Laboratory Manual

On

DATABASE MANAGEMENT SYSTEM LAB

(For 4th Semester CSE/IT)

Prepared by:

Miss Barsha Subudhi Ray

Guest faculty (CSE/IT)

UCP Engineering School

Berhampur.

Pr.4-DATABASE MANAGEMENT SYSTEM LAB

BASIC CONCEPTS OF DBMS

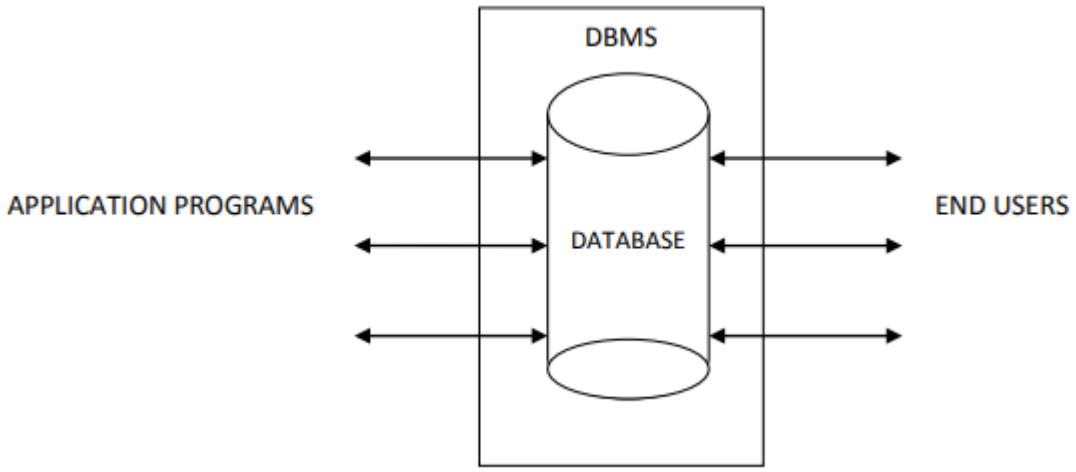
DATABASE:- A database is an integrated collection of related files along with details of the interpretation of the data contained in it. The collection of data usually referred to as the database which contains information related to an enterprise.

DBMS:-

- A database system is nothing more than a computer based record keeping system i.e. whose purpose is to record and maintain the data or information.
- A DBMS is a software system that allows to access data contained in a database.
- The objective of DBMS is to provide a convenient and effective method of defining, storing and retrieving information contained in the database.
- The DBMS interfaces with application programs so that the data contained in the database can be used by multiple applications and users.
- A database system involves four major components namely data, hardware, software and user.
- Data: - The fundamental unit of database is data. A database is therefore nothing but a repository for stored data. Every data must have two properties namely integrated and shared. Integrated means that the data can be uniquely identified in the database and shared means the data can be shared among several different users.
- Hardware: - The hardware consists of secondary storage volumes on which the database resides.
- Software: - Between the physical database and the users of the system there is a layer of software usually called as database management system (DBMS). All requests from users for access to the database are handled by DBMS.
- Users: - The users are the application programmers responsible for writing application programs that use the database. The application programmers operate on the data in all the usual ways, i.e. retrieving information, creating new information, deleting or changing existing information. The second classes of users are the end users whose task is to access the data of a database from a terminal. The end users use a query language to invoke user written application programs as per the commands from the terminal. The third classes of users are the database administrators or DBA who has control over the whole system.

DATADEFINITION LANGUAGE (DDL):-

- A database system provides a data definition language to specify the database schemas.
- For example the following SQL statement defines the account table.



Sql> create table account (accountno varchar2 (10), balance number); Execution of the above DDL statement creates the account table. In addition it updates a special set of tables called data dictionary.

