UNIT-2: Advanced SQL

2.1 Data types:

- Each field has values of the same datatype.
- It is associated with a specific storage format, constraints and range of values.
- At the time of creating the table, there is a need to set datatype for the field.
- Following are the different data types:

No.	Data type	Description
1	NUMBER(p,s)	It is used to store zero, negative, positive, fixed and floating point numbers. The precision (P), determines the maximum length of the data, whereas the scale(S), determines the number of places to the right of the decimal. Number(6) — 123456 Number(6,2) — 1234.56 Number(6,-2) — 123500
2	CHAR(n)	It is used to store fixed length text up to 2000 bytes (characters).
3	VARCHAR(n)	It is used to store variable length text up to 2000 bytes (characters) Ex: varchar (5) and value is “Hi” then extra space (3) is padded to the right side. So in memory 5 bytes are used to store a single value. ANSI SQL standard External datatype
4	VARCHAR2(n)	It is used to store variable length text up to 4000 bytes (characters) Ex: varchar2 (5) and value is “Hi” then it uses only 2 bytes to store “Hi”. Oracle Standard Internal datatype
5	LOB data types	It is used to store and manipulate large block of unstructured data like text, image, audio or video
	CLOB	(Character Large Object) It is used to store large text whose maximum size is 128TB characters. It uses 1 byte/character. It is stored in form of ASCII character set.
	NCLOB	(National Character Large Object) It is used to store Unicode national character set data up to 128TB. It uses multiple bytes/character (Ex: 2 or 3 bytes per character) It is stored in form of Unicode character set.
6	LONG	It is used to store large text whose maximum size is 2GB characters. Normally it is used for internal purpose. Within Data Dictionary, it is used. Prefer to use CLOB in place of LONG
7	DATE	It is used to store date in format ‘dd-mon-yyyy’ or ‘dd-mon-yy’
8	RAW(n)	It is used to store hexadecimal values upto 32767 characters.
9	LONG RAW	It is used to store graphics, audio, text or array of binary data.

2.2 ROWID, Pseudo column & DUAL table

2.2.1 Pseudo Column

- Pseudocolumn behave like column table but it is not actually stored within table.
- We can select the value of Pseudocolumn but we cannot perform insert, update, delete operations on it.
- It is just used as read only.
- Different Pseudocolumn are listed below:
    - CURRVAL: Returns current value of sequence.
    - NEXTVAL: Returns next value of sequence.
    - NULL: Returns NULL value.
    - ROWID: Returns ROWID of row.
    - ROWNUM: Returns the ROWNUM of selected rows.
    - SYSDATE: Returns system date.
    - USER: Returns the name of current user.
    - UID: Returns the unique number assigned to the current user.
- Example:
    - Display the name of current user.
      SELECT USER FROM DUAL;

      ```
      USER
      --------------
      SYSTEM
      ```
    - SELECT SYSDATE FROM DUAL;

      ```
      SYSDATE
      ---------
      13-MAR-21
      ```
    - SELECT UID FROM DUAL;

      ```
            UID
      ---------
            5
      ```

2.2.2 ROWID & ROWNUM

ROWID

- Oracle assigns ROWID to each row of the table.
- ROWID is the column in form of hexadecimal value which returns the address of the row.
- ROWID is used by oracle for internal purpose. Within INDEX table of the table, this ROWID is stored to point the record.
- Once the record is inserted within table, one ROWID is assigned to this record. Now if user deletes any record then the related ROWID is not assigned to any other row.
- Example: SELECT ROWID, RNO, NAME FROM TBLSTUD;

ROWID	RNO	NAME
AAADzWAABAAAKXiAAA	101	Reet
AAADzWAABAAAKXiAAB	102	Ridham
AAADzWAABAAAKXiAAC	103	Maya
AAADzWAABAAAKXiAAD	104	Zankhana

