Introduction to SQL

Data:

Data is a raw fact which describes an attributes of an object (entity)

Information:

The collection of data is called as Information.

Database:

It is a storage medium used to store the data in an organized manner.

Database Management System:

It is a software which is used to manage the database like inserting a data, updating a data and by deleting a data.

In DBMS data is stored in the form of files.

The different models in DBMS are -

1) hierarchical database model
2) Network database model
3) Relational database model

Relational Database Model :-

- This model was introduced by the person called Eadgar F Codd in the year 1970s.
- Using this model we can store the data in the form of Tables.
- Tables :-
  Table is a logical structure of data which consists of rows, columns and cells.

Std

Stdid	Sname	marks
123	Anu	55
456	Banu	70
789	Sonu	59

Columns :-
- columns are also known as Attributes or fields.
- It is used to represent the property of all entities.

Stdid
123
456
789

Rows :-
- Row is also called as Tuples or Records.
- It is used to represent the properties of an individual entity.

456	Banu	70

only | 456 | (x)

Cell :-
Cell is the smallest unit of a table which is used to represent a data.

Sonu

RDBMS (Relational Database Management System)
- It is a s/w which is used to manage the database.
- In this data is stored in the form of Tables.
- We can store n number of Tables and also we can build a relation between the Tables with the mechanism of key attributes.

SQL (Structured Query Language)
- In the years 1970s the Persons called Donald Chamberlin and Raymond Boyce introduced a Language called Sequel with the help of the Organization called IBM.
- Later ANSI (American National Standard Institute) took up the Sequel and they changed the name to Structured Query Language.
- It is a high level language.
- It is used to communicate or interact or access the database.

↳ All the queries in SQL are ended with semicolon.
↳ SQL is not a case sensitive but the data present in the table is case-sensitive.

Data Integrity :-
It is used to restrict the invalid data enter into the table.

Std

Sname	age	DOB
Kavya	22	01-JAN-22
30	30-Dec-93	ANU
04-may-as	Manu	25

Invalid data

We can achieve data integrity in 2 ways:
1) Data Types
2) Constraints.

Data Types :- (belongs to DDL)
↳ It is used to specify the type of data to be stored for the selected columns in the Table.
↳ The different types of DataTypes are-
1) Number
2) char
3) Varchar2
4) Date
↳ Number- It is used to store the data like numeric values.
Syntax: Number (Size)
Here the size represents the value to be stored for the selected column.
We can store the digits in the range of 1 to 38.

↳ • char- using this datatype we can store alphanumeric values which includes alphabets, numbers & special characters.
• characters are case-sensitive.
• It must always be enclosed within a single quotes.
• We can store upto 2000 of characters.
• It is fixed in length.
• Syntax: char (size)

↳ Varchar2
• using this datatype we can store alphanumeric values.
• We can store upto 4000 of characters.
• characters are case-sensitive.
• Whenever we use varchar datatype characters should be enclosed within a single quotes.
• It is variable in length.
• Syntax: varchar2 (size). Varchar - 2000
Varchar2 - 4000

↳ Date -
• It is used to store date kind of data.
• In SQL, we have a standard date format and it is represented by 'DD-MON-YY'.

↳ All the queries in SQL are ended with Semicolon
↳ SQL is not a case sensitive but the data present in the table is case-sensitive.

Data Integrity :-
It is used to restrict the invalid data enter into the table.

Std

Sname	age	DOB
Kavya	22	01-JAN-22
30	30-Dec-93	ANU
04-may-95	Manu	25

Invalid data

We can achieve data integrity in 2 ways:
1) Data Types
2) Constraints.

Data Types :- (belongs to DDL)
↳ It is used to specify the type of data to be stored for the selected columns in the Table.
↳ The different types of DataTypes are-
1) Number
2) Char
3) Varchar2
4) Date
↳ Number- It is used to store the data like numeric values.
Syntax: Number (Size)
Here the size represents the value to be stored for the selected column.
We can store the digits in the range of 1 to 38.

↳ • Char- using this datatype we can store alphanumeric values which includes alphabets, numbers & special characters.
• characters are case-sensitive.
• It must always be enclosed within a single quotes.
• We can store upto 2000 of characters.
• It is fixed in length.
• Syntax: char (size)

↳ Varchar2
• using this datatype we can store alphanumeric values.
• We can store upto 4000 of characters.
• characters are case-sensitive.
• Whenever we use varchar datatype characters should be enclosed within a single quotes.
• It is variable in length.
• Syntax: varchar2 (size). Varchar-2000
Varchar2-4000

↳ Date-
• It is used to store date kind of data.
• In SQL, We have a standard date format and it is represented by 'DD-MON-YY'.

Constraints :-

- Constraints are used to give rules for the selected columns.
- The different types of constraints are-
  - NOT NULL
  - Unique
  - Primary key
  - Foreign key
  - check

- NOT NULL Constraint :-
  It is a constraint which is assigned to columns which shouldn't accept null.

- unique constraint :-
  It is a constraint which is assigned to a column which shouldn't accept duplicate values.

(Imp) Difference bet? NOTNULL & unique Constraints.

NOTNULL	Unique
It doesn't allow null.	It allows null.
It allows a duplicate values.	It doesn't allows duplicate values.
e.g: 4, 5, 6, 6, 6	e.g: 1, 2, 3, 4

- Primary key : (VImp)
  - It is a combination of unique and notnull constraint.
  - It is used to identify the records in an unique way.
  - For 1 table we can create only 1 primary key.
  - Creating PK is not a mandatory but it is highly recommended.
  - It doesn't allows null and duplicate values.
  - The columns which are eligible to become a Primary key, those columns are called as candidate keys.
  - The columns which are eligible to become a pk but not choosen as a PK is called as Alternate keys.

- Foreign key : (VImp)
  - It is used to build a relation bet? the tables.
  - It is also called as referential Integority Constraint.
  - Reference is the keyword we are using to build a relation bet? the tables.
  - It allows null and duplicate values.
  - We can create more than 1 foreign key for a table.
  - If you want to create a foreign key first the column should be a Primary key. then we have to take a reference of the Primary key.

- check :
  - check is a constraint which is used to give additional validations based on client requirement.

SQL Statements

1) Data Query Lang (DQL)
   - Select
   - Joins
2) Data Definition lang (DDL)
   - Create
   - Truncate
   - Drop
   - Rename
   - Alter
3) Data Manipulation lang (DML)
   - update
   - delete
4) Transaction control lang (TCL)
   - Rollback
   - Commit
   - savepoint
5) Data Control lang (DCL)
   - Grant
   - Revoke

Data Query Language :-

- Using this statement we can form the Queries and fetch data from the database.
- In this statement we have Select and Joins.

NOTE :-

↳ To display the table in a proper way we need to set the pagesize and linesize.

eg: setpages 100

Whenever we set the pagesize to 100 it will Consider 100 lines for 1 page.

Set lines 100

Whenever we set the linesize to 100 it will Consider 100 characters for 1 line.

↳ The valid credentials to login to the Oracle database are

username : SCOTT
Password : TIGER

↳ To display the tables present in the database we can make use of the Query

Select * from tab;

o/p →

TNAME	TABTYPE	CLUSTERID
DEPT	TABLE	
BONUS	TABLE	
SALGRADE	TABLE	
EMP	TABLE	

↳ To know the structure of any table we can make use of

Desc TableName ; (Desc - describe)

eg: To know the structure of EMP Table.

desc EMP

o/p →

Name	Null?	Type
EMPNO	NOTNULL	NUMBER(4)
ENAME		VARCHAR2(10)
JOB		VARCHAR2(9)
MGR		NUMBER(4)
HIREDATE		DATE
SAL		NUMBER(7,2)
COMM		NUMBER(7,2)
DEPTNO		NUMBER(2)
		NUMBER(12)

To Know the structure of DEPT Table.

Desc DEPT

O/P → NAME	NULL?	TYPE
DEPTNO	NOTNULL	NUMBER(2)
DNAME		VARCHAR2(14)
LOC		VARCHAR2(13)

↳ To display all the column information we can make use of the syntax

Select * from Tablename;

Execution Flow of the above Syntax.

- Execution starts from "from" clause (Predefined word)
- For the "from" clause we can pass tablename as an argument.
- The function of the "from" clause is to go to the database and search for the table and put that table under execution.
- 'Select' clause will execute after the "from" clause.
- It will select all the columns mentioned from the given table.
- 'Select' clause is responsible for preparing the result table.
- For the "Select" clause we can pass 3 arguments
  - ↳ *
  - ↳ column name
  - ↳ expression

- In "Select" we have two forms
  - ↳ Selection
  - ↳ Projection

- Selection :- It is the Process of retrieving the data from all the columns and records.
- Projection :- The process of retrieving the data by selecting Particular columns

syntax: select colname from Tablename;

- Semicolon - it means end of the Query.

eg :-

1) Write a query to display all the employee details.

Query → Select * from EMP;

2) Write a query to display all the details of dept. Table.

→ Select * from DEPT;

3) Write a query to display details of all the students from student table.

→ Select * from STD;

4) Query to display Empname of all the EMPLOYEES

→ Select ENAME from EMP;

5) Query to display Empname along with their Salary of all the Employees.

→ Select ENAME, SAL from EMP;

6) WAQTD Empnumber along with their designation and deptno.
→ Select EMPNO, JOB, DEPTNO from EMP;

7) QTD Deptname and location of all the departments.
→ Select DNAME, LOC from DEPT;

8) QTD Empname along with their join date.
→ Select ENAME, HIREDATE from EMP;

ALIASING

- Aliasing is an alternative name given to a columns or tables.
- It is a temporary change and it will not effect to original column names.
- 'as' is the optional keyword we are using in aliasing.
- Whenever we want to give a space between two words in aliasing, then those words should be enclosed within double quotes or underscore.

Syntax:
Select colname as aliasname from Tablename;

eg:- Select EMPNO as ENO from EMP;
o/p → ENO
7346
7499

Select SAL SALARY from EMP;
o/p → SALARY
800
1600

Select ENAME EMPLOYEE NAME from EMP;
o/p → Error

Select ENAME "EMPLOYEE NAME" from EMP;
o/p → EMPLOYEE NAME
SMITH
ALLEN
WARD
JONES
MARTIN

Dual :-

- Dual is a dummy table which is automatically created by oracle database alongwith the data dictionary.
- It consists of only one record.

eg: select 2*3 from DUAL;
o/p → 2*3
6

Select 'JSP' from DUAL;
'JS
JSP

Literals :-

- Usage of data directly inside the Query is called as Literals.
- In this we have 3 kinds of data literals.
  1) Number Empno.
  2) Storing Ename, JOB
  3) Date Hiredate, DOB

NOTE:- Whenever we are using string and Date kind of data then it should be enclosed with single quotes. And for number it is not a mandatory.