- DDL is used to define the database. This definition includes all the entity sets and their associated attributes as well as the relationship among the entity sets. It also includes any constraints that have to be maintained.
- The DDL used at the external schema is called the view definition language (VDL) from where the defining process starts.
- A data dictionary contains metadata. A database system consults data dictionary before reading or modifying actual data. The schema of a table is an example of a metadata i.e. data about data.
- We specify the storage structure and access methods used by the database system by a set of statements in a special type of DDL called data storage and definition language (DSDL). These statements define the implementation details of the database schemas which are usually hidden from the users.
- The DDL provides facilities to specify such consistency constraints. The database system checks these constraints every time the database is updated**.**

DATA DICTIONARY:-

- Information regarding the structure and usage of data contained in the database, the metadata maintained in a data dictionary. The term system catalog also describes this metadata. The data dictionary which is a database itself documents the data.

- Each database users can consult the data dictionary to learn what each piece of data and various synonyms of data fields mean.
- In an integrated system (i.e. in system where the data dictionary is a part of the DBMS) the data dictionary stores information concerning the external, conceptual and internal levels of the database. It contains the source of each data field value, the frequency of its use and an audit trail concerning updates, including the who and when of each update.

Data Manipulation Language (DML) in SQL:-

- The **SQL commands** that deal with the **manipulation of data** present in the database belong to **DML** or Data Manipulation Language and this includes most of the **SQL statements.** It is the component of the SQL statement that controls access to data and to the database. Basically, DCL statements are grouped with DML statements.

Important SQL Commands :-

Common DML Commands

Command	Description	Syntax
INSERT	Insert data into a table	`INSERT INTO table_name (column1, column2, ...) VALUES (value1, value2, ...);`
UPDATE	Update existing data within a table	`UPDATE table_name SET column1 = value1, column2 = value2 WHERE condition;`
DELETE	Delete records from a database table	`DELETE FROM table_name WHERE condition;`
LOCK	Table control concurrency	`LOCK TABLE table_name IN lock_mode;`
CALL	Call a PL/SQL or JAVA subprogram	`CALL procedure_name(arguments);`
EXPLAIN PLAN	Describe the access path to data	`EXPLAIN PLAN FOR SELECT * FROM table_name;`

1. **SELECT**: Used to retrieve data from a database.
2. **INSERT**: Used to add new data to a database.
3. **UPDATE**: Used to modify existing data in a database.
4. **DELETE**: Used to remove data from a database.
5. **CREATE TABLE**: Used to create a new table in a database.
6. **ALTER TABLE**: Used to modify the structure of an existing table.
7. **DROP TABLE**: Used to delete an entire table from a database.
8. **WHERE**: Used to filter rows based on a specified condition.
9. **ORDER BY**: Used to sort the result set in ascending or descending order.
10. **JOIN**: Used to combine rows from two or more tables based on a related column between them

Commands

- ALTER TABLE

```
ALTER TABLE table_name
ADD column_name datatype;
```

`ALTER TABLE` lets you add columns to a table in a database.

- AND

```
SELECT column_name(s)
FROM table_name
WHERE column_1 = value_1
  AND column_2 = value_2;
```

`AND` is an operator that combines two conditions. Both conditions must be true for the row to be included in the result set.

- AS

```
SELECT column_name AS 'Alias'
FROM table_name;
```

`AS` is a keyword in SQL that allows you to rename a column or table using an *alias*.

- AVG()

```
SELECT AVG(column_name)
FROM table_name;
```

AVG() is an aggregate function that returns the average value for a numeric column.

- BETWEEN

```
SELECT column_name(s)
FROM table_name
WHERE column_name BETWEEN value_1 AND value_2;
```

The BETWEEN operator is used to filter the result set within a certain range. The values can be numbers, text or dates.

- CASE

```
SELECT column_name,
 CASE
  WHEN condition THEN 'Result_1'
  WHEN condition THEN 'Result_2'
  ELSE 'Result_3'
 END
FROM table_name;
```

CASE statements are used to create different outputs (usually in the SELECT statement). It is SQL's way of handling if-then logic.

