint REF_NAME constraint_name (col_name);

inbow

Date: 22-8-25

* DML (Data Manipulation Lang)

- DML is used to insert / update / delete the data from the database objects.
- There are three command in DML

1) DELETE : It is used to delete the records from the table
   - Syntax

```
DELETE FROM TABLE_NAME
WHERE condition ---- optional
```

2) Update : It is used to update the data from the table
   - syntax

```
UPDATE TABLE NAME
SET COL-Name1 = V1 , col_name2 = V2, ----
COL_Name n = Vn
WHERE condition ---- optional
```

3) INSERT : use to insert the data / Records into the table
   - syntax

```
INSERT INTO TABLE_NAME values (V1, V2, --Vn);
                    or
INSERT INTO TABLE_NAME values (&colName1,
                    & colname2, --- & colnamen);
                    or
INSERT ALL into TABLE NAME1(colname1, colname2,
                    --- colnamen) values (V1,V.N
                    or
Insert All into table_name2(colnam1, CN2, ---
                    colnamen) values (V1, V2, ----n).
select * from dual ;
```

Rainbow

Date :

Note - All the DDL commands are autocommit commands where DML are non Autocommit.

- Autocommit :
The operation will get save automatically
- Non-autocommit
we need to save operations manually.

* 1) while inserting the data into the parent table just we have to look at no. of coln. & datatype of a column.
2) while inserting the data into child table the FK value which we are going to insert that value should be present in the parent table as PK
3) while deleting the data from child table we dont have any Restrictions. but while deleting data from parent table 1st we have to remove the connection between tables.

- while updating parent table's primary key it will become very difficult bz of the reference
- If we delete the FK connection easily we can update PK of parent table and FK of child table

Rainbow

Date:

* TCL (Transaction control language)

- It is used to control the transactions in SQL
- the transactions in SQL are all DML operations (Insert, delete, Update).
- there are 3 commands in TCL

(con.1) a) COMMIT - Use to save the transaction in DB
syntax - COMMIT;

(con.2) b) ROLLBACK - use to Getback transaction Before commit
syntax - ROLLBACK;

c) Savepoint - use to control transaction in level wise
syntax - SAVE_POINT_NAME;

- To control the transaction By savepoints
ROLLBACK / COMMIT TO SAVEPOINT_Name;

Date :

Q.n Create Boys (parent) table & Girls (child) table
insert 6 boys records, 18 girls
- At least 2 boys should not have any girlfriend
- At least 2 girls not have any BF (BID NULL)
- Rest maintain one to many (eg Boy1 - 4 Girls, Boy2 - 3 girls, Boy3 - 5 girls)

→

```
CREATE TABLE Boys (
        Bid number ~~PRIMARY~~,
        Bname varchar(20) NOT NULL
        Age number,
        Height number,
        type varchare(20),   constraints Bid_Pk ~~primary key (Bid),~~
);

CREATE TABLE Girls (
        Gid number ~~NOT NULL~~,
        GName varchar(20),
        Age number,
        Height number,
        type varchar(20),
        Bid number
        CONSTRAINTS Gid_Pk primary key (Gid),
        CONSTRAINTS Bid_Fk Foreign key (Bid)
        References Boys (Bid)
);

INSERT INTO Boys values
(1, 'Amit', 21, 170, 'cool'),
(2, 'Rohan', 22, 175, 'funny'),
(3, 'Kiran', 23, 180, 'serious'),
(4, 'Arjun', 22, 172, 'smart'),
```

ainbow

Date:

(5, 'suresh', 21, 180, 'sporty'),
(6, 'vikram', 23, 178, 'charming');

INSERT INTO girls values
// 2 girls should not have any BF

(1, 'Priya', 20, 160, 'Cute', NULL),
(2, 'sneha', 21, 162, 'Bold', NULL),
// Boy 1 → 4 girls

(3, 'Ankita', 20, 158, 'Sweet', 1),
(4, 'komal', 22, 165, 'kind', 1),
(5, 'meena', 21, 163, 'Friendly', 1),
(6, 'Neha', 22, 159, 'Stylish', 1),

// Boy 2 → 3 girls
(7, 'Rutu', 21, 161, 'calm', 2),
(8, 'shreya', 20, 164, 'Funny', 2),
(9, 'Nisha', 22, 160, 'smart', 2),

// Boy3 → 5 girls
(10, 'Divya', 21, 167, 'chill', 3),
(11, 'Pooja', 22, 162, 'cute', 3),
(12, 'Aarti', 21, 165, 'modern', 3),
(13, 'sonali', 20, 163, 'cool', 3),
(14, 'kajal', 22, 166, 'sweet', 3),

