GR22 Regulations

II B.Tech I Semester

Databases Lab

(GR22A2096)

B.Tech-Computer Science and Business System

GOKARAJU RANGARAJU
INSTITUTE OF ENGINEERING AND TECHNOLOGY
(Autonomous)

GOKARAJU RANGARAJU INSTITUTE OF ENGINEERING & TECHNOLOGY

DATABASES LAB

Course Code : GR22A2096 **L/T/P/C:0/0/2/1**

II Year I Semester

Course Objectives:

1. Develop the logical design of the database using data modeling concepts such as Relational model
2. Infer the commands for retrieving the data.
3. Create a relational database using SQLite
4. Manipulate the data in the tables using SQL.
5. Render the Procedural concepts with SQL

Course Outcomes: After the completion of the course, the student will be able to

1. Construct the schema of the database and modify it.
2. Compile a query to obtain the aggregated result from the database.
3. Speculate the concepts of database objects.
4. Compare the use of procedure and function in database.
5. Use SQLite to connect to database from C programs.

LIST OF EXPERIMENTS:

Task-1 (DDL and DML Commands):

a) Practice queries on DDL Commands
b) Practice queries on DML Commands

Task-2 (SQL Functions):

a) Practice queries using basic SQL operators.
b) Practice queries on between. And, like and not operators.
c) Use various built in SQL Functions and practice queries

Task-3 (Aggregate Operators):

a) Perform aggregate operations and generate queries using them.
b) Implement the group by and having clauses with aggregate operators.

Task-4 (Nested Queries):

a) Write queries to illustrate the use of pair wise sub queries.

b) Practice the single row and multiple row sub queries.

c) Use sub queries in Create, Insert, Update and delete commands

Task-5 (Joins and Set Operators):

a) Practice queries on various kinds of joins.
b) Practice queries on set operators.

Task-6 (Views):

a) Create a simple view and try modifications through it.
b) Create a complex view and understand the restrictions for modifications through it.
c) Practice the creation of sequence and synonym.

Task-7(Indexes, Sequences and Synonyms):

a) Practice the creation of sequence and synonym.
b) Practice creation of function-based indexes.
c) Create an index on attribute of a table.

Task-8 (DCL Commands):

a) Practice grant and revoke of user level privileges.
b) Practice object-level privileges and creation of roles.

Task-9 (PL/SQL Blocks, Named Blocks):

a) Write programs to use the anonymous blocks.
b) Develop PL/SQL named blocks-Procedures, Functions.

Task-10(Cursor and Trigger):

a) Write a PL/SQL program to illustrate the purpose of cursors.
b) Write a PL/SQL program to exemplify the concept of triggers.

Task-11(C Implementation for DB):

a) Write a C program to connect to SQLite Database and perform DDL and DML operations in it.
b) Write a C program to perform all kinds of retrieval operations on SQLite database.

Task-12(Case Study):

a) Download standard data of reasonable size (Unit level data of various rounds of NSS surveys) form internet and implement various SQL commands.

Text Books:

1. Database System Concepts. Abraham Silberschatz, Henry F. Korth and S. Sudarshan.

Reference Books:

1. Principles of Database and Knowledge – Base Systems, Vol 1 by J. D. Ullman.
2. Fundamentals of Database Systems. R. Elmasri and S. Navathe.
3. Foundations of Databases. Serge Abiteboul, Richard Hull, Victor Vianu.

Index

	c. Display the month name of date "14-jul-09" in full. d. Display the Date of joining of all employees in the format"dd-mm-yy". e. Display the date two months after the Date of joining of employees. f. Display the last date of that month in"05-Oct-09". g. Display the rounded date in the year format, month format, day format in the employee h. Display the commissions earned by employees. If they do not earn commission, display it as "No Commission". **DML COMMANDS (Insert, Update, Delete)** a. Insert 5 records into dept Insert few rows and truncate those from the emp1 table and also drop it. b. Insert 11 records into emptable. c. Update the emptable to set the value of commission of all employees to Rs1000/- who are working as managers. d. Delete only those who are working as supervisors. e. Delete the rows whose empno is 7599.	
3	**SQL Aggregate Functions, Group by clause, Having clause** a. Count the total records in the emptable. b. Calculate the total and average salary of the employee. c.Determine the max and min salary and rename the column as max_salary and min_salary. d. Find number of departments in employee table. e. Display job wise sum, average, max, min salaries. f. Display maximum salaries of all the departments having maximum salary >2000 g. Display job wise sum, avg, max, min salaries in department 10 having average salary is greater than 1000 and the result is ordered by sum of salary in descending order. **TCL COMMANDS (Save Point, Rollback Commit)**	15
4	**Nested Queries** a. Find the third highest salary of an employee. b. Display all employee names and salary whose salary is greater than minimum salary of the company and job title starts with'M'. c. Write a query to display information about employees who earn more than any employee in dept30. d. Display the employees who have the same job as Jones and whose salary is greater than or equal to the salary of Ford. e. List out the employee names who get the salary greater than the maximum salaries of dept with deptno20,30. f. Display the maximum salaries of the departments whose maximum salary is greater than9000. g. Create a table employee with the same structure as the table emp and insert rows into the table using select clause.	17

<table>
<tr><td></td><td>h. Create a manager table from the emptable which should hold details only about the managers.

DQL COMMAND (Select)- SQL Operators and Order by Clause
a. List the records in the emptable order by salary in descending order.
b. Display only those employees whose deptnois30.
c. Display deptno from the table employee avoiding the duplicated values.
d. List all employee names, salary and 15% rise in salary. Label the column as pay hike.
e. Display the rows whose salary ranges from 15000 to30000.
f. Display all the employees in dept 10 and 20 in alphabetical order of names.
g. List the employee names who do not earn commission.
h. Display all the details of the records with 5-character names with ‘S’ as starting character.
i. Display joining date of all employees in the year of1998.
j. List out the employee names whose salary is greater than 5000 and less than6000.</td><td></td></tr>
<tr><td>5</td><td>Joins, Set Operators
a. Display all the employees and the departments implementing a left outer join.
b. Display the employee name and department name in which they are working implementing a full outer join.
c. Write a query to display their employee names and their managers’ name and salary for every employee.
d. Write a query to output the name, job, empno, deptname and location for each dept, even if there are no employees.
e. Display the details of those who draw the same salary.</td><td>20</td></tr>
<tr><td>6</td><td>Views
a. Create a view that displays the employee id, name and salary of employees who belong to 10thdepartment.
b. Create a view with read only option that displays the employee name and their department name.
c. Display all the views generated.
d. Execute the DML commands on views created and drop them</td><td>22</td></tr>
<tr><td>7</td><td>Sequence, Indexes and Synonyms</td><td>24</td></tr>
<tr><td>8</td><td>Practice on DCL Commands</td><td>28</td></tr>
<tr><td>9</td><td>PL/SQL Blocks, Named Blocks
anonymous blocks, named blocks, Procedures and Functions</td><td>30</td></tr>
<tr><td>10</td><td>a. Write a trigger on employee table that shows the old and new values of employee name after updating on employee name.
b. Write a PL/SQL procedure for inserting, deleting and updating the employee</td><td>33</td></tr>
</table>

	table. c. Write a PL/SQL function that accepts the department number and returns the total salary of that department.	
11	**C Implementation for DB** a. Write a PL/SQL code to retrieve the employee name, join date and designation of an employee whose number is given as input by the user. b. Write a PL/SQL code to calculate tax of employee. c. Write a PL/SQL program to display top ten employee details based on salary using cursors. d. Write a PL/SQL program to update the commission values for all the employees' with salary less than 2000, by adding 1000 to the existing values.	35
12	Download standard data of reasonable size (Unit level data of various rounds of NSS surveys) form internet and implement various SQL commands.	42

TASK 1 (DDL and DML Commands)

Aim : To use DDL Commands to work on the schema of a database and Practice queries on DML Commands.

i) Create a table EMP with the following structure.

Name	Type
EMPNO	NUMBER(6)
ENAME	VARCHAR2(20)
JOB	VARCHAR2(10)
MGR	NUMBER(4)
DEPTNO	NUMBER(3)
SAL	NUMBER(7,2)

Query:

SQL>create table emp(empno number(6), ename varchar2(20), jobvarchar2(10), mgr number(4), deptno number(3), sal number(7,2));

Table created.

Output:

```
SQL> desc emp;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------

 EMPNO                                              NUMBER(6)
 ENAME                                              VARCHAR2(20)
 JOB                                                VARCHAR2(10)
 MGR                                                NUMBER(4)
 DEPTNO                                             NUMBER(3)
 SAL                                                NUMBER(7,2)
```

Add a column commission to the EMP table. Commission should be numeric with null values allowed.

Query:

SQL>Alter table emp add(commission number(4));

Output:

Table altered.

```
SQL> desc emp
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------

 EMPNO                                              NUMBER(6)
 ENAME                                              VARCHAR2(20)
 JOB                                                VARCHAR2(10)
 MGR                                                NUMBER(4)
 DEPTNO                                             NUMBER(3)
 SAL                                                NUMBER(7,2)
 COMMISSION                                         NUMBER(6)
```

ii) Modify the column width of the job field of emp table.

Query:

SQL>Alter table emp modify(job varchar2(15));

Output:

Table altered.

```
SQL> desc emp
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------

 EMPNO                                              NUMBER(6)
 ENAME                                              VARCHAR2(20)
 JOB                                                VARCHAR2(15)
 MGR                                                NUMBER(4)
 DEPTNO                                             NUMBER(3)
 SAL                                                NUMBER(7,2)
 COMMISSION                                         NUMBER(6)
```

iii) Create dept table with the following structure.