- COUNT()

```
SELECT COUNT(column_name)
FROM table_name;
```

COUNT() is a function that takes the name of a column as an argument and counts the number of rows where the column is not NULL.

- CREATE TABLE

```
CREATE TABLE table_name (
 column_1 datatype,
```

```
 column_2 datatype,
 column_3 datatype
);
```

CREATE TABLE creates a new table in the database. It allows you to specify the name of the table and the name of each column in the table.

- DELETE

```
DELETE FROM table_name
WHERE some_column = some_value;
```

DELETE statements are used to remove rows from a table.

- GROUP BY

```
SELECT column_name, COUNT(*)
FROM table_name
GROUP BY column_name;
```

GROUP BY is a clause in SQL that is only used with aggregate functions. It is used in collaboration with the SELECT statement to arrange identical data into groups.

- HAVING

```
SELECT column_name, COUNT(*)
FROM table_name
GROUP BY column_name
HAVING COUNT(*) > value;
```

HAVING was added to SQL because the WHERE keyword could not be used with aggregate functions.

- INNER JOIN

```
SELECT column_name(s)
FROM table_1
JOIN table_2
  ON table_1.column_name = table_2.column_name;
```

An inner join will combine rows from different tables if the *join condition* is true.

- INSERT

```
INSERT INTO table_name (column_1, column_2, column_3)
VALUES (value_1, 'value_2', value_3);
```

`INSERT` statements are used to add a new row to a table.

- IS NULL / IS NOT NULL

```
SELECT column_name(s)
FROM table_name
WHERE column_name IS NULL;
```

`IS NULL` and `IS NOT NULL` are operators used with the `WHERE` clause to test for empty values.

- LIKE

```
SELECT column_name(s)
FROM table_name
WHERE column_name LIKE pattern;
```

`LIKE` is a special operator used with the `WHERE` clause to search for a specific pattern in a column.

- LIMIT

```
SELECT column_name(s)
FROM table_name
LIMIT number;
```

`LIMIT` is a clause that lets you specify the maximum number of rows the result set will have.

- MAX()

```
SELECT MAX(column_name)
FROM table_name;
```

MAX() is a function that takes the name of a column as an argument and returns the largest value in that column.

- MIN()

```
SELECT MIN(column_name)
FROM table_name;
```

MIN() is a function that takes the name of a column as an argument and returns the smallest value in that column.

- OR

```
SELECT column_name
FROM table_name
WHERE column_name = value_1
  OR column_name = value_2;
```

OR is an operator that filters the result set to only include rows where either condition is true.

- ORDER BY

```
SELECT column_name
FROM table_name
ORDER BY column_name ASC | DESC;
```

ORDER BY is a clause that indicates you want to sort the result set by a particular column either alphabetically or numerically.

- OUTER JOIN

```
SELECT column_name(s)
FROM table_1
LEFT JOIN table_2
  ON table_1.column_name = table_2.column_name;
```

An outer join will combine rows from different tables even if the join condition is not met. Every row in the *left* table is returned in the result set, and if the join condition is not met, then NULL values are used to fill in the columns from the *right* table.

- ROUND()

```
SELECT ROUND(column_name, integer)
FROM table_name;
```

ROUND() is a function that takes a column name and an integer as arguments. It rounds the values in the column to the number of decimal places specified by the integer.

- SELECT

```
SELECT column_name
FROM table_name;
```

SELECT statements are used to fetch data from a database. Every query will begin with SELECT.

- SELECT DISTINCT

```
SELECT DISTINCT column_name
FROM table_name;
```

SELECT DISTINCT specifies that the statement is going to be a query that returns unique values in the specified column(s).

- SUM

```
SELECT SUM(column_name)
FROM table_name;
```

SUM() is a function that takes the name of a column as an argument and returns the sum of all the values in that column.

- UPDATE

```
UPDATE table_name
SET some_column = some_value
WHERE some_column = some_value;
```

UPDATE statements allow you to edit rows in a table.