// Boy4 → 2 girls
(15, 'sapna', 21, 159, 'simple', 4),
(16, 'kiran', 22, 164, 'sporty', 4),
// Boy 5 → 2 girls
(17, 'sara', 20, 161, 'friendly', 5),
(18, 'Nandini', 21, 162, 'cute', 5);

Boy 6 does not have any GF

Rainbow

Date : 25-8-25

* DCL (Data control language):

- This statement is used to control the data flow between multiple users (It is very similar to our gmail access)
- There are two commands in DCL
  - (i) grant:
  - (ii) revoke:

i) grant-

- The statement which is used to give the access to the multiple user is known as grant cmd.
- Syntax:

```
grant sql_statements
on Table_name
to user_name;
```

ii) revoke:

- This command is used to get the permissions from the users.
- syntax:

```
revoke sql_statements
on Table_name
from User_name;
```

Note - 1) select -- For viewer
2) Insert / update / delete -- Editor

Rainbow

: Connect
: show user

Date:

SELECT * FROM USER_TAB_PRIVS ;
(to view that which access we are given to the another user)

1) To create multiple tables and to insure the many-to-many connectivity between all the multiple tables we have to create one junction table which consists of only the foreign keys
2) In the junction table if we are connecting two parents table, 2 foreign should present along with this we can also able to make this col as a primary key in junction table this is called as composite key attribute

- composite :
  including multiple columns if we are creating as a primary key those type of col known as composite key attribute

Date :

Q5 → Film and actor table (many to many)

```
CREATE TABLE Actor
(
 Aid number,
 AName varchar(20),
 Age number,
 MNO number check (length(MNO)=10),
 Net_worth number,
 CONSTRAINT Aid_PK primary key (Aid)
);

CREATE TABLE Films
(
 Fid number,
 FName varchar(30),
 Budget number,
 Duration varchar,
 RDate Date,
 CONSTRAINT Fid_PK primary key (Fid)
);

CREATE TABLE Actor_Film
(
 Aid number,
 Fid number,
 CONSTRAINT Aid_FK Foreign key (Aid)
 References Actor (Aid),
 CONSTRAINT Fid_FK foreign key (Fid)
 References Films (Fid)
);
```

Rainbow

Date: 27-8-25

- Note - i) we can create the table by using another table as a references (same structure of a table column, datatype, and constraints except primary key & FK) the Result table may be as empty table or as same data which is present in the reference table.

- As empty table -

```
CREATE TABLE Table_name
AS
SELECT col1, col2, col3 / * FROM TableNa
WHERE False condition;
```

eg - CREATE TABLE EMPCOPY
AS
SELECT * FROM EMP
WHERE 1 = 0;

- With data -

```
CREATE TABLE Table_name
AS
SELECT * FROM Table_name;
```

eg. CREATE TABLE EMPCOPY
AS
SELECT * FROM EMP;

Date :

* DQL (Data query lang)

- It is the process of Fetching the data from the table and displaying it
- There are 3 methods in a DQL.

1) Projection -
It is a process of Fetching the data from the table by selecting only the columns is known as projection

2) Selection -
It is the process of fetching the data from the table by selecting both rows and columns

3) Joins :
It is the process of Fetching data from multiple table simultaneously is known as joins.

- In the DQL there is a SELECT keyword which is used to display the data from the table.

Rainbow

Date :

* DQL (Data query lang)

- It is the process of fetching the data from the table and displaying it
- There are 3 methods in a DQL.
  1) Projection -
  It is a process of fetching the data from the table by selecting only the columns is known as projection

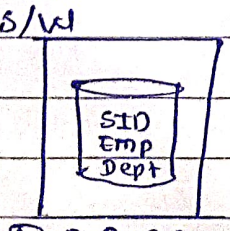
  2) Selection -
  It is the process of fetching the data from the table by selecting both rows and column

  3) Joins :
  It is the process of fetching data from multiple table simultaneously is known as joins.