- IMPORTANT :-

- "where" clause is used to restrict the number of records to be displayed.
- (or) "where" clause is used to display particular record as an output by giving conditions.

Syntax:

Select */column ③ from Tablename ① where ② {condition};

NOTE:↳ We can't use more than 1 "where" clause for 1 select statement.

Select SAL from EMP where SAL=800, where ENAME= 'SMITH';

O/P → Error

↳ The position of "where" clause is always after the "from" clause.

Select ENAME from EMP where DEPT.NO=20;

↳ We can't use "where" clause inside the "select" clause.

Select where Sal=800 from EMP; (Error)

Orderby

↳ Orderby clause is used to arrange the records either in ascending or in descending order.

↳ Syntax :-

Select col/* from Tablename order by colname asc/desc;

NOTE :-

1) ↳ Select sal from emp order by sal asc;

Sal
800
950
1100
1250
1250
1300
1500

↳ Select sal from emp order by sal;

In orderby clause ascending order is not mandatory bydefault if you not mention any order, order by clause will sort the records in ascending order.

↳ Select sal from emp order by sal, order by ename;

We can't use two order by clause for one select statement.

↳ ③ Select col/* ① from Tablename ② where {condition} ④ Order by colname asc/desc;

eg:- WAQID all the employee details those whoare working in the deptno 20 and arrange the employee's name in desc order.

Select * from emp where deptno = 20 orderby ename desc;

Operators

- Operators are used to Perform operations on operands.
- The different types of operators are -
  1) Arithmetic operator
  2) Logical operator
  3) Relational Operator
  4) Special operator
  5) Concatenation operator
  6) Set operator

1) Arithmetic Operators :-

- Arithmetic op includes +, -, *, /.
- We can Perform arithmetic operations only inside the 'select' clause.
- We can't Perform arithmetic operations inside the 'where' clause but we can give condition for the arithmetic Operations.

↳ WAQID employee name alongwith their salary and salary should be increment by 500.

Select ename, sal+500 from emp;

↳ WAQID employee name along with their annual salary.

Select ename, sal * 12 from emp;

↳ QID name of the employees along with 25% hike in their salary.

Select ename, sal+(sal * 0.25) from emp;

↳ QID employee no, employeename, designation along with 10% deduction in their salary

Select empno, ename, Job, sal-(sal * 0.1) from emp;

↳ QID half term salary of the employees.

Select sal/2 * 6 from emp;

↳ Query to display daily salary of the employees.

Select sal/30 from emp;

2) Logical Operator

- Logical Operator includes AND, OR, NOT
- All the logical operators should be used only inside the 'where' clause.
- Whenever we are passing multiple conditions there we have to use logical Operators.

1) AND Operator :-

In this type of Operator rows get selected only if all the conditions are true.

eg:- WAQTD all the employee details those who are working as a manager in the deptno 30.

```
select * from emp where job='manager'
AND deptno=30;
```

2) OR OPERATOR :-

Whenever we use "OR" operator we get selected if any one condition is true.

eg:- WAQTD all the employee details those who are working in the deptno 10 or 20.

```
Select * from emp where deptno=10 OR
deptno=20;
```

3) NOT OPERATOR :-

When we use NOT Operator, it will select the records for which that statement is false.

eg: WAQTD all the employee details except salesman.

```
Select * from emp where NOT job='salesman';
```

3) Relational Operator :-

- This Operator is also called as comparision Operator.
- This Operator includes <, >, <=, >=, !=, =
- All the relational operators should be used inside the where clause.

eg:- WAQTD all the employee details those who are earning a salary more than 2500.

```
Select * from emp where sal>2500;
```

↳ QTD employee name, designation, along with their salary if the employee is working as a clerk and earning a salary of Rs 800.

```
Select ename, job, sal from emp where job='CLERK'
and sal=800;
```

↳ QTD all the employee details those who are working in the deptno = 10.

```
Select * from emp where deptno=10;
```

↳ QTD employee name along with their annual salary of the employees if the annual salary is more than 15000.

```
Select ename, sal*12 from emp where sal*12>15000;
```

↳ QTD all the employee details those who belongs to deptno 20 and earning a salary 2850 and exceed that.

```
Select * from emp where deptno=20 AND sal>=2850;
```

↳ WAQTD the employee name, salary of the employees except deptno 10.

```
Select ename, sal from Emp where dept != 10;
                                  (<>: NOT)
```

↳ WAQTD all the employee details those who are hired in the year 81.

```
Select * from Emp where Hiredate >= '01-JAN-81'
AND Hiredate <= '31-Dec-81';
```

↳ WAQTD all the employee details those who are hired after the years 81.
Select * from Emp where Hiredate > '31-DEC-81';

↳ QTD employee name and hiredate of the employees those who are hired before 87.
Select Ename, hiredate from emp where Hiredate < '01-JAN-87';

↳ QTD the details of the employees those who are earning a commission more than salary.
Select * from emp where comm > Sal;

↳ QTD all the employee details those who are earning a salary more than 800 but less than 5000.
Select * from emp Sal > 800 AND Sal < 5000;

↳ QTD all the employee details those who are working as a saleman or manager or analyst.
Select * from emp where Job = 'Saleman' OR Job = 'Manager' OR Job = 'Analyst';

SPECIAL OPERATORS :- (VImp)

- Special Operators includes IN, BETWEEN, IS, LIKE.
- All the special Operators should be used inside the where clause.

↳ IN OPERATOR :-
To evaluate multiple values and instead of using OR operator more than once we can go for IN operator.
Syntax:
Select */colname from Tablename where colname IN (value1, value2, value3);

eg: WAQTD all the employee details those who belongs to department number 10, 20 or 30.
Select * from emp where deptno IN (10, 20, 30);

↳ BETWEEN OPERATOR :-

- Whenever user want to display the output in a range of values then we use BETWEEN Operator
- In order to use BETWEEN Operator we must know the starting and ending value.
- Between the starting value and ending value we can make use of "AND".
- While displaying the output this Operator includes the starting value and ending value.

Syntax:
Select */colname from Tablename where column between SV and EV;

eg. QTD employee name along with their salary those who are earning a salary in the range of 1000 to 3000.
Select ename, salary from emp where Sal between 1000 AND 3000;

6 Q) all the employee details except those who are hired in the year 81.
Select * from emp where hiredate NOT between '01-JAN-81' AND '31-DEC-81';

IS OPERATOR :-
This operator is used to compare NULL and NOTNULL values.
Syntax: Select */colname from Tablename where colname is NULL/NOTNULL;

6 INAQTD all the employee details those who are not earning a commission.
Select * from emp where comm is NULL;

LIKE OPERATOR :-
- LIKE Operator is used for the Purpose of Pattern matching.
- We can achieve LIKE Operator with the help of two wild cards such as '%', '_'.
- '%' using this we can match one to n number of characters.
- '_' :- using this we can match single character.

eg:- 6 INAQTD the employee names whose name starts with A.
Select ename from emp where ename like 'A%';

6 INAQTD the employee name whose name is having a letter A in the second Position.
Select Ename from emp where Ename like '_A%';

6

1) INAQTD all the employee details those who are working as a salesman or clerk or President.
Select * from emp where job in ('salesman', 'clerk', 'President');

2) INAQTD all the employee details whose name ends with D.
Select * from Emp where ename like '%D';

3) INAQTD all the managers details those who are working in the deptno 20 and their salary ends with 5.
Select * from emp where JOB='MANAGER' and Dept-No=20 and sal like '%5';

4) INAQTD employee name, designation of the employees whose name starts with S and their designation starts with K.
Select ename, job from emp where ename like 'S%' and Job like '%K';

5) INAQTD all the employee details those who are not earning a salary.
Select * from emp where sal is null;

6) QID all the employee details those who doesn't have reporting managers.
Select * from emp where MGR is null;

7) QID all the employee details those who are hired in the year 81.
Select * from emp where hiredate between '01-JAN-81' and '31-DEC-81';
or Select * from emp where hiredate '%81';
or Select * from emp where hiredate '-------81';

8) INAQID the employee name whose name doesn't starts with M.
Select ename from emp where ename NOT LIKE 'M%';

9) QID employee name, salary and annualsalary of the employees those who are earning annual salary in the range of 12000 to 18500.
Select ename, sal, sal*12 from emp where sal*12 between 12000 and 18500;

10) QID the employee names whose name starts with T and ends with R.
Select ename from emp where ename like 'T%R';

11) INAQID all the employee details those who are working as a salesman and earning a salary of exactly 4 digits and their name contains last but 1 character is E.