- WHERE

```
SELECT column_name(s)
FROM table_name
WHERE column_name operator value;
```

WHERE is a clause that indicates you want to filter the result set to include only rows where the following *condition* is true.

- WITH

```
WITH temporary_name AS (
  SELECT *
  FROM table_name)
SELECT *
FROM temporary_name
WHERE column_name operator value;
```

WITH clause lets you store the result of a query in a temporary table using an alias. You can also define multiple temporary tables using a comma and with one instance of the WITH keyword.

The WITH clause is also known as common table expression (CTE) and subquery factoring.

1.1 Create a database of a company.

CREATE DATABASE CompanyDB;

Output:

Query OK, 1 row affected (0.01 sec)

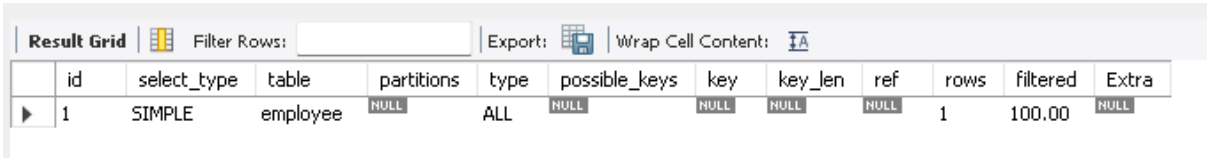
SHOW DATABASES;

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| book_shop          |
| employee_directory |
| geeksforgeeks      |
| ig_clone           |
| information_schema |
| inst_clone         |
| mysql              |
| performance_schema |
| sakila             |
```

1.2Create a query to create a table department and employee.

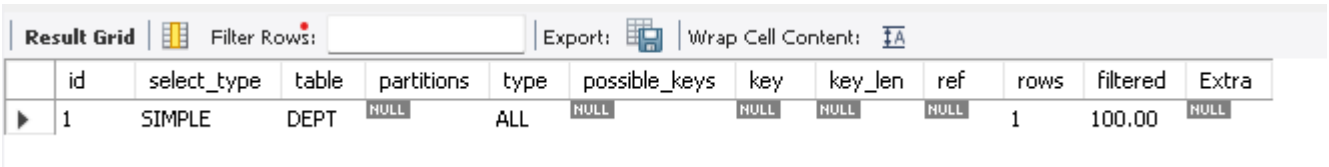
```
CREATE TABLE DEPT(
Dept_no int ,
Dept_Name varchar(20)
);


CREATE TABLE EMPLOYEE(
Emp_no int,
Name varchar (40),
titel varchar(30),
Job varchar (20),
Dept_code int,
Salary int,
Comm varchar(21),
Age int,
DOJ date,
LOC varchar(20)
);


desc table EMPLOYEE;
desc table DEPT;
```

Output:



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	employee	NULL	ALL	NULL	NULL	NULL	NULL	1	100.00	NULL



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	DEPT	NULL	ALL	NULL	NULL	NULL	NULL	1	100.00	NULL

1.3. Create a query to insert data into table department and employee.

```
INSERT INTO DEPT
(Dept_no,Dept_Name)
VALUES
(123423,"MECHANICLE"),
(123424,"CHEMICAL"),
(123425,"CSE"),
(123426,"ELETRICAL"),
(123427,"BIOTECH"),
(6,"ETC");

INSERT INTO EMPLOYEE
(Emp_no,Name,titel,Job,Dept_code,Salary,Comm,Age,DOJ,LOC)
VALUES
(100,"GOURAB ","SWAIN","cleark",123423,1000,"2%",17,'1981-02-23',"DALLAS"),
(101,"ADARSH ","HOTA","Manager",123424,1800,"5%",18,'1981-03-26',"DALLAS"),
(102,"RAKESH ","PATRA","Manager",123425,3200,"3%",19,'1981-05-10',"GOPALPUR"),
(7566,"RAM ","SINGH","peon",123426,500," ",20,'1981-10-04',"DALLAS"),
(104,"SATYJIT","SETHI","cleark",123427,4100," ",21,'1981-01-20',"ASKA");

