CONTENTS

Exp. no: 1 24-01-2022

FAMILIARIZATION OF SQL COMMANDS

Data Definition Language (DDL) Statements

1. Create

To create tables, views, synonyms, sequences, functions, procedures, packages etc.

CREATE TABLE <tablename>(attribute1 datatype, attribute2 datatype, attribute n datatype);

2. Alter

To alter the structure of a table.

- To add new columns

ALTER TABLE <tablename> add (attribute1 datatype);

- To modify a column.

ALTER TABLE <tablename> modify (attribute datatype);

- To drop columns

ALTER TABLE <tablename> drop column <column name>;

3. Rename

To rename a table, view, sequence, or private synonym for a table, view, or sequence.

Oracle automatically transfers integrity constraints, indexes, and grants on the old object to the new object.

RENAME <oldtablename> to <newtablename>

4. Drop

To permanently delete the values and the structure of tables, functions, procedures, views, synonym, sequences etc.

DROP table <tablename>;

5. Truncate

Use the Truncate statement to delete all the rows from table permanently.

TRUNCATE table <tablename>;

Data Manipulation Language (DML) Statements

1. Insert

Used to add new rows to a table.

INSERT INTO <table name>VALUES (<value 1>, ... <value n>);

2. Update

The update statement is used to change values that are already in a table.

UPDATE <table name> SET <attribute> = <expression> WHERE <condition>;

3. Delete

To delete a particular row from the table based on a condition.

DELETE FROM <table name> WHERE <condition>;

If the WHERE clause is omitted, then every row of the table is deleted.

4. Select

A SELECT statement can include the following:

1. The name of each column you want to include
2. The name of the table or view that contains the data
3. A search condition to uniquely identify the row that contains the information you want
4. The name of each column used to group your data
5. A search condition that uniquely identifies a group that contains the information you want
6. The order of the results so a specific row among duplicates can be returned.

SELECT column names
FROM table or view name
WHERE search condition
GROUP BY column names
HAVING search condition
ORDER BY column-name

The SELECT and FROM clauses must be specified. The other clauses are optional.

To retrieve all columns (in the same order as they appear in the table's definition), use an asterisk (*) instead of naming the columns:

SELECT *FROM <TABLENAME>;

The FROM clause specifies the table that you want to select data from. You can select columns from more than one table.

Transaction Control Statements (TCL)

TCL statements are used to manage the changes made by DML statements. A transaction is a set of SQL statements which Oracle treats as a Single Unit i.e. all the statements should execute successfully or none of the statements should execute.

1. Commit

To make the changes done in a transaction permanent, issue the COMMIT statement.

2. Rollback

To rollback the changes done in a transaction, give ROLLBACK statement. Rollback restores the state of the database to the last commit point.

3. Savepoint

To specify a point in a transaction to which later you can roll back.
SAVEPOINT savepointname;

Data Control Language (DCL) Statements

DCL statements are used to control access to data stored in a database.

1. Grant

To provide privileges and accesses on database objects to the users.

```
GRANT <privilege name> ON <object name> TO <user name/role  name/public>;
```

2. Revoke

To remove the previously granted privileges from the user.

Exp. no: 2 01-12-2022

EMPLOYEE DATABASE

AIM

Create a table employee. The attributes of employee is empno, empname, place, desig, salary, dept_no, dept_name.

Set empno as primary key and also set 100000 as a default salary of employee

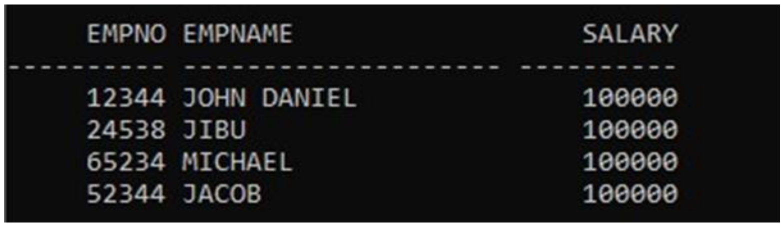
Create a constraint where the dept_no ranges from 1 -10

a. Display the empno, empname and salary.
b. Display all records of employee table
c. Display the details of the employee whose salary is greater than 10000
d. Increment the salary of 'officer' by 1000
e. Display the name and id of employee whose salary ranges between 10000 and 15000
f. Display the employee details whose name start with 'A'
g. Change the salary of officer to 15000
h. Add a column district to the table
i. Delete the column district
j. Modify the column name desig to designation

QUERY AND OUTPUT:

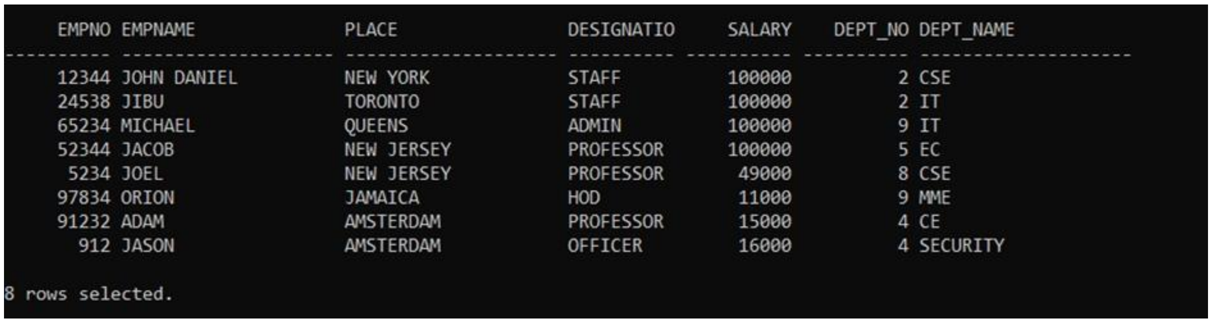
SQL> create table employee(empno number primary key,empname varchar(10),place varchar(10),desig varchar(10),salary number default 10000,dept_no number check(dept_no between 0 and 10),dept_name varchar(10));

a) SQL> SELECT empno, empname, salary FROM employee;



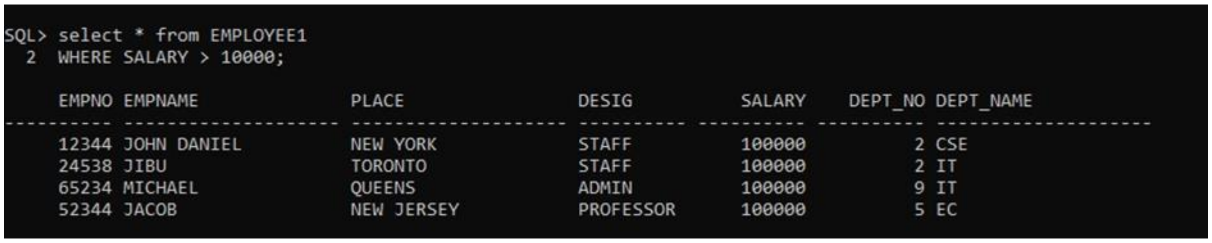
```
     EMPNO EMPNAME                  SALARY
---------- -------------------- ----------
     12344 JOHN DANIEL              100000
     24538 JIBU                     100000
     65234 MICHAEL                  100000
     52344 JACOB                    100000
```

b) SQL> SELECT * FROM employee;



```
     EMPNO EMPNAME              PLACE                DESIGNATIO     SALARY    DEPT_NO DEPT_NAME
---------- -------------------- -------------------- ---------- ---------- ---------- --------------------
     12344 JOHN DANIEL          NEW YORK             STAFF          100000          2 CSE
     24538 JIBU                 TORONTO              STAFF          100000          2 IT
     65234 MICHAEL              QUEENS               ADMIN          100000          9 IT
     52344 JACOB                NEW JERSEY           PROFESSOR      100000          5 EC
      5234 JOEL                 NEW JERSEY           PROFESSOR       49000          8 CSE
     97834 ORION                JAMAICA              HOD             11000          9 MME
     91232 ADAM                 AMSTERDAM            PROFESSOR       15000          4 CE
       912 JASON                AMSTERDAM            OFFICER         16000          4 SECURITY

8 rows selected.
```

c) SQL> select * from EMPLOYEE1 WHERE SALARY > 10000;

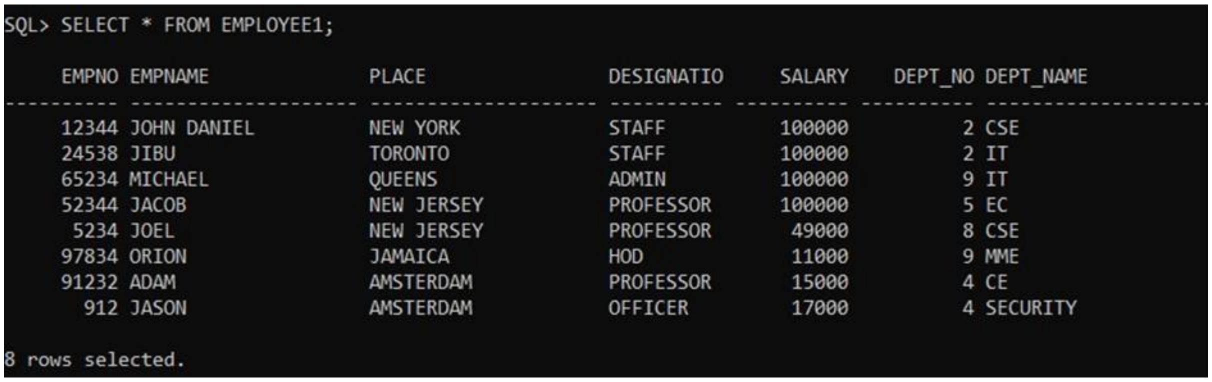


```
SQL> select * from EMPLOYEE1
  2  WHERE SALARY > 10000;

     EMPNO EMPNAME              PLACE                DESIG          SALARY    DEPT_NO DEPT_NAME
---------- -------------------- -------------------- ---------- ---------- ---------- --------------------
     12344 JOHN DANIEL          NEW YORK             STAFF          100000          2 CSE
     24538 JIBU                 TORONTO              STAFF          100000          2 IT
     65234 MICHAEL              QUEENS               ADMIN          100000          9 IT
     52344 JACOB                NEW JERSEY           PROFESSOR      100000          5 EC
```

d) SQL> UPDATE employee

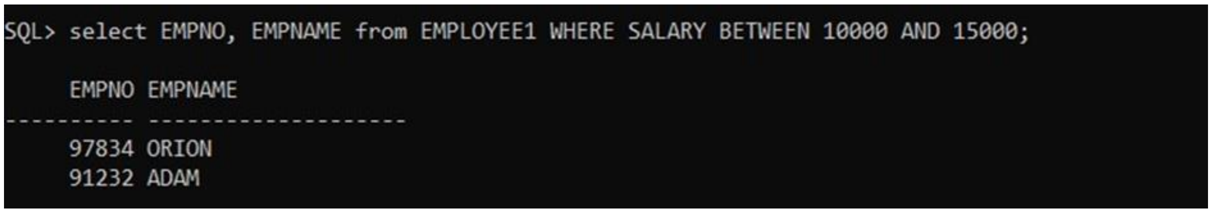
SET salary=salary+10000

WHERE desig = 'Officer';