ROWNUM

- Oracle also assigns ROWNUM to the selected records.
- ROWNUM can be changed query to query.
- Ex: SELECT ROWNUM, RNO, NAME FROM TBLSTUD;

ROWNUM	RNO	NAME
1	101	Reet
2	102	Ridham
3	103	Maya
4	104	Zankhana

- Ex: SELECT ROWNUM, RNO, NAME FROM TBLSTUD WHERE NAME NOT LIKE 'R%';

ROWNUM	RNO	NAME
1	103	Maya
2	104	Zankhana

- Before ORDER BY clause, the total numbers of records are selected so ROWNUM is assigned before ORDER BY clause. After ORDER BY clause, the ROWNUM are not modified.
- Ex: SELECT ROWNUM, RNO, NAME FROM TBLSTUD ORDER BY NAME;

ROWNUM	RNO	NAME
3	103	Maya
1	101	Reet
2	102	Ridham
4	104	Zankhana

- ROWNUM can also be used within condition using < and <= operators.
- Ex: SELECT ROWNUM, RNO, NAME FROM TBLSTUD WHERE ROWNUM<=3;

```
 ROWNUM        RNO NAME
------- ---------- -------
      1        101 Reet
      2        102 Ridham
      3        103 Maya
```

2.2.3 DUAL table

- DUAL is a small Oracle worktable with one row and one column within oracle database. DUAL table is created by SYSTEM owner but it is accessed by all other users.
- It is to perform mathematical calculation or retrieve date and it's formatting without using table, this a dummy table called DUAL is used.
- Description of DUAL table is: (DESC DUAL)

```
Name                                      Null?    Type
----------------------------------------- -------- ----------------
DUMMY                                              VARCHAR2(1)
```

- Ex: SELECT * FROM DUAL;

```
D
-
X
```

- Total numbers of records in DUAL table: (SELECT COUNT (*) FROM DUAL ;)

```
 COUNT(*)
---------
        1
```

- Ex: SELECT SYSDATE FROM DUAL;

```
SYSDATE
---------
22-MAR-21
```

- SELECT ROUND(12.45) FROM DUAL;

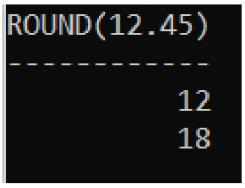
```
ROUND(12.45)
------------
          12
```

- User doesn't have rights to delete record of DUAL table.

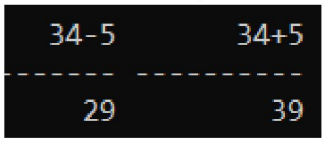
```
delete from dual
            *
ERROR at line 1:
ORA-01031: insufficient privileges
```

- Multiple records can also be displayed using DUAL.
- Ex: SELECT ROUND(12.45) FROM DUAL
  UNION ALL

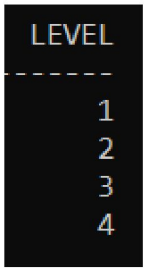
SELECT 5+6+7 FROM DUAL;



- Multiple columns can also be displayed using DUAL table.
- Ex: SELECT 34-5, 34+5 FROM DUAL;



- Multiple rows can also be displayed using DUAL.
- Ex: SELECT LEVEL FROM DUAL CONNECT BY LEVEL<5;



2.3 DATE Functions

No.	Function with Syntax
1	**SYSDATE** It is used to return the system date and time. Date arithmetic is possible using number constants or other dates. Only addition and subtraction are supported. Ex: SELECT SYSDATE FROM DUAL; =>22-MAR-21 SELECT SYSDATE +1 FROM DUAL;
2	**SYSTIMESTAMP** It is used to return the system date and time. Ex: SELECT SYSTIMESTAMP FROM DUAL;=>22-MAR-21 02.02.07.370000 PM +05:30
3	**TO_CHAR(date, [,fmt])** Converts a value of a DATE datatype to CHAR value.TO_CHAR() accepts a date, as well as the format(fmt) in which the date has to appear.Frm must be a date format.If format(fmt) is omitted, the date is converted to a character value using the default date format, i.e. “DD-MON-YY”. **Format must be following** ‘DD’ – day ‘MM’ – month ‘YY’ or ‘YYYY’ – year ‘HH’ – hours (0 to 11)