SELECT * FROM EMPLOYEE;
SELECT * FROM DEPT;
```

Output:

Result Grid | Filter Rows

Dept_no	Dept_Name
123423	MECHANICLE
123424	CHEMICAL
123425	CSE
123426	ELETRICAL
123427	BIOTECH
6	ETC

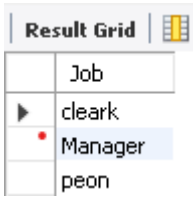
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Emp_no	Name	titel	Job	Dept_code	Salary	Comm	Age	DOJ	LOC
100	GOURAB	SWAIN	cleark	123423	1000	2%	17	1981-02-23	DALLAS
101	ADARSH	HOTA	Manager	123424	1800	5%	18	1981-03-26	DALLAS
102	RAKESH	PATRA	Manager	123425	3200	3%	19	1981-05-10	GOPALPUR
7566	RAM	SINGH	peon	123426	500		20	1981-10-04	DALLAS
104	SATYJIT	SETHI	cleark	123427	4100		21	1981-01-20	ASKA

1.4. Create a query to display unique jobs from the EMP table.

SELECT DISTINCT Job FROM employee;

Output:



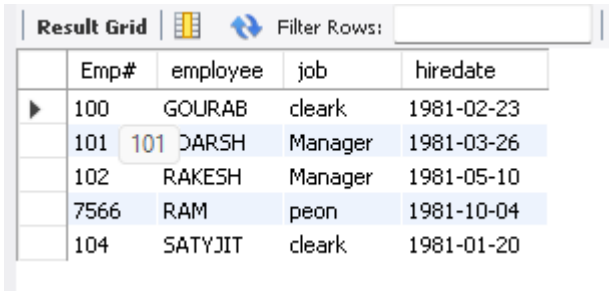
Result Grid

Job
cleark
Manager
peon

2. Write a query to Name the column headings EMP#, Employee, Job and Hire date, respectively. Run the query.