Name	Type
DEPTNO	NUMBER(2)
DNAME	VARCHAR2(10)
LOC	VARCHAR2(10)

Query:

SQL>create table dept(deptno number(2), dname varchar2(10), loc varchar2(10));

Output:

Table created.

```
SQL> desc dept
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------

 DEPTNO                                             NUMBER(2)
 DNAME                                              VARCHAR2(10)
 LOC                                                VARCHAR2(10)
```

iv) Add constraint to the emp table that is empno as primary key and deptno as foreign key.

Query:

SQL>alter table emp add constraint emp_id_pk primary key(empno);

SQL>alter table dept add constraint pk primary key(deptno);

Output:

Table altered

```
SQL> alter table dept add constraint pk primary key(deptno);

Table altered.
```

SQL>Alter table emp add constraint emp_deptno_fk foreign key(deptno) references dept(deptno);

```
SQL> alter table emp add constraint emp_id_pk primary key(empno);

Table altered.

SQL> alter table emp add constraint emp_deptno foreign key(deptno) references de
pt(deptno);

Table altered.
```

v). Add constraints to the emp table to check the empno value while entering i.eempno>100.

Query:

SQL>alter table emp add check (empno>100);

Output:

```
SQL> alter table emp add check(empno>100);
Table altered.
```

vi) Salary value by default is 5000, otherwise it should accept the values from the user.

Query:

SQL>alter table emp modify sal default 5000;

```
SQL> alter table emp modify sal default 5000;
Table altered.
```

vii) Add column DOB to the emp table Add and drop a column DOJ to the emp table.

Query:

SQL>alter table emp add(dob date);

SQL>alter table emp add(doj date);

SQL>alter table emp drop(doj);

Output:

```
SQL> alter table emp add(dob date);
Table altered.
SQL> alter table emp add(doj date);
Table altered.
SQL> alter table emp drop(doj);
Table altered.
```

b) Practice queries on DML Commands

DML COMMANDS (Insert, Update, Delete)

a. Insert 5 records into dept table.Insert few rows and truncate those from emp1 table and also drop it. .

Query:

SQL>Insert into dept values(&deptno,'&dname','&loc');

SQL>create table emp1 as select * from emp;

SQL>insert into emp1 values(7000, 'King', 'Pres', 10, 20,10000,500, '12-Jan-92');

SQL>insert into emp1 values(7010, 'Jack', 'VP', 10, 30, 9000, 300, '19-Jul-92');

SQL>Truncate table emp1;

SQL>Drop table emp1;

Output:

```
SQL> create table emp1 as select * from emp;

Table created.

SQL> insert into emp1 values(7000,'King','Pres',10,20,10000,500,'12-Jan-92');

1 row created.

SQL> insert into emp1 values(7010,'Jack','VP',10,30,9000,300,'19-Jul-92');

1 row created.

SQL> truncate table emp1;

Table truncated.

SQL> drop table emp1;

Table dropped.

SQL>
```

b. Insert 11 records into the emp table.

Query:

SQL>insert into empvalues(&no, '&name', '&job', &mgr, &deptno, &sal, &comm, '&dob');

Note: Repeat execution of this statement for 11 times for 11 record insertions

Output:

```
SQL> select * from emp;

     EMPNO ENAME                JOB                MGR     DEPTNO        SAL
---------- -------------------- --------------- ---------- ---------- ----------
COMMISSION DOB
---------- ---------
      7000 King                 President            7500         20      10000
       500 12-JAN-88

      7200 Whalen               Supervisor           7580         10       8000
       200 04-FEB-91

      7500 OConnell             Manager                           30       9000
      1000 07-JUL-89

     EMPNO ENAME                JOB                MGR     DEPTNO        SAL
---------- -------------------- --------------- ---------- ---------- ----------
COMMISSION DOB
---------- ---------
      7580 Jane                 SWManager                         10       8000
      1000 09-DEC-91

      7599 Mary                 Advisor              7500         33       9000
       300 13-FEB-89

      7600 Birch                Clerk                7800         20       6000
           26-JAN-94

     EMPNO ENAME                JOB                MGR     DEPTNO        SAL
---------- -------------------- --------------- ---------- ---------- ----------
COMMISSION DOB
---------- ---------
      7650 SPaul                GM                   7580         10      10000
       300 19-SEP-89

      7680 Kochhar              AsstHead             7850         10      10000
           15-AUG-92

      7850 Hartstein            Manager                           20       5000
      1000 13-AUG-90

     EMPNO ENAME                JOB                MGR     DEPTNO        SAL
---------- -------------------- --------------- ---------- ---------- ----------
COMMISSION DOB
---------- ---------
      7700 Russell              Clerk                7800         10       9000
       300 29-JAN-93

      7800 Grant                ExeManager                        33       9000
      1000 18-NOV-91

11 rows selected.
```

c. Update the emp table to set the default commission of all employees to Rs.1000 /- who are working as managers.

Query:

SQL>update emp set commission=1000 where job like '%Manager%';

Output:

```
SQL> update emp set commission=1000 where job like '%Manager%';

4 rows updated.
```

d. Delete only those who are working as Supervisors.

Query:

SQL>delete from employee where job like '%Supervisor';

Output:

```
SQL> delete from employee where job like '%Supervisor';
1 row deleted.
```

e. Delete the rows whose empno is 759**9.**

Query:

SQL>delete from employee where empno=7599;

Output:

```
SQL> delete from employee where empno=7599;
1 row deleted.
```

TASK 2 (SQL Functions):

OPERATORS USED IN SQL

Arithmetic operator	Relational Operators	Logical Operators	SQL Special operators
+	=	**AND**	**BETWEEN**..- checks between two values **IN** - checks to match with any of a list of values
-	<	**OR**	**LIKE** - checks for pattern matching
*	>	**NOT**	**IS** - checks for nulls. **ALL** - checks for all values in the list
/	**!= or <>**		**ANY** - checks for any of the value in the list
	>=		**EXISTS** - checks if a sub query returns at least one row
	<=		**ROLLUP** - calculate multiple levels of subtotals
			CUBE - fraction of possible subtotals

Aim: Write Commands on SQL Operators

a. List the records in the emp table order by salary in descending order.

Query:

SQL>select * from emp order by sal desc;

Output:

```
SQL> select * from emp order by sal desc;

     EMPNO ENAME                JOB              MGR     DEPTNO        SAL
---------- -------------------- --------------- ---------- ---------- ----------
COMMISSION DOB
---------- ---------
      7000 King                 President             7500         20      10000
       500 12-JAN-88

      7680 Kochhar              AsstHead              7850         10      10000
            15-AUG-92

      7650 SPaul                GM                    7580         10      10000
       300 19-SEP-89


     EMPNO ENAME                JOB              MGR     DEPTNO        SAL
---------- -------------------- --------------- ---------- ---------- ----------
COMMISSION DOB
---------- ---------
      7800 Grant                ExeManager                         33       9000
      1000 18-NOV-91

      7700 Russell              Clerk                 7800         10      9000
       300 29-JAN-93

      7500 OConnell             Manager                            30      9000
      1000 07-JUL-89


     EMPNO ENAME                JOB              MGR     DEPTNO        SAL
---------- -------------------- --------------- ---------- ---------- ----------
COMMISSION DOB
---------- ---------
      7599 Mary                 Advisor               7500         33      9000
       300 13-FEB-89

      7200 Whalen               Supervisor            7580         10      8000
       200 04-FEB-91

      7580 Jane                 SWManager                          10      8000
      1000 09-DEC-91


     EMPNO ENAME                JOB              MGR     DEPTNO        SAL
---------- -------------------- --------------- ---------- ---------- ----------
COMMISSION DOB
---------- ---------
      7600 Birch                Clerk                 7800         20      6000
            26-JAN-94

      7850 Hartstein            Manager                            20      5000
      1000 13-AUG-90


11 rows selected.
```

b. Display only those employees whose deptno is 30.

Query:

SQL>select * from emp where deptno=30;

Output:

```
SQL> select * from emp where deptno=30;

     EMPNO ENAME                JOB              MGR     DEPTNO        SAL
---------- -------------------- --------------- ---------- ---------- ----------
COMMISSION DOB
---------- ---------
      7500 OConnell             Manager                            30      9000
      1000 07-JUL-89
```

c. Display deptno from the table employee avoiding the duplicate values.

Query:

SQL>select distinct deptno from emp;

Output:

```
SQL> select distinct deptno from emp;

    DEPTNO
----------
        30
        20
        33
        10
```

d. List all employee names, salary and 15% rise in salary.Label the column as New Sal.

Query:

SQL>select ename, sal, (sal*1.15) "New Sal" from emp;

Output:

```
SQL> select ename, sal, sal*1.15 "New Sal" from emp;

ENAME                     SAL     New Sal
-------------------- ---------- ----------
King                      10000      11500
Whalen                     8000       9200
OConnell                   9000      10350
Jane                       8000       9200
Mary                       9000      10350
Birch                      6000       6900
SPaul                     10000      11500
Kochhar                   10000      11500
Hartstein                  5000       5750
Russell                    9000      10350
Grant                      9000      10350
```

e. Display the rows whose empno ranges from 7500 to 7600.

Query:

SQL>select empno, ename, sal from emp where empno between 7500 and 7600;

Output:

```
SQL> select empno, ename, sal from emp where empno between 7500 and 7600;

     EMPNO ENAME                     SAL
---------- -------------------- ----------
      7500 OConnell                  9000
      7580 Jane                      8000
      7599 Mary                      9000
      7600 Birch                     6000
```

f. Display all the employees in dept 10 and 20 in alphabetical order of names.

