Oracle SQL

Agenda:

1. Introduction to DBMS
2. Introduction to SQL.
3. Oracle setup and installation (not in notes refer official documentation)
4. Relational model
5. Miscellaneous topics
6. Operators and Keywords ***
7. Functions ***
8. Clauses ***
9. Nested query ***
10. Joins ***
11. SQL commands ***
12. Database Objects and SP

1. # Introduction to DBMS

DBMS stands for DataBase Management System.

DBMS stores the information / data in organized / structured manner, DBMS also lets us retrieve the stored data by writing the required queries.

* By writing SQL queries we can retrieve the exact data from a huge database in other words SQL queries lets us filter the data and it helps us to retrieve the exact data which we want.

Back in the day in hotels or hospitals they used to use ledgers (A big book) to store the information / data.

Let us some problems associated in managing data in Ledgers:

1. Security challenges: The ledgre could be stolen or lost and entire information is also lost
2. Backup issues
3. Space issues
4. Human resource issues
5. inefficiency in retrieving facts

All the above problems can be overcomed using DBMS and SQL.

Introduction to SQL:

* SQL stands for Structured Query Language.
* SQL lets us access and manipulate the data.

What can SQL do?
SQL can execute queries against databases.
Using SQL we can perform below operations.

- Create Database
- Create table within DB
- Insert data into DB
- Update data
- Retrieve data
- Set permissions.

RDBMS:

RDBMS stands for Relational Data Base Management System.
RDBMS stores data in table format.

Some of the most important SQL keywords / Commands

- SELECT
- UPDATE
- DELETE
- INSERT INTO
- CREATE DATABASE
- ALTER DATABASE
- CREATE TABLE
- ALTER TABLE
- DROP

We will see all in detail.

Before learning the syntax of the commands it is important to know the below points:

- SQL keywords, Table name, Database names are case insensitive
- The Data within the table are case sensitive

Select Statement:

```
SELECT  Column_name/ function /*
FROM    table_name;
```

Database creation

```
CREATE DATABASE DB_name;
```

Table creation:

```
• CREATE TABLE table_name (
      Column1   data type   constraint,
      Column 2  data type   constraints,
        ⋮
      Column n  data type   constraint);
```

- `Clear screen;` can be used to clear the CLI screen
- `DROP TABLE table_name;` used to delete table.

4

Relational Model

PK

Roll Number	Student Name
1	Rohit
2	Mohan
3	Sanjay
4	Kiran

FK

Roll Number	Marks Obtained
1	45
2	78
3	89
4	56

Above is two different tables.
Now lets say we need to fetch marks of individual student. We can retrive this data as we have relation b/w them two table ie Roll no column.
We use Primary key & Foriegn key.

*** **Primary key**: Primary key is a constraint, It defines the unique columns Eg: Phone number, RollNo.
- Primary key will have all unique records
- Primary key is always Not NULL.

Foregin key: Foregin key is that column of second table or outside table that refers to main table or first table and establish relationship among them.
- The primary key in main table is foregin key in another.
"Refers" keyword can be used.

NOTE: A Single table can have both primary & foregin key columns.

Data types:

- Number, Float
- Char → Fixed length
- Varchar → Variable length
- Date

Operators:

- +
- −
- *
- /
- %.

Eg: Query to add 5000 to all emp salary.

```
SELECT Name, Salary + 5000 From Employee;
```

Aliasing:

Aliasing means a temporary nick name for a specific column. We can use "as" keyword (OR) we could just write alias name by leaving space.

Eg: Select USN as RNo, Name n, Gender g ...

DISTINCT Keyword:

Distinct will manipulate & retrieve the data that are Distinct in other words data that are not repeated.

Concatenation Operator:

"||" is used as Concatenation operator & is used to combine multiple data or column.

Eg: Select Name || Age as Name & age From Employee

Select Name || 'Aged' || Age as Name & Age From Employee

- Description of table

DESC table_name.