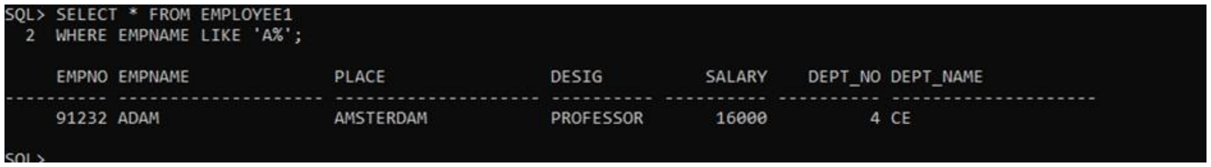
```
SQL> SELECT * FROM EMPLOYEE1;

     EMPNO EMPNAME              PLACE                DESIGNATIO     SALARY    DEPT_NO DEPT_NAME
---------- -------------------- -------------------- ---------- ---------- ---------- --------------------
     12344 JOHN DANIEL          NEW YORK             STAFF          100000          2 CSE
     24538 JIBU                 TORONTO              STAFF          100000          2 IT
     65234 MICHAEL              QUEENS               ADMIN          100000          9 IT
     52344 JACOB                NEW JERSEY           PROFESSOR      100000          5 EC
      5234 JOEL                 NEW JERSEY           PROFESSOR       49000          8 CSE
     97834 ORION                JAMAICA              HOD             11000          9 MME
     91232 ADAM                 AMSTERDAM            PROFESSOR       15000          4 CE
       912 JASON                AMSTERDAM            OFFICER         17000          4 SECURITY

8 rows selected.
```

e) SQL> empno, empname FROM employee WHERE salary>10000 AND salary<15000;



```
SQL> select EMPNO, EMPNAME from EMPLOYEE1 WHERE SALARY BETWEEN 10000 AND 15000;

     EMPNO EMPNAME
---------- --------------------
     97834 ORION
     91232 ADAM
```

f) SQL> SELECT * FROM employee WHERE empname LIKE 'A%';

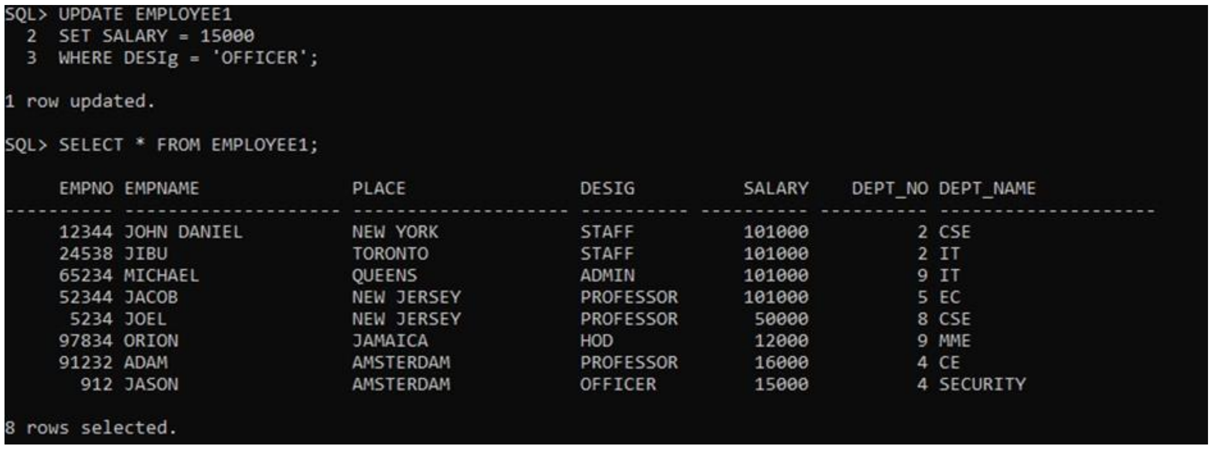


```
SQL> SELECT * FROM EMPLOYEE1
  2  WHERE EMPNAME LIKE 'A%';

     EMPNO EMPNAME              PLACE                DESIG          SALARY    DEPT_NO DEPT_NAME
---------- -------------------- -------------------- ---------- ---------- ---------- --------------------
     91232 ADAM                 AMSTERDAM            PROFESSOR       16000          4 CE

SQL>
```

g) SQL> UPDATE employee

SET salary=15000

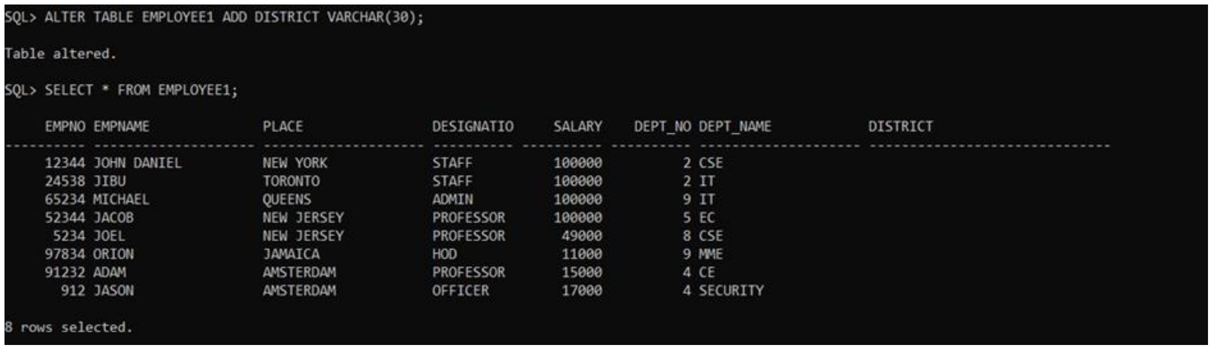
WHERE desig = 'Officer';



```
SQL> UPDATE EMPLOYEE1
  2  SET SALARY = 15000
  3  WHERE DESIg = 'OFFICER';

1 row updated.

SQL> SELECT * FROM EMPLOYEE1;

     EMPNO EMPNAME              PLACE                DESIG          SALARY    DEPT_NO DEPT_NAME
---------- -------------------- -------------------- ---------- ---------- ---------- --------------------
     12344 JOHN DANIEL          NEW YORK             STAFF          101000          2 CSE
     24538 JIBU                 TORONTO              STAFF          101000          2 IT
     65234 MICHAEL              QUEENS               ADMIN          101000          9 IT
     52344 JACOB                NEW JERSEY           PROFESSOR      101000          5 EC
      5234 JOEL                 NEW JERSEY           PROFESSOR       50000          8 CSE
     97834 ORION                JAMAICA              HOD             12000          9 MME
     91232 ADAM                 AMSTERDAM            PROFESSOR       16000          4 CE
       912 JASON                AMSTERDAM            OFFICER         15000          4 SECURITY

8 rows selected.
```

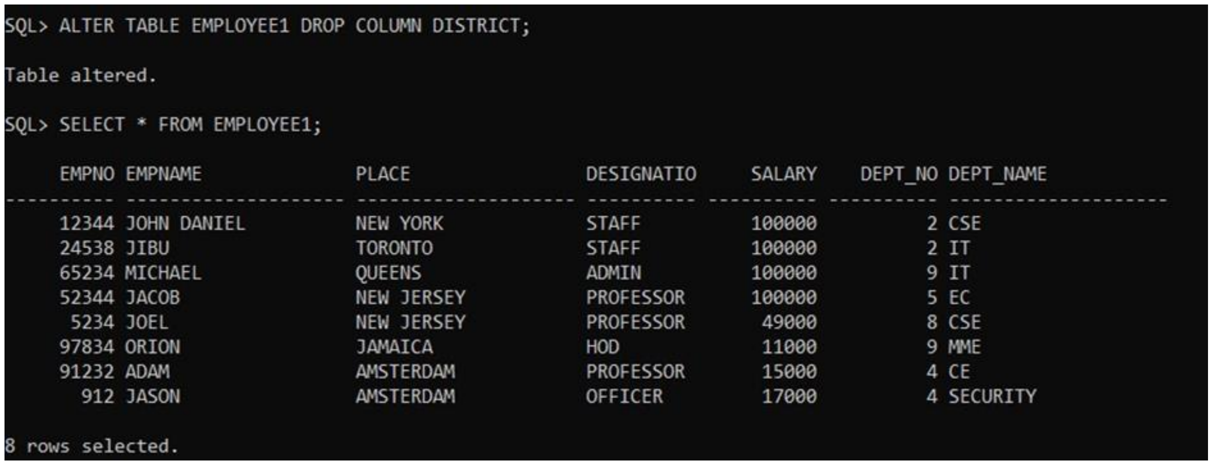
h) SQL> ALTER TABLE employee

ADD district varchar(20);



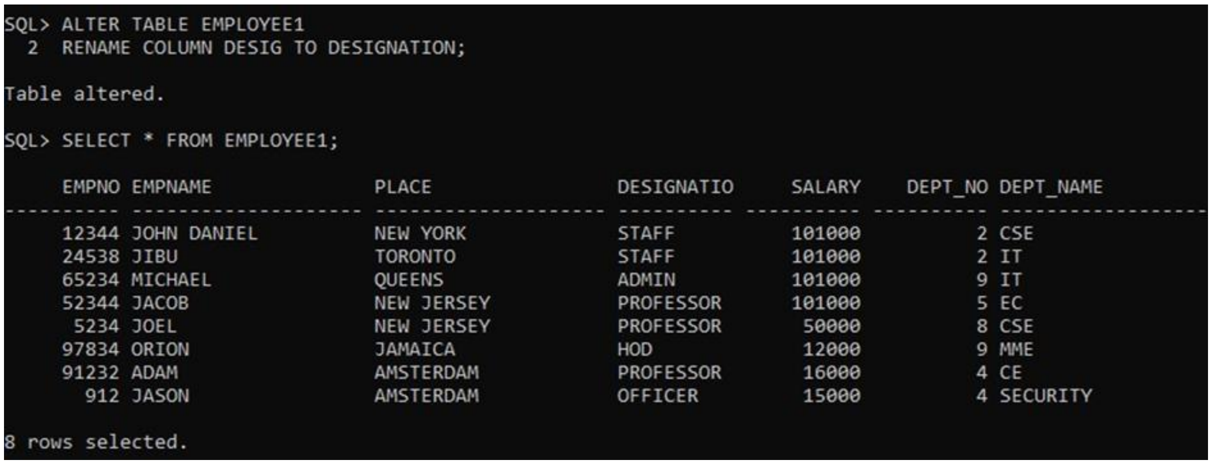
```
SQL> ALTER TABLE EMPLOYEE1 ADD DISTRICT VARCHAR(30);

Table altered.

SQL> SELECT * FROM EMPLOYEE1;

     EMPNO EMPNAME              PLACE                DESIGNATIO     SALARY    DEPT_NO DEPT_NAME            DISTRICT
---------- -------------------- -------------------- ---------- ---------- ---------- -------------------- ------------------------------
     12344 JOHN DANIEL          NEW YORK             STAFF          100000          2 CSE
     24538 JIBU                 TORONTO              STAFF          100000          2 IT
     65234 MICHAEL              QUEENS               ADMIN          100000          9 IT
     52344 JACOB                NEW JERSEY           PROFESSOR      100000          5 EC
      5234 JOEL                 NEW JERSEY           PROFESSOR       49000          8 CSE
     97834 ORION                JAMAICA              HOD             11000          9 MME
     91232 ADAM                 AMSTERDAM            PROFESSOR       15000          4 CE
       912 JASON                AMSTERDAM            OFFICER         17000          4 SECURITY

8 rows selected.
```