Select * from emp where Job='SALESMAN' AND sal LIKE '----' AND ENAME LIKE '%E_';

12) INAQID name of the employees along with their deptno and Job those who are working as a clerk in the deptno 10 or 20.
Select ename, deptno, job from emp where job='CLERK' and deptno in(10,20);

13) Details of the employees those who are working as a manager in the deptno 10, 20, 30 or 40.
Select * from emp where Job='MANAGER' and deptno in(10,20,30,40);

14) QID the employee names along with their hiredate those who are hired after the year 81 and before the year 87.
Select ename, hiredate from emp where hiredate between '01-JAN-82' and '31-DEC-86';

15) QID name of the employees along with the annual salary for the employees those who are working as a manager or clerk in the deptno 10 or 20.
Select ename, sal*12 from emp where job in('MANAGER', 'CLERK') and deptno in(10,30);

16) QTD all the employee details those who are earning a commission.
Select * from emp where comm IS NOT NULL;

17) List all the employees whose name starts with S or starts with A.
Select * from emp where ename like 'S%' or ename like 'A%';

18) QTD all the employee details those who are having reporting managers in the deptno 10.
Select * from emp where MGR is not null and deptno = 10;

19) QTD all the employee details those who are not earning a commission and working as a clerk.
Select * from emp where comm is not null and JOB = 'CLERK';

20) List all the employees with their annual salary except those who are working in the deptno 30.
Select emp.*, sal*12 from emp where deptno != 30;

21) QTD all the employee details those who are having reporting managers in the deptno 10 along with 10% hike in salary.
Select emp.*, sal+(sal*0.1) from emp where MGR is not null and deptno = 10;

22) QTD all the employee details those who are joined in the month of Feb.
Select emp.* from emp where Hiredate like '---FEB---';

23) QTD all the employee details whose salary is not in the range of 800 to 2500 in the deptno 10 or 20 except salesman.
Select * from emp where sal not between 800 and 2500 and deptno in (10, 20) and job != 'SALESMAN';

24) QTD the deptname and the locations which are having letter O in their dept names as well as locations.
Select dname, Loc from dept where dname like '%O%' and Loc like '%O%';

25) INAQTD all the employee details those who contains "man" in their designation.
Select * from emp where job like '%MAN%';

26) QTD name and annual salary of the employees if the annual salary ends with 0.
Select ename, sal*12 from emp where sal*12 like '%0';

27) QTD name of the employees those who are having 2 L's in their name.
Select ename from emp where ename like '%L%L%';

28) QTD name of the employees those who are having two consecutive LL's in their names.
Select ename from emp where ename like '%LL%';

29) QTD name of the employees whose name starts with vowels and the ends with consonents.
Select ename from emp where ename like 'A%' OR ename like 'E%' OR ename like 'I%' OR ename like 'O%' OR ename like 'U%' and ename not like '%A' AND ename not like '%E' AND ename not like AND ename not like '%O' AND ename not like '%U';

SET OPERATOR :-

- Whenever the user wants to merge the results of two or more there we use set operator.
- The different types of set operators are UNION, UNION ALL, INTERSECT, MINUS.
- Syntax: (can't use *)

```
Select colname from Tablename
    SET OPERATOR
Select colname from Tablename;
```

UNION :-

- This operator will merge two or more select statement, after merging the select statement it will give unique values as an output.
- Between the select statement we make use of same datatype columns.

eg-1) Select Deptno from emp
UNION
Select Deptno from dept;

o/p →

Deptno
10
20
30
40

eg-2) Select ename from emp
Union
Select deptno from emp;

o/p → error: expression must have same datatype as corresponding expression.

eg-3) Select sal, deptno from emp
union
Select deptno from dept;

o/p → Error: Query block has incorrect number of result columns.

UNION ALL :-

After merging two or more select statement union all display the output including the duplicate values.

eg :- Select deptno from emp
UNION ALL
Select deptno from dept;

o/p →

Deptno
20
20
30
20
30
30
10

INTERSECT :-

After merging two or more select statement this operator displays only the common values between the select statements.

eg :- Select deptno from emp
intersect
Select deptno from dept;

o/p →

deptno
10
20
30

MINUS :-

This operator will subtract the common values between the select statements and display the output as only uncommon value from first select statement.

eg-1) Select deptno from emp minus
Select deptno from dept;
o/p → no rows selected

eg-2) Select deptno from dept minus
Select deptno from emp;
o/p → 40

CONCATENATION OPERATOR

- Using this operator we can merge two or more columns or literals.
- We can achieve concatenation with the symbol '||'.
- eg-) WAQTD the output in the below format.
  Format : Smith is earning 800
  query → Select ename||' is earning '||sal from emp where ename='SMITH';
  o/p → ename||' is earning '||sal
  SMITH is earning 800

eg-2) WAQTD out put in the below format for all the employees :-
Format : 1) My name is Smith (for all) :
2) Smith is working as clerk.
3) Smith employee number is 7369.
4) Smith is working in deptno 10 as clerk

1) Select 'My name is '||ename from emp;
2) Select ename||' is working as '||Job from emp;
3) Select ename||' employee number is '||empno from emp;
4) Select ename||' is working in '||deptno||' as '||Job from emp;

FUNCTIONS

- Functions are the important features of SQL.
- It is used to perform specific task like calculations on data or to perform modification on data.
- The different types of functions are -
  i) Single row function
  ii) Multi row function
  iii) Date function

Single Row Functions :-

- In this type of functions if you give n no. of inputs you will get corresponding output for each input.
- The different types of single row functions are length(), substr(), mod(), Replace(), upper(), concat(), Lower(), Initcap(), Trunc(), Reverse()

Length()

- Using this function we can derive the number of characters in a given column or literal.
- Syntax :-
  Select length(column/literal) from Tablename;

5 WAQTD the length of all the employee names.
Select length(ename) from emp;

Q WAQTD the employee names those who are having exactly 5 characters in their name.
select ename from emp where length(ename)=5;

Q WAQTD all the employee details whose name starts with M and ends with N and exactly contains 6 characters in their name.
Select emp.* from emp where ename like 'M%' and ename like '%N' and length(ename)=6;

Q QTD all the employee details those who are working as a manager and length of their salary will be exactly 4 digits.
Select emp.* from emp where Job='MANAGER' and length(Sal)=4;

Q QTD employee name, designation and deptno of the employees those who are working in the deptno 20 and their designation contains 5 characters.
Select ename, job, deptno from emp where deptno=20 and length(JOB)=5;

Q WAQTD the employee names whose name contains 4 characters and the 3rd character is R
Select ename from emp where length(ename)=4 and ename like '__R_';

Substr()
- Substring is used to extract particular part of a given string.
- In substring extraction is always from left hand side to right hand side.
- Syntax:
  Select substr(arg1, arg2, arg3) from Tablename;
  arg1 → colname / literal
  arg2 → starting position
  arg3 → length
- For this function minimum we need to pass 2 arguments and maximum we can pass 3 arguments.
- If we not specify the length then it will extract from starting position to the last character.

eg:-1) Select substr('QSPIDER', 1, 3) from dual;
SUB
QSP

eg-2) Select substr('QSPIDER', 3) from dual;
SUBSTRC
QSPIDER

eg-3) Select substr('QSPIDER', -4, 2) from dual;
(-7 -6 -5 -4 -3 -2 -1)
SU
ID

eg-4) Select substr('QSPIDER', -1, -1) from dual;
S

1) WAQTD the first character of the employee name.
Select substr(ename, 1, 1) from emp;

2) QTD last character of the employee name.
Select substr(ename, -1, 1) from emp;

3) QTD the first and last character of employee name.
Select substr(ename, 1, 1), substr(ename, -1, 1)
from emp;

4) QTD the employee names whose name starts with M.
Select ename from emp where substr(ename, 1, 1) = 'M';

5) QTD the employee names whose name ends with R.
Select ename from emp where substr(ename, -1) = 'R';

6) QTD first 3 characters of your name.
Select substr('PALLAVI', 1, 3) from dual;

7) QTD last 3 characters of your name.
Select substr('PALLAVI', -3, 3) from dual;

8) QTD all the employee details whose designations ends with 'CLERK'.
Select emp.*, from emp where substr(JOB, -3, 3) = 'CLERK';

9) QTD first half of the employee name.
Select substr(ename, 1, length(ename)/2) from emp;

10) QTD second half of the employee name.
Select substr(ename, length(ename)/2 + 1) from emp;

11) QTD the employee name whose name starts with vowels and ends with consonents.
Select ename from emp where substr(ename, 1, 1) IN ('A', 'E', 'I', 'O', 'U') and select ename substr(ename, -1) NOT IN ('A', 'E', 'I', 'O', 'U');

12) QTD all the employee details whose last but one character in the name should be equals to middle letters of the month.
Select emp.* from emp where substr(ename, -2, 1) = substr(hire date, 5, 1);

REPLACE()

- This function is used to replace old data with new data.
- For this function we can pass minimum of 2 arguments and maximum of 3 arguments.
- Syntax:-

```
select Replace (arg1, arg2, arg3) from Tablename;
```

arg1 → colname, literals
arg2 → character to be replace.
arg3 → Replacing character.

eg-1) `select REPLACE('JAVA','J','L') from dual;`

```
REPL
LAVA
```

eg-2) `select REPLACE('JAVA','J') from dual;`

```
REP
AVA
```

eg-3) `Select REPLACE('JAVA','j','m') from dual;`

```
REPL
JAVA
```

eg-4) `select REPLACE('JAVA','V',' ') from dual;`

```
REPL
JA A
```

eg-5) `select REPLACE('JAVA','A','B') from dual;`

```
REPL
JBVB
```

eg-6) `select REPLACE(JOB,'SALESMAN','SUPERMAN') from emp;`

```
REPLACE(JOB,'SALESMAN','SUPERMAN')
CLERK
SUPERMAN
SUPERMAN
MANEGER
```

1) WAQTD Replace all the employee details those who contains 'man' in their designation.

```
select * from emp where job = replace(JOB,'MAN','');
```

2) QTD all the employee details those who contains 'man' in their designation.

```
select emp.* from emp where Job like '%MAN%';
```

or=

```
Select * from emp where job != replace(JOB,'MAN');
```

3) WAQTD the number of times letter A is present in the String MALAYALAM.

```
Select length('MALAYALAM') - length(replace('MALAYALAM','A')) from dual;
```

4) WAQTD number of times letter L is present in the employee names.

```
select length(ename) - length(replace(ename,'L')) from emp;
```

Concat()

Using this function we can merge two arguments. The arguments should be either colname or literals.

Using this function we can merge only two columns or literals. If you want to merge more than 2 columns then we have to use nested concat.

- Syntax: select concat(arg1, arg2) from Tablename;
  arg1, arg2 → colname/literals

eg) select concat(ename, sal) from emp;

- Nested concat Syntax:-
  select concat(concat(arg1, arg2), arg3) from Tablename;

1) WAQTD the output in the below format for all the employees using concat function.
   i) Smith is working as clerk
   select concat(concat(ename, ' is working as '), JOB) from emp;
   ii) Smith is earning 800
   Select concat(concat(ename, ' is earning '), Sal) from emp;
   iii) Smith is working in deptno 10 as clerk
   Select concat(concat(concat(concat(ename, ' is working in deptno '), Deptno), 'as'), Job) from emp;
   iv) Employee number of Smith is 7369.
   select concat(concat(concat('Employee number of ', ename), 'is'), empno) from emp;

Mod()

- This function is used to find the remainder of a given number.
- Syntax: Select mod(divident, divisor) from Tablename;
- eg: select mod(5,2) from dual;

  mod(5,2)
  1

1) WAQTD empnum along with the empname those who have odd employee number.
   select empno, ename from emp where mod(empno, 2) != 0;

2) QTD all the employee details those who have even employee numbers.
   Select emp.* from emp where mod(empno, 2) = 0;

3) WAQTD employee name along with the salaries of the employees who earns the salary in the multiple of 3.
   Select ename, sal from emp where mod(sal, 3) = 0;

UPPER()

- This function converts the lowercase strings to uppercase.
- Syntax: Select upper(colname/literals) from Tablename;

LOWER()

- This function converts uppercase characters to lowercase.
- Syntax: Select lower(colname/literals) from Tablename;

initcap()
This function converts only the first characters into Uppercase and followed by lowercase characters.
Syntax: select initcap(colname/literal) from Tablename;

Reverse()
Using this function we can reverse the given string.
Syntax: select Reverse (colname literal) from Tablename;

Trunc()
This function will round off the given number to a lower value.
select Trunc(value) from Tablename;

Multirow function or Aggregate function or Group function:—
- In this type of functions if you give n number of inputs we will get only single output.
- The different kinds of multi-row functions are - min(), max(), avg(), sum(), count().