System date:

Select SYSDATE From table_name

Relational operators:

Operator	Description
=	Equal to
>	Greater than
<	Lesser than
>=	Greater than or equal to
<=	Lesser than or equal to
!=	Not equal to
< >	Not equal to
^ =	Not equal to

Eg: Select Name FROM EMPLOYEE
WHERE Salary >= 5000;

Operators as keywords:

- Between:

Select Name From Student
where Marks between 80 and 100;

- NOT:

Select Name From Student
where Marks NOT Between 80 and 100;

- **IN:**

```
Select * FROM student
WHERE Age IN (20,22);
```

It is not range.

- **LIKE:**

Like is used for pattern matching.

There are two main operators

- % → Matches 0 or more
- _ → matches only one.

Eg:

```
Select Name From Student
WHERE Name like 'A%';
```

Logical Operators:

OPERATORS	DESCRIPTION
AND	TRUE if all the conditions separated by AND is TRUE
OR	TRUE if any of the conditions separated by OR is TRUE
NOT	Displays a record if the condition(s) is NOT TRUE

Q. Write a query to display all the details of the students whose marks is greater than 85 and their names should not start with "R" from STUDENT table.

```
SELECT Name
FROM Student
WHERE Marks >85 AND Name NOT like 'R%';
```

7. Functions:

There are two types of functions

- Single row functions
- Multiple row functions
  - Aggregate functions (a.k.a.).

- Single row functions:

Q. Write a query to display NAME of all the students in lowercase letters from the table STUDENT.

```
SELECT LOWER(Name)
 From  Student.
```

Q. Write a query to display Initial letter capital of data 'RITWIK'.

```
SELECT INITCAP('RITWIK') AS "INITIAL" FROM DUAL;
```

Q. Write a query to display all the names and gender of the students in lowercase where gender is 'MALE'.

```
SELECT LOWER(Name), LOWER(Gender)
FROM   Student
WHERE  gender = 'Male';
```

Q. Write a query to concatenate data 'Ritwik' & 'Raj'.

Soln 1:

```
Select 'Ritwik' || 'Raj' as Name
 From  DUAL;
```

Sol 2:

```
Select CONCAT (Ritwik, 'Raj') as Name
 From  DUAL;
```

Q. Write a query to find the length of the string 'Ritwik'.

```
Select LENGTH ('Ritwik') From Dual;
```

Q. Write a query to display the substring of the string 'Sudeshna' from 3rd position extract 6 characters.

SELECT SUBSTR ('Sudeshna', 3, 6) FROM Dual;

O/P: deshna

Q. Write a query to display the position of the character 'a' in the string 'Chatterjee'.

SELECT INSTR ('Chatterjee', 'a') FROM Dual;

- TRIM

Eg: SELECT TRIM (LEADING 'a' from 'akash') From Dual;

SELECT TRIM (TRAILING 'a' from 'angelina') From Dual;

- ROUND (95.528, 2)

O/P : 95.53

- TRUNC (95.528, 2

O/P : 95.52

- mod (60, 2) → remainder

O/P 0

- MONTHS_Between ('24-MAR-2018', '24-DEC-2018')

- Add_Months ('24-May-2018', 7)

**** Aggregate Functions: (Multiple row functions)

- Count()
- Max()
- Min()
- Sum()
- Avg()

SQL Clauses:

- SELECT
- FROM
- WHERE
- ORDER BY
- HAVING
- GROUP BY
- JOIN

Eg: SELECT * FROM student Order By Age;
SELECT * FROM student order By Age DESC;

Note: Ascending By default.

Q. Write a query to display AGE and the least marks scored by each UNIQUE aged student from the table STUDENT.

```
SELECT Age, MIN(Marks)
FROM Student GROUP BY Age;
```

Q. Write a query to display GENDER and the least marks scored by each UNIQUE gender student from the table STUDENT.

```
SELECT Gender, Min(Marks)
FROM Student
GROUP BY Gender;
```

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
HAVING condition
ORDER BY column_name(s);
```

← Having Syntax

Q6. Write a query to display AGE and the MAXIMUM marks scored by all student whose age is greater than 19 and having maximum marks greater than 85 for each age, while displaying the data in ascending order with respect to age.