i) SQL> ALTER TABLE employee
DROP COLUMN district;



```
SQL> ALTER TABLE EMPLOYEE1 DROP COLUMN DISTRICT;

Table altered.

SQL> SELECT * FROM EMPLOYEE1;

     EMPNO EMPNAME              PLACE                DESIGNATIO     SALARY    DEPT_NO DEPT_NAME
---------- -------------------- -------------------- ---------- ---------- ---------- --------------------
     12344 JOHN DANIEL          NEW YORK             STAFF          100000          2 CSE
     24538 JIBU                 TORONTO              STAFF          100000          2 IT
     65234 MICHAEL              QUEENS               ADMIN          100000          9 IT
     52344 JACOB                NEW JERSEY           PROFESSOR      100000          5 EC
      5234 JOEL                 NEW JERSEY           PROFESSOR       49000          8 CSE
     97834 ORION                JAMAICA              HOD             11000          9 MME
     91232 ADAM                 AMSTERDAM            PROFESSOR       15000          4 CE
       912 JASON                AMSTERDAM            OFFICER         17000          4 SECURITY

8 rows selected.
```

j) SQL> ALTER TABLE employee
RENAME COLUMN desig TO designation;



```
SQL> ALTER TABLE EMPLOYEE1
  2  RENAME COLUMN DESIG TO DESIGNATION;

Table altered.

SQL> SELECT * FROM EMPLOYEE1;

     EMPNO EMPNAME              PLACE                DESIGNATIO     SALARY    DEPT_NO DEPT_NAME
---------- -------------------- -------------------- ---------- ---------- ---------- --------------------
     12344 JOHN DANIEL          NEW YORK             STAFF          101000          2 CSE
     24538 JIBU                 TORONTO              STAFF          101000          2 IT
     65234 MICHAEL              QUEENS               ADMIN          101000          9 IT
     52344 JACOB                NEW JERSEY           PROFESSOR      101000          5 EC
      5234 JOEL                 NEW JERSEY           PROFESSOR       50000          8 CSE
     97834 ORION                JAMAICA              HOD             12000          9 MME
     91232 ADAM                 AMSTERDAM            PROFESSOR       16000          4 CE
       912 JASON                AMSTERDAM            OFFICER         15000          4 SECURITY

8 rows selected.
```

RESULT

The SQL Queries are executed and output is obtained.

Exp. no: 3 01-12-2022

CUSTOMER DATABASE

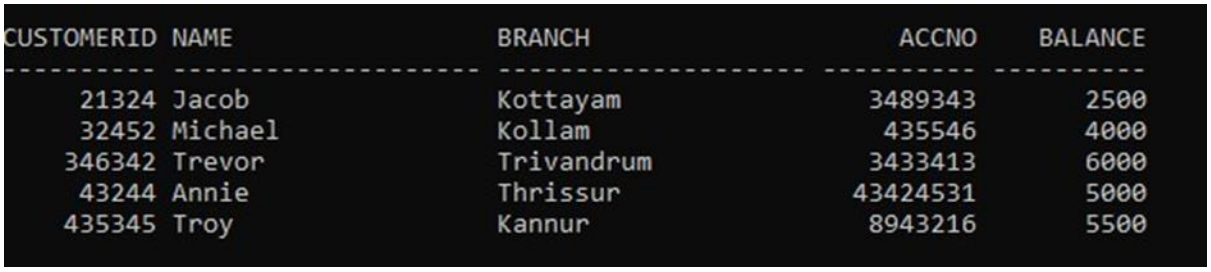
AIM

Create a table customer with the following fields : customerid, name, branch, accno, balance. Customerid is the primary key. In all other fields, we cannot enter null value. The balance should not be less than 500.

a. Find out the details of all customers whose balance is between 2000 and 3000.
b. Show all branches of the bank (duplicates eliminated).
c. Find out the details of all customers whose branch is kottayam and balance>5000.
d. Show the details of all customers whose name start with A.
e. Retrieve the branch name values as city.
f. Find the total balance of the bank.
g. Find the average balance of the bank.
h. Find the max value for balance.
i. Find the min balance of the bank.
j. Count number of records in the table.
k. Modify the size of name in the table to 50
l. Add a new column address to the table with data type varchar(10) and insert values into it.

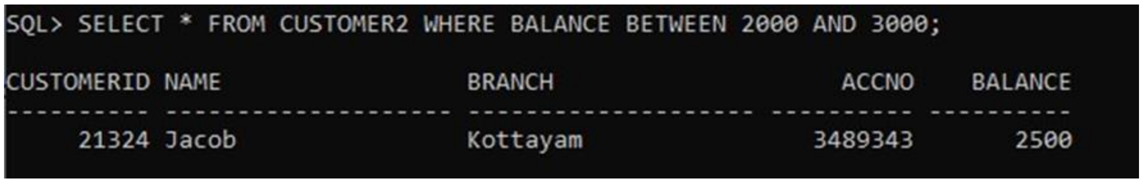
QUERY AND OUTPUT:

```
SQL>   create table customer(
          customerid integer PRIMARY KEY,
          name varchar(20) NOT NULL,
          branch varchar(20) NOT NULL,
          accno numeric(5,0) NOT NULL,
          balance numeric(8,2) NOT NULL CHECK(balance>500));
```



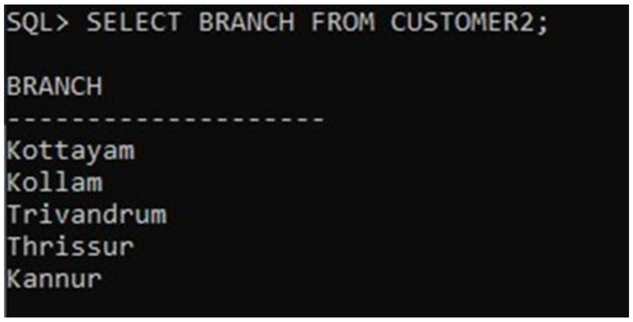
```
CUSTOMERID NAME                 BRANCH                    ACCNO    BALANCE
---------- -------------------- -------------------- ---------- ----------
     21324 Jacob                Kottayam                3489343       2500
     32452 Michael              Kollam                   435546       4000
    346342 Trevor               Trivandrum              3433413       6000
     43244 Annie                Thrissur               43424531       5000
    435345 Troy                 Kannur                  8943216       5500
```

a) SQL> select * from customer where balance between 2000 and 3000;



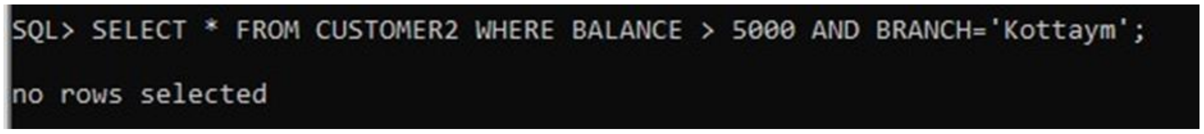
```
SQL> SELECT * FROM CUSTOMER2 WHERE BALANCE BETWEEN 2000 AND 3000;

CUSTOMERID NAME                 BRANCH                    ACCNO    BALANCE
---------- -------------------- -------------------- ---------- ----------
     21324 Jacob                Kottayam                3489343       2500
```

b) SQL> select distinct branch from customer;



```
SQL> SELECT BRANCH FROM CUSTOMER2;

BRANCH
--------------------
Kottayam
Kollam
Trivandrum
Thrissur
Kannur
```

c) SQL> select * from customer where branch='kottayam' and balance>5000;



```
SQL> SELECT * FROM CUSTOMER2 WHERE BALANCE > 5000 AND BRANCH='Kottaym';

no rows selected
```

d) SQL> select * from customer where name like 'a%';

```
CUSTOMERID       NAME              BRANCH      ACCNO    BALANCE
------------------ -------------------- -------------------- ------------ -------------
43244                 Annie             Thrissur         43424531       5000
```

e) SQL> select branch as city from customer;

```
CITIES
--------------------
Kottayam
Kollam
Trivandrum
Thrissur
Kannur
```

f) SQL> select sum(balance) from customer;

```
TOTAL_BALANCE
-------------------------
23000
```

g) SQL> select avg(balance) from customer;

```
AVG_BALANCE
----------------------
4600
```

h) SQL> select max(balance) from customer;

```
MAX_VALUE
-------------------
6000
```

i) SQL> select min(balance) from customer;

```
MIN_VALUE
----------
2500
```

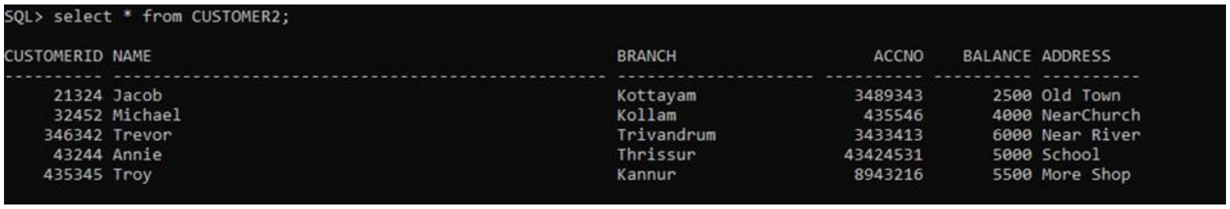
j) SQL> select count(*) from customer;

```
NAME
----------
5
```

k) SQL> alter table customer
            modify name varchar(50);

```
Table altered.
```

l) SQL> ALTER TABLE CUSTOMER2 ADD ADDRESS VARCHAR(10);
SQL> UPDATE CUSTOMER2 SET ADDRESS='Old Town' WHERE NAME='Jacob';
SQL> UPDATE CUSTOMER2 SET ADDRESS='NearChurch' WHERE NAME='Michael';
SQL> UPDATE CUSTOMER2 SET ADDRESS='Near River' WHERE NAME='Trevor';
SQL> UPDATE CUSTOMER2 SET ADDRESS='School' WHERE NAME='Annie';
SQL> UPDATE CUSTOMER2 SET ADDRESS='More Shop' WHERE NAME='Troy';



```
SQL> select * from CUSTOMER2;

CUSTOMERID NAME                                               BRANCH                   ACCNO    BALANCE ADDRESS
---------- -------------------------------------------------- -------------------- ---------- ---------- ----------
     21324 Jacob                                              Kottayam               3489343       2500 Old Town
     32452 Michael                                            Kollam                  435546       4000 NearChurch
    346342 Trevor                                             Trivandrum             3433413       6000 Near River
     43244 Annie                                              Thrissur              43424531       5000 School
    435345 Troy                                               Kannur                 8943216       5500 More Shop
```

<u>RESULT</u>

The SQL Queries are executed and output is obtained.