Query:

SQL>select empno, ename, deptno from emp where deptnoin(10,20) group by ename;

Output:

```
SQL> select empno, ename, deptno from emp where deptno in (10,20) order by ename;

     EMPNO ENAME                  DEPTNO
---------- -------------------- ----------
      7600 Birch                        20
      7850 Hartstein                    20
      7580 Jane                         10
      7000 King                         20
      7680 Kochhar                      10
      7700 Russell                      10
      7650 SPaul                        10
      7200 Whalen                       10
```

g. List the employe names who do not earn commission.

Query:

SQL>select empno, ename, sal from emp where commission is null;

Output:

```
SQL> select ename from emp where commission is null;

ENAME
--------------------
Birch
Kochhar
```

h. Display all the details of the records with 5 character names with ‘S’ as starting character**.**

Query:

SQL>select * from employees where lengty(last_name)=5 and last_name like ‘s%’;

Output:

```
SQL> select * from employees where length(last_name)=5 and last_name like 'S%';

EMPLOYEE_ID FIRST_NAME           LAST_NAME
----------- -------------------- -------------------------
EMAIL                     PHONE_NUMBER         HIRE_DATE JOB_ID         SALARY
------------------------- -------------------- --------- ---------- ----------
COMMISSION_PCT MANAGER_ID DEPARTMENT_ID
-------------- ---------- -------------
        159 Lindsey              Smith
LSMITH                    011.44.1345.729268   10-MAR-97 SA_REP           8000
            .3        146             80

        171 William              Smith
WSMITH                    011.44.1343.629268   23-FEB-99 SA_REP           7400
           .15        148             80

EMPLOYEE_ID FIRST_NAME           LAST_NAME
----------- -------------------- -------------------------
EMAIL                     PHONE_NUMBER         HIRE_DATE JOB_ID         SALARY
------------------------- -------------------- --------- ---------- ----------
COMMISSION_PCT MANAGER_ID DEPARTMENT_ID
-------------- ---------- -------------

        157 Patrick              Sully
PSULLY                    011.44.1345.929268   04-MAR-96 SA_REP           9500
           .35        146             80
```

i.Display joining date of all employees in the year of 1998.

Query:

SQL>select employee_id ,hire_date from employees where hire_date between ‘1-jan-1998’ and ’31-dec-1998.

Output:

```
SQL> select sysdate from dual
  2  ;

SYSDATE
---------
10-FEB-19

SQL> select employee_id , hire_date from employees where hire_date between '1-jan-1998' and '31-dec-1998';

EMPLOYEE_ID HIRE_DATE
----------- ---------
        106 05-FEB-98
        112 07-MAR-98
        118 15-NOV-98
        126 28-SEP-98
        134 26-AUG-98
        139 12-FEB-98
        140 06-APR-98
        143 15-MAR-98
        144 09-JUL-98
        153 30-MAR-98
        154 09-DEC-98

EMPLOYEE_ID HIRE_DATE
----------- ---------
        161 03-NOV-98
        169 23-MAR-98
        170 24-JAN-98
        176 24-MAR-98
        177 23-APR-98
        180 24-JAN-98
        181 23-FEB-98
        186 24-JUN-98
        190 11-JUL-98
        194 01-JUL-98
        196 24-APR-98

EMPLOYEE_ID HIRE_DATE
----------- ---------
        197 23-MAY-98

23 rows selected.
```

j. List out the employee names whose salary is greater than 5000and greater than6000.

Query:

SQL>select ename from emp where sal>5000 and sal>6000;

Output:

```
SQL> select ename from emp where sal>5000 and sal>6000;

ENAME
--------------------
King
Whalen
OConnell
Jane
Mary
SPaul
Kochhar
Russell
Grant
```

Aim: Write commands on SQL Functions

a. Display the employee name concatenated with empno.

Query:

SQL>select concat(empno, concat(' ', ename)) from emp;

Output:

```
SQL> select concat(empno, concat(' ', ename)) from emp;

CONCAT(EMPNO,CONCAT('',ENAME))
--------------------------------------------------------------------------------
7000 King
7200 Whalen
7500 OConnell
7580 Jane
```

b. Display half of employee name in upper case and half in lower case.

Query:

SQL>Selectupper(substr(ename,0,length(ename)/2))||lower(substr(ename,(length(ename)/2)+1,length(ename))) “Name” from emp;

Output:

```
SQL> select upper(substr(ename,0,length(ename)/2))||lower(substr(ename,(length(ename)/2)+1,le
ngth(ename))) "Name" from emp;

Name
--------------------------------------------------------------------------------
KIng
WHAlen
OCONnell
JAne
MAry
BIrch
SPaul
KOChhar
HARTstein
RUSsell
GRant
```

c. Display the month name of date “14-jul-09” in full.

Query:

SQL>select to_char(to_date('14-jul-09'),'MONTH') “Month” from dual;

Output:

```
SQL> select to_char(to_date('14-jul-09'),'MONTH')  "Month" from dual;

Month
---------
JULY
```

d. Display the DOB of all employees in the format 'dd-mm-yy'.

Query:

SQL> select to_char(dob,'dd-mm-yy') from emp;

Output:

```
SQL> select to_char(dob,'dd-mm-yy') from emp;

TO_CHAR(
--------
12-01-88
04-02-91
07-07-89
09-12-91
13-02-89
26-01-94
19-09-89
15-08-92
13-08-90
29-01-93
18-11-91
```

e. Display the date two months after the DOB of employees.

Query:

SQL> select add_months(dob,2) from emp;

Output:

```
SQL> select add_months(dob,2) from emp;

ADD_MONTH
---------
12-MAR-88
04-APR-91
07-SEP-89
09-FEB-92
13-APR-89
26-MAR-94
19-NOV-89
15-OCT-92
13-OCT-90
29-MAR-93
18-JAN-92
```

f. Display the last date of that month in "05-Oct-09".

Query:

SQL>select last_day(to_date('05-oct-09')) "Last" from dual;

Output:

```
SQL> select last_day(to_date('05-oct-09')) "Last" from dual;

Last
---------
31-OCT-09
```

7. Display the rounded date in the year format, month format, day format in the employee.

Query:

SQL> select round(dob, 'dd'), round(dob, 'month'), round(dob, 'year') from emp;

Output:

```
SQL> select round(dob, 'dd'), round(dob, 'month'), round(dob, 'year') from emp;

ROUND(DOB ROUND(DOB ROUND(DOB
--------- --------- ---------
12-JAN-88 01-JAN-88 01-JAN-88
04-FEB-91 01-FEB-91 01-JAN-91
07-JUL-89 01-JUL-89 01-JAN-90
09-DEC-91 01-DEC-91 01-JAN-92
13-FEB-89 01-FEB-89 01-JAN-89
26-JAN-94 01-FEB-94 01-JAN-94
19-SEP-89 01-OCT-89 01-JAN-90
15-AUG-92 01-AUG-92 01-JAN-93
13-AUG-90 01-AUG-90 01-JAN-91
29-JAN-93 01-FEB-93 01-JAN-93
18-NOV-91 01-DEC-91 01-JAN-92
```

8. Display the commissions earned by employees. If they do not earn commission, display it as "No Commission".

Query:

SQL>selectemployee_id,last_name,nvl(to_char(commission_pct),'NoCommission')"commission" from employees;

Output:

```
SQL> select employee_id,last_name ,nvl(to_char(commission_pct),'No Commission')"commission" from employees;

EMPLOYEE_ID LAST_NAME                 commission
----------- ------------------------- ---------------------------------------
        100 King                      No Commission
        101 Kochhar                   No Commission
        102 De Haan                   No Commission
        103 Hunold                    No Commission
        104 Ernst                     No Commission
        105 Austin                    No Commission
        106 Pataballa                 No Commission
        107 Lorentz                   No Commission
        108 Greenberg                 No Commission
        109 Faviet                    No Commission
        110 Chen                      No Commission

EMPLOYEE_ID LAST_NAME                 commission
----------- ------------------------- ---------------------------------------
        111 Sciarra                   No Commission
        112 Urman                     No Commission
        113 Popp                      No Commission
        114 Raphaely                  No Commission
        115 Khoo                      No Commission
        116 Baida                     No Commission
        117 Tobias                    No Commission
        118 Himuro                    No Commission
        119 Colmenares                No Commission
        120 Weiss                     No Commission
        121 Fripp                     No Commission
```

TASK 3 (Aggregate Operators):

Aim: Write Commands on SQL Aggregate Functions, Group By clause, Having clause

a. Count the total records in the emp table.

Query: select count(*) from emp;

Output:

```
SQL> select count(*) from emp;

  COUNT(*)
----------
        11
```

b. Calculate the total and average salary of the employees.

Query: select sum(sal) “Total”, avg(sal) “Average” from emp;

Output:

c. Determine the maximum and minimum salary of the employees and rename the columns max_salalary and min_salary.

Query: select max(sal) “max_salary”, min(sal) “min_salary” from emp;

Output:

```
SQL> select max(sal) "max_salary", min(sal) "min_salary" from emp;

max_salary min_salary
---------- ----------
     10000       5000
```

d. Find the no.of departments in employee table.

Query: select deptno, count(deptno) from emp group by deptno;

Output:

```
SQL> select deptno, count(deptno) from emp group by deptno;

    DEPTNO COUNT(DEPTNO)
---------- -------------
        30             1
        20             3
        33             2
        10             5
```

e. Display job wise sum, avg, max, min salaries.

Query: select job, sum(sal), avg(sal), max(sal), min(sal) from emp group by job;

Output:

```
SQL> select job, sum(sal), avg(sal), max(sal), min(sal) from emp group by job;

JOB              SUM(SAL)   AVG(SAL)   MAX(SAL)   MIN(SAL)
--------------- ---------- ---------- ---------- ----------
Manager             14000       7000       9000       5000
Advisor              9000       9000       9000       9000
Clerk               15000       7500       9000       6000
Supervisor           8000       8000       8000       8000
President           10000      10000      10000      10000
ExeManager           9000       9000       9000       9000
AsstHead            10000      10000      10000      10000
SWManager            8000       8000       8000       8000
GM                  10000      10000      10000      10000
```

f. Display maximum salaries of all departments having maximum salary>2000.