```
SELECT Age, Max(Marks)
FROM Student
WHERE age > 19
GROUP BY Age
HAVING Max(Marks) > 85
ORDER BY Age ;
```

9 Nested Query:

USN	NAME	GENDER	AGE	MARKS
120	Ram	Male	21	85
121	Ravi	Male	20	88
122	Kavitha	Female	22	90
123	Arjun	Male	22	87
124	Arjuna	Male	21	75
125	Manisha	Female	25	85
126	Sonal	Female	26	89
127	Rohan	Male	26	95

Q. Write a query to display USN, NAME and GENDER of all students whose marks is more than RAVI's marks from the table STUDENT.

```
SQL> SELECT usn,name,gender
  2  FROM student
  3  Where marks>(
  4  SELECT Marks FROM student WHERE name='Ravi'
  5  );

       USN NAME            GENDER
---------- --------------- ------
       122 Kavitha         Female
       126 Sonal           Female
       127 Rohan           Male
```

Inner query will get executed first.

Q. Write a query to display the USN and NAME for all the student whose age is same as ARJUN's age from the table STUDENT.

```
SQL> SELECT USN,Name
  2  FROM student
  3  WHERE age = (
  4  SELECT age FROM student WHERE name = 'Arjun');

       USN NAME
---------- ---------------
       122 Kavitha
       123 Arjun
```

NOTE: ORDER BY Cannot be used in subquery.

Q. Write a query to display NAME, AGE whose age is equal to student named RAM and marks is greater than student named ANJUNA from the table STUDENT.

```
SQL> SELECT name,age
  2  FROM student
  3  WHERE age = (SELECT age from student where name = 'Ram')
  4  AND
  5  Marks>(Select marks from student where name = 'Arjuna');

NAME                 AGE
-------------- ----------
Ram                    21

SQL>
```

Stu-dept table

DEPT_ID	USN	COURSE	CITY	PINCODE
21	125	Operating system	Banglore	560076
22	123	Management system	Pune	426520
23	124	Engineer system	Mumbai	460548
24	122	Microprocessor	Banglore	560076
25	121	Microcontroller	Banglore	560076
26	126	Semiconductor	Banglore	560076

6 rows selected.

Q6. Write a query to display AGE and the MAXIMUM marks scored by all student whose age is greater than 19 and having maximum marks greater than 85 for each age, while displaying the data in ascending order with respect to age.

```
SQL> SELECT age, max(marks)
  2  FROM student
  3  WHERE age>19
  4  GROUP BY age
  5  Having max(marks)>85
  6  ORDER BY age;

       AGE MAX(MARKS)
---------- ----------
        20         88
        22         90
        26         95
```

Q. Write a query to display the DEPT_ID of all the students from STU-DEPT table whose USN in STUDENT table is equal to USN in STU-DEPT.

```
SQL> SELECT dept_id
  2  FROM stu_dept
  3  WHERE USN IN (SELECT USN FROM student);

   DEPT_ID
----------
        21
        22
        23
        24
        25
        26
```

IN can be used if sub query return multiple rows

10

JOINS

Very important

PK

EMP_ID	NAME	GENDER	AGE	SALARY
206	AMAN	MALE	24	350000
208	TANISHA	FEMALE	27	460000
203	HARPREET	FEMALE	29	890000
201	RAM	MALE	23	358000
205	VISHAL	MALE	25	560000
204	SHUBHAM	MALE	29	679000
207	ROHAN	MALE	28	700000
210	PRIYANKA	FEMALE	27	650000

EMPLOYEE

FK

DEPT_ID	EMP_ID	DEPARTMENT	CITY	PINCODE
21	205	HR	BANGALORE	560076
23	203	DEVELOPER	DELHI	879009
23	206	DEVELOPER	PUNE	476501
22	207	SALES	BANGALORE	568077
21	201	HR	PUNE	564487
22	210	SALES	MUMBAI	656009
24	209	ANALYST	AHEMDABAD	678879
25	200	ANALYST	PUNE	238977

EMP_DEPT

Write a query to display NAME, GENDER, AGE and MARKS of all employees whose EMP_ID in the EMPLOYEE table is equal to the EMP_ID in the EMP_DEPT table.