	'W' – Week of month Ex: SELECT TO_CHAR(SYSTIMESTAMP,'DD') FROM DUAL => 22 SELECT TO_CHAR(SYSTIMESTAMP,'MM') FROM DUAL => 03 SELECT TO_CHAR(SYSTIMESTAMP,'YY') FROM DUAL => 21 SELECT TO_CHAR(SYSTIMESTAMP,'YYYY') FROM DUAL => 2021 SELECT TO_CHAR(SYSTIMESTAMP,'HH') FROM DUAL => 02 SELECT TO_CHAR(SYSTIMESTAMP,'W') FROM DUAL => 4
4	**TRUNC(date,[fmt])** The TRUNC (date) function returns *date* with the time portion of the day truncated to the unit specified by the format model *fmt*.The value returned is always of datatype DATE, even if you specify a different datetime datatype for *date*. If you omit *fmt*, then *date* is truncated to the nearest day. Ex: SELECT TRUNC(TO_DATE('22-SEP-2014'),'MONTH') "Trunc_date" FROM DUAL; // 01-SEP-14 SELECT TRUNC(TO_DATE('12-MAR-2014'),'MONTH') "Trunc_date" FROM DUAL; // 01-MAR-14
5	**ROUND(date,[fmt])** Returns a date rounded to a specific unit of measure. If the second parameter is omitted, the ROUND function will round the date to the nearest day. Ex: SELECT ROUND(TO_DATE('22-SEP-2014'),'MONTH') "Round_date" FROM DUAL; // 01-OCT-14 SELECT ROUND(TO_DATE('12-MAR-2014'),'MONTH') "Round_date" FROM DUAL;// 01-MAR-14
6	**NEXT_DAY (date, char)** It returns the next date of given day available after given date. Ex: SELECT NEXT_DAY('20-MAR-21','THURSDAY') FROM DUAL; => 25-Mar-21
7	**LAST_DAY(DATE)** It is used to return the last day of month of specified date. Ex: SELECT LAST_DAY('20-MAR-2021') FROM DUAL; =>31-Mar-21
8	**MONTHS_BETWEEN(d1,d2)** Returns number of months between d1 and d2 as per 31 days per month. Also it returns the fractional part as a result. Ex: SELECT MONTHS_BETWEEN('21-mar-21','25-mar-21') FROM DUAL; -.12903226 Ex: SELECT MONTHS_BETWEEN('1-mar-21','1-apr-21') FROM DUAL; -1

	Ex: SELECT MONTHS_BETWEEN('1-apr-21','1-mar-21') FROM DUAL; 1
9	**ADD_MONTHS(date, no_months)** It returns the date after adding given numbers of month in specified date. Ex: SELECT ADD_MONTHS('20-MAR-21',5) FROM DUAL; =>20-AUG-21

2.4 Concepts of Index

- Index is a data structure associated with table which helps in improving performance of query processing.
- The primary key and unique key which are available within table are used as Index for table. (Automatically created by system).
- For foreign key, there is a need to explicitly create Index.
- Index is normally created for the columns which are frequently used during query processing. (Means the column which is frequently used within WHERE clause of query)
- One separate index file is generated from table. (If necessary, then and only then the index is created for table.)
- By default, oracle index store ROWID to identify each row uniquely.
- When the data in updated or deleted, the respectively the index file is also modified.
- Structure of Index file:

Search Key	Pointer

  - Search key: values prepared using one or more columns of the tables in ascending order.
  - Pointer: Store the address of the value where it is actually store in memory (ROWID).
- Ex: tblstudent table:

Address	Rno	Name	Phone_no
1	201	Manthan	123
2	205	Mohit	543
3	210	Meera	345
4	203	Ankit	765
5	206	Darshan	912

- Index file:

Name as search key	
Name	**Pointer**
Ankit	4
Darshan	5
Manthan	1

Phone_no as search key	
Phone_no	**Pointer**
123	1
345	3
543	2

Meera	3
Mohit	2

765	4
912	5

- Why Index is required?
  - It helps in searching specific information from table.
  - It increases the query processing performance.
  - It provides uniqueness.