Exp. no: 4 01-12-2022

STUDENT DATABASE

AIM:

Create a table student with following fields roll no int (primary key), Name char (20) not null (first letter as either B,S E,P), sex char (1) accept only m or f, dob date not null, course (values must be MCA, CSE ME), sem(values must be S3, S4), Date_of_Join.
Create second table marks with following data Mid in (primary key), roll no int (foreign key) referencing student tables). Sub_code char (5) not null and marks int not null (>=0 &<=100). Insert the data into these tables.

a. List the name of students joined in mca after 10-10-1990.
b. List the name of students who are not in CS department.
c. List the names of students whose names start with ‘E’ and ‘P as 3rd character
d. List all marks of the students Robert from MCA.
e. List all roll no from two table (avoid duplicate roll no).
f. List all roll no which is common in both tables.
g. List name from student table and all marks from marks of roll no 23 in student table.
h. List the roll no and total marks of each roll no from mark table.
i. Display name and roll no of students, where marks are entered in marks table.
j. Display the name, roll no, sex, dob, sub_code and mark of highest subject mark.
k. List the student name and Date of Join in format dd/mm/yy
l. List all students joined during the year 1998
m. List the minimum mark of various students in various having minimum mark greater than 60.
n. List all the students in the college other than CS Department
o. Count the number of students in each department whose mark in greater than 60

QUERY AND OUTPUT

```
SQL>  CREATE TABLE STUDENT (
        ROLLNO INT PRIMARY KEY,
        NAME CHAR(20) NOT NULL
        CHECK(NAME LIKE 'B%' OR NAME LIKE 'S%' OR NAME LIKE 'E%' OR NAME LIKE 'P%'),
        SEX CHAR(1) CHECK(SEX ='M' OR SEX='F'),
        DOB DATE NOT NULL,
        COURSE VARCHAR(20) CHECK(COURSE='MCA' OR COURSE='CSE' OR COURSE ='ME'),
        SEM VARCHAR(20) CHECK(SEM = 'S3' OR SEM='S4'),
        DATE_OF_JOIN DATE);

        rollno  name     sex       dob         course     sem     date_of_join
        ------- -------- -----  -------------- ----------- ---------- -----------------
        23      Travis   M    1967-05-10    cse        s4      1992-06-07
        24      Goku     M    1970-05-10    mca        s3      1998-05-10
        25      Prakash  M    1972-07-23    cse        s4      1987-09-21
        26      Jessica  F    1989-08-17    me         s3      2000-07-13

SQL>  CREATE TABLE MARKS(
        MID INT PRIMARY KEY,
        ROLLNO INT NOT NULL,
        SUB_CODE CHAR(5) NOT NULL,
        MARK INT CHECK(MARK BETWEEN 0 AND 100),
        CONSTRAINT FK FOREIGN KEY (ROLLNO) REFERENCES STUDENT(ROLLNO));

        mid     rollno  sub_code   mark
        -----   --------- ----------- ---------
        1       12      C01        70
        2       13      M03        65
        3       14      C07        90
```

a) SELECT NAME FROM STUDENT WHERE DATE_OF_JOIN > '1990-10-10' AND COURSE='MCA';

```
name
--------
Travis
Goku
Jessica
```

b) SELECT NAME FROM STUDENT WHERE COURSE<>'CSE';

```
Name
-------
Goku
Jessica
```

c) SELECT NAME FROM STUDENT WHERE NAME LIKE 'E_p%';

```
name
---------
Prakash
```

d) SELECT M.MARK FROM STUDENT S, MARKS M WHERE S.ROLLNO=M.ROLLNO AND S.NAME='Saurav' AND S.COURSE='MCA';

```
mark
-----
65
```

e) SELECT DISTINCT(STUDENT.ROLLNO) FROM STUDENT,MARKS;

```
rollno
-------
23
24
25
26
```

f) SELECT STUDENT.ROLLNO FROM STUDENT,MARKS WHERE STUDENT.ROLLNO=MARKS.ROLLNO;

```
rollno
-------
23
24
25
```

g) SELECT STUDENT.NAME, MARKS.MARK FROM STUDENT,MARKS WHERE STUDENT.ROLLNO=MARKS.ROLLNO AND MARKS.ROLLNO=23;

```
name        mark
---------   --------
Travis      70
```

h) SELECT ROLLNO,MARK AS TOTAL_MARKS FROM MARKS;

```
rollno      total_marks
-------     ---------------
23           70
24           65
25          90
```

i) SELECT STUDENT.NAME,MARKS.ROLLNO FROM STUDENT,MARKS WHERE STUDENT.ROLLNO= MARKS.ROLLNO;

```
name        rollno
----------- ---------
Travis      23
Goku        24
Prakash     25
```

j) SELECT NAME,S.ROLLNO,SEX,DOB,SUB_CODE,MARK FROM STUDENT S,MARKS M WHERE S.ROLLNO=M.ROLLNO AND MARK=(SELECT MAX(MARK) FROM MARKS);

```
name        rollno  sex    dob      sub_code   mark
--------- --------- ----- -------------- ------------ -------
Prakash    25        M   1972-07-23    C07      90
```

k) SELECT TO_CHAR(DATE_OF_JOIN,'DD-MM-YYYY') AS DATE_OF_JOIN FROM STUDENT;

```
date_of_join
---------------
1992-06-07
```

```
1998-05-10
1987-09-21
2000-07-13
```

l) SELECT * FROM STUDENT WHERE TO_CHAR(DATE_OF_JOIN,'YYYY')='1998';

```
rollno  name    sex      dob          course   sem     date_of_join
------- -------- ------ -------------- ---------- ------- -------------------
24      Goku     M   1970-05-10     mca      s3      1998-05-10
```

m) SELECT SUB_CODE,MIN(MARK) FROM MARKS WHERE MARK>60 GROUP BY SUB_CODE;

```
sub_code   min(mark)
----------- --------------
C01              70
M03              65
C07              90
```

n) SELECT * FROM STUDENT WHERE COURSE <> 'CSE';

```
rollno  name     sex           dob          course  sem     date_of_join
-------- --------- ------ ----------------- --------- ------- -----------------
24      Goku     M      1997-05-10     mca       s3      1998-05-10
26     Jessica    F      1989-08-17      me      s3      2000-07-13
```

o) SELECT SUB_CODE,COUNT(*) FROM MARKS,STUDENT WHERE MARK>60 GROUP BY SUB_CODE;

```
sub_code  count(*)
----------- ----------
C07            4
M03            4
C01            4
```

RESULT

The SQL Queries are executed and output is obtained.

Exp. no: 5 08-12-2022

DATA MODELER FAMILIARIZATION

AIM

Use Data Modeler to design a Database schema for a customer-sale scenario as shown below.

customer(cust_id,cust_name) primary key(cust_id).

Item(item_id,item_name,price) primary key(item_id)

Sale(bill_no,bill_date,cust_id,item_id,qty_sold) primary key(bill_no),foreign key(cust_id),foreign key(item_id).

Find:

a. Create the tables using data modeler. Insert around 10 records in each of the tables.

b. List all the bills for the current date with the customer name and item_no.

c. List the total bill detail with the quantity sold, price of the item and final amount.

d. List the details of the customer who have brought a product which has price>200.

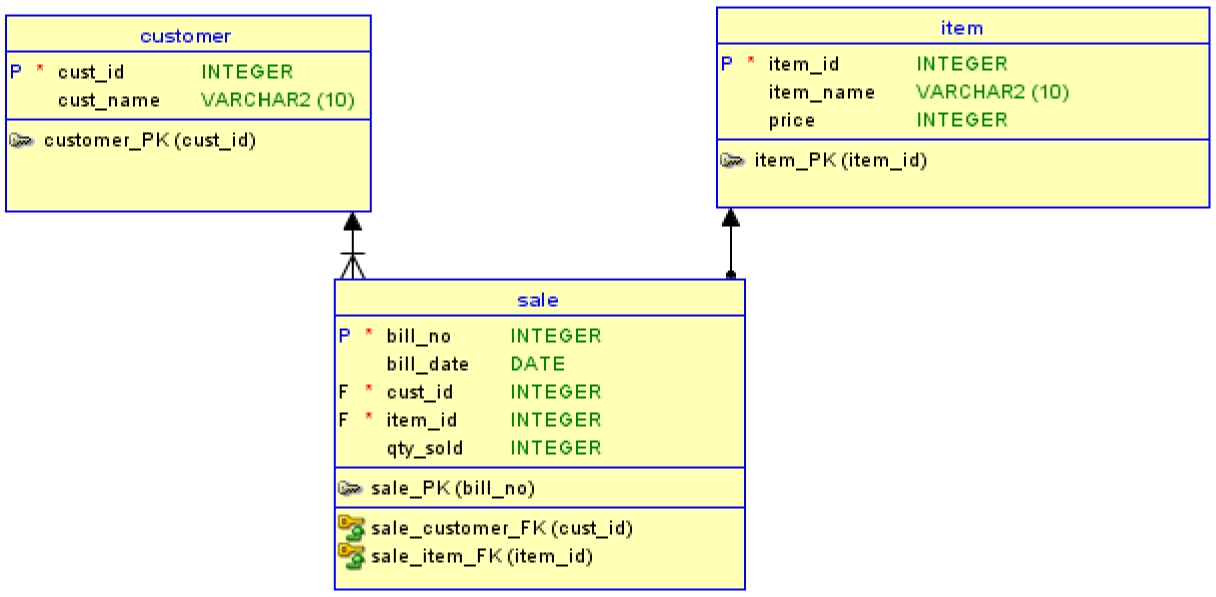
e. Give a count of how many product have been brought by each customer.

f. Give a list of product brought by a customer having cust_id 5.

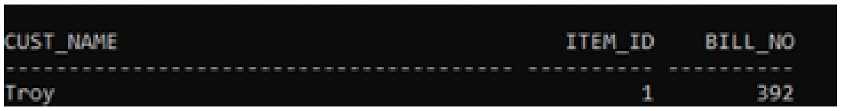
g. List the item details which are sold as of today.

QUERY AND OUTPUT

a)

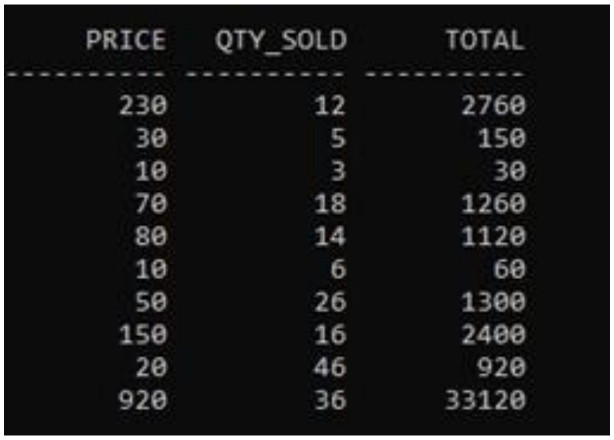


b) SELECT B.BILL_NO,B.BILL_DATE,C.CUST_NAME,I.ITEM_ID FROM SALE B, CUSTOMER C, ITEM I WHERE B.CUST_ID=C.CUST_ID AND B.ITEM_ID=I.ITEM_ID AND BILL_DATE=TO_CHAR(sysdate);