G min():—
- It is used to obtain the minimum value present in the given column.
- eg: select min(sal) from emp;

min(sal)
800

G max():—
- using this function we can obtain the highest value present in the given column.
- eg: select max(sal) from emp;

G sum():—
- It is used to obtain summation of values present in the given column.
- eg: select sum(sal) from emp;

G avg():—
- It is used to find the average of values from the given columns.
- eg: select avg(sal) from emp;

G count():—
- Using this function we can count the no. of values present in the given column.
- eg: select count(sal) from emp;
- Only the count() will accepts * as an argument.
- eg: select count(*) from emp;

count(*)
14

- eg: select comm(*) from emp;

comm(*)
4

NOTE:—
1) Multi-row functions can accept only one argument.
eg: select max(ename, sal) from emp;
error: invalid no. of arguments
2) We can't use multi-row functions inside the where clause.
eg: select * from emp where max(sal) > 3000;
error: group function is not allowed here (where).

3) We can't use the combination of column names and multi-row functions in the select clause.
eg: select ename, max(sal) from emp;
o/p → error.

4) We can't use string and date kind of data for avg() and sum().
eg: select avg(job) from emp;
o/p → error

i) WAQTD the highest salary given to a manager
Select max(sal) from emp where job='MANAGER';

ii) QTD the total salary given to deptno 10 employees
Select sum(sal) from emp where deptno=10;

iii) QTD average salary needed to pay for all the employees.
Select avg(sal) from emp;

iv) QTD no. of employees earning a salary more than 1500 in the deptno 20.
Select count(*) from emp where sal >= 1500 and deptno=20;

v) QTD the oldest and latest hiredate present in the employee table.
Select min(hiredate), max(hiredate) from emp;
(oldest) (latest)

vi) QTD the highest and lowest sal given to the salesman.
Select min(sal), max(sal) from emp where job='SALESMAN';

vii) QTD no. of employees having letter 'E' in their name
Select count(*) from emp where ename like '%E%';

viii) QTD no. of employees those who are earning commission.
Select count(comm) from emp;
(or) Select count(*) from emp where comm is not null;

ix) QTD the highest salary given to the employees if the employee has letter 's' in their names and working as a manager in the deptno 10 with a sal more than 1800.
Select max(sal) from emp where ename like '%S%' and job='MANAGER' and deptno=10 and sal>1800;

➡ Difference bet'n single row and multirow function.

Single row func'	Multirow func'
↳ For n no. of input it will give corresponding o/p.	↳ For n no. of inputs it will get single output.
↳ We can use single row func's inside where clause.	↳ We can't use multirow func's inside the where clause.
↳ It allows the combination of columnname & single row func'.	↳ It doesn't allow the comb'n of columnname & multirow fun'.

eg: 2) Select min(ename) from emp;

min(ename)
ADAMS

eg: 2) Select max(ename) from emp;

max(ename)
WARD

Whenever we use strings for min and max() it will display the output based on the ascii value.

Group by

- Group by clause is used to group the records.
- It collect the data from multiple records and group the results and display it as an output.
- Syntax: Select colname from Tablename
  group by colname;
- Between the group by and select clause we can make use of common columns.
  eg: select job from emp group by job;
  eg: select sal from emp group by job;
  o/p → error: not a group by expression.
- If you want to use the combination of columnname and multi-row function then we have to use with the help of group by clause.
  Syntax: Select ③ multirowfunction(colname), colname
  ① from Tablename ② group by colname;

eg: 1) WAQTD the maximum salary given to each job.

Select max(sal), job from emp group by job;

2) WAQTD highest salary given to each job except President.

Select max(sal), job from emp where job != 'PRESIDENT'
group by job;

- non-grouped data cond^n is given using where clause.
- grouped data cond^n is given using having clause

Execution flow
select ④
from ①
where ②
group by ③
order by ⑤

having clause

- having clause is used to filter the groups if you want to give any condition to the group by clause then we use having clause.
- Syntax:
  Select max(colname), colname from Tablename group by colname having <condition based on max>;
- Without group by clause we can't use having clause.

Difference between where and having clause.

where	having
• It is used to filter the records.	• It is used to filter the groups.
• It doesn't allow multi-row function.	• It allows multi-row function.
• Without groupby also we can use where clause.	• without groupby clause we can't use having clause.
• Position of where clause is always before the group by clause.	• Position of having clause is after the group by clause.

select max(colname), colname from Tablename where <condition> group by colname having <condition based on max> order by colname asc/desc;

6. Execution flow
select ⑤
from ①
where ②
group by ③
having ④
order by ⑥

1) WAQ ID highest salary of each dept except president
select max(sal), deptno from emp where job != 'PRESIDENT' group by deptno;

2) QID no. of employees having letter A in their names in each job.
select count(*), job from emp where ename like '%A%' group by Job;

3) QID no. of employees and average salary needed to pay the employees whose salary is > 2000 in each dept.
Select count(*), avg(sal), deptno from emp where sal > 2000 group by deptno;

4) QID no. of employees working as a manager in each dept if their average salary is less than 3000
Select count(*), deptno from emp where job = 'MANAGER' groupby deptno having avg(sal) < 3000;

5) QID no. of employees having letters A in their names.
Select count(ename) from emp where ename like '%A%';

6) QID total salary of each job if the total salary is more than 2600.
Select sum(sal), Job from emp group by job having sum(sal) > 2600;

(Imp)7) QID name of the employees that are repeated exactly twice.
Select count(*), ename from emp group by ename having count(*) = 2;

(Imp)8) QID the salary that are repeated.
Select count(*), sal from emp group by sal having count(*) > 1;

9) WAQ ID no. of employees getting same salary in the same dept.
Select count(*), sal, deptno from emp group by sal, deptno having count(*) > 1;

10) QID no. of employees hired on the same days into the same dept.
Select count(*), hiredate, deptno from emp where group by hiredate, deptno having count(*) > 1;

11) QTD deptwise max salary of the employee excluding deptno 20 and arrange the highest salary in ascending order.
Select max(sal), deptno from emp where deptno != 20 group by deptno order by max(sal) asc;

12) QTD total salary needed to pay for the employee in each job if the total salary of each job exceeds 2450.
Select sum(sal), job from emp group by job having sum(sal) > 2450;

13) QTD no. of employees in each job and atleast 4 employee should be working in each dept except President.
Select count(*), job, count(*), deptno from emp where job != 'PRESIDENT' group by job, deptno having count(deptno) >= 4;

14) QTD average salary needed to pay for the employees in each dept excluding the employee of deptno 20.
Select avg(sal), deptno from emp where deptno != 20 group by deptno;

15) QTD maximum salary, lowest salary, total salary and average salary given to 'President'
Select max(sal), min(sal), sum(sal), avg(sal), job from emp where job = 'PRESIDENT' group by job;

DATE FUNCTION :-

- Date function is used to know the present date or to change the date format.
- SYSDATE or CURRENT_DATE :-
  using this funct we can find the present date.

eg 1) Select current_Date from dual;

- To_char :-
  - Using this function we can change the date-format according to users choice.
  - It is a temporary change.
  - Syntax :-
    Select to_char(colname, 'Dateformat') from Tablename;

eg :- Select to_char(sysdate, 'MON-DD-YY') from dual;

1) WAQTD all the employee details those who are hired in the month of September.
Select * from emp where to_char(hiredate, 'MON') = 'SEP';

(Imp)
2) WAQTD empname along with the no. of years of experience of the employees.
Select ename, to_char(sysdate, 'YYYY') - to_char(hiredate, 'YYYY') from emp;

(Imp)
3) QTD all the employee details those who have atleast 36 years of experience.
Select * from emp where to_char(sysdate, 'YYYY') - to_char(hiredate, 'YYYY') > 36;

SUBQUERY

- A query within an another query is called as subquerie.
- Whenever there is an indirect condition there we use subqueries.
- In subquery we have inner query and outer query.
- Inner query will execute first the output of the inner query will be the input of the outer query. Based on that outer query will give you the final output.
- In subquery inner query will execute only once.
- In this type of query outer query is dependent on inner query and inner query is independent.
- In subquery there are 2 types :-
  i) correlated subquery
  ii) Non-co-related subquery

Non-correlated Subquery -:

In this we have single row subquery and multirow subquery.