- In the DQL there is a SELECT keyword which is used to display the data from the table.

a) Projection

- Projection is the process of fetching the data from the table by selecting only the columns
- Syntax



SELECT * / [DISTINCT] colname / Expres [ALIAS] ③
FROM TABLE Name;
① ②

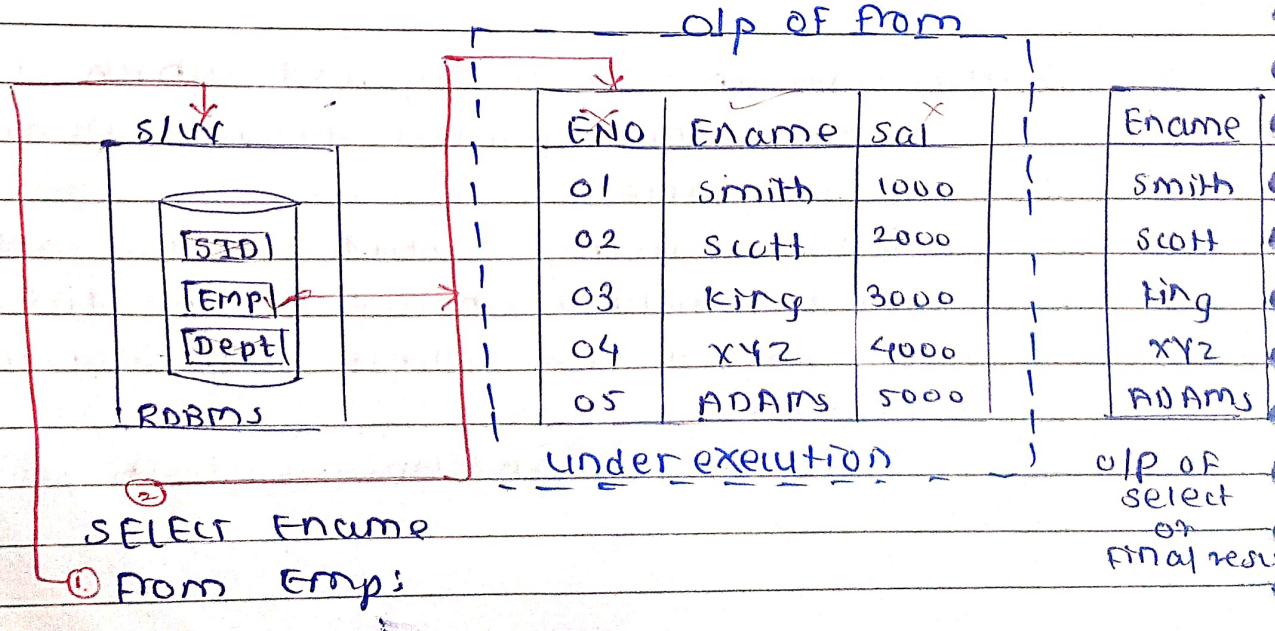
Date : 28-5-25

- Working procedure.

1) The From clause will execute
2) The From clause will goes to database and search for the table
   - If the table is present in database it will select the table and proceed table to next process
   - If the table is not present in the database From clause will rise error msg. "the table or view does not exist"
3) → after the from clause the select clause will execute, and it will display the final result
4) If the table consist an any data default the complete data will selected as a final result
5) If the table is completely empty the final Result will be no rows selected.

eg write a query to display all the Employees name from emp. table.

→



Rainbow

Date :

Q1 Write a q·t·D· salary of all the employees from employee table

→

```
SELECT Sal
FROM EMP ;    // display salary column
```

Q2 W·A·Q·T·D all the ename & Emp disignation and emp salary from emp table

→

```
SELECT EName, Job, Sal FROM EMP ;
```

Q3 WAQTD all Film names from Film table

→

```
SELECT FName FROM Films ;
```

Q4 WAQTD All details of all Books

→

```
SELECT * FROM Books ;
```

Q5 WAQTD Actors name, Actor age and networth of all actors

→

```
SELECT AName, Age, NetWorth FROM ACTOR ;
```

- Distinct : Distinct is a keyword which is used to remove the repeated records from the result table.
  1. while using distict keyword in the select clause distinct should be the first argument
  2. we can use distinct only one in a select clause
  3. the distinct and unique both are different

Rainbow

Date :

Q) W.A.Q.T.D The job of all the employees but there should not be any repeated records

→ SELECT Distinct Job FROM EMP ;

Q) W.A.Q.T.D. Employees job and their DNO. but there should not be any repeated records

→ SELECT Distinct Job, DNO FROM EMP;

Q) W.A.Q.T.D. All the details of Emp. but their should be any repeated records

→ SELECT Distinct * FROM EMP;