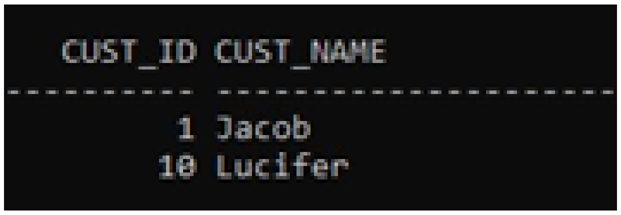
```
CUST_NAME                                ITEM_ID    BILL_NO
---------------------------------------- ---------- ----------
Troy                                              1        392
```

c) SELECT I.ITEM_NAME,I.PRICE,B.QTY_SOLD,B.QTY_SOLD*I.PRICE FINAL_AMOUNT FROM SALE B, ITEM I WHERE B.ITEM_ID=I.I



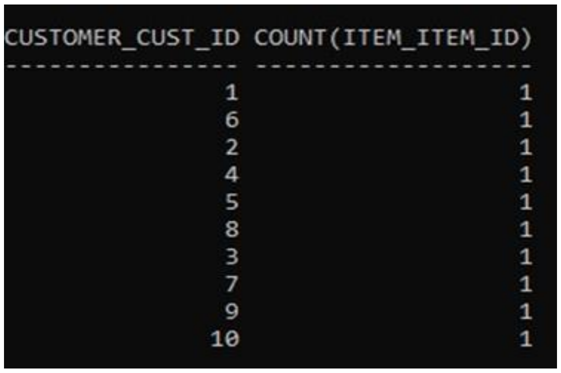
```
     PRICE   QTY_SOLD      TOTAL
---------- ---------- ----------
       230         12       2760
        30          5        150
        10          3         30
        70         18       1260
        80         14       1120
        10          6         60
        50         26       1300
       150         16       2400
        20         46        920
       920         36      33120
```

d) SELECT DISTINCT(C.CUST_NAME) FROM CUSTOMER C, SALE B, ITEM I WHERE B.ITEM_ID=I.ITEM_ID AND B.CUST_ID=C.CUST_ID AND I.PRICE>200;



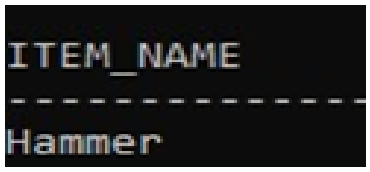
```
   CUST_ID CUST_NAME
---------- --------------------
         1 Jacob
        10 Lucifer
```

e) SELECT C.CUST_NAME,COUNT(I.ITEM_ID) FROM ITEM I, CUSTOMER C, SALE B WHERE B.ITEM_ID=I.ITEM_ID AND B.CUST_ID=C.CUST_ID GROUP BY CUST_NAME;



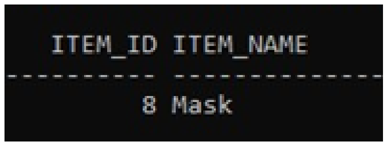
```
CUSTOMER_CUST_ID COUNT(ITEM_ITEM_ID)
---------------- -------------------
               1                   1
               6                   1
               2                   1
               4                   1
               5                   1
               8                   1
               3                   1
               7                   1
               9                   1
              10                   1
```

f) SELECT I.ITEM_NAME FROM ITEM I, CUSTOMER C, SALE B WHERE B.ITEM_ID=I.ITEM_ID AND B.CUST_ID=C.CUST_ID AND C.CUST_ID=5;



```
ITEM_NAME
-------------
Hammer
```

g) SELECT B.BILL_NO,B.BILL_DATE,C.CUST_NAME,I.ITEM_ID FROM SALE B, CUSTOMER C, ITEM I WHERE B.CUST_ID=C.CUST_ID AND B.ITEM_ID=I.ITEM_ID AND BILL_DATE=TO_CHAR(sysdate);



```
   ITEM_ID ITEM_NAME
---------- ------------
         8 Mask
```

<u>RESULT</u>

The SQL Queries are executed and output is obtained.

Exp. no: 6 20-12-2022

VIEW CREATION

AIM

To create a table with following fields roll number(primary key), name, age, place.

a. Create a view which contains only roll number and name.
b. Describe the view.
c. Insert a value into the view and list the view and table.
d. Update a value in the view and list the view and table.
e. Delete a value in the table and list the view and table.
f. Create another view which contains name and age column and check whether the above operations are possible in this view.

QUERY

```
SQL> create table stud(rno number(3) primary key, name char(10), age number(3),place char(10));

        RNO      NAME      AGE       PLACE
        ---------- ---------- ---------- ---------------------
        1          Norah          17       Sharjah
        2          Alex           18        Dubai
        3          Emily          17       Dubai
```

a)
```
SQL> create view stud1 as select rno, name from stud
View created.
```

b)
```
SQL> desc stud1;
Name                              Null?          Type
--------------------------------- -------------- ----------------------------
RNO                               NOT NULL NUMBER(3)
NAME                                        CHAR(10)
```

c)
```
SQL> insert into stud1 values(4,'Bridget');
1 row created.
SQL> select *from stud1;
```

```
  RNO       NAME
  ---------- ----------
    1       Norah
    2       Alex
    3       Emily
    4       Bridget
```

d) SQL> update stud1 set name='Serena' where rno=4;

```
1 row updated.
SQL> select *from stud1;
```

```
 RNO        NAME
 ---------- -----------------
    1       Norah
    2       Alex
    3       Emily
    4       Serena
```

e) SQL> delete from stud1 where rno='1';

```
1 row deleted.
SQL> select *from stud1;
```

```
 RNO        NAME
----------  ----------
    2       Alex
    3       Emily
    4       Serena
```

f) SQL> create view stud2 as select name, age from stud;

```
View created.
```

```
SQL> select *from stud2;
NAME            AGE
---------- ----------
Alex            18
Emily            17
Serena

SQL> insert into stud2 values(‘anu’,22);
        insert into stud2 values(‘anu’,22)
        *
        ERROR at line 1:
        ORA-01400: cannot insert NULL into (“SANJANA”.”STUD”.”RNO”)
```

RESULT

SQL queries are executed and output obtained.

Exp. no: 7 20-12-2022

SEQUENCE CREATION

AIM

To create a table student with the following fields: roll number, name, mark. Create a sequence that will increment value by 1, start with 101, max value up to 200 and comes in a cycle manner.

a) Display the next value of sequence.
b) Display current value of sequence.
c) Alter the sequence by updating increment value by 2.
d) Insert values into table and use sequence to enter values in roll number field.

QUERY

```
SQL> create table studseq(rollno number(6), name char(10), mark number(3));

Table created.
```

a)

```
SQL> create sequence seq1 increment by 1 start with 101 maxvalue 200 minvalue 101 cycle;

Sequence created.
```

b)

```
SQL> select seq1.currval from dual;

CURRVAL
--------------
 101
```

c)

```
SQL> alter sequence seq1 increment by 2;

Sequence altered.

SQL> select seq1.nextval from dual;

NEXTVAL
----------
103
```

d)
```
SQL> insert into studseq values(seq1.nextval, 'Reshma', 76);

1 row created.


SQL> select *from studseq;


ROLLNO   NAME    MARK
------------ ------------ ----------
105          Reshma       76
```

RESULT

SQL queries are executed and output obtained.

Exp. no: 8 10-01-2022

PRODUCT OF TWO NUMBERS

AIM

Write a PL/SQL program to print the product of two numbers.

ALGORITHM

1. Enter number a
2. Enter number b
3. Compute c = a*b
4. Display c
5. End

Query:

```
DECLARE
        a number;
        b number;
        c number;
BEGIN
        a:=&a;
        b:=&b;
        c:=a*b;
        dbms_output.put_line('PRODUCT = '||c);
END;
```

OUTPUT

```
Enter value of a: 5
Enter value of b: 4
PRODUCT=20
```

RESULT

The PL/SQL Program executed and output is obtained.

Exp. no: 9 10-01-2022

LARGEST AND SMALLEST OF THREE NUMBERS

AIM

Write a PL/SQL program to find largest and smallest of three numbers.

ALGORITHM

```
1. Declare variables a, b, c, big, small.
2. Enter the three numbers to a, b, c.
3. If  a >b then
        a. If  a>c then
               Assign big as a.
        b. Otherwise
               Assign big as c.
4. Otherwise
        4.1 If  b>c then
               Assign big as b.
        4.2 Otherwise
               Assign big as c.
5. Print big.
        a. If  a<c then
               Assign small as a.
        b. Otherwise
               Assign small as c.
6. Otherwise
        6.1 If  b<c then
               Assign small as b.
        6.2 Otherwise
               Assign small as c.
7. Print small.
8. End.
```

PROGRAM

```
DECLARE
        a  number;
        b  number;
        c  number;
        big number;
        small number;
BEGIN
```

```
        a:=&a;
        b:=&b;
        c:=&c;
        if a>b and a>c then
                big:=a;
        elsif b>c then
                big:=b;
        else
                big:=c;
        end if;
        dbms_output.put_line('largest is'||big);
        if a<b and a<c then
                small:=a;
        elsif b<c then
                small:=b;
        else
                small:=c;
        end if;
        dbms_output.put_line('smallest is'||small);
END;
```

OUTPUT

```
Enter value for a: 6
Enter value for b: 3
Enter value for c: 17
Largest is 17.
Smallest is 3.
```

RESULT

PL/SQL queries are executed and output obtained.

Exp. no: 10 10-01-2022

AREA OF CIRCLE AND SQUARE

AIM

To write a PL/SQL program to find area of circle and square.

ALGORITHM

1. Enter side of square a
2. Enter radius of circle r
3. Area of square = a*a
4. Area of circle = 3.14*r*r
5. End

PROGRAM

```
DECLARE
        a number;
        r number;
        areas number;
        areac number;
        pi constant number:=3.14;
BEGIN
        dbms_output.put_line('enter side of square: ');
        a:=&a;
        dbms_output.put_line('enter radius of circle: ');
        r:=&r;
        areas :=a*a;
        areac :=pi*r*r;
        dbms_output.put_line('area of square = '||areas);
        dbms_output.put_line('area of circle = '||areac);
END;
```

OUTPUT

```
Enter side of square: 2   Enter radius of circle: 2
Area of square=4    Area of circle=12.56
```

RESULT

PL/SQL queries are executed and output obtained.

Exp No: 11 24-01-2022

SEQUENCE GENERATION

AIM

To write a PL/SQL program to obtain the sequence 1, 4, 9, 16...

ALGORITHM

1. Enter limit n
2. Compute product for each number from 1 to limit
3. Display the sequence
4. End

PROGRAM

```
DECLARE
        n number;
        i number;
        s number;
BEGIN
        dbms_output.put_line('enter limit: ');
        n:=&n;
        for i in 1..n
        loop
        s:=i*i;
        dbms_output.put_line(s);
        end loop;
END;
```

OUTPUT

```
Enter value of n: 3
1
4
9
```

RESULT

PL/SQL queries are executed and output obtained.

Exp. no: 12 24-01-2022

STRING REVERSAL

AIM

To write a PL/SQL program to reverse a given string.

ALGORITHM

1. Read the string.
2. Find the length of the string a
3. Enter the limit from 1 to a and repeat step 4.
4. Find the substring of given string
5. Print the reversed string
6. End

PROGRAM

```
DECLARE
        s1 varchar(10);
        s2 varchar(10);
        a varchar(10);
BEGIN
        s1:='&s1';
        a:=length(s1);
        for i in reverse 1..a
        loop
        s2:=s2||substr(s1,i,1);
        end loop;
        dbms_output.put_line(s2);
END;
```

OUTPUT

```
Enter string: reshma
Amhser
```

RESULT

PL/SQL queries are executed and output obtained.

Exp No: 13 24-01-2022

ARMSTRONG NUMBER

AIM

write a PL/SQL program to check given number is Armstrong or not.

ALGORITHM:

```
1. Enter the number n.
2. While n greater than 0 perform
       i=n mod 10
       s=s+(i*i*i)
       n=n/10
3. if n=s then print 'armstrong no else print 'not armstrong'
4. end
```

PROGRAM

```
DECLARE
        n number;
        m number;
        i number:=1;
        s number:=0;
BEGIN
        n:=&n;
        m:=n;
        while(i>0)
        loop
                i:=n mod(10);
                s:=s+(i*i*i);
                n:=n/10;
                n:=trunc(n);
        end loop;
        dbms_output.put_line(s);
                if(s=m) then
                        dbms_output.put_line('AMSTRONG NUMBER!');
                else
                        dbms_output.put_line('NOT AMSTRONG NUMBER!');
                end if;
END;
```

RESULT

The PL/SQL Program executed and output is obtain.