Single row subquery :-

- Whenever the inner query returns single record as an input to the outer query then we call that as single row subquery.
- To connect inner query and outer query we can make use of Relational Operators.

eg :- WAQTD all the employee details those who are working in the same designation as smith

```
select * from emp where job = (outer query)
(select job from emp where ename = 'SMITH');
        ↓
   innerquery
   Subquery.
```

Multirow subquery :-

- Whenever the inner query returns multiple records as an input to the outer query then we call that as multirow subquery.
- To connect inner query and outer query we can make use of special operators as in, not in, all, any.

eg : WAQTD all the employee details whose designation is same as smith or allen.

```
Select * from emp where job in (select job from
emp where ename in ('SMITH', 'ALLEN');
```

1) WAQTD name of the employee earning more than Adams.

```
Select ename from emp where sal > (Select sal
from emp where ename = 'ADAMS');
```

2) WAQTD name and salary of the employees earning less than king.

```
Select ename, sal from emp where sal < (select
sal from emp where ename = 'KING');
```

3) WAQTD name and deptno of the employees if they are working in the same dept as Jones.

```
Select ename, deptno from emp where deptno =
(select deptno from emp where ename = 'JONES');
```

4) WAQTD name and job of all the employees working in the same designation as James.

```
Select ename, job from emp where job = (select job
from emp where ename = 'JAMES');
```

hired before → <
hired after → >

5) WAQTD empno and ename along with annual salary of all the employees if their annual salary is greater than wards annual salary.
Select empno, ename, sal*12 from emp where sal*12 > (select sal*12 from emp where ename = 'WARD');

6) WAQTD name and hiredate of the employees if they are hired before SCOTT.
Select ename, hiredate from emp where hiredate < (select hiredate from emp where ename = 'SCOTT');

7) WAQTD name and hiredate of the employees if they are hired after the president.
Select ename, hiredate from emp where hiredate > (select hiredate from emp where job = 'PRESIDENT');

8) WAQTD name and sal of the employees if they are earning sal less than the employee whose empno is 7839.
Select ename, sal from emp where sal < (select sal from emp where empno = 7839);

9) WAQTD all the details of the employees if the employees are hired before miller.
Select * from emp where hiredate < (select hiredate from emp where ename = 'MILLER');

10) WAQTD all the emp details those who are senior to king.
Select * from emp where hiredate < (select hiredate from emp where ename = 'KING');

(Imp)11) QTD all the employee details those who are earning highest salary.
Select * from emp where sal = (select max(sal) from emp);

15) QTD all the employee details those who are junior to miller.
Select * from emp where hiredate > (Select hiredate from emp where ename = 'MILLER');

16) QTD all the employee details whose salary should be less than the average sal of deptno 20.
Select * from emp where sal < (select avg(sal) from emp where deptno=20);

17) WAQTD ename and salary of all the employees who are earning more than miller but less than allen.
Select ename, sal from emp where sal > (Select sal from emp where ename='MILLER') and sal < (Select sal from emp where ename='ALLEN');

18) WAQTD all the details of the employees working in dept 20 and working in the same designation as smith.
Select * from emp where deptno=20 and job = (select job from emp where ename='SMITH');

19) WAQTD all the details of the employees working as managers in the same dept as turner.
Select * from emp where job='MANAGER' and deptno = (Select deptno from emp where ename="TURNER");

20) WAQTD name and hiredate of the employees hired after 1980 and Before king.
Select ename, hiredate from emp where hiredate > '1980' and hiredate < (Select hiredate from emp where ename = 'KING');

21) WAQTD name and sal along with annual sal for all employees whose sal is less than blake and more than 3500.
Select ename, sal, sal*12 from emp where sal > 3500 and sal < (select sal from emp where ename = 'BLAKE');

22) WAQTD all the details of employees who earn more than scott but less than king.
Select * from emp where sal > (Select sal from emp where ename = 'SCOTT') and sal < (Select sal from emp where ename = 'KING');

23) WAQTD name of the employees whose name starts with 'A' and works in the same dept as Blake.
select ename from emp where ename like 'A%' and deptno = (select deptno from emp where ename = 'BLAKE');

24) WAQTD name and comm if employees earn comission and work in the same designation as Smith.
Select ename, comm from emp where comm is not null and job = (Select job from emp where job = 'SMITH');

25) WAQTD details of all the employees working as clerk in the same dept as turner.
Select * from emp where job = 'CLERK' and deptno = (select deptno from emp where ename = 'TURNER');

26) WAQTD ename, sal and designation of the employees whose annual salary is more than Smith and less than king.
Select ename, sal, job from emp where sal*12 > (Select sal*12 from emp where ename = 'SMITH') and sal*12 < (Select sal*12 from emp where ename = 'KING');

Case-II

- WAhenever the data to be selected and condition to be executed are present in different tables then we have to use subqueries.
- In the above condition between the inner query and outer query we can make use of common columns.

eg-1) WAQTD the location of ADAMS.
Select deptno from dept where deptno = (Select deptno from emp where ename = 'ADAMS');

2) QTD deptname of the employee whose name is MILLER.
Select Dname from dept where deptno = (Select deptno from emp where ename = 'MILLER');

3) QTD deptname and location of the employees those who are working as a Salesman.
Select Dname, Loc from dept where deptno in (Select deptno from emp where job = 'SALESMAN');

4) QTD details of the employees along with their annual salary those who are located at Newyork.

Select emp.*, sal*12 from emp where deptno = (select deptno from dept where loc = 'NEWYORK');

5) WAQTD name of the employees those who are working in operations dept.
Select ename from emp where deptno = (select deptno from dept where Dname = 'OPERATIONS');

6) WAQTD all the employee details those who are hired before all the employees.
Select * from emp where hire_date = (Select min (hiredate) from emp;

7) WAQTD name of the employee along with the hiredate those who hired in the last.
Select ename, hiredate from emp where hiredate = (select max (hiredate) from emp);

8) WAQTD employee name along with the annual salary those who are earning the least annual salary
Select ename, sal*12 from emp where sal*12 = (Select min(sal*12) from emp);

(Imp)
9) WAQTD the jones managers details.
Select * from emp where empno = (Select MGR from emp where ename = 'JONES');

(Imp)
10) WAQTD the employee details those who are working under Blake.
Select * from emp where MGR = (Select empno from emp where ename = 'BLAKE');

lowest → >
highest → <

Nested subqueries :-
A subquery inside an another subquery is called as nested subquery.

1) WAQTD all the employee details those who are earning 2nd highest salary.
select * from emp where sal = (Select max(sal) from emp where sal < (Select max(sal) from emp));

2) WAQTD all the employee details those who are earning highest salary in the sales dept.
Select * from emp where sal = (Select max(sal) from emp where deptno = (select deptno from emp where dname = 'SALES'));

3) WAQTD the employee name those who have longest name in the employee table.
Select ename from emp where (select max(length (ename))) from emp;

4) WAQTD 5th maximum salary.
Select max(sal) from emp where sal ≤ (Select max(sal) from emp where sal < (Select max(sal) from emp where sal < (Select max(sal) from emp where sal < (Select max(sal) from emp)))));

5) WAQTD empno of the employee earning 2nd minimum salary.
Select empno from emp where sal = (Select min(sal) from emp where sal > (Select min(sal) from emp));

6) INAQTD SMITH MANAGER'S MANAGER Details.
Select * from emp where empno = (select MGR from emp where ename = (select ename from emp where empno = (select MGR from emp where ename = 'SMITH')));

7) INAQTD Allen manager's manager sal.
Select sal from emp where empno = (select MGR from emp where ename = (select ename from emp where empno = (select MGR from emp where ename = 'ALLEN')));

8) INAQTD Ford manager's department name:
Select Dname from dept where deptno = (select deptno from emp where empno = (select MGR from emp where ename = 'FORD'));

9) INAQTD details of the employee who has hired 2nd.
Select * from emp where hiredate = (select min(hiredate) from emp where hiredate > (select min(hiredate) from emp));

10) INAQTD name of the employee hired before the last employee.
Select ename from emp where hiredate = (select max(hiredate) from emp where hiredate < (select max(hiredate) from emp));

11) INAQTD Loc of the employee who was hired first
select loc from dept where deptno = (select deptno from emp where hire date = (select min(hiredate) from emp));

Multirow subqueries :-

All Operator
- All operator is a special subqueery operator which has to be used along with the relational op which will compare the value present at the L.H.S with all the values of R.H.S.
- All operator returns the records if all the values at the R.H.S have satisfied the condition.

Any Operator
- Any operator is a special subqueery operator which has to be used along with the relational operator which will compare the value present at the L.H.S with all the values of R.H.S.
- Any operator returns records if any one of the values at R.H.S have satisfied the condition.

eg-1) INAQTD all the employee details those who are earning a salary more than MILLER and FORD.
Select * from emp where sal > all (Select sal from emp where ename in ('MILLER', 'FORD'));

2) INAQTD all the employee details those who are earning a salary more than MILLER or FORD.
Select * from emp where sal > any (Select sal from emp where ename in ('MILLER', 'FORD'));

3) QTD name and salary of the employees those who are earning more than the employees of deptno=20
Select ename, sal from emp where sal > all (Select sal from emp where deptno = 20);

5) Q ID name and hiredate of the employees those who were hired before a clerk.
Select ename, hiredate from emp where sal < any (select sal from emp where job = 'CLERK');

6) Q ID name and hiredate of the employees those who are hired after all the managers.
Select ename, hiredate from emp where hiredate > all (select hiredate from emp where job = 'MANAGER');

6) Q ID, empname along with the designation those who are working in the same designation as FORD or WARD.
select ename, job from emp where job in (select job from emp where ename in ('FORD', 'WARD'));

7) Q ID all the employee details those who are working as a salesman or manager.
Select * from emp where job in ('SALESMAN', 'MANAGER');

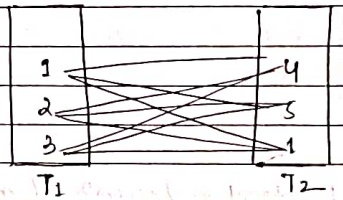
8) Q ID the details of all the reporting managers of those who have at least 3 employees working under them.
select * from emp where empno in (select mgr from emp group by mgr having count(*) >= 3);

JOINS

- Joins are used to merge one or more tables and fetch a data from multiple Tables.
- Different types of Joins are -
  i) cross join
  ii) inner join
  iii) outer join → left outer join, right outer join, full outer join
  iv) self join

Cross Joins :-
- It is also called as Cartesian joins.
- Each and every record from one table matches with each and every record of another table. and displays the o/p as both matched and unmatched data.
- Syntax :
  Select */colname from T_1, and T_2;

eg:



[3 x 3 = 9 records]

o/p →

1	4
1	5
1	1 → matched
2	4
2	5
2	1
3	4
3	5
3	1

unmatched

eg-1) Select * from emp, dept;
o/p → 56 rows selected.

INNER JOINS

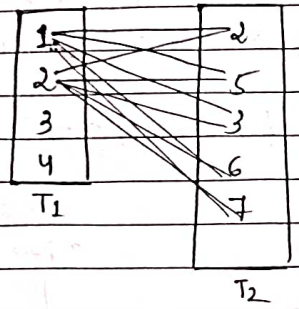
- It is also called as equijoins.
- Each and every record of one table matches with each and every record of another table and displays the output as only matched records.
- To apply the inner joins condition both the tables common column should be present.

Syntax: select * / colname from T_1, T_2 where $T_1.cc = T_2.cc$;

Syntax

- To merge more than 2 tables.