Query: select deptno, max(sal) from emp group by deptno having max(sal)>2000;

Output:

```
SQL> select deptno, max(sal) from emp group by deptno having max(sal)>2000;

    DEPTNO   MAX(SAL)
---------- ----------
        30       9000
        20      10000
        33       9000
        10      10000
```

g. Display job wise sum, avg, max and min salaries in department 10 having average salary>1000 and result is orederd by sum of salary in desc order.

Query: select job, sum(sal), avg(sal), max(sal), min(sal) from emp where deptno=10 group by job having avg(sal)>1000 order by sum(sal) desc;

Output:

```
SQL> select job, sum(sal), avg(sal), max(sal), min(sal) from emp where deptno=10
 group by job having avg(sal)>1000 order by sum(sal) desc;

JOB              SUM(SAL)   AVG(SAL)   MAX(SAL)   MIN(SAL)
--------------- ---------- ---------- ---------- ----------
AsstHead            10000      10000      10000      10000
GM                  10000      10000      10000      10000
Clerk                9000       9000       9000       9000
SWManager            8000       8000       8000       8000
Supervisor           8000       8000       8000       8000
```

TASK 4 (Nested Queries):

Aim: Write Commands on Nested Queries

a. Find the third highest salary of the employees.

Query:

SQL>select max(sal) from emp where sal<(select max(sal) from emp where sal<(select max(sal) from emp));

Output:

```
SQL> select max(sal) from emp where sal<(select max(sal) from emp where sal<(select max(sal)
from emp));

  MAX(SAL)
----------
      8000
```

b. Display all the employee names and salary whose salary is greater than the minimum salary and job title starts with 'M'.

Query:

SQL>select ename, sal from emp where sal>(select min(sal) from emp) and job like 'M%';

Output:

```
SQL> select ename, sal from emp where sal>(select min(sal) from emp) and job like 'M%';

ENAME                    SAL
-------------------- ---------
OConnell                 9000
```

c. Write a Query to display information about employees who earn more than any employee in department 30.

Query:

SQL>select empno, ename, sal, deptno from emp where sal>any (select sal from emp where deptno=30);

Output:

```
SQL> select empno, ename, sal, deptno from emp where sal>any (select sal from emp where deptn
o=30);

     EMPNO ENAME                     SAL     DEPTNO
---------- -------------------- ---------- ----------
      7000 King                      10000         20
      7650 SPaul                     10000         10
      7680 Kochhar                   10000         10
```

d. Display the employees who have the same job as Jones and whose salary>=Fords.

Query:

SQL>select empno, ename, sal, job from emp where job= (select job from emp where ename='Jones') and sal>= (select sal from emp where ename='FORDS');

Output:

```
SQL> select empno, ename, sal, job from emp where job= (select job from emp where ename='Jone
s') and sal>= (select sal from emp where ename='Fords');

no rows selected
```

e. List out the employee names who get the salary> maximum salary of dept with deptno 20,30.

Query:

SQL>select ename from emp where sal>(select max(sal) from emp where deptno in(20,30));

Output:

```
SQL> select ename from emp where sal>(select max(sal) from emp where deptno in(20,30));

no rows selected
```

f. Display the maximum salaries of the departments whose maximum salary>9000.

Query:

SQL>select max(sal) from emp group by deptno having max(sal)>9000;

Output:

```
SQL> select max(sal) from emp group by deptno having max(sal)>9000;

  MAX(SAL)
----------
     10000
     10000
```

g. Create a table employee with the same structure as the table emp and insert rows into the

table using select clauses.

Query:

SQL>create table employee as (select * from emp);

Output:

```
SQL> create table employee as (select * from emp);

Table created.

SQL> select * from employee;

     EMPNO ENAME                JOB                 MGR     DEPTNO        SAL
---------- -------------------- --------------- ---------- ---------- ----------
COMMISSION DOB
---------- ---------
      7000 King                 President             7500         20      10000
       500 12-JAN-88

      7200 Whalen               Supervisor            7580         10       8000
       200 04-FEB-91

      7500 OConnell             Manager                            30       9000
      1000 07-JUL-89

11 rows selected.
```

h. Create a manager table from the emp table which should hold details only about managers.

Query:

SQL>create table manager as (select * from emp where job like '%Manager%');

Output:

```
SQL> create table manager as (select * from emp where job like '%Manager%');

Table created.

SQL> select * from manager
  2  ;

     EMPNO ENAME                JOB                 MGR     DEPTNO        SAL
---------- -------------------- --------------- ---------- ---------- ----------
COMMISSION DOB
---------- ---------
      7500 OConnell             Manager                            30       9000
      1000 07-JUL-89

      7580 Jane                 SWManager                          10       8000
      1000 09-DEC-91

      7850 Hartstein            Manager                            20       5000
      1000 13-AUG-90

     EMPNO ENAME                JOB                 MGR     DEPTNO        SAL
---------- -------------------- --------------- ---------- ---------- ----------
COMMISSION DOB
---------- ---------
      7800 Grant                ExeManager                         33       9000
      1000 18-NOV-91
```

TASK 5 (Joins and Set Operators):

Aim: Write a Programs on Joins, Set Operators

a. Display all the employees and departments implementing left outer join.

Query:

SQL>select e.empno, e.ename, d.deptno, d.dname from emp e left outer join dept d on(e.deptno=d.deptno);

Output:

```
SQL> select e.empno, e.ename, d.deptno, d.dname from emp e left outer join dept d on(e.deptno
=d.deptno);

     EMPNO ENAME                    DEPTNO DNAME
---------- -------------------- ---------- ----------
      7000 King                         20 Marketing
      7200 Whalen                       10 Executive
      7500 OConnell                     30 Production
      7580 Jane                         10 Executive
      7599 Mary                         33 Despatch
```

b. Display the employee name and department name in which they are working implementing a full outer join.

Query:

SQL> select e.ename,d.dname from emp e full outer join dept d on(e.deptno=d.deptno);

Output:

```
SQL> select e.ename,d.dname from emp e full outer join dept d on(e.deptno=d.deptno);

ENAME                DNAME
-------------------- ----------
Russell              Executive
Kochhar              Executive
SPaul                Executive
Jane                 Executive
Whalen               Executive
Hartstein            Marketing
Birch                Marketing
King                 Marketing
OConnell             Production
Grant                Despatch
Mary                 Despatch

ENAME                DNAME
-------------------- ----------
                     Packaging
```

c. Write a Query to display the employee name and manager's name and salary for all employees.

Query:

SQL> select e.ename, m.ename "MGR", m.sal "MGRSAL" from emp e, emp m where e.mgr=m.empno;

Output:

```
SQL> select e.ename, m.ename "MGR", m.sal "MGRSAL" from emp e, emp m where e.mgr=m.empno;

ENAME                MGR                    MGRSAL
-------------------- -------------------- ----------
King                 OConnell                 9000
Whalen               Jane                     8000
Mary                 OConnell                 9000
Birch                Grant                    9000
SPaul                Jane                     8000
Kochhar              Hartstein                5000
Russell              Grant                    9000
```

d. Write a Query to Output the name, job, employee number, department name, location for each department even if there are no employees.

Query:

SQL> select e.eno, e.ename, e.job,d.depno, d.dname, d.loc from emp e left outer join dept d on(e.depno=d.depno);

Output:

```
SQL> select e.eno, e.ename, e.job,d.depno, d.dname, d.loc from emp e left outer join dept d on(e.depno=d.depno);

       ENO ENAME                JOB                    DEPNO DNAME                LOC
---------- -------------------- -------------------- ---------- -------------------- ----------
       101 vamsi                sales                         5 production           UK
       103 sai                  hrm                           6 sales                Germany
       201 arun                                              10 Executive            US
         1 pantu                                             10 Executive            US
       101 vamsi
       104 raju                 manager
       104 raj
       501 tarun

8 rows selected.
```

e. Display the details of those who draw the same salary.

Query:

SQL> select e.eno, e.ename, e.job, e.sal,d.sal from emp e,emp d where e.sal=d.sal and e.eno<>d.eno;

Output:

```
SQL> select e.eno, e.ename, e.job, e.sal,d.sal from emp e,emp d where e.sal=d.sal and e.eno<>d.eno;

       ENO ENAME                JOB                      SAL        SAL
---------- -------------------- -------------------- ---------- ----------
       103 sai                  hrm                     90000      90000
       101 vamsi                sales                   90000      90000

SQL>
```

TASK 6 (Views):

Aim: Write a Commands on Views

a.Create a view that displays the employee id, name and salary of employees who belong to 10th department.