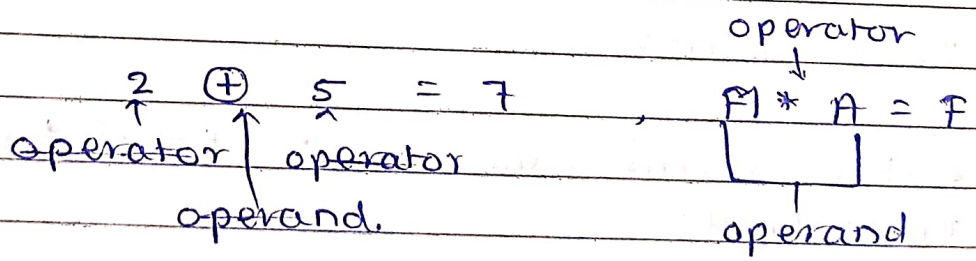
~~b)~~ Expressions :-

- The combination of operands and operator is known as expression

or

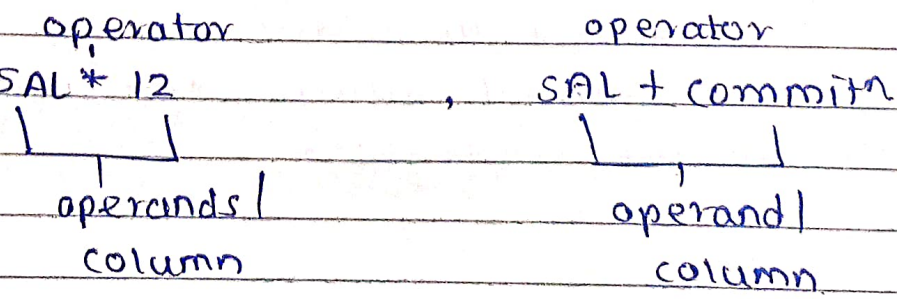
- The statement which will gives the result is known as expression

- eg



- In SQL instead of using operands we have to use the columns

- eg



Date :

eg · WAQTD emp name and emp annual salary from the emp table.

⟹

```
SELECT Ename, sal, sal*12 FROM EMP;
                or
SELECT Ename, sal*12 From EMP;
```

Qn W.Q.T.D Emp. Job, Emp salary, Name including 10% Hike in the salary

⟹

```
SELECT Ename, Job, salary, sal + salary*0.1
FROM EMP;
```

Qn W.a.Q.T.D. All the empNames, Ep DeptNO., salary 10% increment in Employes annual salary.

⟶

```
SELECT Ename, DeptNO, Salary, salary*12,
salary*12 + (sal + sal*0.1)
FROM EMP;

SELECT Ename, DNO, (sal + sal*0.1)*12
FROM EMP;
```

Qn W.a.Q.T.D. EmpName, Edisgnation, actual salary, emp Annual salary, Emp Half term salary including 10% increment in actual salary
10% in annual salary, 20% incr in Half term salary for all the employees.

Exp = sysDate - HDate

Date:

→ SELECT Ename, Job, salary, salary * 12,
Salary * 6, (sal + sal * 0.1), (sal + sal * 0.1) * 12,
(sal + sal * 0.2) * 6
FROM EMP;

Qn Write a Q.T.D. Employees HDate, Emp Experence, DeptNO, 20% duduction in Emp annual salary

→
SELECT HDate, Exp /* sysDate - HDate */ /365 (in term of years),
DeptNO, (sal - sal * 0.2) * 12
FROM EMP;

Qn W.A.Q.T.D. EName, Ejob, Esalary, 12% increment in actual salary, 6% diduction in Half term salary 20% diduction in quarter term salary.

→
SELECT Ename, Job, sal, sal + sal * 12/100,
(sal - sal * 6/100) * 6, (sal - sal * 0.2) * 8
FROM EMP;

Rainbow

Date :

- **ALIAS :**
- ALIAS is a alternative name which we can use for any column, expression or table inorder to print user friendly result
1) The alias name we can assign in 3 ways
   - using As keyword
   - using underline (_)
   - using double Quotes (" ")
2) Among the 3 ways " " way is most common usable one.

Qn1 W.a.Q.T.D Ename, Esalary, Emp annual salary, 20% diduct in annual salar, use ALIAS Name For to print user Friendly result

→

```
SELECT Ename, Salary, Salary*12 "Annual Sal",
       (Sal-Sal*0.2)*12 "Diducted Salary",
FROM EMP;
```

Qn2 W.A.Q.T.D. Esalary, E Annual salary, E experi in year 10% increment in annual, Edisignatn For all Emp

→

```
SELECT Salary, Salary*12 "Annual Sal",
(SysDate-HDate)/365 "Experience", (Sal+Sal*0.1)*12
Job FROM EMP;
```

inbow

Date: 1-9-25

b) Selection :-

- selection is the process of Fetching the data from the table by selecting both rowes and columns
- Syntax

SELECT * / [DISTINT] column / Expression [Alias]
FROM Table_name WHERE Filter_condition ;

- process
  1) FROM clause will execute ~~ptu~~ first.
  2) After the from clause, the where clause will execute.
  3) The where clause will executes row by row & check the condition for each row.
     - IF the condition is true then the record will be selected
     - IF the condition is false then the record will be rejected
  4) The output of the where clause is in the form of boolean value (true/False)
  5) After the where clause select clause will execute and it will print the final value or result
  6) In the where clause we can also use multiple filter conditions with the help of logical operators.