Select * from T_1, T_2, T_3 where $T_1.cc = T_2.cc$ and $T_2.cc = T_3.cc$;



O/P → 2 2
3 3

1) INNER JOIN employee and dept inform of all the employee
Select * from emp, dept where emp.deptno = dept.deptno;

2) INNER JOIN employee name along with their dept name
Select ename, dname from emp, dept where emp.deptno = dept.deptno;

3) JOIN empname, salary and location of all the employees.
Select ename, sal, loc from emp, dept where emp.deptno = dept.deptno;

4) JOIN empname, deptname and annual salary of the employees those who are working in the sales dept.

5) JOIN empname, deptname, salary and designation of the employees those who are working as a salesman except those who are located at DALLAS.

6) JOIN empname, deptname, deptno of the employees those who are working in the deptno 30.
Select ename, dname, emp.deptno from dept, emp where emp.deptno = dept.no and emp.deptno = 30;
or Select ename, dname, e.deptno from emp e, dept d where e.deptno = d.deptno and d.deptno = 30;

7) JOIN deptname and average salary of each dep
Select dname, avg(sal) from emp, dept where emp.deptno = dept.deptno group by dname;

8) JOIN only the employee information of those wh are working in RESEARCH dept.

9) JOIN only the dept information of the employe whose name is martin.

10) QID empname, salary, deptname of the employees those who are earning a salary more than CLARK.

11) QID empname, deptname and hiredate of the employees those who are hired before king.

12) QID empname, commission and deptname of the employees those who are getting commission in the deptno 30.

13) QID deptname and designation of the employees whose designation and deptname starts with the letters.

14) QID empname, deptname and location of the employees those who are located at NEWYORK and earning a salary less than 3000.

OUTER JOINS

In outer joins there are 3 types
- left outer joins
- right outer joins
- full outer joins

LEFT OUTER JOINS :-

- Each and every record of one table matches with each and every record of another table and display the output as matched data from both the table and unmatched data from only left side of the table.
- Syntax :
  Select */colname from T_1, T_2 where $T_1.cc = T_2.cc(+)$;
- eg :

T_1
4
5
6
7
8
9

T_2
1
2
3
4
5
5
7

o/p →

4	4
5	5
5	5
7	7
6	–
8	–
9	–

RIGHT OUTER JOINS :-

- Each and every record of one table matches with each and every record of another table and display the output as matched data from both the table and unmatched data from only right side of the table.
- Syntax :
  Select */colname from T_1, T_2 where $T_1.cc(+) = T_2.cc$;

NOTE (ANSI RULE SYNTAX)

1) Cross join:
   select */colname from T1 cross join T2;

2) Inner join:
   select */colname from T1 join T2 on T1.cc = T2.cc;
   (Inner join OR)

3) Left outer join:
   select */colname from T1 left outer join T2 on
   T1.cc = T2.cc;

4) Right outer join:
   select */colname from T1 right Outer join T2 on
   T1.cc = T2.cc;

5) Full outer join:
   select */colname from T1 full outer join T2 on
   T1.cc = T2.cc;

6) Self join:
   select */a.colnames from T1 a join T1 b on
   a.cc = b.cc; a & b → alias name.

1) QTD empname and deptname of the employees and departments even though the employee dont working in any department and the dept having no employees.
   select ename, dname from emp full outer join
   dept on emp.deptno = dept.deptno;

2) QTD empname and deptname of the employees even though there are no employee in a dept.
   select ename, dname from emp right Outer
   join dept on emp.deptno = dept.deptno;

3) QTD empname and deptname of the employees even though the employees donot working in any dept.

select ename, dname from emp left outer join dept on emp.deptno = dept.deptno;

eg-1)

A
10
20
30
50
70

B
40
10
20
10
50
80
90
10

1) cross join: 40 records

2) inner join:

10	10
10	10
10	10
20	20
50	50

(5 records)

3) left outer join:

10	10
10	10
10	10
20	20
50	50
30	-
70	-

(7 records)

4) right outer join:

10	10
10	10
10	10
20	20
50	50
-	40
-	80
-	90

(8 records)

5) full outer join:

10	10
10	10
10	10
20	20
50	50
30	-
70	-
-	40
-	80
-	90

(10 records)

ii) $S \rightarrow \varepsilon \mid \varepsilon + S \mid \varepsilon * S$

$S_1 \rightarrow \varepsilon S \mid \varepsilon$

$\varepsilon \rightarrow b$

$F(S) = \{\varepsilon, a\}$ $\quad \varepsilon \neq \phi$

$E(S_1) = \{\varepsilon, \ldots\}$ $\quad \varepsilon, a \neq \phi$.

$Fo(S_1) = \{Fo(S)$

$Fo(S) = E(S_1) - \{\varepsilon\} \cup Fo(S_1)$

$\therefore$ Not LL(1)

SELF JOINS :-

- Joining or merging the table itself is called as self joins.
- Why we go for self joins?
  Whenever the data to select is in the same table but present in different records at that time we go for self joins.
- To achieve self joins we can make use of aliasing.
- syntax:
  select */a.colname from T1 a, T1 b where a.cc = b.cc;
  (or diff column a.c = b.c) - data should be same.
- eg:

empno	ename	sal	MGR
123	Allen	100	789
456	smith	200	123
789	ward	300	456

Emp a

empno	ename	sal	M
123	Allen	100	7
456	smith	200	1
789	ward	300	4

Emp b

1) WAQTD all the employee details those who are earning same salary.

```
select * from emp a, emp b where
a.deptno = b.deptno;
a.ename != b.ename;
a.empno != b.empno;
```

(sal)

select * from emp a, emp b where a.sal = b.sal and a.empno != b.empno;

1) QTD name of the employees those who have similar names.

Select a.ename from emp a, emp b where a.ename = b.ename and a.empno != b.empno;

2) QTD ename and hiredate of the employees those who are hired on the same date.

Select a.ename, a.hiredate from emp a, emp b where a.hiredate = b.hiredate and a.empno != b.empno;

3) QTD ename and their manager name. (Imp)

Select a.ename, b.ename from empa, empb where a.mgr = b.empno;

4) QTD ename and their manager name along with their dept numbers those who are working in the deptno 30.

Select a.ename, b.ename, a.deptno, b.deptno from emp a, emp b where a.mgr = b.empno and a.deptno = 30;

→ ANSI RULE SYNTAX FOR INNER JOIN :—

Select * from T_1 join T_2 on $T_1.CC = T_2.CC$ join T_3 on $T_1.CC = T_3.CC$;

NOTE

Explain about joins.

1. def^n
2. syntax
3. e.g
4. O/P based on the type of join.

DISTINCT

- It is used to display the unique data from the given column.
- Syntax: select distinct(colname) from Tablename;

1) Select distinct(deptno) from emp;

deptno
30
20
10

- We can't use 2 distinct for one select statement.

2) select distinct(ename), distinct(deptno) from emp;

o/p → error.

- We can't pass 2 arguments for 1 distinct.

3) Select distinct(sal, deptno) from emp;

o/p → error

- We can't use the combination of columnname and distinct columnname.

4) Select ename, distinct(job) from emp;

o/p → error

- We can use the combination of distinct columnname and columnname. Whenever we use this combination it will display the o/p based on the column.

5) Select distinct(job), ename from emp;

1) WAQTD name of the employee and his manager's name if employee is working as clerk.

Select a.ename, b.ename from emp a, emp b where a.mgr = b.empno and a.job = 'CLERK';

2) WAQTD name of the employee and manager's designation if manager works in dept 10 or 20.

Select a.ename, b.job from emp a, emp b where a.mgr = b.empno and b.deptno in (10,20);

3) WAQTD name of the emp and managers salary if employee and manager both earn more than 2300.

Select a.ename, b.sal from emp a, emp b where a.mgr = b.empno and a.sal > 2300 and b.sal > 2300;

4) WAQTD empname and manager's hiredate if employee was hired before 82.

Select a.ename, b.hiredate from emp a, emp b where a.mgr = b.empno and a.hiredate < '01-JAN-82';

5) WAQTD empname and manager's comm if employee works as saleman and manager works in dept 30.

Select a.ename, b.comm from emp a, emp b where a.mgr = b.empno and a.job = 'SALESMAN' and b.deptno = 30;

6) WAQTD empname and managername and their salaries if employee earns more than manager.

Select a.ename, b.ename, a.sal, b.sal from emp a, emp b where a.mgr = b.empno and a.sal > b.sal;

7) WAQTD empname and hiredate, managername and hiredate if manager was hired before employee.

```
select A.ename, A.hiredate, B.ename. B.hiredate
from EMP A, EMP B where A.mgrc = B.empno and
B.hiredate < A.hiredate;
```

8) WAQTD empname and manager name if both are working in same job.

```
select a.ename, b.ename from emp a, emp b where
a.mgrc = b.empno and a.job = b.job;
```

9) WAQTD empname and managerc name if manager is working as actual manager.

```
Select a.ename, b.ename from emp a emp b where
a.mgrc = b.empno and b.job = 'MANAGER';
```

10) WAQTD empname and managerc name along with their annual salaries if employee works in deptno 10,20 and manager's sal is greater than employee's salary.

```
select a.ename, a.sal*12, b.ename, b.sal*12 from
emp a, emp b where a.mgrc = b.empno and a.deptno in
(10,20) and b.sal > a.sal;
```

11) WAQTD employee's name and manager's designation for all the employees.

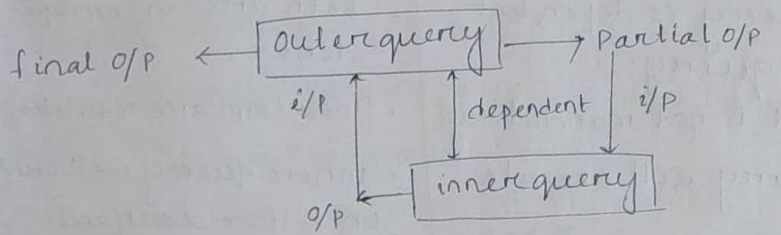
```
Select a.ename, b.job from emp a, emp b where
a.mgrc = b.empno;
```

12) WAQTD employee's name and manager's salary for all the employees if manager's salary ends with 50.

```
select a.ename, b.sal from emp a, emp b where a.mgrc =
b.empno and b.sal like '%50';
```

CORELATED SUBQUERIES

- Co-related subqueries are used for row-by-row processing.
- A query written inside an another query so that the inner query and outer query both are interdependent on eachother.