Query:

SQL>Create view emp_view as select employee_id,last_name,salary from employees where department_id=10;

Output:

```
SQL> create view emp_view as select employee_id,last_name,salary from employees where department_id=10;

View created.

SQL> select * from emp_view;

EMPLOYEE_ID LAST_NAME                     SALARY
----------- ------------------------- ----------
        200 Whalen                          4400

SQL>
```

b.Create a view with read only option that displays the employee name and their department name

Query:

SQL> create view emp_dept as select employee_id,last_name,department_id from employees with read onlyh constraint emp_dept_readonly;

Output:

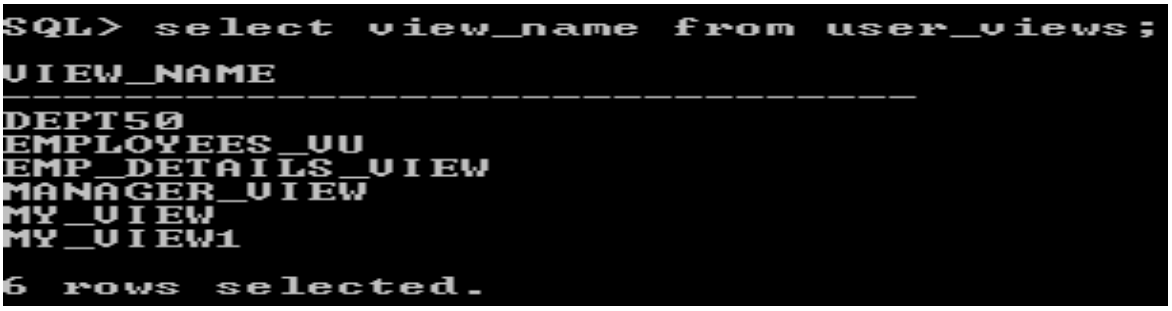
```
SQL> create view emp_dept as select employee_id,last_name,department_id from employees with read only constraint emp_dept_readonly;

View created.

SQL> select * from emp_dept
  2  ;

EMPLOYEE_ID LAST_NAME                 DEPARTMENT_ID
----------- ------------------------- -------------
        100 King                                 90
        101 Kochhar                              90
        102 De Haan                              90
        103 Hunold                               60
        104 Ernst                                60
        105 Austin                               60
        106 Pataballa                            60
        107 Lorentz                              60
        108 Greenberg                           100
        109 Faviet                              100
        110 Chen                                100

EMPLOYEE_ID LAST_NAME                 DEPARTMENT_ID
----------- ------------------------- -------------
        111 Sciarra                             100
        112 Urman                               100
        113 Popp                                100
        114 Raphaely                             30
        115 Khoo                                 30
        116 Baida                                30
        117 Tobias                               30
        118 Himuro                               30
        119 Colmenares                           30
        120 Weiss                                50
        121 Fripp                                50
```

c. Display all the views generated.

Query:

SQL> select view_name from user_views;

Output:



d. Execute the DML commands on the view created.and drop them.

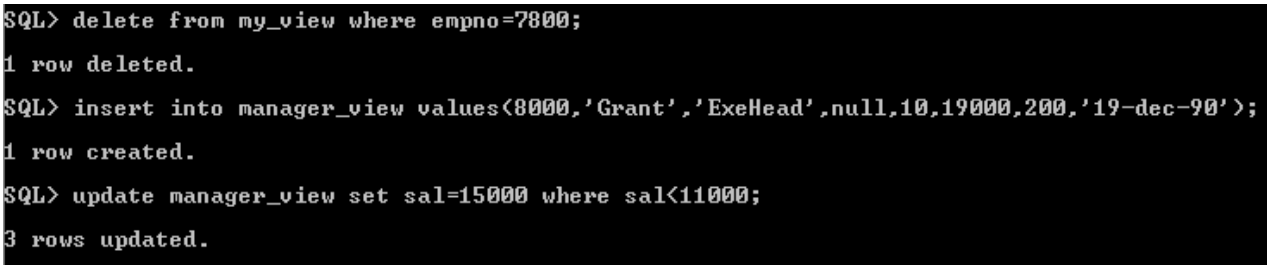
Query:

SQL>delete from my_view where empno=7900;

SQL> insert into manager_view values(8000, 'Grant', 'ExeHead', null, 10, 19000, 200, '19-dec-90');

SQL> update manager_view set sal=15000 where sal<11000;

Output:

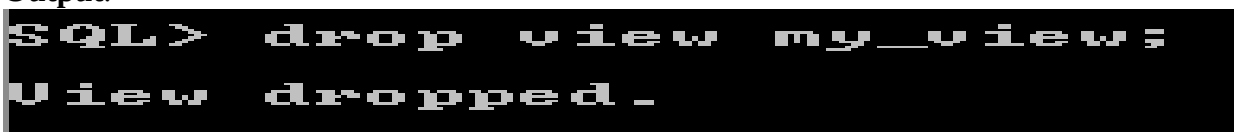


e.Drop a view.

Query:

SQL> drop view my_view;

Output:



TASK 7 (Indexes, Sequences and Synonyms):

.

INDEXES

Aim: Practices on Function based indexes

SQL>create index emp_index on emp (upper(ename));

Output:

```
SQL> create index emp_index on emp (upper(ename));

Index created.
```

SQL>select employee_id,last_name,job_id from employees where last_name between 'N' and 'P';

Output:

```
SQL> select ename from emp where ename between 'N' and 'P';

ENAME
--------------------
OConnell
```

2. Creating Index while creating Table.

SQL>create table emp2 (empnonumber(6) PRIMARY KEY USING INDEX (CREATE INDEX emp_idx ON emp2(empno)) ,ename varchar2(20),job varchar2(20));

Output:

```
SQL> create table emp2 (empno number(6) PRIMARY KEY USING INDEX (CREATE INDEX emp_idx ON emp2
(empno)) ,ename varchar2(20),job varchar2(20));

Table created.
```

User-defined indexes:

SQL>select index_name,table_name from user_indexes where table_name='EMP2';

Output:

```
SQL> select index_name,table_name from user_indexes where table_name='EMP2';

INDEX_NAME                     TABLE_NAME
------------------------------ ------------------------------
EMP_IDX                        EMP2
```

3. create table emp3(empno number(6) primary key, ename varchar2(20), job varchar2(10));

Output:

```
SQL> create table emp3(empno number(6) primary key, ename varchar2(20), job varchar2(10));

Table created.
```

Default indexes.

SQL>select index_name,table_name from user_indexes where table_name='EMP3';

Output:

```
SQL> select index_name,table_name from user_indexes where table_name='EMP3';

INDEX_NAME                     TABLE_NAME
------------------------------ ------------------------------
SYS_C007173                    EMP3
```

4. Displaying all the indexes.

SQL>Select index_name, table_name from user_indexes;

Output:

```
SQL> select index_name, table_name from user_indexes;

INDEX_NAME                     TABLE_NAME
------------------------------ ------------------------------
REG_ID_PK                      REGIONS
LOCATIONS_PK_IDX               LOCATIONS_NAMED_INDEX
LOC_ID_PK                      LOCATIONS
LOC_CITY_IX                    LOCATIONS
LOC_STATE_PROVINCE_IX          LOCATIONS
LOC_COUNTRY_IX                 LOCATIONS
JHIST_EMP_ID_ST_DATE_PK        JOB_HISTORY
```

5. SQL>select table_name, index_name, column_name from user_ind_columns where table_name='EMPLOYEES';

Output:

```
SQL> select table_name, index_name, column_name from user_ind_columns where table_name='EMPLO
YEES';

TABLE_NAME                     INDEX_NAME
------------------------------ ------------------------------
COLUMN_NAME
--------------------------------------------------------------------------------
EMPLOYEES                      EMP_EMAIL_UK
EMAIL

EMPLOYEES                      EMP_EMP_ID_PK
EMPLOYEE_ID

EMPLOYEES                      EMP_DEPARTMENT_IX
DEPARTMENT_ID
```

6. Dropping an index:

SQL> drop index emp_index;

Output:

```
SQL> drop index emp_index;

Index dropped.
```

SEQUENCE

1. SQL>create sequence my_seq start with 10 increment by 10 maxvalue 100 nocache;

Output:

```
SQL> create sequence my_seq start with 10 increment by 10 maxvalue 100 nocache;

Sequence created.
```

2. SQL>select my_seq.nextval from dual;

Output:

```
SQL> select my_seq.nextval from dual;
   NEXTVAL
----------
        10
```

3. SQL>select my_seq.currval from dual;

Output:

```
SQL> select my_seq.currval from dual;
   CURRVAL
----------
        10
```

4. SQL>create table dept(deptno number(6),dname varchar2(20),loc varchar2(10));

SQL>insert into dept values(my_seq.nextval,'Executive','US');

SQL>insert into dept values(my_seq.nextval,'Marketing','UK');

Output:

```
SQL> create table dept1(id number(3), dname varchar2(10));
Table created.
SQL> insert into dept1 values(my_seq.nextval, 'Admin');
1 row created.
```

```
SQL> select * from dept1;
        ID DNAME
---------- ----------
        20 Admin
```

5. SQL>drop sequence my_seq;

Output:

```
SQL> drop sequence my_seq;
Sequence dropped.
```

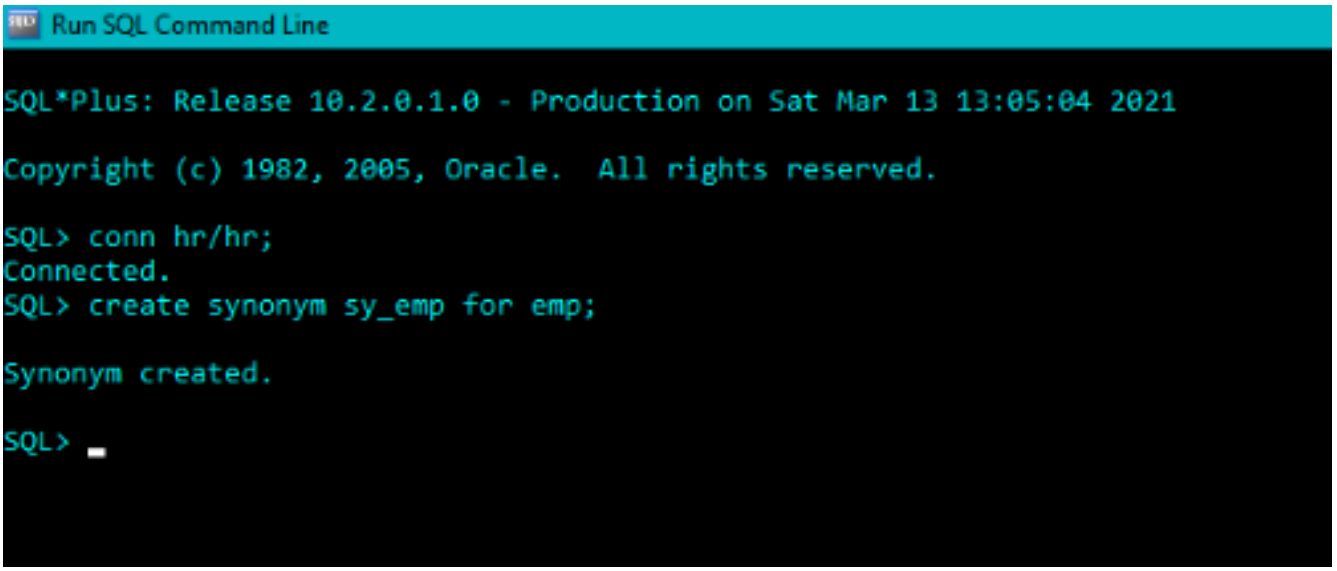
SYNONYM

A Synonym is another name defined for a table, view, sequence, procedure, function, and package. It provides an alternative and easier way of referring to the database objects.