- Note - selection is also known as filteration method
  - Format of filter condition

colname / Exp	operator	values
• DNO	=	10
• sal	>	2000

Rainbow

Date :

Qn W.A.Q.T.D All the employees name, Emp. who are working in DeptNO 10

→

o/p of FROM

ENAME	SAL	DNO
smith	1000	10 ✓
scott	2000	30 X
king	3000	40 X
Allen	4000	null X
Adams	5000	10 ✓

o/p of WHERE

Ename	SAL	DNO
smith	1000	10
Adams	5000	10

DNO = 10
10 = (10) - T
30 = (10) - F

Result o/p of select

Ename
smith
Adams

③ SELECT Ename
① FROM Emp
② WHERE DNO = 10;

Qn W.A.Q.T.D all the details for emp. the emp who's salary are more than 3000

→

```
SELECT * FROM EMP
WHERE salary > 3000 ;
```

Qn All details of emp, the emp who are working as mager

→

```
SELECT * FROM Emp
WHERE Job = 'Manager';
```

- Note - The SQL is not a case sensitive lang - but data which we are going to use is case sensitive
- In SQL string data & date data, while perperforming CRUD operation we to mention within single coat (' ')

inbow

Date :

and for the date we must follow formcat.

Qn W.R.T.D all details of emp, the emp who Hired from 9th june 1981

→
```
SELECT * FROM Emp
WHERE HIREDATE = '9-jun-1981' ;
```

Qn Emp name, actual salary and 10% diduction in annual salary from for all emp, emp who hired after 1981

→
```
SELECT Ename, Sal, (Sal + Sal * 0.1) * 12 FROM Emp
WHERE HIREDATE > '31-Dec-1981' ;
```

Qn All details of emp the emp who hired befor 1985

→
```
SELECT * FROM Emp
WHERE HIREDATE <= '1-Jan-1985' ;
                <= '31-Dec-1984' ;
```

Qn W.R.T.D all the details OF smith

→
```
SELECT * FROM Emp
WHERE Ename = 'smith' ;
```

Qn W.R.T.D. All the details of employees, the emp who Reporting manager no. is 7839

→
```
SELECT * FROM Emp
WHERE Report_MGR = 7839 ;
```

Qn All the details of emp, emp who are working in Dept 10 and Dept-20

→
```
SELECT * FROM Emp
WHERE DeptNO = 10 OR DeptNO = 20 ;
```

Rainbow

Date :

Q1 all the details of emp. emp who are work as manager in depNo 30

→

```
SELECT * FROM Emp
WHERE Job = 'Manager' AND DeptNO = 30 ;
```

Q2 WAQTD. All details of emp, emp who's salarys are 3000 & 5000

→

```
SELECT * FROM Emp
WHERE sal = 5000 OR sal = 3000 ;
```

Q3 WAQTD. All details of emp. emp who's salarys are more than 1000 & less than 3000

→

```
SELECT * FROM Emp
WHERE sal > 1000 AND sal < 3000 ;
```

Q4 Display all details of emp. the emp who work as manager who work in Dept 10 & 20

→

```
SELECT * FROM Emp
WHERE Job = 'Manager' AND (DeptNO = 10,
 OR DeptNO = 20);
```

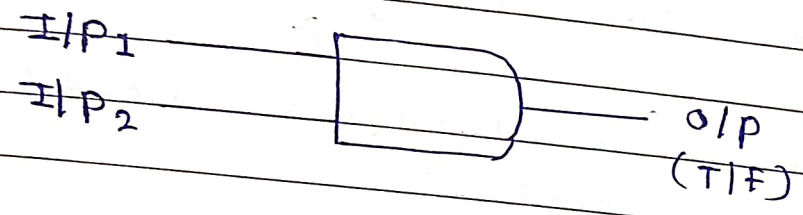
Rainbow

* Logical operators:

Date: 2-9-25

- The logical operators which used to pass multiple filter conditions in the where clause
- In SQL there are three logical operators
  A) AND
  B) OR
  C) NOT

A) AND - (*)