2.4.1 CREATE index

- Using Create statement, the index can be generated.
- **Syntax:**

```
1. Single column index:
        CREATE INDEX index_name ON table_name (column_name);

2. Composite index:
        CREATE INDEX index_name ON table_name (column1, column2,…);

3. Unique index:
        CREATE UNIQUE INDEX index_name ON table_name (column_name);
```

- **Single column Index:**
  - It is created using single table column.
  - Ex:
    CREATE INDEX idxstud ON tblstud (name);
    It will create an index named idxstud on the column named ‘Name’ based on tblstud table.

- **Unique index:**
  - It is used for performance improvement as well as for data integrity also.
  - It doesn’t allow duplicate values to be inserted into the table.
  - Unique index is automatically created for the columns where primary key or Unique key constraint is set.
  - Ex:
    CREATE UNIQUE INDEX idxstud ON tblstud (name)
    It doesn’t allow to insert duplicate name within table.

- **Composite Index:**
  - It is used to set the index on two or more columns of a table. The most commonly used column is set first and then another choice is set.
  - Ex:
    Table name: tblstudresult(rno, name, cid, percentage)
    CREATE INDEX idxstud ON tblstudresult (cid, rno);

Here, the index is created with the combination of cid and rno.

2.4.2 DROP index

- If the index is no longer required then we have to delete the index file.
- To remove Index of table, DROP statement is used.
- Syntax:

  DROP INDEX index_name;
- Ex:

  DROP INDEX idxstud;

2.4.3 Disadvantages of index:

- When the table is updated, with each transaction, the index is also modified.
- If the created index is never used then it unnecessarily consumes resources.
- If many indexes are created for single table then also the performance is decreased with INSERT and DELETE.
- Extra space occupies within system because separate index file is created for each index.

2.5 Join Queries

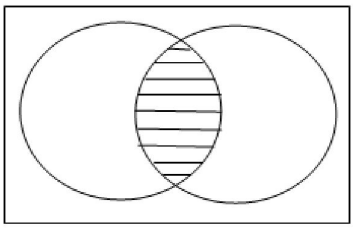
- SQL joins are used to retrieve data from multiple tables without repeating all the data. Tables are joined on columns that have the same data type and data width in the tables.
- Tables in a database can be related to each other with keys. A primary key is a column with a unique value for each row.
- **Tables:**

tblstudinfo			
Rno	**Name**	**Marks**	**Did**
1	Mayank	35	1
2	Maya	25	2
3	Mohit	45	1
4	Raghav	10	10
5	Ankit	49	3

tbldept	
Did	**Dname**
1	IT
2	Management
3	Science
4	Commerce
5	Arts

2.5.1 Inner Join

- When the tables are combined using Inner Join then it returns all those records which are matched with both tables. It is also known as Equi Joins.



- **ANSI Style (Only two tables can join)**

Syntax

```
SELECT <ColumnName1><ColumnName2>,<ColumnName N>
FROM <TableName1>
INNER JOIN<TableName2>
ON <TableName1>.<ColumnName1>= <TableName2>.<ColumnName2>
WHERE <Condition>
ORDER BY <ColumnName1>,<ColumnName2>,<ColumnNameN>
```