Exp No: 14 28-01-2022

SALARY INCREMENT

AIM

An employee is given 25% increase in salary if salary is above Rs. 25000 and 20% increase in salary if salary is above Rs. 30000. Write a PL/SQL program to calculate new salary.

ALGORITHM

1. Enter salary s
2. If s greater then 25000, increment by 25%
3. If s greater than 30000, increment by 20%
4. Else no change in salary
5. Display new salary
6. End

PROGRAM

```
DECLARE
        s number(5);
BEGIN
        dbms_output.put_line(‘enter salary: ‘);
        s:=&s;
        if(s>25000) then
                s:=s+s*0.25;
        elsif(s>30000) then
                s:=s+s*0.2;
        else
                dbms_output.put_line(‘no change in salary’);
                s:=s;
        end if;
        dbms_output.put_line(‘new salary= ‘||s);
END;
```

OUTPUT

```
Enter salary: 15000
No change in salary
New salary =15000
```

RESULT

PL/SQL queries are executed and output obtained.

Exp No: 15 28-01-2022

ODD OR EVEN NUMBER

AIM

To write a PL/SQL program to insert first 15 odd numbers into a table ODD and first 15 even numbers into a table EVEN.

ALGORITHM

```
1. Create table even with a field even_number
2. Create table odd with a field odd_number
3. For count i from 1 to 30 repeat steps 4 to 5
4. If i mod 2 is 0, insert i into table even
5. Else insert i into table odd
6. End
```

PROGRAM

```
SQL>Create table even (even number(6));

SQL>Create table odd (odd number(6));


DECLARE

BEGIN

        for i in 1..30

        loop

                if(i mod 2=0) then

                        insert into even values(i);

                else

                        insert into odd values(i);

                end if;

        end loop;

END;
```

OUTPUT

```
EVEN

---------

2

4

6

8

10

12

14

16

18

20

22

24

26

28

30


ODD

---------

1

3

5

7

9

11

13

15

17

19
```

```
21

23

25

27

29
```

<u>RESULT</u>

PL/SQL queries are executed and output obtained.

Exp. no: 16 31-01-2022

IMPLICIT CURSOR – SALARY UPDATION

AIM

Write a PL/SQL program to update salary of Sindhu by 30% if she is earning salary>10000, otherwise update by 20% if salary>8000, otherwise update by 10%. (Table name: income(ename, salary).

ALGORITHM

1. Create table income with fields ename and salary and insert values into it
2. Select ename into ename and salary into sal for employee Sindhu
3. If sal greater than 10000, increment by 30%
4. If sal greater than 8000, increment by 20%
5. Else increment by 10%
6. Update the values in table income
7. End

PROGRAM

```
SQL>create table income (ename char(10), salary number(5));

Table created.

SQL>insert into income values (‘Sindhu’,10000);

1 row created.

SQL>insert into income values(‘Indhu’,18000);

1 row created.

DECLARE

        ename income.ename %type;

        sal income.salary %type;

BEGIN

        Select salary into sal from income where ename=’Sindhu’;

        if(sal>10000) then

                sal:=sal+sal*0.3;

        elsif(sal>8000) then

                sal:=sal+sal*0.2;
        else
                sal:=sal+sal*0.1;
        end if;
        update income set salary=sal where ename =’Sindhu’;
END;
```

OUTPUT

```
ENAME        SALARY
---------------------------
Sindhu       10000
Indhu        18000

ENAME        SALARY
-----------------------------
Sindhu       263644
Indhu        18000
```

RESULT

PL/SQL queries are executed and output obtained.

Exp. no: 17 31-01-2022

IMPLICIT CURSOR – SALARY CHECKING & ROLL BACKING

AIM

To write a PL/SQL program to update salary of all employees by 20%. If total salary>100000 then rollback else commit.

ALGORITHM

1. Create table income with fields ename and salary and insert values
2. Fetch salary into sal
3. Increment sal by 20% for all employees
4. Update the values in table
5. If sum of salaries is greater than 100000 then perform rollback, else commit
6. End

Ans:

```
SQL>create table income (ename char(10), salary number(5));

Table created.

SQL>insert into income values (‘Sindhu’,10000);

1 row created.

SQL>insert into income values (‘Indhu’,10000);

1 row created.

SQL>insert into income values (‘shintu’,10000);

1 row created.

SQL>insert into income values (‘mintu’,10000);

1 row created.
```

```
DECLARE
        sal income.salary %type;
        s varchar(10);
BEGIN
        sal:=sal+sal*0.2;
        update income set salary=sal;
        select sum(sal) into s from income;
        if(s>100000) then
                rollback;
        else
                commit;
        end if;
END;
```

OUTPUT

```
ENAME       SALARY
-----------------------------
Sindhu      10000
Indhu       10000
Shintu      10000
Mintu       10000

ENAME       SALARY
----------------------------
Sindhu      10000
Indhu       10000
Shintu      10000
Mintu       10000
```

RESULT

The PL/SQL Program executed and output is obtained.

Exp. no: 18 31-01-2022

IMPLICIT CURSOR – EXCEPTION HANDLING

AIM

To create a table student with fields - rollno, stud_name, sessionals, univ_mark. If sessionals + univ_mark >150, raise an error message. Also handle all the possible exceptions.

ALGORITHM

1. Create table studp with fields rno, stdname, sessionals and unimark
2. Declare exception e1
3. If unimark + sessionals greater than 150 then raise exception e1- mark exceeds 150
4. End

PROGRAM

```
SQL>Create table studp( rno number(3), stdname char(10),  sessionals number(10), unimark number(10));

Table created.

SQL>insert into studp values(1, Jem, 48, 90);

1 row created.

SQL> select *from studp;

      RNO       STUDNAME   SESSIONALS    UNIMARK
---------- ---------- ---------- ---------------------------------------
        1       Jem                  48            90
        2       Scout                35            65
        3       Jo                   41            78
        4       Neville              50            101

DECLARE
          s studp.sessionals%type;
          u studp.unimark%type;
          e1 exception;
BEGIN
          select sessionals,unimark into s,u from studp where rno=&rno;
          if (s+u > 150) then
                    raise e1;
          end if;
EXCEPTION
          when e1 then
          dbms_output.put_line('mark exceeds 150!');
          when no_data_found then
          dbms_output.put_line('roll no. not found!');
          when others then
          dbms_output.put_line('invalid input');
END;
```

OUTPUT

```
Enter value for rno: 2
PL/SQL procedure successfully completed.
Enter value for rno: 4
mark exceeds 150!
PL/SQL procedure successfully completed.
```

RESULT

PL/SQL queries are executed and output obtained.

Exp. no: 19 31-01-2022

IMPLICIT CURSOR – ELECTRICITY BILL

AIM

Write a PL/SQL program to accept the customer_no and print the electricity bill for the same. The charge is calculated as follows:

UNITS CONSUMED	CHARGE
<20	Nil
20-100	50ps per unit
101-300	75ps per unit
301-500	150ps per unit
>500	225ps per unit

Print the electricity bill in the form:

ELECTRICITY BILL

CONSUMER NO
PRESENT READING
PAST READING
UNITS TAKEN
CHARGE

ALGORITHM

1. Create table ebill with fields cno, present, past, units, charge and insert values cno and past
2. Enter customer number cno and present reading pre
3. Compute units = pre-past
4. Compute charge as follows:
    a. If units less than 20, charge=0
    b. If units from 20-100, charge =50/unit
    c. If units from 101-300, charge= 75/unit
    d. If units from 301-500, charge =150/unit
    e. If unit greater than 500, charge =225/unit
5. Display the bill
6. End

PROGRAM

```
SQL> create table ebill(cno number(3), present number(5), past number(5), units number(5), charge
number(5));

Table created.

SQL> insert into ebill values(1,NULL, 20,NULL,NULL);

1 row created.

SQL> insert into ebill values(2,NULL, 10,NULL,NULL);

1 row created.
```

```
SQL> select *from ebill;
CNO    PRESENT       PAST      UNITS     CHARGE
---------- ---------- ---------- ---------- ----------
        1                      20
        2                      10
DECLARE
   c ebill.cno%type;
   p ebill.past%type;
   ch ebill.charge%type;
   pr ebill.present%type;
   u ebill.units%type;
BEGIN
         select cno,past into c,p from ebill where cno=&cno;
         pr:=&pr;
         u:=pr-p;
         if(u<20) then
         ch:=0;
         elsif(u between 20 and 100) then
         ch:=50*u;
         elsif(u between 101 and 300) then
         ch:=50*100+(u-100)*75;
         elsif(u between 301 and 500) then
         ch:=50*100+200*75+150*(u-300);
         else
         ch:=50*100+200*75+150*200+225*(u-500);
         end if;
         dbms_output.put_line('          ELECTRICITY BILL');
         dbms_output.put_line('CONSUMER NO'||c);
         dbms_output.put_line('PRESENT READING'||pr);
         dbms_output.put_line('PAST READING'||p);
         dbms_output.put_line('UNITS TAKEN'||u);
         dbms_output.put_line('CHARGE'||ch);
         update ebill set charge=ch, units=u, present=pr  where cno=c;
EXCEPTION
         when no_data_found then
         dbms_output.put_line('cno. not found!');
END;
```

OUTPUT

```
Enter value for cno: 1
Enter value for pr: 150
        ELECTRICITY BILL
CONSUMER NO =1
PRESENT READING =150
PAST READING =20
UNITS TAKEN =130
CHARGE =7250
PL/SQL procedure successfully completed.
Enter value for cno: 2
Enter value for pr: 500
        ELECTRICITY BILL
CONSUMER NO =2
PRESENT READING =500
PAST READING =10
UNITS TAKEN =490
CHARGE =48500
PL/SQL procedure successfully completed.
SQL> select *from  ebill;
       CNO    PRESENT       PAST      UNITS     CHARGE
---------- ---------- ---------- ---------- -------------------------
         1        150         20        130       7250
         2        500         10        490       48500
```

RESULT

PL/SQL queries are executed and output obtained.

Exp. no: 20 14-02-2022

EXPLICIT CURSOR – MESS FEE INCREMENT

AIM

Create a hostel mess database with fields (stud_no, name, messfee, veg/nonveg). Write a PL/SQL program to increase the mess fee of vegetarians by 10% and non vegetarians by 20%. Also create tables vegetarian and non_vegetarian which includes fields: stud_no, name, raise_in_fee and date on which raise was given. Insert values into these tables through PL/SQL program.