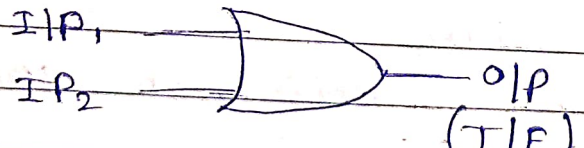
I/P1	I/P2	O/P
0	0	0
0	1	0
1	0	0
1	1	1

DNO = 10 AND sal > 2000

A	20	5000	X
B	10	1000	X
C	10	3000	✓

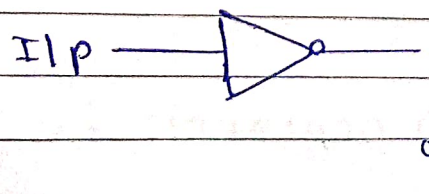
10 (T) & 3000 (T)

B) OR - (+)



I/P1	I/P2	O/P
0	0	0
0	1	1
1	0	1
1	1	1

C) NOT -



I/p	O/p
0	1
1	0

Date :

eg WRTD All the details of emp, the emp whos names are scott and smith

→

```
SELECT * FROM Emp
WHERE Ename = 'scott' AND Ename = 'smith';
```

eg WRTD All details of emp, emp who work as mang in Dept No. 30

→

```
SELECT * FROM emp
WHERE Job = 'manager' AND DeptNO = 30;
```

eg WRTD All details of emp except all emp who are work- ing in deptNo 30

→

```
SELECT * FROM Emp
WHERE NOT DeptNO = 30;
      DeptNo Not = 30;
      DeptNO != 30;
      DeptNo <> 30;
```

eg display All details of emp. emp who are work in Dept No 30 as manager earning salary less than 2900

→

```
SELECT * FROM Emp
WHERE (DeptNo = 30 AND Job = 'manager')
      AND Sal < 2900;
```

eg All emp details, except all emp who are work in Dept No 10 & Dept No 20

→

```
SELECT * FROM Emp
WHERE DeptNo != 20 AND DeptNo != 10;
```

Date :

Q) All details emp who's salary's are 3000, 5000, 1500 and 1600

→

```
SELECT * FROM Emp
WHERE Sal = 3000 OR sal = 5000 OR
      sal = 1500 OR sal = 1600 ;
```

Q) All details of emp, emp who work as cleark in DeptNO 30 and DeptNO 10

→

```
SELECT * FROM Emp
WHERE Job = 'cleark' AND (DeptNO = 30 OR
                          DeptNO = 10) ;
```

Date : 8/9/25

* Special Operators :

- The special operators which are used to perform special kind of task in SQL.
- In SQL we have several types of special operators.

1) IN vs NOT IN
2) IS vs IS NOT
3) like vs NOT like
4) Between vs NOT Between
5) Escape character
6) Regular Expression

1) IN vs NOT IN operator :

- In is a special operator which we can used to compare the values at the LHS side it will allow the column and RHS side it will allow the multiple values and it will generates final result as true if anyone value from RHS will satisfy the condn
- syntax

WHERE colname / exp IN ($V_1, V_2, V_3 \dots V_n$)

- eg DNO IN (10, 20, 30 ---)

- NOT IN - It is similar to IN operator instead of selecting the records it will reject.

WHERE colname/exp NOT IN ($V_1, V_2 \dots V_n$) ;

Date :

Q1 WAQD all the details of emp, emp who work as salesman. manager & clerk

→

```
SELECT * FROM Emp
WHERE Job IN ('salesman', 'manager', 'clerk');
```

Q2 WAQD all details of emp. emp whos names are smith, scott, allen & king

→

```
SELECT * FROM Emp
WHERE EName IN ('Smith', 'Scott', 'Allen', 'king');
```

Q3 WAQD all details of emp, except all emp who work in DNO 10 and 30

→

```
SELECT * FROM Emp
WHERE DeptNO NOT IN (10, 30);
```

Q4 all details emp, except all emp who Hired on 2nd April 81 and 8 Dec 81 and 23 May 87

→

```
SELECT * FROM
WHERE HiredDate NOT IN ('2-Apr-81',
                        '3-Dec-81',
                        '23-May-87');
```

Q5 All emp details emp work in deptNO 10,20,30 as salesman

→

```
SELECT * FROM Emp
WHERE Job = 'salesman' AND
deptNO IN (10, 20, 30);
```

Rainbow

Date :

Q) All details of emp. emp who work as manager except dNo. 10, 20, the emp who sal less than 5000

→

```
SELECT * FROM Emp
WHERE Job IN('manager') AND
      DepNO NOT IN (10,20) AND
      Sal < 5000;
```

Q) All emp details except all emp who's sal are 3000, 5000, 1500, 1600, emp who work as sales, manager & clerk in D-10, 20 & 30. except all emp Hired on 3-Dec-81