```
SQL> SELECT name,gender,age,salary
  2  FROM Employee e, Emp_dept ed
  3  WHERE e.emp_id = ed.emp_id;

NAME            GENDER          AGE      SALARY
--------------- ------ ---------- ----------
Aman            Male            24      350000
Harpreet        Male            29      890000
Ram             Male            23      358000
Vishal          Male            25      560000
Rohan           Male            28      700000
Priyanka        Female          27      650000
```

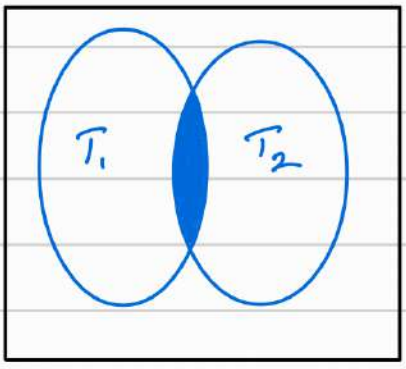
Very imp

Joins: Joins is a means for combining fields from two tables (or more) by using values common to each.

Types of Join:

- Inner Join
- Left Join
- Right Join
- Full Join
- Natural Join
- Cross Join

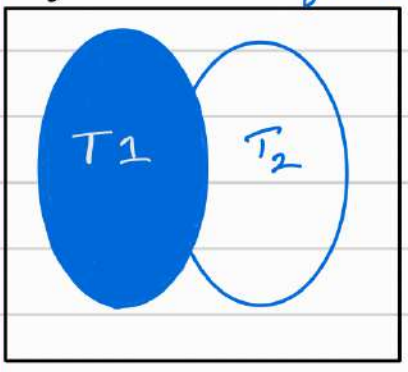
- **Inner Join:** Returns all the rows when there are at least one match in both the table



The dark blue shaded part represents the Inner Join. Common column will print twice.

Keyword: Join / Inner Join

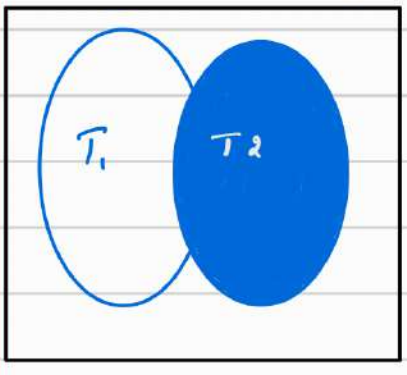
- **Left Join:** Left Join selects all rows from the left and common rows from the right table.



The dark blue shaded part represents the data that will be retrieved

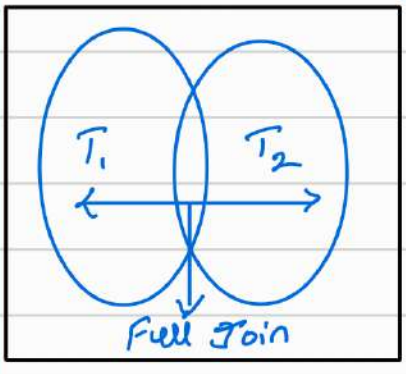
Keyword: Left Join

- **Right Join:** Right Join is similar to left join. It selects all rows from the Table 2 and common rows from table 1.



Keyword: Right Join

- **Full Join:** Full join returns all rows when there is a match in one of the tables.



Keyword: Full Join

```
SQL> SELECT *
  2  FROM EMPLOYEE LEFT JOIN EMP_dept
  3  ON employee.emp_id = emp_dept.emp_id;

    EMP_ID NAME           GENDER       AGE     SALARY    DEPT_ID     EMP_ID
---------- -------------- ------ ---------- ---------- ---------- ----------
DEPARTMENT CITY         PINCODE
---------- ---------- ----------
       205 Vishal         Male          25     560000         21        205
HR         Banglore       560076