- **Theta style: (Can join multiple tables)**
- **Syntax**

```
SELECT <ColumnName1>,<ColumnName2><ColumnName N>
FROM <TableName1l><TableName2>
WHERE <TableName1>.<ColumnName1>= <TableName2>.<ColumnName2>
AND <Condition>
ORDER BY <ColumnName1><ColumnName2>,<ColumnNameN>
```

- **In the above syntax:**
  - ColumnName1 in TableName1 is usually that table's Primary Key
  - ColumnName2 in TableName2 is a Foreign Key in that table
  - ColumnName1 and ColumnName2 must have the same Data Type and for certain data types, the same size

Examples:

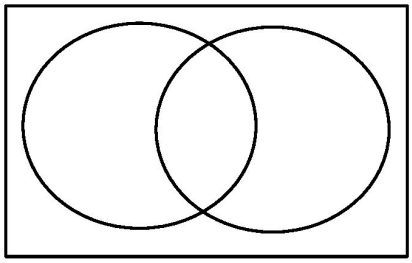
```
SELECT NAME, DNAME FROM TBLDEPT
INNER JOIN TBLSTUDINFO
ON TBLDEPT.DID= TBLSTUDINFO.DID;

SELECT NAME, DNAME FROM TBLDEPT, TBLSTUDINFO
WHERE TBLDEPT.DID= TBLSTUDINFO.DID;
```

NAME	DNAME
Mayank	IT
Maya	Management
Mohit	IT
Ankit	Science

2.5.2 Outer Join (Left, Right, Full)

1. **Left Join (Left Over Join)**
   - When the tables are combined using Left Join then all the records of Left tables are displayed.
   - If the related record's information is not available within another table then it returns NULL to those fields



   - **Syntax**

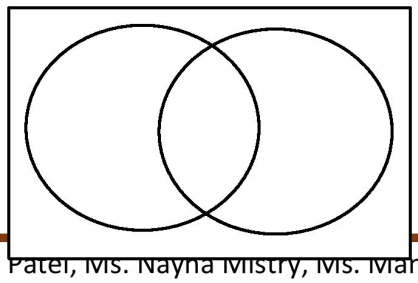
```
SELECT<ColumnName1><ColumnName2>,<ColumnName N>
FROM <TableName1>
LEFT JOIN<TableName2>
ON <TableName1>.<ColumnName1>= <TableName2>.<ColumnName2>
```

   - **Example:**

```
SELECT NAME, DNAME FROM TBLDEPT
LEFT JOIN TBLSTUDINFO
ON TBLDEPT.DID= TBLSTUDINFO.DID;
```

NAME	DNAME
Mayank	IT
Maya	Management
Mohit	IT
Ankit	Science
	Arts
	Commerce

2. **Right Join (Right Over Join)**
   - When the tables are combined using Right Join then all the records of Right tables are displayed.
   - If the related record's information is not available within another table then it returns NULL to those fields.



Syntax

```
SELECT<ColumnName1><ColumnName2>,<ColumnName N>
FROM <TableName1>
RIGHT  JOIN<TableName2>
ON <TableName1>.<ColumnName1>= <TableName2>.<ColumnName2>
```

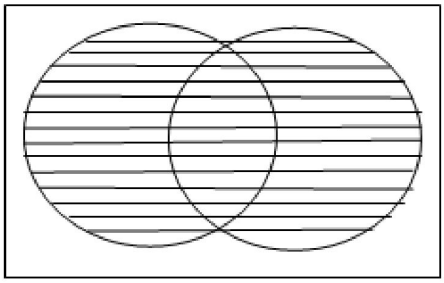
- **Example:**

```
SELECT NAME, DNAME FROM TBLDEPT
RIGHT JOIN TBLSTUDINFO
ON TBLDEPT.DID=TBLSTUDINFO.DID;
```

```
NAME       DNAME
---------- -------------
Mohit      IT
Mayank     IT
Maya       Management
Ankit      Science
Raghav
```

3. Full Join

- When tables are combined using Full Join then it displays all the records of both the tables and returns NULL to the fields where the respective information is missing.
- It gives combine result of LEFT and RIGHT Joins.