→

```
SELECT * FROM Emp
WHERE sal NOT IN (3000,5000,1500,1600) AND
      Job IN ('salesman','manager','clerk') AN
      Dept NO (10,20,30) AND
      HiredDate IN ('3-Dec-81');
```

NOT

Ans) IS VS IS NOT operator -

- These operator we can use to compare null values
- syntax

```
Colname / Exp  IS / IS NOT NULL ;
```

- Note
- Null means may be empty may not be
- Not null means not empty

Date :

eg WAID All the details of emp, emp who earning commission

→

```
SELECT * FROM Emp
WHERE COMM IS NOT NULL ;
```

Qn All details of emp, emp who are not having any reporting manager

→

```
SELECT * FROM Emp
WHERE MGR IS NULL ;
```

Qn All details of emp. emp who working in any dept (any)

→

```
SELECT * FROM Emp
WHERE DeptNo IS NOT NULL ;
```

Qn all emp detail, emp who work in Dept 10,20,30, exlept manager, clerk & salesman. if emp not earn any comm.

→

```
SELECT * FROM Emp
WHERE DeptNO IN (10,20,30) AND
      Job NOT IN ('manager','clerk','salesm')
      AND
      COMM IS NULL ;
```

Qn AH Ename emp sal & FAnnual salary For all the emp, emp whos annual sal are more than $6000

→

```
SELECT * FROM Emp
SELECT Ename, Sal, AnnualSal FROM Emp
```

Date : 9/9/25

3) Like vs Not Like :

- It is used to perform the pattern matching operations
- syntax

WHERE colname/Exp Like/Not like 'pattern';

1) In SQL to create patterns there are two special characters
   - % (modu.)
   - _ (underscore)

* % (properties)
- The % will allow n number of characters
- The % will allow any type of characters including %, _, no characters.

* _ (properties)
- It will allow only one digit of character
- It will allow any type of character including %, _, no character

Note :-
1) Like operator or Not like operator will not allow multiple conditions, to pass the multiple condition we can use logical operators

Qn WAQTD all emp details emp whos name ends with 'H'

→
```
SELECT * FROM Emp
WHERE EName Like '%H';
```

Rainbow

Date :

```
SELECT Ename, Salary, Salary*12 "Annualsal"
FROM Emp
WHERE Salary*12 > 36000;
```

OR

```
SELECT EName, Sal, Sal*12 "Annual"
FROM Emp
WHERE "Annual" > 36000;   // Wrong
```

- Note
In SQL Alias which we used in select clause we can able to access that alias name as condition in WHERE clause with the help of Inline subquery

eg

```
SELECT * FROM (
    SELECT Emp.*, Sal*12 Annual From Emp )
WHERE Annual > 36000;
```

Rainbow

Date :

Q) all details of emp. emp whos name starts with A and start with B

→
```
SELECT * FROM Emp
WHERE EName like 'A%' OR
      EName like 'B%' ;
```

Q) All emp details except all emp the emp whos name starts with A & B

→
```
SELECT * FROM Emp
WHERE EName NOT like 'A%' AND
      EName NOT like 'B%' ;
```

Q) All emp details. emp whos name does not starts with S and M

→
```
SELECT * FROM Emp
WHERE EName NOT like 'S%' AND
      EName NOT like 'M%'
```

Q) All details. emp whos name ends with R and end with H & G

→
```
SELECT * FROM Emp
WHERE Ename Like '%R' OR
      Ename like '%H' OR
      Ename like '%G' ;
```

Q) All emp details. emp whos name starts with vowels

→
```
SELECT * FROM Emp
WHERE Ename Like 'A%' OR
Ename like 'E%' OR
Ename like 'I%' OR Ename Like 'U%' ;
Ename like 'O%' OR
```

Rainbow

Date :

Q) All emp details emp whos name starts with T & ends with R

→
```
SELECT * FROM Emp
WHERE EName Like 'T%R'
      EName like 'T%' AND EName like '%R';
```

Q) WAFD all emp details the emp who earning 3 digits of salary

→
```
SELECT * FROM Emp
WHERE sal like '___' ;
```

Q) All emp details. the emp whos name Having A character

→
```
SELECT * FROM Emp
WHERE EName like '%A%' ;
```

Q) All emp details emp who hired in the month of march

→
```
SELECT * FROM Emp
WHERE Hiredate like '%Mar%' ;
```

Q) All emp detail excluding all emp who name start with S

→
```
SELECT * FROM Emp
WHERE EName NOT Like 'S%' ;
```

Q) All details. except all emp. who name having A char

→
```
SELECT * FROM Emp
WHERE EName NOT Like '%A%' ;
```

Rainbow

Assignment (special character)