       203 Harpreet       Male          29     890000         23        203
Developer  Delhi          879009

       206 Aman           Male          24     350000         23        206
Developer  Pune           476501


    EMP_ID NAME           GENDER       AGE     SALARY    DEPT_ID     EMP_ID
---------- -------------- ------ ---------- ---------- ---------- ----------
DEPARTMENT CITY         PINCODE
---------- ---------- ----------
       207 Rohan          Male          28     700000         22        207
Sales      Banglore       568077

       201 Ram            Male          23     358000         21        201
HR         Pune           564487

       210 Priyanka       Female        27     650000         22        210
Sales      Mumbai         656009


    EMP_ID NAME           GENDER       AGE     SALARY    DEPT_ID     EMP_ID
---------- -------------- ------ ---------- ---------- ---------- ----------
DEPARTMENT CITY         PINCODE
---------- ---------- ----------
       204 Shubham        Male          29     679000


       208 Tanisha        Female        27     460000
```

```
SQL> SELECT *
  2  FROM employee RIGHT JOIN emp_dept
  3  ON employee.emp_id = emp_dept.emp_id;

    EMP_ID NAME           GENDER       AGE     SALARY    DEPT_ID     EMP_ID
---------- -------------- ------ ---------- ---------- ---------- ----------
DEPARTMENT CITY         PINCODE
---------- ---------- ----------
       206 Aman           Male          24     350000         23        206
Developer  Pune           476501

       203 Harpreet       Male          29     890000         23        203
Developer  Delhi          879009

       201 Ram            Male          23     358000         21        201
HR         Pune           564487


    EMP_ID NAME           GENDER       AGE     SALARY    DEPT_ID     EMP_ID
---------- -------------- ------ ---------- ---------- ---------- ----------
DEPARTMENT CITY         PINCODE
---------- ---------- ----------
       205 Vishal         Male          25     560000         21        205
HR         Banglore       560076

       207 Rohan          Male          28     700000         22        207
Sales      Banglore       568077

       210 Priyanka       Female        27     650000         22        210
Sales      Mumbai         656009

6 rows selected.
```

Left Join

Right Join

```
SQL> SELECT *
  2  FROM Employee e RIGHT JOIN Emp_dept ed
  3  ON e.emp_id = ed.emp_id
  4  WHERE e.salary>500000;

    EMP_ID NAME           GENDER       AGE     SALARY    DEPT_ID     EMP_ID
---------- -------------- ------ ---------- ---------- ---------- ----------
DEPARTMENT CITY         PINCODE
---------- ---------- ----------
       205 Vishal         Male          25     560000         21        205
HR         Banglore       560076

       203 Harpreet       Male          29     890000         23        203
Developer  Delhi          879009

       207 Rohan          Male          28     700000         22        207
Sales      Banglore       568077


    EMP_ID NAME           GENDER       AGE     SALARY    DEPT_ID     EMP_ID
---------- -------------- ------ ---------- ---------- ---------- ----------
DEPARTMENT CITY         PINCODE
---------- ---------- ----------
       210 Priyanka       Female        27     650000         22        210
Sales      Mumbai         656009
```

Full Join:

```
SQL> SELECT *
  2  FROM employee e FULL JOIN emp_dept ed
  3  ON e.emp_id = ed.emp_id;

    EMP_ID NAME           GENDER       AGE     SALARY    DEPT_ID     EMP_ID
---------- -------------- ------ ---------- ---------- ---------- ----------
DEPARTMENT CITY         PINCODE
---------- ---------- ----------
       206 Aman           Male          24     350000         23        206
Developer  Pune           476501

       208 Tanisha        Female        27     460000


       203 Harpreet       Male          29     890000         23        203
Developer  Delhi          879009


    EMP_ID NAME           GENDER       AGE     SALARY    DEPT_ID     EMP_ID
---------- -------------- ------ ---------- ---------- ---------- ----------
DEPARTMENT CITY         PINCODE
---------- ---------- ----------
       201 Ram            Male          23     358000         21        201
HR         Pune           564487