Synonym(It is purely logical object) :=> Single Table=>Multiple names.

CREATE [PUBLIC] SYNONYM <SNAME> FOR OBJECT;

SQL> create synonym sy_emp for emp;



TASK 8 (DCL Commands):

Aim: Practices on DCL Commands

1. SQL>Create user test identified by pswd;

Output:

```
SQL> create user test identified by pswd;
User created.
```

2. SQL> Grant create session, create table, create sequence, create view to test;

c) Output:

```
SQL> Grant create session, create table, create sequence, create view to test;
Grant succeeded.
```

3. SQL>Create role manager;

SQL>Grant create table, create view to manager;

SQL>Grant manager to test;

Output:

```
SQL> create role manager;
Role created.
SQL> grant create table, create view to manager;
Grant succeeded.
SQL> grant manager to test;
Grant succeeded.
```

4. SQL>Alter user test identified by qwerty;

Output:

```
SQL> alter user test identified by qwerty;
User altered.
```

5. SQL>Grant select on employees to test;

Output:

```
SQL> grant select on hr.emp to test;
Grant succeeded.
```

6. SQL>Grant update (department_name,location_id) on departments to test;

Output:

```
SQL> grant update (dname, loc) on hr.dept to test;
Grant succeeded.
```

7. SQL>Grant select,insert on hr.locations to test;

Output:

```
SQL> grant select, insert on hr.dept to test;
Grant succeeded.
```

8. SQL>Revoke select,insert on departments from test;

Output:

```
SQL> revoke select, insert on hr.dept from test;
Revoke succeeded.
```

ROLE: A role is a collection of related privileges that an administrator can grant collectively to database users and other roles. Roles are nothing but a set of privileges instead of granting individual privileges to each user, the privileges can be granted to roles and the role is granted to each user.

Syntax:

Syntax for creating a role is

CREATE ROLE <role-name> [IDENTIFIED BY <password>];

Syntax for granting a role to users is:

GRANT <role-name> TO <user-list>;

Syntax to enable/ disable role is

SET ROLE <role-name>

TASK 9 (PL/SQL Blocks, Named Blocks):

Aim: To write a PL/SQL code to retrieve the employee name, join date and designation from employee database of an employee whose number is input by the user.

Program:

```
/*Employee details*/
DECLARE
v_name varchar2(25);
v_joindate date;
v_dsgn employees.job_id%type;
BEGIN
elect last_name,hire_date,job_id into v_name,v_joindate,v_dsgn from employees where
employee_id=&id;
DBMS_OUTPUT.PUT_LINE('Name:'||v_name||' Join
Date:'||v_joindate||'Designation:'||v_dsgn);
END;
/
```

Output:

```
Enter value for id: 193
Name:Everett Join Date:03-MAR-05 Designation:SH_CLERK

PL/SQL procedure successfully completed.
```

b.Write a PL/SQL code to calculate tax for an employee of an organization.

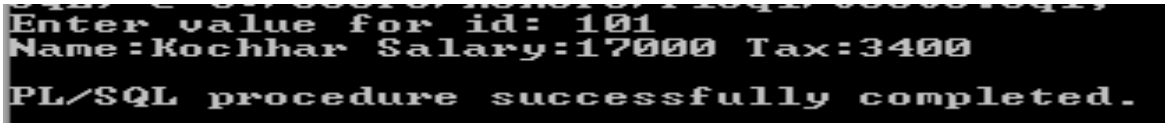
Aim: To write a PL/SQL code to calculate tax for an employee of an organization.

Program:

```
/*Calculate Tax*/
DECLARE
v_sal number(8);
v_tax number(8,3);
v_name varchar2(25);
BEGIN
select salary,last_name into v_sal,v_name from employees where employee_id=&id;
if v_sal<10000 then
```

```
 v_tax:=v_sal*0.1;
elsif v_sal between 10000 and 20000 then
 v_tax:=v_sal*0.2;
else
                v_tax:=v_sal*0.3;
               END IF;
               DBMS_OUTPUT.PUT_LINE('Name:'||v_name||' Salary:'||v_sal||'Tax:'||v_tax);
               END;
               /
```

Output:



b. Write a PL/SQL procedure for inserting, deleting and updating in employee table.

Aim: To Write a PL/SQL procedure for inserting, deleting and updating in employee table.

Program:

```
create or replace procedure proc_dml (p_id emp.employee_id%type, p_sal number,p_case
number) is
BEGIN
case p_case
        when 1 then
           DBMS_OUTPUT.PUT_LINE('Insertion...');
                insert into emp(employee_id,last_name,email,hire_date,job_id)
values(p_id,'Franco',’FJames’,'12-JAN-02','ST_CLERK');
        when 2 then
                DBMS_OUTPUT.PUT_LINE('Deletion...');
                delete from emp where employee_id=p_id;
        when 3 then
                DBMS_OUTPUT.PUT_LINE('Updation...');
                update emp set salary=p_sal where employee_id=p_id;
        end case;
DBMS_OUTPUT.PUT_LINE('DML operation performed on '||SQL%rowcount||' rows');
END;
/
DECLARE
v_id employees.employee_id%type:=&id;
v_sal employees.salary%type:=&sal;
v_case number:=&case1or2or3;
begin
proc_dml(v_id,v_sal,v_case);
END;
/
```

Output:

```
SQL> @C:/Users/Kshore/Plsql/test1.sql
Enter value for employee_id: 210
Enter value for salary: 20000
Enter value for case1or2or3: 1
Insertion...
DML operation performed on 1 rows

PL/SQL procedure successfully completed.

SQL> @C:/Users/Kshore/Plsql/test1.sql
Enter value for employee_id: 101
Enter value for salary: 21000
Enter value for case1or2or3: 3
Updation...
DML operation performed on 1 rows

PL/SQL procedure successfully completed.

SQL> @C:/Users/Kshore/Plsql/test1.sql
Enter value for employee_id: 210
Enter value for salary: 20000
Enter value for case1or2or3: 2
Deletion...
DML operation performed on 2 rows

PL/SQL procedure successfully completed.
```

c. Write a PL/SQL function that accepts department number and returns the total salary of the department.

Aim: To write a PL/SQL function that accepts department number and returns the total salary of the department.

Program:

```
create function func_dept (p_dept number) return number is
v_total number;
BEGIN
select sum(salary) into v_total from employees where department_id=p_dept;
return v_total;
END;
/
DECLARE
v_dept number:=&department_id;
v_total number;
BEGIN
v_total:=func_dept(v_dept);
DBMS_OUTPUT.PUT_LINE('Total salary in Department '||v_dept||' is '||v_total);
END;
/
```

Output:

```
SQL> @C:/Users/Kshore/Plsql/temp1.sql

Function created.

SQL> @C:/Users/Kshore/Plsql/test1.sql
Enter value for department_id: 40
Total salary in Department 40 is 6500

PL/SQL procedure successfully completed.
```

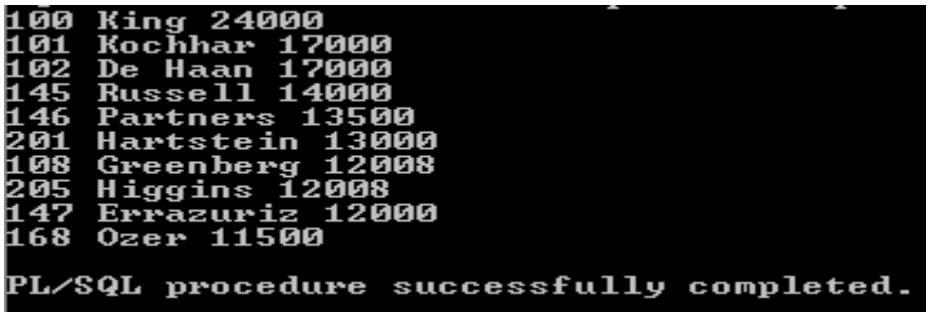
TASK 10 (Cursor and Trigger):

Aim: To write a PL/SQL program to display top 10 employee details based on salary using cursors.

Program:

```
/*Top 10 salary earning employee details*/
DECLARE
cursor c_emp_cursor is select employee_id, last_name, salary from employees order by
salary desc;
v_rec c_emp_cursor%rowtype;
v_i number(3):=0;
BEGIN
open c_emp_cursor;
loop
v_i:=v_i+1;
fetch c_emp_cursor into v_rec;
exit when v_i>10;
DBMS_OUTPUT.PUT_LINE(v_rec.employee_id||' '||v_rec.last_name||' '||v_rec.salary);
END LOOP;
close c_emp_cursor;
END;
                /
```

Output:



d). Write a PL/SQL program to update the commission values for all employees with salary less than 5000 by adding 1000 to existing employees.