- **Syntax**

```
SELECT<ColumnNamel><ColumnName2>,<ColumnName N>
FROM <TableNamel>
FULL JOIN<TableName2>
ON <TableNamel>.<ColumnNamel>= <TableName2>.<ColumnName2>
```

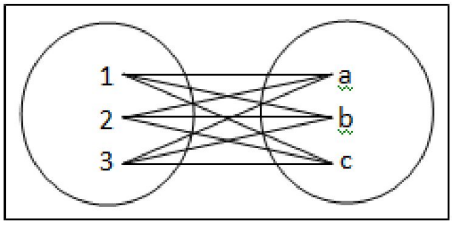
- **Example:**

```
SELECT NAME, DNAME FROM TBLDEPT
FULL JOIN TBLSTUDINFO
ON TBLDEPT.DID=TBLSTUDINFO.DID;
```

NAME	DNAME
Mayank	IT
Maya	Management
Mohit	IT
Ankit	Science
	Arts
	Commerce
Raghav	

2.5.3 Cross Join

- When tables are combined with Cross Join then it displays all records.
- Each record of first table is joined with all records of another table.
- If first table contains N records and second table contains M records then Cross Join table displays N*M records.



- **Syntax**

```
SELECT<ColumnNamel>,<ColumnName2><ColumnName N>
FROM <TableNamel>
CROSS JOIN<TableName2>
```

- **Example:**

```
SELECT NAME, DNAME FROM TBLSTUDINFO
CROSS JOIN TBLDEPT;
```

NAME	DNAME
Mayank	IT
Mayank	Management
Mayank	Science
Mayank	Commerce
Mayank	Arts
Maya	IT
Maya	Management
Maya	Science
Maya	Commerce
Maya	Arts
Mohit	IT

NAME	DNAME
Mohit	Management
Mohit	Science
Mohit	Commerce
Mohit	Arts
Raghav	IT
Raghav	Management
Raghav	Science
Raghav	Commerce
Raghav	Arts
Ankit	IT
Ankit	Management

NAME	DNAME
Ankit	Science
Ankit	Commerce
Ankit	Arts

2.5.4 Self Join (Not in Syllabus)

- Self Join combines the table with itself.
- Same table is renamed as twice and then these two tables are combined with each other.
- Example:
  SELECT A.RNO AS ARNO, B.NAME AS BNAME, A.MARKS AS AMARKS FROM STUD_INFO A, STUD_INFO BWHERE A.MARKS<B.MARKS;

ARNO	BNAME	AMARKS
1	Mohit	35
1	Ankit	35
2	Mayank	25
2	Mohit	25
2	Ankit	25
3	Ankit	45
4	Mayank	10
4	Maya	10
4	Mohit	10
4	Ankit	10

Important: Foreign key constraint:

- FOREIGN KEY constraint is used to specify the relation between multiple tables.
- This constraint is specified to the column of CHILD table which has relation between MASTER table's fields.
- This master table's field must contain either UNIQUE or PRIMARY KEY constraint.
- Records cannot be inserted into a detail table if corresponding records in the master table do not exist
- Records of the master table cannot be deleted if corresponding records in the detail table actually exist.
- The column in which we set the FOREIGN KEY constraint can be NULL or must contain the value which is available in MASTER table's respective field.