Execution flow of co-related subquery :-

- Outer query executes first but Partially.
- The partially executed o/p is the given as an i/p to the inner query.
- The inner query executes and generate an o/p.
- The o/p of inner query is given as the i/p of the outer query, and outer query genereates the result.
- Therefore we can state that both outer query and inner query are inter-dependent.

NOTE :-

- In correlated subquery joins condition is must and that joins condition should be present in the inner query.
- correlated subquery works with the Principles of both subquery and joins.

Difference bet^n subquery and correlated subquery

Subquery	correlated Subquery
• Inner query executes first.	• outer query executes first
• Outer query is dependent on inner query.	• Both are interdependent.
• Joins cond^n is not mandatory.	• Joins cond^n are mandatory.
• inner query will execute once.	• inner query will execute once for each and every record Present in outer query.

1) WAQTD the employee name along with 4th highest salary.

select ename, sal from emp where sal = (select max(sal) from emp where sal < (select max(sal) from emp where sal < (select max(sal) from emp where sal < (select max(sal) from emp))));

2) 4th highest salary with employee name using correlated subquery.

Select distinct (e1.sal), e1.ename from emp e1 where 4 = (select count (distinct (e2.sal)) from emp 2 where e2.sal >= e1.sal);

1st step :-

distinct (sal)
800
900
950
1000
2000

Partial o/p from outer query

2nd step :- The Partial o/p is given to inner query.

where e2.sal >= e1.sal

5	800, 900, 950, 1000, 2000 >= 800
4	800, 900, 950, 1000, 2000 >= 900
3	800, 900, 950, 1000, 2000 >= 950
2	800, 900, 950, 1000, 2000 >= 1000
1	800, 900, 950, 1000, 2000 >= 2000

count (distinct (sal))

4 = count (distinct (sal))

3) find 5th highest salary.

1st step :- sal	step-1: distinct (sal)
3000	3000
3550	3550
4000	4000
800	800
1000	1000
5000	5000
	9000

Step-2:-

3000, 3550, 4000, 800, 1000, 5000, 9000 = 3000

3000, 3550, 4000, 800, 1000, 5000, 9000 = 3550

3000, 3550, 4000, 800, 1000, 5000, 9000 = 4000

3000, 3550, 4000, 800, 1000, 5000, 9000 = 800

3000, 3550, 4000, 800, 1000, 5000, 9000 = 1000

3000, 3550, 4000, 800, 1000, 5000, 9000 = 5000

3000, 3550, 4000, 800, 1000, 5000, 9000 = 9000

Step-3: 5 = count (distinct (sal)

5	⟶ 3000
4	
3	
7	
6	
2	
1	

4) QTD 4th lowest salary.

Select & distinct (sal e1.sal) from emp e1 where 4 = (select count (distinct (e2.sal)) from emp e2 where e2.sal <= e1.sal);

5) WAQTD 3rd highest and 5th highest salary.

Select distinct (e1.sal) from emp e1 where (select count (distinct (e2.sal)) from emp e2 where e2.sal <> = e1.sal) in (3,5);

6) QTD 3rd highest, and 2nd lowest salary.

Select distinct (e2.sal) & from emp e1 where 3 = (select count (distinct (e2.sal)) from emp e2 where e2.sal >= e1.sal) or 2 = (select count (distinct (sal)) from emp e2 where e2.sal <= e1.sal);

PSEUDOCOLUMNS

In this we have 2 types i) row num
ii) row id

rownum :-

- rownum is a hidden column or virtual columns which are present in the table which gives numbers for each and every record present in the table.
- By default rownum starts from 1 and it works for '<' or '<=' operators.
- only for '1' eq equal to (=) operator will work.

eg: i) select * from ~~dept~~ dept where rownum = 1;

o/p → From the above query it will display the first record.

ii) select rownum from dept;

o/p →

rownum
1
2
3
4

iii) select * from emp where rownum = 2;

o/p → no rows selected.

iv) select * from emp where rownum <= 2;

o/p → It will display from the first record to second record.

v) INAQTD to display 3rd record of emp table.

Select * from (select emp.*, rownum as a from emp) where a = 3;

vi) INAQTD 5th record of emp table.

Select * from (select emp.*, rownum a from emp) where a=5;

vii) INAQTD 4th and 6th record of emp table.

select * from (select emp.*, rownum a from emp) where a in (4,6);

viii) INAQTD last record of emp table.

~~Select emp.*, max (sal) from (select emp.*, rownum a from emp) group by~~

Select * from (select emp.*, rownum a from emp) where a = (select count(*) from emp);

or Select * from (select emp.*, rownum a from emp) where a=1 order by a desc;

ix) INAQTD 2nd last record of emp table.

Select * from (select emp.*, rownum a from emp) where a (select count(*)-1 from emp);

x) INAQTD 3rd highest salary.

Select min(sal) from (select distinct (sal) from emp order by sal desc) where rownum <=3;

xi) INAQTD 2nd least salary.

Select max(sal) from (select distinct (sal) from emp order by sal) where rownum <=2;

↳ nth highest → desc → min

↳ nth least → asc → max

xii) INAQTD all the Team Lead details who has minimum 3 employee working under them.

Select * from emp where empno in (select mgr from emp group by mgr having count(*) >=3);

Interview Question

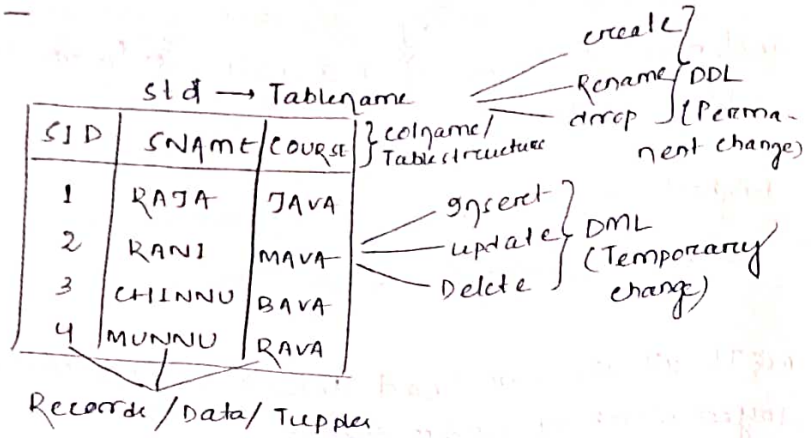
Select comm from emp order by comm asc;

comm
0
300
500
1400
.
.
.
.

Select comm from emp order by comm desc;

comm
.
.
.
.
1400
500
300

NOTE :-



DDL

- By using DDL commands we will be able to create a table, read a table and drop a table inform" from database.
- All DDL commands are permanent change.
- To create a table we should know two things :- datatypes and constraints.

Syntax for creating a table :-

```
create Table Tablename
(colname. datatype(size), constraint,
 colname datatype(size) constraint,
 - - - - - - - - );
```

eg :-
```
create table hebbal
(uno number(5) unique,
 uname char(10) not null);
```

```
select * from tab/cat ;
```

ALTER

Alter command is used to add a column to an existing column or for renaming a column or dropping a column from an existing table.

Syntax for adding a column :

```
Alter table tablename
add(columnname, datatype(size), constraint);
```

eg :-
```
alter table hebbal
add (mobno. number(10) unique);

desc hebbal;
```

RENAME

Syntax :-
```
ALTER Table Tablename
Rename column old colname to new colname;
```

eg :-
```
Alter Table Hebbal
Rename column mobno to mno;
```

DROP

Syntax :-
```
ALTER Table Tablename
drop column colname;
```

eg:
```
alter table hebbal
drop colum mno;
```

Permanently change
columname - Alter rename
Temporary change - aliasing

Syntax for Renaming a table

```
Rename old Tablename to new Tablename;
```

eg:- Select rename hemnat to Jspiders.

Syntax for dropping a Table :-

Drop table tablename;

eg:- Drop table Jspiders;

Difference:-

Drop	Delete	Truncate

- drop and truncate are DDL commands whereas delete is a DML command.
- drop will delete the complete data from the database whereas truncate will delete all the records from the tables but table structure remains same. and delete will delete the particular record from the table.

Queries using rownum

1) INA@TD 7th record of student table.
Select * from (select ~~student~~ *, rownum rn from student)
where a = 7;

2) INA@TD last record of employee table
Select * from (select emp.*, rownum rn from emp) where rn = (select count(*) from emp;

3) INA@TD last record of student table.
Select * from (select student.*, rownum rn from student)
where rn = (select count(*) from student);

4) INA@TD Last but one record of dept table.
Select * from (select dept.*, rownum rn from dept)
where rn = (select count(*) -1 from dept);

5) @TD Last 4 records of employee table.
Select * from (select emp.*, rownum rn from emp)
where rn > (select count(*) -4 from emp);

6) @TD first 50% records of employee table.
Select * from (select emp.*, rownum rn from emp) where
rn <= (select count(*)/2 from emp);

7) @TD last 50% records of emp table.
Select * from (select emp.*, rownum rn from emp where
rn > = (select count(*)/2 from emp);

8) @TD even records of employee table.
Select * from (select emp.*, rownum rn from emp)
where mod(rn, 2) = 0;

9) @TD odd records of employee table.
Select * from (select emp.*, rownum rn from emp)
where mod(rn, 2) != 0;

10) @TD first 4 records of employee table.
Select * from (select emp.*, rownum rn from emp)
where a <= (select count(*) from emp);

or: Select * from emp where rownum <= 4;

11) INA@TD 3rd highest salary with emp name.
Select ename, sal from emp where sal = (select min(sal)
from (select distinct (sal) from emp order by sal desc)
where rownum <= 3);

12) @TD first 3 highest salaries in the employee table.
Select sal from (select distinct (sal) from emp order by
sal desc) where rownum <= 3;

13) Q TO employee name, deptname along with 4th highest salary.