Aim: To write a PL/SQL program to update the commission values for all employees with salary less than 5000 by adding 1000 to existing employees.

Program:

```
/*Updation*/
declare
cursor c_emp is select salary,commission_pct from employees;
v_emp c_emp%rowtype;
v_temp number(7,2);
v_temp1 number;
BEGIN
open c_emp;
loop
 fetch c_emp into v_emp;
 exit when c_emp%notfound;
```

```
  v_temp1:=v_emp.commission_pct;
  v_temp:=(v_emp.salary*v_emp.commission_pct)+1000;
  v_temp:=v_temp/v_emp.salary;
  if(v_emp.salary<5000) then
  update employees set commission_pct=v_temp where employee_id=v_temp.employee_id;
  end if;
  DBMS_OUTPUT.PUT_LINE('Commission % updated from '||v_temp1||' to '||v_temp);
 end loop;
END;
/
```

Output:

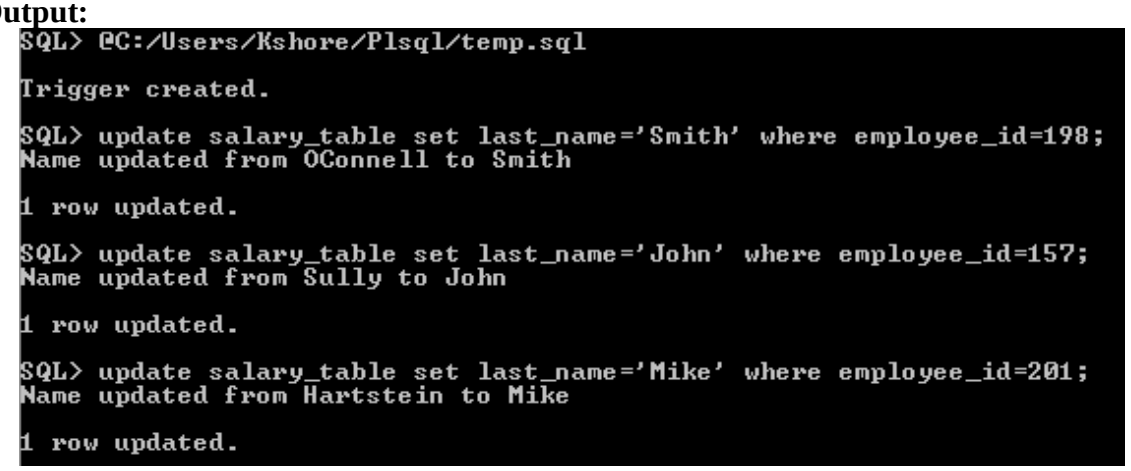
e. Write a trigger on the employee table which shows the old values and new values of ename after any updations on ename on Employee table.

Aim: To write a trigger on the employee table which shows the old values and new values of ename after any updations on ename on Employee table.

Program:

```
create or replace trigger t_emp_name after update of last_name on salary_table FOR EACH
ROW
begin
DBMS_OUTPUT.PUT_LINE('Name updated from '||:OLD.last_name||' to
'||:NEW.last_name);
END;
/
```

Output:

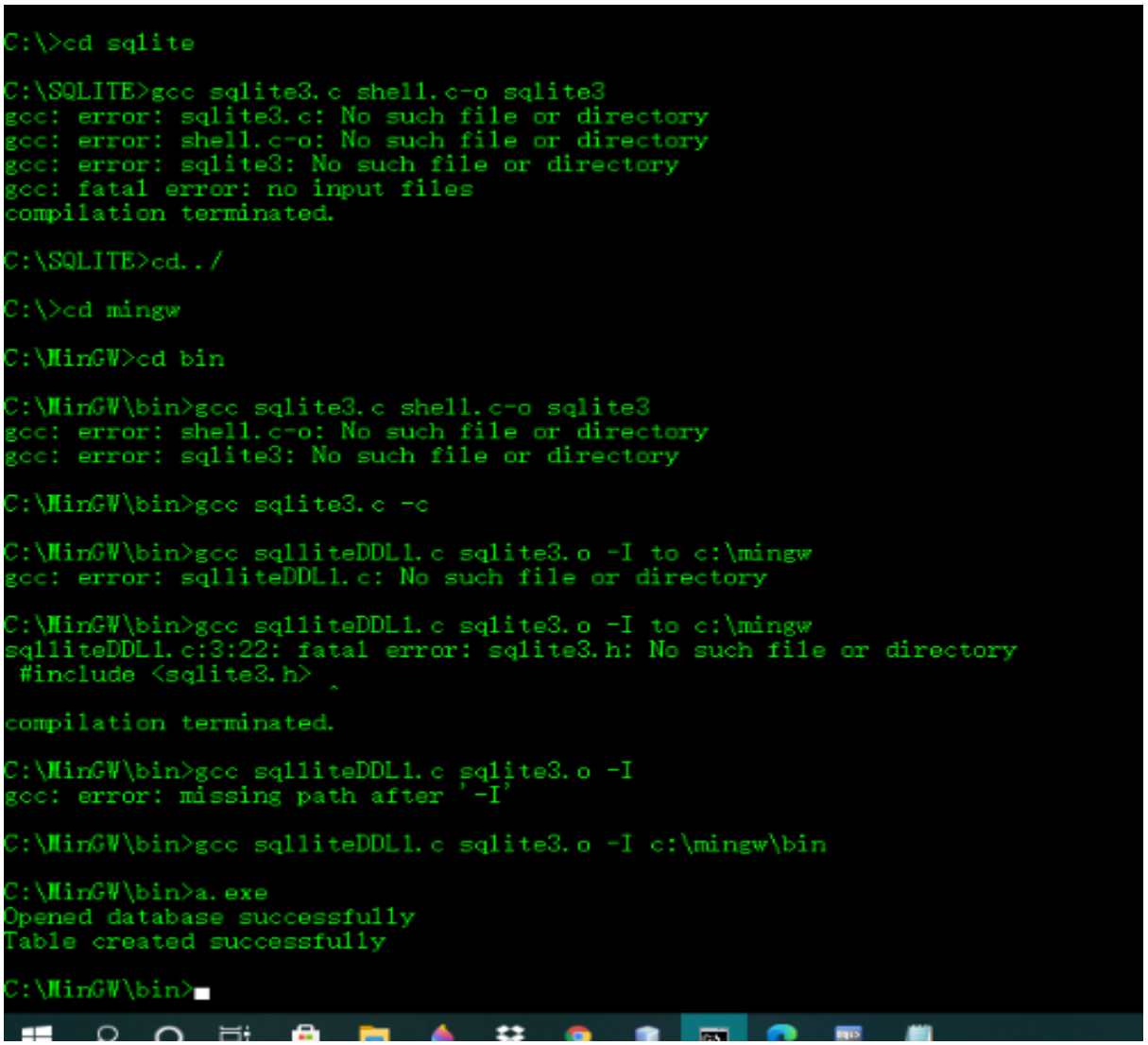


TASK 11 (C Implementation for DB):

Write a C program to connect to SQLite Database and perform DDL and DML operations in it. Write a C program to perform all kinds of retrieval operations on SQLite database.

```
#include    <stdio.h>
#include   <stdlib.h>
#include <sqlite3.h>

static int callback(void *NotUsed, int argc, char **argv, char **azColName) {
   int i;
   for(i = 0; i<argc; i++) {
      printf("%s = %s\n", azColName[i], argv[i] ? argv[i] : "NULL");
   }
   printf("\n");
   return 0;
}

int main(int argc, char* argv[]) {
   sqlite3 *db;
   char *zErrMsg = 0;
   int rc;
   char *sql;
const char* data = "Callback function called";

   /* Open database */
   rc = sqlite3_open("test.db", &db);

   if( rc ) {
      fprintf(stderr, "Can't open database: %s\n", sqlite3_errmsg(db));
      return(0);
   } else {
      fprintf(stdout, "Opened database successfully\n");
   }
```

```
/* Create SQL statement */
sql = "CREATE TABLE COMPANY(" \
   "ID INT PRIMARY KEY     NOT NULL," \
   "NAME           TEXT    NOT NULL," \
   "AGE            INT     NOT NULL," \
   "ADDRESS        CHAR(50)," \
   "SALARY         REAL );";

/* Execute SQL statement */
rc = sqlite3_exec(db, sql, callback, 0, &zErrMsg);

if( rc != SQLITE_OK ){
   fprintf(stderr, "SQL error: %s\n", zErrMsg);
   sqlite3_free(zErrMsg);
} else {
   fprintf(stdout, "Table created successfully\n");
}
```