Syntax:
1. With CREATE statement:

```
CREATE TABLE tablename
(column definitions,
….,
FOREIGN KEY (columnname) REFERENCES tablename(columnname));
```

Example:

- DEPT table:
  create table dept (did int primary key, dname varchar2(15));
- STUD_INFO table with foreign key (Pointing to DID field of DEPT table)

```
create table stud_info
(rno int primary key, name varchar2(10),
marks int, did int,
FOREIGN KEY (did) REFERENCES dept(did));
```

OR

```
create table stud_info
(rno int primary key, name varchar2(10),
marks int, did int REFERENCES dept(did));
```

- DEPT table data entry:

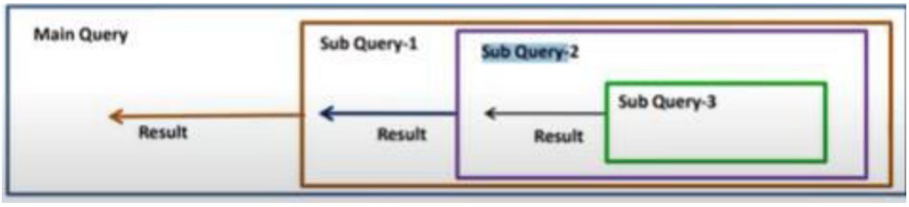
```
insert into dept values (1,'IT');
insert into dept values (2,'Management');
```

- STUD_INFO table data entry:

```
insert into stud_info values (1,'Mayank',35,1);
insert into stud_info values (2,'Maya',25,2);
insert into stud_info values (3,'Mohit',45,1);
insert into stud_info values (4,'Raghav',10,10);        //Error-Not allowed
insert into stud_info values (5,'Ankit',49,3);          //Error-Not allowed
```

2.6 Sub Queries /Nested Queries

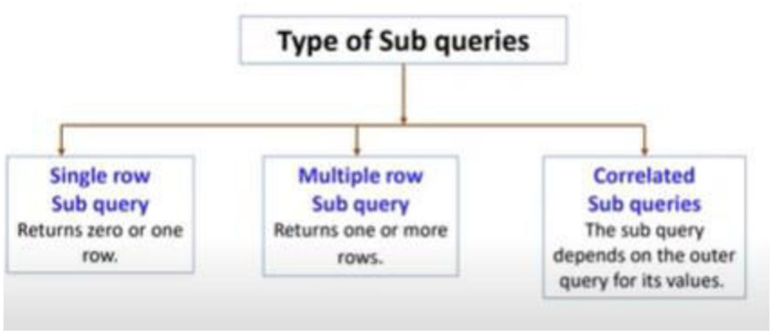
- The query which is written within another query is called **inner query or sub query.**
- The main query is called outer query and sub-query is called inner query. Inner query is written within parenthesis ().
- The inner query is evaluated first and then outer query is evaluated during execution.



- It works with INSERT, SELECT, UPDATE and DELETE.
- The Subquery can be written within WHERE, HAVING and FROM clauses. An order by clause cannot be used in a sub query.
- **Inline view:** If the query is nested within FROM clause of SELECT query then it is called inline view.
- Example: select a.* ,(select deptname from tbldept b where b.deptid=a.deptid) deptname from tblemp a;
- **Nested query:** If the query is nested within WHERE clause of SELECT statement then it is called nested subquery.
- **Syntax:**

  SELECT Columns FROM table_name WHERE Column_name Comparison_operator (SELECT Columns FROM table_name WHERE Condition);

Types of Subquery:



1. **Single row Subquery:**
   Inner query returns zero or one row.
   Single row comparison operators work with this. (=, <, >, <=, >=, <>).
   Example:
   Display the detail of students having highest marks
   SELECT * FROM TBLSTUDINFO WHERE MARKS = (SELECT MAX(MARKS) FROM TBLSTUDINFO);

RNO	NAME	MARKS	DID
5	Ankit	49	3

2. **Multi row Subquery:**
   Inner query returns one or more rows .
   Multi row comparison operators work with this. (IN, ANY, ALL).
   Example:
   Display the students detail of 'IT' department.
   SELECT * FROM TBLSTUDINFO WHERE DID IN (SELECT DID FROM TBLDEPT WHERE DNAME='IT');