ALGORITHM

```
1.  START
2.  Create table mess, veg and nonveg
3.  Initialise no, nm, fee, tp, r
4.  create cursor c1
5.  fetch c1 into no, nm, fee, tp
6.  if tp = 'v' then
7.  r = fee * 0.10, fee = fee + r
8.  insert it into table veg
9.  else do
10. r = fee * 0.20, fee = fee + r
11. insert it into table nonveg
12. STOP
```

PROGRAM

```
SQL> create table mess(stud_no number, name varchar(10), messfee number, vegnon varchar(10));
Table created.
SQL> create table veg(stud_no number, name varchar(10),raise_in_fee number, rsdate date);
SQL> create table nonveg(stud_no number, name varchar(10),raise_in_fee number, rsdate date);
SQL> insert into mess values(&stud_no,'&name',&messfee,'&vegnon');
Enter value for stud_no: 1
Enter value for name: Sohesh
Enter value for messfee: 2600
Enter value for vegnon: non
old   1: insert into student values(&stud_no,'&name',&messfee,'&vegnon')
new   1: insert into student values(1,'Sohesh',2600,'veg')
1 row created.
```

```
SQL> /
Enter value for stud_no: 2
Enter value for name: Vimal
Enter value for messfee: 2500
Enter value for vegnon: veg
old   1: insert into student values(&stud_no,'&name',&messfee,'&vegnon')
new   1: insert into student values(2,'Vimal',2500,veg)
1 row created.
SQL> /
Enter value for stud_no: 3
Enter value for name: Ruben
Enter value for messfee: 2600
Enter value for vegnon: non
old   1: insert into student values(&stud_no,'&name',&messfee,'&vegnon')
new   1: insert into student values(4,'Ruben',2600,'non')
1 row created.
SQL> /
Enter value for stud_no: 4
Enter value for name: Sanish
Enter value for messfee: 2500
Enter value for vegnon: veg
old   1: insert into student values(&stud_no,'&name',&messfee,'&vegnon')
new   1: insert into student values( 4,'Sanish',2500,'non')
1 row created.
SQL> /
Enter value for stud_no: 5
Enter value for name: Kevin
Enter value for messfee: 2600
Enter value for vegnon: non
old   1: insert into student values(&stud_no,'&name',&messfee,'&vegnon')
new   1: insert into student values(5,'Kevin',2600,'non')
1 row created.
```

```
SQL> select* from mess;
---------------------------------------------------------------------------------------
STUD_NO           NAME        MESSFEE           VEGNON
---------------------------------------------------------------------------------------
1                 Sohesh      2600              non
2                 Vimal       2500              veg
3                 Ruben       2600              non
4                 Sanish      2500              veg
5                 Kevin       2600              non
```

```
DECLARE
        cursor c1 IS select * from mess;
        no number;
        nm varchar(20);
        fee number(8);
        tp varchar(1);
        r number;
BEGIN
        open c1;
        loop
        fetch c1 into no,nm,fee,tp;
        if tp='v' then
        r:=fee*.1;
        fee:=fee+r;
        insert into veg values(no,nm,r,SYSDATE);
        else
        r:=fee*.2;
        fee:=fee+r;
        insert into non values(no,nm,r,SYSDATE);
        end if;
        exit when c1%NOTFOUND;
        end loop;
END;
```

OUTPUT

```
SQL> select * from veg;
----------------------------------------------------------------------------------------------------
STUD_NO              NAME                 RAISE_IN_FEE            DATE
----------------------------------------------------------------------------------------------------
2                    Vimal                250                     22-AUG-2014
4                    Sanish               250                     22-AUG_2014

SQL> select * from nonveg;
----------------------------------------------------------------------------------------------------
STUD_NO              NAME                 RAISE_IN_FEE            DATE
----------------------------------------------------------------------------------------------------
1                    Sohesh               520                     22-AUG-2014
3                    Ruben                520                     22-AUG-2014
4                    Kevin                520                     22-AUG-2014
```

RESULT

The PL/SQL Program executed and output is obtained

Exp. no: 21 14-02-2022

MODERATION USING CURSOR

AIM

To create a table T1 having 3 fields(rollno, univ_mark and sessionals). Write a PL/SQL program to do the following: If sessionals is in between 30 and 34, then give necessary moderation so that it comes up to 35. If univ_mark+sessionals>75, then insert those tuples into another table T2.

ALGORITHM

1. Create two tables hostel t1, t2.
2. Create a cursor t_cur
3. Declare the variables
4. Open the cursor t_cur
5. Fetch the roll no, unimark and sessionals to the cursor.
6. Update the table depending on sessionals between 30 &34.
7. If unimark greater than 75 insert into table t2.
8. Display the resultant table.
9. Stop

PROGRAM

```
SQL>create table t1(rollno number(3), unimark number(3), sessionals number(3));

Table Created.

 SQL>create table t2(rollno number(3), unimark number(3), sessionals number(3));

Table Created.

SQL> insert into t1 values(3, 70, 32);

1 row created.

SQL> select *from t1;

   ROLLNO    UNIMARK   SESSIONALS
----------------   ---------------   ------------------
        1              45             34
        2              70             40
        3              70             32
        4              40             29


DECLARE
        t t1%rowtype;
        cursor t_cur is select *from t1;
BEGIN
        open t_cur;
        if(t_cur%isopen) then
        loop
        fetch t_cur into t;
        exit when t_cur%notfound;
        if(t.sessionals between 30 and 34) then
        update t1 set sessionals=35 where (sessionals>30 and sessionals<35);
        end if;
        if(t.unimark+t.sessionals>75) then
        insert into t2 values(t.rollno, t.unimark, t.sessionals);
        end if;
        end loop;
        end if;
END;
```

OUTPUT

```
SQL> select *from t2;
   ROLLNO    UNIMARK   SESSIONALS
----------------    --------------- --------------------
       1                  45              35
       2                  70              40
       3                  70               35
SQL> select *from t1;
ROLLNO    UNIMARK   SESSIONALS
    -------------   -------------     -----------------
        1               45                35
        2               70               40
        3                70               35
       4                40               29
```

RESULT

PL/SQL queries are executed and output obtained.

Exp. no: 22 21-02-2022

STUDENT MARK USING FUNCTION

AIM

Write a function which accepts the reg_no and print the total marks. The student table has the fields: reg_no, name, physics_mark, chemistry_mark and maths_mark.

ALGORITHM

```
1. Create a table student with fields regno, phy, chem, math and insert values into it.
2. Create a function f which accept regno as argument.
   a .Let total = phy+chem+math
   b. Return total
3. Enter the regno and print the total marks by calling the function f(reg) in
   the main program.
4. End
5.
```

PROGRM

```
SQL> create table student4(regno number(5), phy number(3), chem number(3), math
number(3));
Table created.
SQL> insert into student4 values(1, 79, 83, 81);
1 row created.
SQL> insert into student4 values(2, 97, 93, 100);
1 row created.
SQL> insert into student4 values(3, 54, 67, 44);
1 row created.
SQL> select *from student4;

     REGNO        PHY       CHEM       MATH
---------- ---------- ---------- ----------------
         1         79         83         81
         2         97         93        100
         3         54         67         44
```

```
create or replace function f(no number) return number is
p student4.phy%type;
c student4.chem%type;
m student4.math%type;
tot number(5);
BEGIN
        select phy,chem,math into p,c,m from student4 where regno=no;
        tot:=p+c+m;
        return(tot);
END;
Function Created.
DECLARE
  r student4.regno%type;
  p number(10);
BEGIN
  r:=&r;
  p:=f(r);
  dbms_output.put_line('total mark'||p);
END;
```

OUTPUT

```
Enter value for r: 1
total mark= 243
```

RESULT

The PL/SQL Program executed and output is obtained

Exp. no: 23 21-02-2022

SUM OF FIRST N EVEN NUMBERS USING FUNCTION

AIM

Write a function to find sum of first N even numbers

ALGORITHM

1. Create a function f which accept limit as argument.
2. Let s=0 and i=2
3. Go to step 4 and repeat till ab is equal to limit.
4. Enter the limit and print the total marks by calling the function f1(no) in the main program.
5. End

PROGRAM

```
create or replace function f1(no in number) return number is
s number:=0;
i number:=2;
ab number:=1;
begin
   while(ab<=no)
   loop
          s:=s+i;
          i:=i+2;
          ab:=ab+1;
   end loop;
   return(s);
end;
Function Created.
DECLARE
no number(10):=&no;
 s number(10);
 BEGIN
 s:=f1(no);
 dbms_output.put_line('sum'||s);
```

```
END;
```

OUTPUT

```
Enter value for no: 3
sum=12
```

RESULT

The PL/SQL Program executed and output is obtained.

Exp. no: 24 21-02-2022

GRADE DISPLAY USING PROCEDURE

AIM

Write a PL/SQL program to display the grade of a particular student from student database. Use a stored procedure to display the grade.

TOTAL MARK	GRADE
>100	A
70-100	B
50-70	C
<50	Fail

ALGORITHM

```
1. Create a table student5 with fields rno, name, mark and insert values into it.
2. Create a procedure pr which accept rno as argument.
3. Fetch mark into tot.
    a. If tot>100 grade=A
    b. If tot from 70-100 grade=B
    c. If tot from 50-70 grade =C
    d. If tot<50 grade= Fail
4. Enter the rno and print the grade by calling the procedure pr(no) in the main program.
5. End
```

PROGRAM

```
SQL> create table student5(rno number(3), name char(10), mark number(3));
Table created.
SQL> insert into student5 values(1, 'harry', 81);
1 row created.
SQL>select *from student5;
RNO   NAME        MARK
-----------------------------------
1     Harry       81
2     Fred        60
```

```
create or replace procedure pr(no  number) is
tot number;
begin
        select mark into tot from student5 where  rno=no;
        if(tot>100) then
                dbms_output.put_line('A');
        elsif(tot>70 and tot<100) then
                dbms_output.put_line('B');
        elsif(tot>50 and tot<70) then
                dbms_output.put_line('C');
        else
                dbms_output.put_line('Fail');
        end if;
end;

DECLARE
  r number(10);
BEGIN
  r:=&r;
  pr(r);
END;
```

OUTPUT

```
Enter value for r: 1
B
```

RESULT

The PL/SQL Program executed and output is obtained

Exp. no: 25 28-02-2022

ACCOUNT TABLE USING TRIGGER

AIM

To create an account table(acc_no, cname, balance, branch_name), loan table(loan_no, amt, branch_name), borrower table(cname, loan_no). Create a trigger to perform the following operations: Whenever the balance becomes negative, create a loan in the amount of overdraft. The loan_no is given same as acc_no.

ALGORITHM

1. Create a table account with fields accno, cname, bal and branchname.
2. Create a table loan with fields loanno, amount, branchname.
3. Create a table borrower with fields cname, loanno.
4. Create a trigger t which executes after update or insert on each row of account table.
5. Insert into loan table new.accno,new.bal-1 and new.branchname.
6. Insert into borrower table new.cname, new.loanno
7. End the trigger.
8. Insert values into the account table
9. End.

PROGRAM

```
SQL> create table account(accno number(10), cname char(10), balance number(10),
branch char(10));
Table created.
SQL> create table loan(loanno number(10), amt number(10), branch char(10));
Table created.
SQL> create table borrower(cname char(10), loanno number(10));
Table created.
SQL> insert into account values(1, 'tibin', 200, 'ktpna');
1 row created.

 create or replace trigger ac_tri after update on account for each row when(new.balance<0)
  begin
```

```
   insert into loan values(:new.accno,:new.balance*-1,:new.branch);
  insert into borrower values(:new.cname,:new.accno);
  dbms_output.put_line('Loan created');
   end;
Trigger created.
SQL> update account set balance=-20 where accno=1;
1 row updated.
```

OUTPUT

```
PL/SQL procedure successfully completed.
SQL> select *from loan;
   LOANNO        AMT BRANCH
---------- ---------- ----------
         1         20 ktpna
SQL> select *from borrower;
CNAME          LOANNO
---------- ----------
tibin               1
```

RESULT

PL/SQL queries are executed and output obtained.