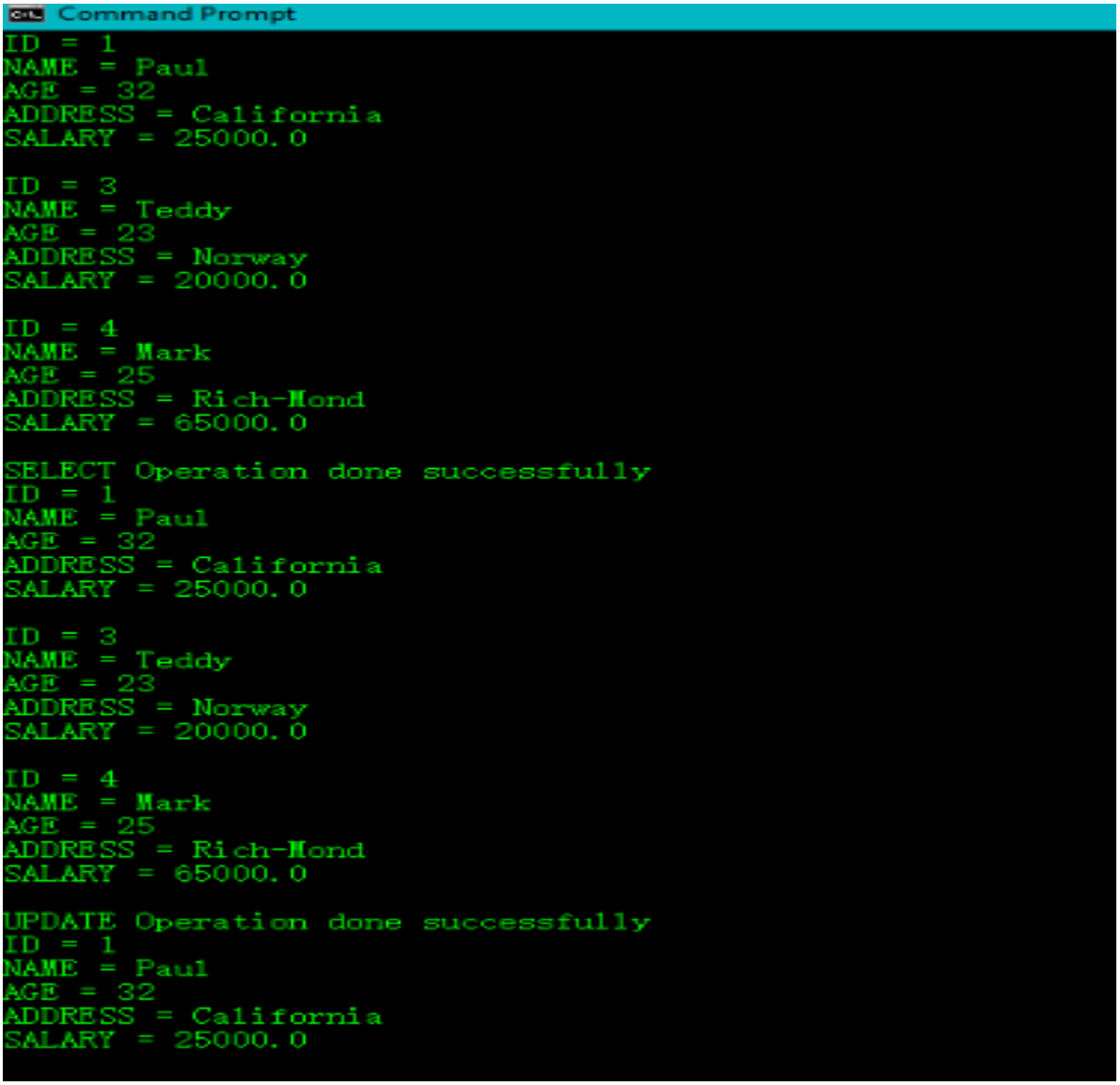

```
/* Create SQL statement */
   sql = "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) " \
         "VALUES (1, 'Paul', 32, 'California', 20000.00 ); " \
         "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) " \
         "VALUES (2, 'Allen', 25, 'Texas', 15000.00 ); "     \
         "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY)" \
         "VALUES (3, 'Teddy', 23, 'Norway', 20000.00 );" \
         "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY)" \
         "VALUES (4, 'Mark', 25, 'Rich-Mond ', 65000.00 );";

   /* Execute SQL statement */
   rc = sqlite3_exec(db, sql, callback, 0, &zErrMsg);
```

```
   if( rc != SQLITE_OK ){
      fprintf(stderr, "SQL error: %s\n", zErrMsg);
      sqlite3_free(zErrMsg);
   } else {
      fprintf(stdout, "Records created successfully\n");
   }
```

```
Operation done successfully
ID = 1
NAME = Paul
AGE = 32
ADDRESS = California
SALARY = 25000.0

ID = 3
NAME = Teddy
AGE = 23
ADDRESS = Norway
SALARY = 20000.0

ID = 4
NAME = Mark
AGE = 25
ADDRESS = Rich-Mond
SALARY = 65000.0

Operation done successfully

C:\MinGW\bin>gcc sqlliteDDL1.c sqlite3.o -I c:\mingw\bin

C:\MinGW\bin>a.exe
Opened database successfully
SQL error: table COMPANY already exists
SQL error: UNIQUE constraint failed: COMPANY.ID
ID = 1
NAME = Paul
AGE = 32
ADDRESS = California
SALARY = 25000.0

ID = 3
NAME = Teddy
AGE = 23
ADDRESS = Norway
SALARY = 20000.0

ID = 4
NAME = Mark
AGE = 25
ADDRESS = Rich-Mond
SALARY = 65000.0
```

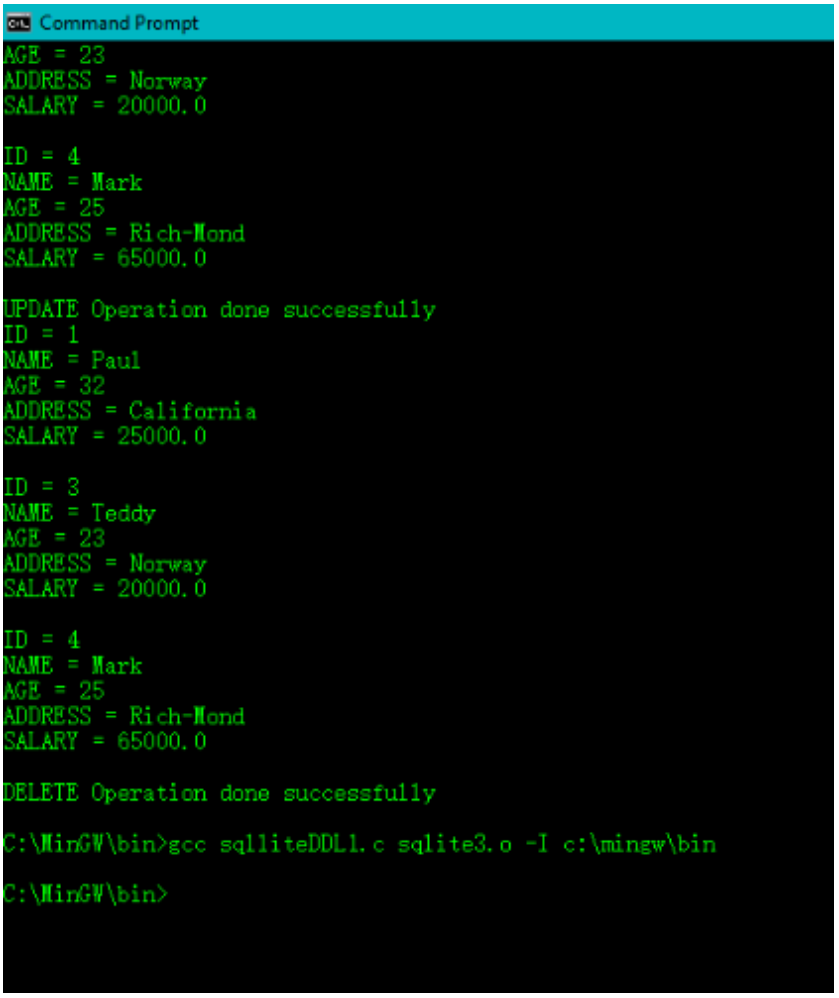
```
/* Create SQL statement */
   sql = "SELECT * from COMPANY";

   /* Execute SQL statement */
   rc = sqlite3_exec(db, sql, callback, (void*)data, &zErrMsg);
```

```
   if( rc != SQLITE_OK ) {
      fprintf(stderr, "SQL error: %s\n", zErrMsg);
      sqlite3_free(zErrMsg);
   } else {
      fprintf(stdout, "SELECT Operation done successfully\n");
   }
```




```
/* Create merged SQL statement */
   sql = "UPDATE COMPANY set SALARY = 25000.00 where ID=1; " \
         "SELECT * from COMPANY";

   /* Execute SQL statement */
   rc = sqlite3_exec(db, sql, callback, (void*)data, &zErrMsg);
```

```
   if( rc != SQLITE_OK ) {
      fprintf(stderr, "SQL error: %s\n", zErrMsg);
      sqlite3_free(zErrMsg);
   } else {
      fprintf(stdout, "UPDATE Operation done successfully\n");
   }
/* Create merged SQL statement */
   sql = "DELETE from COMPANY where ID=2; " \
         "SELECT * from COMPANY";

   /* Execute SQL statement */
   rc = sqlite3_exec(db, sql, callback, (void*)data, &zErrMsg);

   if( rc != SQLITE_OK ) {
      fprintf(stderr, "SQL error: %s\n", zErrMsg);
      sqlite3_free(zErrMsg);
   } else {
      fprintf(stdout, "DELETE Operation done successfully\n");
   }
   sqlite3_close(db);
   return 0;
}
```


Command Prompt
AGE = 23
ADDRESS = Norway
SALARY = 20000.0
ID = 4
NAME = Mark
AGE = 25
ADDRESS = Rich-Mond
SALARY = 65000.0
UPDATE Operation done successfully
ID = 1
NAME = Paul
AGE = 32
ADDRESS = California
SALARY = 25000.0
ID = 3
NAME = Teddy
AGE = 23
ADDRESS = Norway
SALARY = 20000.0
ID = 4
NAME = Mark
AGE = 25
ADDRESS = Rich-Mond
SALARY = 65000.0
DELETE Operation done successfully
C:\MinGW\bin>gcc sqlliteDDL1.c sqlite3.o -I c:\mingw\bin
C:\MinGW\bin>

TASK 12 (Case Study):

Download standard data of reasonable size (Unit level data of various rounds of NSS surveys) form internet and implement various SQL commands.