```
select Emp_no as"Emp#",
Name as "employee",
Job as "job",
DOJ as "hiredate"
from employee;
```

Output:



Result Grid Filter Rows:

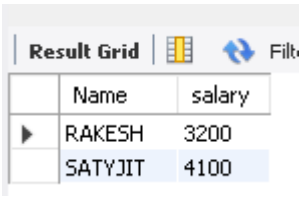
Emp#	employee	job	hiredate
100	GOURAB	cleark	1981-02-23
101	DARSH	Manager	1981-03-26
102	RAKESH	Manager	1981-05-10
7566	RAM	peon	1981-10-04
104	SATYJIT	cleark	1981-01-20

3. Create a query to display the Name and salary of employees earning more than Rs.2850.Save the query and run it.

select Name,salary from employee

where salary>2850;

Output:



Result Grid | Filt

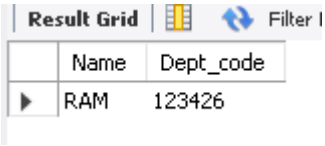
	Name	salary
▶	RAKESH	3200
	SATYJIT	4100

4. Create a query to display the employee name and department no. for employee no. 7566.

select Name,Dept_code from employee

where Emp_no=7566;

Output:



Result Grid | Filter

	Name	Dept_code
▶	RAM	123426

5. Display the employee name, job and start date of employees hire date between Feb.20.1981 and May 1, 1981. Order the query in ascending order of start date.

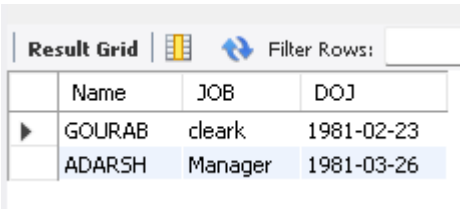
select Name,JOB,DOJ from employee

where

DOJ between "1981-02-20"and"1981-05-01"

order by DOJ asc;

Output:



Result Grid | Filter Rows:

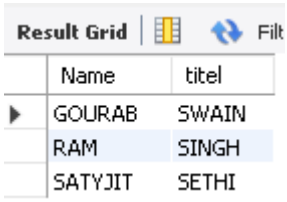
	Name	JOB	DOJ
▶	GOURAB	cleark	1981-02-23
	ADARSH	Manager	1981-03-26

6. Display the name and title of all employees who don't have a Manager.

select Name,titel from employee

where JOB!="manager";

Output:



Result Grid | Filt

Name	titel
GOURAB	SWAIN
RAM	SINGH
SATYJIT	SETHI

7. Display the name, salary and comm. For all employee who earn comm. Sort data in descending order of salary and comm.

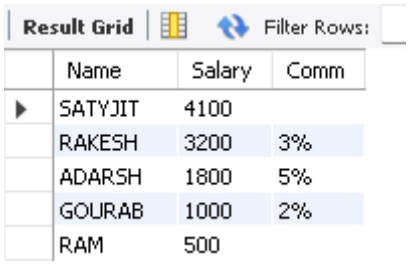
select Name,Salary,Comm from employee

order by salary desc;

select Name,Salary,Comm from employee

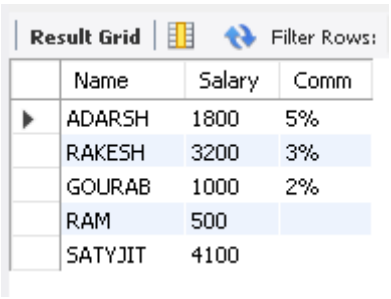
order by Comm desc;

Output:



Result Grid | Filter Rows:

Name	Salary	Comm
SATYJIT	4100	
RAKESH	3200	3%
ADARSH	1800	5%
GOURAB	1000	2%
RAM	500	



Result Grid | Filter Rows:

Name	Salary	Comm
ADARSH	1800	5%
RAKESH	3200	3%
GOURAB	1000	2%
RAM	500	
SATYJIT	4100	

8. Display the name job, salary for all employees whose job is Clerk or Analyst their salary is not equal to Rs.1000, Rs.3000, Rs.5000.

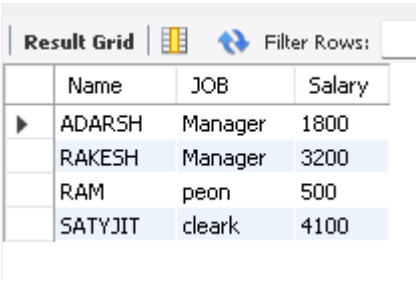
select Name,JOB,Salary from employee

where

JOB in("Clerk"or"Analyst")and

Salary not in(1000,3000,5000);

Output:



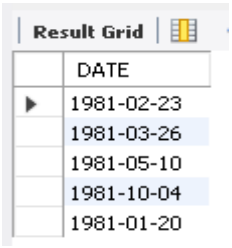
Result Grid | Filter Rows:

	Name	JOB	Salary
▶	ADARSH	Manager	1800
	RAKESH	Manager	3200
	RAM	peon	500
	SATYJIT	cleark	4100

9. Write a query to display the date. Label the column DATE.

select DOJ as DATE from employee;

Output:



Result Grid

	DATE
▶	1981-02-23
	1981-03-26
	1981-05-10
	1981-10-04
	1981-01-20

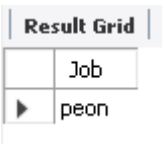
10. Create a unique listing of all jobs that are in department 30.

select distinct Job from EMPLOYEE

where

Dept_code=30;

Output:



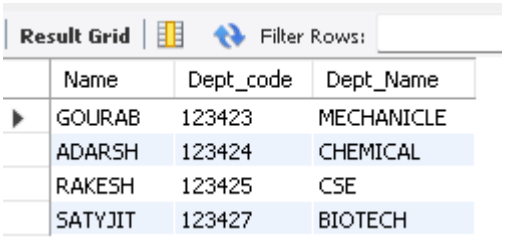
Result Grid

	Job
▶	peon

11. Write a query to display the name, department number and department name for all employees.