Date:

Qn 1 All details of emp who work in Dept 10 & 20

```
SELECT * FROM Emp
WHERE DeptNo IN (10,20);
```

Qn 2 All details of emp who work as managers & salesmen

```
SELECT * FROM Emp
WHERE JOB IN (manager, salesman);
```

Qn 3 All details of emp who's sal is more than 1000 and less than 5000

```
SELECT * FROM Emp
WHERE sal>1000 AND sal<5000;
```

Qn 4 All details of emp who's DeptNo is 20 And working As manager

```
SELECT * FROM Emp
WHERE DeptNo=20 AND JOB='manager';
```

5) All details of emp. Name is scott and king

```
SELECT * FROM Emp
WHERE EName IN (SCOTT, KING);
```

6) Ename & Salary Who Hired After 81 into Dept 10

```
SELECT * FROM Emp
WHERE HireDate>81 AND DeptNo IN (10);
```

7) All emp details who sal is 3000 and 5000 ?

```
SELECT * FROM Emp
WHERE sal IN (3000,5000);
```

Rainbow

Date:

```
SELECT * FROM Emp
WHERE REGEXP_like (Ename, '^[A,E,I,O,U]');
```

(NOT) or

```
SELECT * FROM Emp
WHERE SUBSTR (Ename,1,1) IN ('A','E'---);
```

(Not)

Qn All details, emp whos name starts with consonants

```
SELECT * FROM Emp
WHERE EName NOT like 'A%' AND
      EName NOT like 'E%' AND
      EName NOT like 'I%' AND
      EName NOT like 'O%' AND
      EName NOT like 'U%';
```

Rainbow

Date :

8) Emp who work in DeptNo 10 as manager & Hired on 17-Dec-82

→
```
SELECT * FROM Emp AND
WHERE DeptNO = 10 ^ JOB = 'Manager' AND
      HiredDate = '17-Dec-82' ;
```

9) Emp who work as presidant in DeptNo 10 & Having salary of 5000.

→
```
SELECT * FROM Emp
WHERE JOB = 'Presidant' AND
      DeptNO = 10 AND sal = 5000 ;
```

10) Emp who work in DeptNo 10, 20 as manager

→
```
SELECT * FROM Emp
WHERE DeptNO IN (10, 20) AND
      JOB = 'Manager' ;
```

11) All emp Name and job emp who hired After 81 & Before 87

→
```
SELECT EName, Job FROM Emp.
WHERE HiredDate > 81 AND HiredDate < 87
```

12) All details of emp who's salary more than 1000 & less than 5000 in DeptNO 20

→
```
SELECT * FROM Emp
WHERE sal > 1000 AND sal < 5000 AND Dept = '20';
```

13) Emp who working As manager in DeptNO 20 & 10 And earn commission 400, 1600 ?

→
```
SELECT * FROM Emp
WHERE Job = 'Manager' AND
      DeptNO IN (10, 20) AND comm IN (400,
      1600);
```

Date :

14) Annual salary & Ename with Alias name for emp whos annual sal is more than 800 in Dept 30 work as salesman & Hired Before 88 ?

→
```
SELECT Sal*12 "Annual salary", Ename
FROM Emp
WHERE (sal*12) > 800 AND DeptNO = 30 AND
Job = 'salesman' AND HiredDate < 88 ;
```

15) EName as A, DeptNO as B and job as C for emp whose comm is more than 100 in Dept 10,20,30

→
```
SELECT Ename "A", DeptNO "B", Job "C"
FROM Emp
WHERE comm > 100 AND Dept IN (10,20,30);
```

16) Emp whose HireD. '03-Dec-81' And 17-Dec-81 in Dept 30

→
```
SELECT * FROM Emp
WHERE HiredDate IN ('03-Dec-81', '17-Dec-81')
AND DeptNO = 30;
```

17) Job & sal for emp whose comm is less than 2000 and salary are more than 1000 in dept 10,20

→
```
SELECT * FROM Emp
WHERE comm < 2000 AND sal > 1000
AND DeptNO IN (10,20);
```

18) Ename whose names are king, smith, Allen

→
```
SELECT Ename FROM Emp
WHERE Ename IN ('king', 'Smith', 'Allen');
```

19) Annual sal, Half term with 25% deduct, Emp who Annual sal is more than 6000 ?

→
```
SELECT sal*12 "Annual", sal * 0.25) AS
(sal-
"Half" FROM Emp WHERE (sal*12) > 6000 ;
```

Rainbow