       205 Vishal         Male          25     560000         21        205
HR         Banglore       560076

       204 Shubham        Male          29     679000


    EMP_ID NAME           GENDER       AGE     SALARY    DEPT_ID     EMP_ID
---------- -------------- ------ ---------- ---------- ---------- ----------
DEPARTMENT CITY         PINCODE
---------- ---------- ----------
       207 Rohan          Male          28     700000         22        207
Sales      Banglore       568077

       210 Priyanka       Female        27     650000         22        210
Sales      Mumbai         656009

8 rows selected.
```

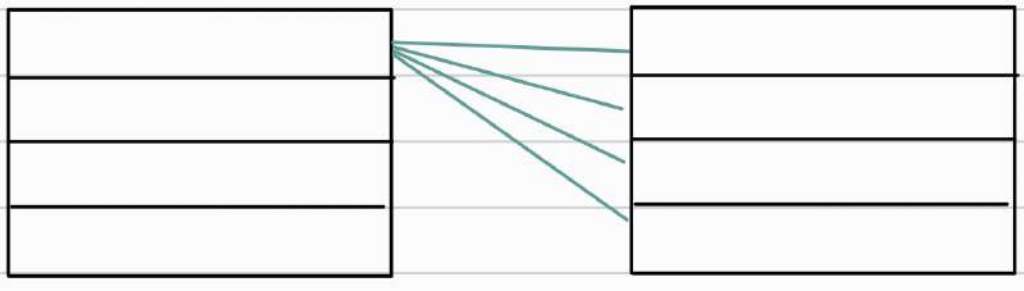
```
SQL> SELECT * FROM employee natural join emp_dept;

    EMP_ID NAME            GENDER        AGE     SALARY    DEPT_ID DEPARTMENT
---------- --------------- ------ ---------- ---------- ---------- ----------
CITY          PINCODE
---------- ----------
       206 Aman            Male           24     350000         23 Developer
Pune            476501

       203 Harpreet        Male           29     890000         23 Developer
Delhi           879009

       201 Ram             Male           23     358000         21 HR
Pune            564487


    EMP_ID NAME            GENDER        AGE     SALARY    DEPT_ID DEPARTMENT
---------- --------------- ------ ---------- ---------- ---------- ----------
CITY          PINCODE
---------- ----------
       205 Vishal          Male           25     560000         21 HR
Banglore        560076

       207 Rohan           Male           28     700000         22 Sales
Banglore        568077

       210 Priyanka        Female         27     650000         22 Sales
Mumbai          656009

6 rows selected.
```

No duplicate column.

We can use both
- Natural left Join
- Natural Right Join

- No condition is required for Natural Join

- Cross Join:



Similar to matrix multiplication.

- SELF Join: used to join table to itself

Select from multiple tables without join:

Write a query to display all the details of employees whose EMP_ID in the EMPLOYEE table is equal to the EMP_ID in the EMP_DEPT table without using Joins.

```
SQL> SELECT *
  2  FROM employee e, emp_dept ed
  3  WHERE e.emp_id = ed.emp_id;

    EMP_ID NAME            GENDER        AGE     SALARY    DEPT_ID     EMP_ID
---------- --------------- ------ ---------- ---------- ---------- ----------
DEPARTMENT CITY          PINCODE
---------- ---------- ----------
       206 Aman            Male           24     350000         23        206
Developer  Pune           476501

       203 Harpreet        Male           29     890000         23        203
Developer  Delhi          879009

       201 Ram             Male           23     358000         21        201
HR         Pune           564487


    EMP_ID NAME            GENDER        AGE     SALARY    DEPT_ID     EMP_ID
---------- --------------- ------ ---------- ---------- ---------- ----------
DEPARTMENT CITY          PINCODE
---------- ---------- ----------
       205 Vishal          Male           25     560000         21        205
HR         Banglore       560076

       207 Rohan           Male           28     700000         22        207
Sales      Banglore       568077

       210 Priyanka        Female         27     650000         22        210
Sales      Mumbai         656009


6 rows selected.
```

Great Learning

Join Multiple Tables with Conditions

Write a query to display salary, employee id from EMPLOYEE table and department, city from EMP_DEPT table whose EMP_ID in the EMPLOYEE table is equal to the EMP_ID in the EMP_DEPT table and salary is greater than 400000.