```
SELECT e.Name, e.Dept_code, d.Dept_Name

FROM EMPLOYEE e

JOIN DEPT d

ON e.Dept_code = d.Dept_no;
```

Output:



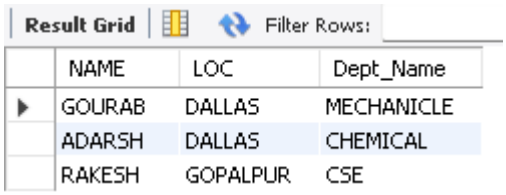
Result Grid | Filter Rows:

Name	Dept_code	Dept_Name
GOURAB	123423	MECHANICLE
ADARSH	123424	CHEMICAL
RAKESH	123425	CSE
SATYJIT	123427	BIOTECH

12. Write a query to display the employee name, department name, and location of all employee who earn a commission.

```
SELECT e.NAME,e.LOC,d.Dept_Name

from employee e

join DEPT d

on e.Dept_code = d.Dept_no

where

e.Comm>0;
```

Output:



Result Grid | Filter Rows:

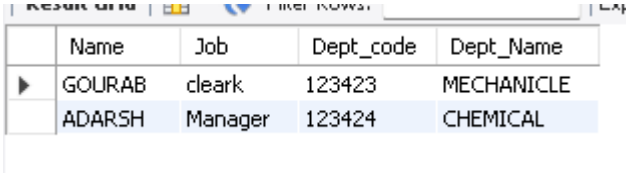
NAME	LOC	Dept_Name
GOURAB	DALLAS	MECHANICLE
ADARSH	DALLAS	CHEMICAL
RAKESH	GOPALPUR	CSE

13. Write a query to display the name, job, department number and department name for all employees who works in DALLAS.

```
select e.Name,e.Job,e.Dept_code,d.Dept_Name

from employee e
```

```
join DEPT d
on e.Dept_code = d.Dept_no
where
e.LOC='DALLAS';
```

Output:

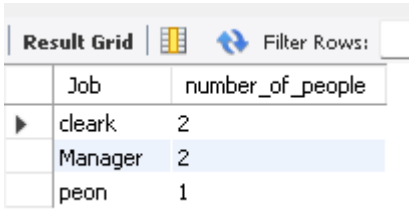


Name	Job	Dept_code	Dept_Name
GOURAB	cleark	123423	MECHANICLE
ADARSH	Manager	123424	CHEMICAL

14. Write a query to display the number of people with the same job. Save the query @ run it.

```
SELECT Job, COUNT(*) AS number_of_people
FROM employee
GROUP BY Job;
```

Output:



Result Grid | Filter Rows:

Job	number_of_people
cleark	2
Manager	2
peon	1

15. Create a query to display the employee name and hire date for all employees in same department.

```
SELECT e1.Name AS Employee_Name, e1.DOJ AS Hire_Date, d.Dept_Name AS Department_Name
FROM EMPLOYEE e1
JOIN DEPT d ON e1.Dept_code = d.Dept_no;
```

Output:

Employee_Name	Hire_Date	Department_Name
GOURAB	1981-02-23	MECHANICLE
ADARSH	1981-03-26	CHEMICAL
RAKESH	1981-05-10	CSE
SATYJIT	1981-01-20	BIOTECH

16. Display the employee name and salary of all employees who report to KING.

```
SELECT e.Employee_Name, e.Salary
FROM Employee e
JOIN Employee m ON e.Manager_ID = m.Employee_ID
WHERE m.Employee_Name = 'KING';
```

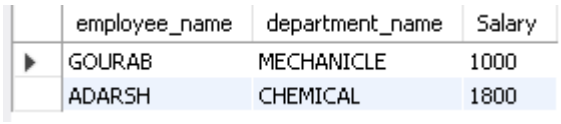
Employee_Name	Salary
GOURAB	1000
ADARSH	1800

17. Display the mane, department name and salary of any employee whose salary and commission matches both the salary and commission of any employee located in DALLAS.