RNO	NAME	MARKS	DID
1	Mayank	35	1
3	Mohit	45	1

3. **Correlated Subquery:**
   - Correlated sub query depends on data provided by outer query.
   Example:
   Display the departments' detail who has atleast one students.
   SELECT A.DNAME FROM DEPT A WHERE EXISTS (SELECT *FROM STUD_INFO WHERE DID=A.DID);

```
DNAME
------------
IT
Management
Science
```

Subquery with INSERT, UPDATE and DELETE:

- Subquery can also be written within INSERT, UPDATE and DELETE statements.
- Ex: Create STUDENT table:
  CREATE TABLE TBLSTUDENT (RNO INT PRIMARY KEY, NAME VARCHAR2(10), MARKS INT);

1. **INSERT:**
   Insert the detail in Student table from the Stud_info table.
   INSERT INTO TBLSTUDENT (SELECT RNO, NAME, MARKS FROM TBLSTUDINFO WHERE RNO<4);

```
 RNO NAME           MARKS
---- ---------- ----------
   1 Mayank             35
   2 Maya               25
   3 Mohit              45
```

2. **UPDATE:**
   Modify the students' marks with 25 whose detail is available in Stud_info table and which belongs to DID 1.
   UPDATE TBLSTUDENT SET MARKS= 25 WHERE RNO IN (SELECT RNO FROM TBLSTUDINFO WHERE DID=1);

```
 RNO NAME           MARKS
---- ---------- ----------
   1 Mayank             25
   2 Maya               25
   3 Mohit              25
```

3. **DELETE:**
   Delete the records of 'IT' department from student table.
   DELETE FROM TBLSTUDENT WHERE RNO IN(SELECT RNO FROM TBLSTUDINFO WHERE DID=(SELECT DID FROM TBLDEPT WHERE DNAME ='IT'));

```
 RNO NAME           MARKS
---- ---------- ----------
   2 Maya               25
```

Operators use with Subquery:

1. Comparison operators (<,<=, >, >=, =, <>)

- Comparison operators are used when user want to compare outer query with single value of inner query.
- Ex:SELECT * FROM TBLDEPT WHERE DID >(SELECT DID FROM TBLSTUDINFO WHERE rno=2);

```
 DID DNAME
---- ----------
   3 Science
   4 Commerce
   5 Arts
```

2. IN

- IN operator is used when you want to compulsory check the value which is available within inner query.
- Ex:SELECT * FROM TBLDEPT WHERE DID IN (SELECT DID FROM TBLSTUDINFO WHERE RNO<4);

```
 DID DNAME
---- -------------
   1 IT
   2 Management
```

3. ANY

- ANY operator is used when you want to compare a value with a list.
- It is followed by comparison operator (<,<=,…).
- With list, it performs like OR operator. (If any one of the conditions is satisfied from the list then it is selected as correct value.
- Ex: SELECT * FROM TBLDEPT WHERE DID >ANY(SELECT DID FROM TBLSTUDINFO WHERE RNO<4);

```
 DID DNAME
---- -------------
   2 Management
   3 Science
   4 Commerce
   5 Arts
```

4. ALL

- ALL operator is used when you want to compare a value with a list.
- It is followed by comparison operator (<,<=,…).
- With list, it performs like AND operator. (If all the values are satisfied with given condition then and only then the value is accepted.
- Ex:SELECT * FROM TBLDEPT WHERE DID >ALL(SELECT DID FROM TBLSTUDINFO WHERE RNO<4);

DID	DNAME
3	Science
4	Commerce
5	Arts

5. EXISTS& NOT EXISTS

- EXISTS and NOT EXISTS operators are used when user wants to find result based on sub-query.
- Outer query executes only if inner query has at least one record.
- Example:
  Select * from tblstudinfo where exists (select * from tbldept where dname='Management');

RNO	NAME	MARKS	DID
1	Mayank	35	1
2	Maya	25	2
3	Mohit	45	1
4	Raghav	10	10
5	Ankit	49	3

Select * from tblstudinfo where exists (select * from tbldept where dname='BA');

```
no rows selected
```