```
SQL> SELECT e.salary, e.emp_id, ed.department,ed.city FROM employee e, emp_dept ed
  2  WHERE e.emp_id = ed.emp_id and e.salary>400000;

    SALARY     EMP_ID DEPARTMENT CITY
---------- ---------- ---------- ----------
    560000        205 HR         Banglore
    890000        203 Developer  Delhi
    700000        207 Sales      Banglore
    650000        210 Sales      Mumbai
```

Joining 3 tables in SQL:

EMP_ID	EMP_FIRSTNAME	EMP_LASTNAME
1	RAM	KUMAR
2	ROHAN	DAS
3	SAURABH	GOYAL

EMPLOYEE

EMP_ID	DEPT_ID
1	2
1	3
2	1
2	2
2	3
2	3
3	1

EMP_DEPT

EMP_ID	DEPT_NAME	DEPT_HEAD_ID
1	HR	1
2	DEVELOPER	2
3	SALES	1

DEPARTMENT

Consider the above 3 tables

```
SELECT EMPLOYEE.EMP_FIRSTNAME, EMPLOYEE.EMP_LASTNAME,
DEPARTMENT.DEPT_NAME
FROM EMPLOYEE
JOIN EMP_DEPT
ON EMPLOYEE.EMP_ID = EMP_DEPT.EMP_ID
JOIN DEPARTMENT
ON DEPARTMENT.EMP_ID = EMP_DEPT.EMP_ID
```

11. # SQL Commands:

DDL → Create, Alter, Drop truncate
DML → Insert, Update, delete
DCL → Grant Revoke
TCL → Commit, Rollback, Save point.

DDL:

- DDL stands for Data Defination Language
- DDL changes the structure of the table like creating, Altering, deleting a table.
- All DDL commands are autocommited ie automatically saved

Alter:

Alter table tablename Add (Column_name datatype);

Alter table tablename Modify (Column_new datatype);

Alter table table_name Rename column
old_column_name to new_column name

- ## DML:

DML stands for Data Manipulation Language
DML commands are not autocommitted
DML commands are used to modify the database

Update tablename Set Col1 where condition

DCL:

DCL stands for Data control Language
Grant
Revoke

CREATE USER Student Identified by Student;
(Student: user name; Student: Password)

GRANT ALL Privileges to student;
Revoke All Privileges from Student;

How to list users in Oracle:
SELECT * FROM all_users;

TCL:
- Transaction Control Language

Database Objects

Database objects are created to make query writing easy.
Anything we make from Create command is called Object.

- Table
- View → Represents subsets of data
- Sequence → Generates primary key value
- Index → Improves performance
- Synonim → Alternative name

View:

```
CREATE VIEW view_name AS
  Select col1, col2 ...
  From table_name
  where condition
```

View is nothing but a temporary table containing a part of data from main table.

Sequence:

Sequence is set of integer 1, 2, 3.... generated and supported by some database systems.

```
CREATE SEQUENCE Seq_name
START WITH initial_value
INCREMENT By value
MIN VALUE value
MAX VALUE value
CYCLE |NOCYCLE;
```

Indexes are **special lookup tables** that the database search engine can use to speed up data retrieval. Simply put, an index is a pointer to data in a table.

An index helps to speed up **select** queries and **where** clauses, but it slows down data input, with the **update** and the **insert** statements. Indexes can be created or dropped with no effect on the data.

Single-Column Indexes Syntax:

```
CREATE INDEX index_name ON table_name (column_name);
```

Multiple-Column Indexes Syntax:

```
CREATE INDEX index_name ON table_name (column1, column2);
```

Unique Indexes Syntax:

```
CREATE UNIQUE INDEX index_name ON table_name (column_name);
```