Select ename, dname, sal from emp, dept where emp.deptno = dept.deptno and sal = (select min(sal) from (select distinct(sal) from emp order by sal desc) where rownum <= 4);

→ create a table for the following data

Product		Orders	
Pname	NOT NULL	Oname	NN
Pid	PK	Oid	P.K
Qty	>4	Pid	FK
mdate		Addr	unique
		Contactno	

Create Table Product
(Pname varchar2(15) not null,
Pid number(5) Primary key,
Qty number(3) check(Qty>4),
mdate date);

create Table orderes
(oname, varchar2(10) not null,
Oid number(5) Primary key,
Pid number(6) References product(pid);
Addre varchar2(50) unique,
contactnum number(12));

7

elg	std
elgname	Sname
elgcode	Sid
Loc	age
contactnum	marks > 35
	elgid

create table elg(
elgname varchar2(15) not null,
elgcode number(6) primary key,
Loc varchar2(25),
contactnum number(12));

create table std(
Sname varchar2(15) not null,
Sid number(4) unique,
age number(2),
marks number(2) check(marks<735),
elgid number(4) references elg(elgcode));

P.K & F.K
↓
datatype must be same
size may be different
colname may be different

Syntax to create a duplicate Table:-

create table Newtablename
as
Select * from oldtablename;

{Difference bet" duplicate table and original table is, in duplicate table the constraints will not be visible}

TRUNCATE

Using this statement we can remove all the records from the table but structure remains same.

Syntax :-

```
TRUNCATE table Tablename;
```

Once we truncate the table we can't restore the data.

eg:> truncate table emp24;

> select table truncated.

> Select * from emp24;

> O/P → no rows selected.

{ We can't restore the lost data but can insert new record a/c to the existing table structure }

DROP :-

Using this statement it will remove all the data along with the structure of the table.

Syntx: Drop table tablename;

Once you drop the table, the table is present in the recycle bin.

FLASHBACK :-

This statement is used to restore the table that is present in Recycle Bin.

Syntax: flashback table tablename to before drop;

PURGE :-

This statement is used to remove the table Permanently from the database.

Syntax:- Purge table tablename;

We can Purge the table only if the table is present in Recycle bin.

ALTER

This statement is used to change the structure of the table like by adding a column, by removing a column, by changing the column name and by modifying the column datatype.

Syntax to add multiple columns for the existing table.

```
Alter table tablename
add (columname datatype(size),
     columname datatype(size));
```

Syntax to remove multiple columns from the existing table

```
Alter table tablename
drop (coln1, coln2, coln3);
```

eg:- alter table jsp
drop(sal, comm);

{ To add not null constraint to any table first the column needs to be empty, other constraints we can add even though the column is not empty. }

Syntax to modify the column datatype:-

```
Alter table tablename
modify colname Newdatatype(size);
```

If we want to modify the column datatype the column should be empty to change the datatype.

DML STATEMENTS

- Using this statement we can manage the database by inserting a data, deleting a data and by updating a data.
- All the DML statements are temporary change.

↳ INSERT :-

using this statement we can add a new record for the existing table.

→ Syntax : Insert into Tablename
values (v_1, v_2, v_3);

→ Syntax to add a data to a particular column.
Insert into Tablename
(colname1, colum2) values (v_1, v_2);

→ Syntax to add multiple records for Particular column.
Insert into Tablename (col1, col2, col3)
values (&v_1, '&v_2', '&v_3');

→ Syntax to add multiple records
Insert into Tablename
values (&v_1, &v_2, '&v_3');

↳ UPDATE :-

using this statement we can modify the existing data with a new data.

update tablename set colname = 'Newvalue';

→ To update particular data we can make use of where clause.
update tablename set colname = 'Newvalue' where <condition>;

→ Syntax to update multiple columns
update tablename set colname = 'Newvalue', colname = 'Newvalue'
where <condition>;

↳ Delete :-
using this statement we can delete all the data or we can delete particular records.

{ from Delete can be used for select and Delete }

→ Syntax :- Delete Tablename;
or Delete from Tablename;

Delete Tablename where <condition>;

→ Syntax to delete duplicate records. (Imp)
Delete from Tablename where rowid not in
(select min(rowid) from Tablename group by colname);

↳ Differences between Truncate and Delete.

Truncate	Delete
• It belongs to DDL statement	• It belongs to DML statement.
• It is a Permanent change.	• It is a temporary change.
• It remove all the records Permanently	• We can delete particular record.
• We can't use where clause.	• We can use where clause.
• Once we truncate we can't restore the records.	• Once we delete we can restore the records.

↳ Difference between:

Drop	Delete
• It is a DDL statement.	• It is a DML statement.
• It's a permanent change.	• It's a temporary change.
• It remove structure along with the data.	• It deletes particulare record but structure remains same.
• We can't use where clause.	• We can use where clause.
• once we drop we can restore using flashback.	• once we delete we can restore using rollback.

TRANSACTION CONTROL LANGUAGE (TCL)

- All the DML statements are temporary change in order to make them permanent we use TCL.
- It includes commit, rollback and savepoint.

↳ COMMIT :-

- This statement permanently save all the changes made in the transaction of a database.
- once you execute commit the database cannot go back to the Previous state.

Syntax :- commit ;

↳ ROLLBACK :-

- It is similare like an undo function using this statement we can restore the data to the Previous state.
- Rollback will not work after the commit.
- Syntax :- Rollback ; (Rollback is applicable for all the DML operations done Previously)

↳ SAVEPOINT :-

- This statement name and mark the current transaction.
- using rollback we can restore a particular transaction instead of the whole transaction.
- Syntax :- Save savepointname ;
  Rollback to savepointname ;

DATA CONTROL LANGUAGE :-

using this statement we can give the access permission and we can take back the access permission from the other users.

i) GRANT :-

- using this statement we can give the access permission to other users.
- Syntax :-
  grant select on Tablename to user ;
  ~~Revoke~~ select on

ii) REVOKE :-

- using this statement we can take back the access Permission from the user.
- Syntax :- Revoke select on Tablename from user ;

NORMALIZATION

The need of Normalization are -:

i) To reduce the data redundancy.

ii) To remove the anamolies like insertion, deletion and updation.

DATA REDUNDANCY :-

- Data redundancy is nothing but when same data exists in multiple locations.

ANAMOLY :-

It refers to the problem occures due to the DML changes operations.

INSERTION ANAMOLY :-

This refers to when one cannot insert a new record to the table due to lack of data.

UPDATION ANAMOLY :-

It refers to an update of single data value requires multiple rows of data to be updated.

DELETION ANAMOLY :-

This refers to the situation where deletion of particular data results in the loss of some other important data.

Table without Normalization

Stdid	Sname	marks	subcode	Sub
123	Anu	55	101	SQL
345	Manu	65	111	Java
567	banu	55	121	Python
123	Anu	55	101	SQL
789	Monu	57	101	SQL

Normalization defination :-

It is a process of decomposing a larger table into less redundant table without losing any information.

or

- Normalization is a database design technique where we can organize the data.
- Here larger table is divided into smaller table and we can build a relation between the table with the help of key attributes.

Stdid	Sname	marks	subcode
123	Anu	55	101
345	Manu	65	111
567	banu	55	121
789	Monu	57	101

↓ foreign Key

Subcode	Sub
101	SQL
111	Java
121	Python

↓ Primary Key

Different types of Normalization are

1NF – 1st Normalization form

2NF – Second Normalization form

3NF – Third Normalization form

3.5 NF or BCNF – Boyce codd Normalization Form.

First Normalization form :–

Rules :–

1) Each cell should consist of single value or automic value.
2) Each record should be unique.
3) The attribute name should be unique.

Stdid	sub
123	SQL, C
345	Python
789	INeb
123	SQL, C

[Table Not in 1NF]

Stdid	sub
123	SQL
123	C
345	Python
789	INeb

[Table in 1NF]

Second Normalization Form :–

Rules :–

1) Table should be in 1NF.
2) There should no partial dependency.

PARTIAL DEPENDENCY

A non-primary key column depends on part of the candidate primary key is called as partial dependency.

Table not in 2NF

Empno	deptno	officeloc
101	10	Bangalore
102	20	pune
103	30	pune
(1st candidate key)	(2nd candidate key)	(non-primary key)

Table with 2NF

deptno	officeloc
10	Bangalore
20	pune
30	pune
(Primary key)	

Empno	deptno
101	10
102	20
103	30
	(foreign key)

Third Normalization form :–

Rules :–

1) Table should be in 2NF.
2) There is no transitive functional dependency.

Transitive functional dependency

A non-primary key column depending on another non-primary key column is called as Transitive functional dependency.

Stdid	Sname	marks	Subcode	Sub
21	Dinga	55	10	SQL
22	Dingi	55	20	java
23	Sheela	56	30	python
24	Mala	60	20	java
			(Non-P.K)	(Non-P.K)

(Sub dependent on Subcode)

[Table not in 3NF]

Subcode	Sub
10	SQL
20	java
30	Python
(P.K)	

(Sub dependent on Subcode)

Stdid	Sname	marks	subcode
21	Dinga	55	10
22	Dingi	55	20
23	Sheela	56	30
24	Mala	60	20
			(F.K)

[Table in 3NF]

BCNF :—

↳ Rules — This normalization form was introduced by Raymond Boycee and E.F codd. Hence it is called as BCNF.

↳ After applying 3NF to the table if still data redundency is existing in the table, we are using 3.5 NF.

↳ Using this normal form we can provide strict rules to the table.

Rule :-

1) Table should be in 3NF.
2) If a column determines another column then determinent column should be super key column.

Super key

It is a set of all the key attributes which consists of only unique records.

Stdid	Sub	Trainer
10	SQL	x
20	java	y
30	Inleb	z

(Trainer dependent on Stdid; Stdid dependent on Trainer)

(candidate key is subset of super key)

[Table in 3.5NF]

Functional Dependency :-

- A Relation exists such that a column determines another column uniquely is called as functional dependency.
- There are 3 types in f.D
  - i) Partial F.D
  - ii) Transitive F.D
  - iii) Total F.D.

Total functional Dependency

If all the attribute of a relation determined by one column is called as total functional dependency.

Sl. No.	Date	Title	Page No.	Teacher's Sign/ Remarks

Select * from emp where empno in (select mgr from emp where ename in (select ename from emp where empno in (select mgr from emp group by mgr having count(*) <=3)));

① Display the empno, ename, job, hiredate, exp of all managers.

Select empno, ename, job, hiredate, to-char(sysdate, 'yyyy') - to-char(hiredate, 'yyyy') exp from emp where empno in (select mgr from emp);

job = manager

② List the empno, ename, sal, exp of all emps working for mgr 7369;

③ List the emps along with their exp and daily sal is more than Rs 100.

④ List the empno, ename, sal, dname of all the mgrs and analyst working in New York, Dallas with an exp more than 7 years without receiving the comm asc order of loc.

job in

Select empno, ename, sal, dname from Emp

manager

mgr is Null

Computer Academy