```
SELECT e.Name AS employee_name, d.Dept_Name AS department_name, e.Salary
FROM EMPLOYEE e
JOIN DEPT d ON e.Dept_code = d.Dept_no
WHERE e.Salary IN (
   SELECT Salary
   FROM EMPLOYEE
   WHERE LOC = 'DALLAS'
)
AND e.Comm IN (
   SELECT Comm
   FROM EMPLOYEE
```

```
   WHERE LOC = 'DALLAS'
)
```

Output:



	employee_name	department_name	Salary
▶	GOURAB	MECHANICLE	1000
	ADARSH	CHEMICAL	1800

18. . Create a student database table using create command using Regd. No as Primary Key , insert data of your class.

```
CREATE TABLE STUDENTCSE(
SL_NO INT,
REBD_NO INT PRIMARY KEY,
Name  VARCHAR(20),
Roll_no INT ,
Address VARCHAR(30)
);

INSERT INTO STUDENTCSE(SL_NO,REBD_NO,Name,Roll_no,Address)
VALUES
(1,7001,"ADERSH HOTA",01,"BERHAMPUR"),
(2,7002,"ADITYA PANDA",02,"BERHAMPUR"),
(3,7003,"GOURAB SWAIN",15,"BBSR"),
(4,7004,"RAKESH PATRA",04,"BAM"),
(5,7005,"SANJU MISHRA",05,"BAKESWAR");
```

SELECT * FROM STUDENTCSE;

Output:

SL_NO	REBD_NO	Name	Roll_no	Address
1	7001	ADERSH HOTA	1	BERHAMPUR
2	7002	ADITYA PANDA	2	BERHAMPUR
3	7003	GOURAB SWAIN	3	BBSR
4	7004	RAKESH PATRA	4	BAM
5	7005	SANJU MISHRA	5	BAKESWAR
NULL	NULL	NULL	NULL	NULL

19. Delete the information of student having roll No -15 and City- Bhubaneswar. Rename the Student database table to STUDENT INFORMATION.

DELETE FROM STUDENTCSE

WHERE Roll_no = 15 AND Address = 'BBSR';

ALTER TABLE STUDENTCSE

RENAME TO STUDENT__INFORMATION; select * from STUDENT__INFORMATION;

Output:

SL_NO	REBD_NO	Name	Roll_no	Address
1	7001	ADERSH HOTA	1	BERHAMPUR
2	7002	ADITYA PANDA	2	BERHAMPUR
4	7004	RAKESH PATRA	4	BAM
5	7005	SANJU MISHRA	5	BAKESWAR
NULL	NULL	NULL	NULL	NULL

20. Practice of all Data Retrieval, DML, DDL, TCL and DCL commands used in Oracle by writing queries.

Create Table & Insert Data (DDL & DML)

CopyEdit

CREATE TABLE employees (

  emp_id NUMBER PRIMARY KEY,

  name VARCHAR2(50),

  salary NUMBER(10,2)

);

INSERT INTO employees VALUES (101, 'John Doe', 60000);

COMMIT;

2. **Retrieve Data (SELECT - Data Retrieval)**

CopyEdit

SELECT * FROM employees;

SELECT name, salary FROM employees WHERE salary > 50000;

3.**Modify & Delete Data (DML & TCL)**

CopyEdit

UPDATE employees SET salary = salary + 5000 WHERE emp_id = 101;

DELETE FROM employees WHERE emp_id = 101;

ROLLBACK;

COMMIT;

4.**Manage Permissions (DCL)**

CopyEdit

GRANT SELECT ON employees TO user1;

REVOKE SELECT ON employees FROM user1;