Exp. no: 26 28-02-2022

STAFF DATABASE USING TRIGGER

AIM

Create a transparent audit system for a table clientmaster. The system has to keep track of records that have been removed or modified and when they have been removed or modified. Table details are given below:

AuditClient: name, bal_due, operation, Op_date

ALGORITHM

```
1. Create a table cli with fields cno, name, adrs and due.
2. Create a table auditcli with fields name, baldue, op, date.
3. Create a trigger trig which executes after update or delete on each row of cli table.
4. If updating, insert into auditcli table :old.name, :old.due, 'Update',sysdate
5. If deleting, insert into auditcli :old.name,:old.due,'Delete',sysdate;
6. End the trigger.
7. Insert values into the account table
8. End.
```

PROGRAM

```
SQL> create table cli(cno number(10), name char(10), adrs char(10), due number(10));
Table created.
SQL> create table auditcli(name char(10), baldue number(10), op char(10), dt date);
Table created.
SQL> select *from cli;
      CNO NAME       ADRS              DUE
---------- ---------- ---------- ----------
        1 reshma     ktm               300
        2 neenu      ktm               400

SQL> select *from auditcli;
no rows selected
```

```
create or replace trigger trig after delete or update on cli for each row
 begin
 if updating then
insert into auditcli values(:old.name,:old.due,'Update',sysdate);
 else
 insert into auditcli values(:old.name,:old.due,'Delete',sysdate);
 end if;
 end;
Trigger created.
SQL> update cli set adrs='ekm' where cno=1;
1 row updated.
```

OUTPUT

```
SQL> select *from cli;
      CNO        NAME           ADRS            DUE
---------- ---------- ---------- -----------------------------
        1          reshma          ekm            300
        2          neenu           ktm             400

SQL> select *from auditcli;
NAME          BALDUE           OP                DT
---------- ---------- ---------- ----------------------------
reshma             300            Update           30-OCT-13

SQL> delete from cli where cno=2;
1 row deleted.
SQL> select *from cli;

CNO     NAME       ADRS             DUE
---------- ---------- ---------- -------------------
   1         reshma         ekm              300
```

```
SQL> select *from auditcli;
NAME          BALDUE   OP              DT
---------- ---------- ---------- ----------------------
reshma         300          Update       30-OCT-13
neenu          400          Delete       30-OCT-13
```

RESULT

The PL/SQL Program executed and output is obtained

Exp. no: 27 28-02-2022

NUMBER USING TRIGGER

AIM

To create a table with two number fields a and b and to write a trigger that satisfies the following condition:

a. a+b>75.
b. If value of b is changed, it should not be changed to a smaller value.
c. Also the tuples that violate these conditions should not be entered.

ALGORITHM

```
1. Create a table with fields a,b.
2. Create a trigger test_trigger which executes before insert or update on each row
   of the test table.
3. If a+b<=75, then raise exception1
4. If new.b < old.b then raise exception2
5. End the trigger.
```

PROGRAM

```
create or replace trigger test_trigger
before insert or update on test
referencing old as o new as n
for each row
when (n.a+n.b <=75 or n.b < o.b)
DECLARE
        excep1 exception;
        excep2 exception;
BEGIN
        If ((:n.a +:n.b) <= 75) then
                 raise excep1;
        end if;
        If (:n.b <:o.b) then
                  raise excep2;
EXCEPTION
         when excep1 then
                    raise_application_error (-20000,'Not allowed: a +b <=75');
         when excep2 then
         raise_application_error (-20001,'Present Value of b less than previous value');
```

```
END;
```

OUTPUT

```
SQL> create table test (a number (4), b number (4));

Table created.

SQL> select * from test;
        A             B
-----------    ------------
       70            75
SQL> @z:\S5\plsql\cy3_6.sql;

Trigger created.
SQL> update test set b=70 where a=70;
update test set b=70 where a=70;
                     *
ERROR at line 1:
ORA-20001: Present Value of b less than previous value
ORA-06512: at "M.TEST_TRIGGER", line 15
ORA-04088: error during execution of trigger' M.TEST_TRIGGER'

SQL> insert into test values (1, 2);
insert into test values (1, 2);
                  *
ERROR at line 1:
ORA-20000: Not allowed: a+b <=75
ORA-06512: at "M.TEST_TRIGGER", line 13
ORA-04088: error during execution of trigger' M.TEST_TRIGGER'
SQL> select * from test;
                  A             B
          -----------    ------------
                 70            75
```

RESULT

The PL/SQL Program executed and output is obtained.

Exp. no: 28 02-03-2022

CRUD OPERATIONS IN MONGO DB

AIM

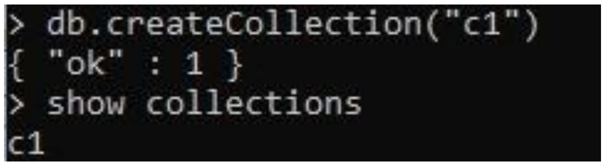
To perform the following CRUD operations:

1) CREATE
2) INSERT
3) READ
4) UPDATE
5) DELETE

QUERY

Creating a collection:

db.createCollection("c1")



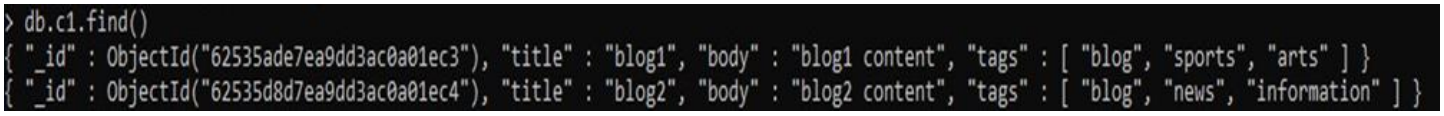
```
> db.createCollection("c1")
{ "ok" : 1 }
> show collections
c1
```

Inserting into a collection:

db.c1.insert({title:'blog1', body:'blog1 content',tags: ['blog','sports','arts'] }) db.c1.insert({title:'blog2', body:'blog2 content',tags: ['blog','news','information'] })

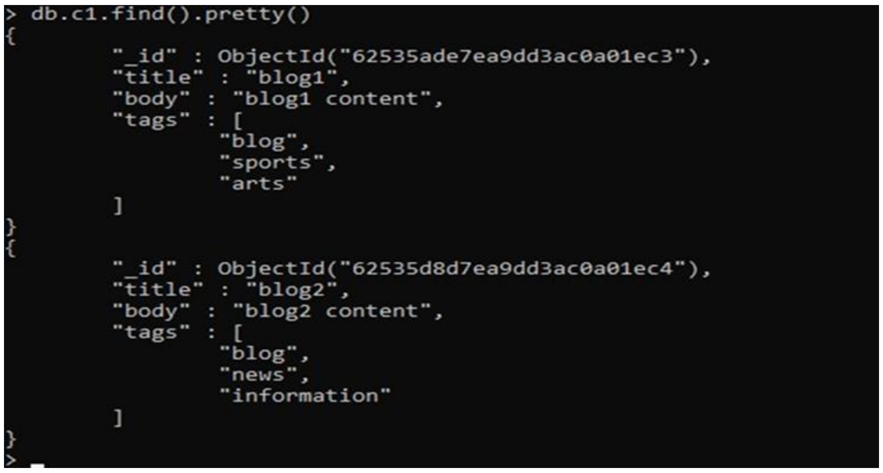
Reading the collection:

db.c1.find()



```
> db.c1.find()
{ "_id" : ObjectId("62535ade7ea9dd3ac0a01ec3"), "title" : "blog1", "body" : "blog1 content", "tags" : [ "blog", "sports", "arts" ] }
{ "_id" : ObjectId("62535d8d7ea9dd3ac0a01ec4"), "title" : "blog2", "body" : "blog2 content", "tags" : [ "blog", "news", "information" ] }
```

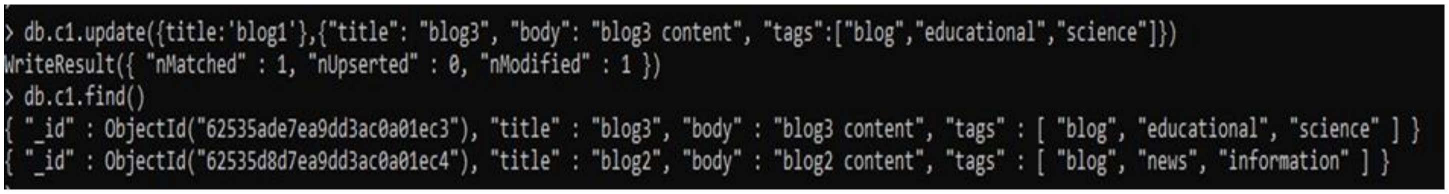
db.c1.find().pretty()



```
> db.c1.find().pretty()
{
        "_id" : ObjectId("62535ade7ea9dd3ac0a01ec3"),
        "title" : "blog1",
        "body" : "blog1 content",
        "tags" : [
                "blog",
                "sports",
                "arts"
        ]
}
{
        "_id" : ObjectId("62535d8d7ea9dd3ac0a01ec4"),
        "title" : "blog2",
        "body" : "blog2 content",
        "tags" : [
                "blog",
                "news",
                "information"
        ]
}
>
```

Updating the Document:

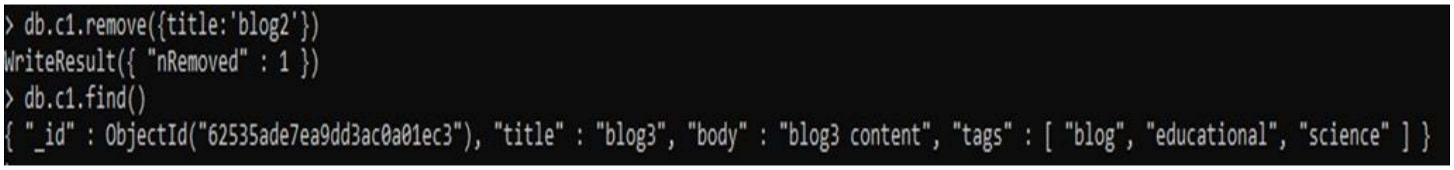
db.c1.update({title:'blog1'},{"title": "blog3", "body": "blog3 content", "tags":["blog","educational","science"]})



```
> db.c1.update({title:'blog1'},{"title": "blog3", "body": "blog3 content", "tags":["blog","educational","science"]})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.c1.find()
{ "_id" : ObjectId("62535ade7ea9dd3ac0a01ec3"), "title" : "blog3", "body" : "blog3 content", "tags" : [ "blog", "educational", "science" ] }
{ "_id" : ObjectId("62535d8d7ea9dd3ac0a01ec4"), "title" : "blog2", "body" : "blog2 content", "tags" : [ "blog", "news", "information" ] }
```

Deleting an item:

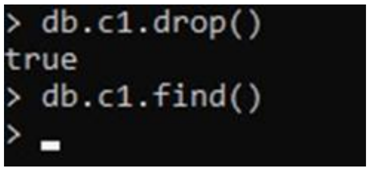
db.c1.remove({title:'blog2'})



```
> db.c1.remove({title:'blog2'})
WriteResult({ "nRemoved" : 1 })
> db.c1.find()
{ "_id" : ObjectId("62535ade7ea9dd3ac0a01ec3"), "title" : "blog3", "body" : "blog3 content", "tags" : [ "blog", "educational", "science" ] }
```

Deleting a collection:

db.c1.drop()



```
> db.c1.drop()
true
> db.c1.find()
> _
```

RESULT

The MongoDB operations executed and output is obtained.

Exp. no: 29 02-03-2022

JAVA DATABASE CONNECTIVITY

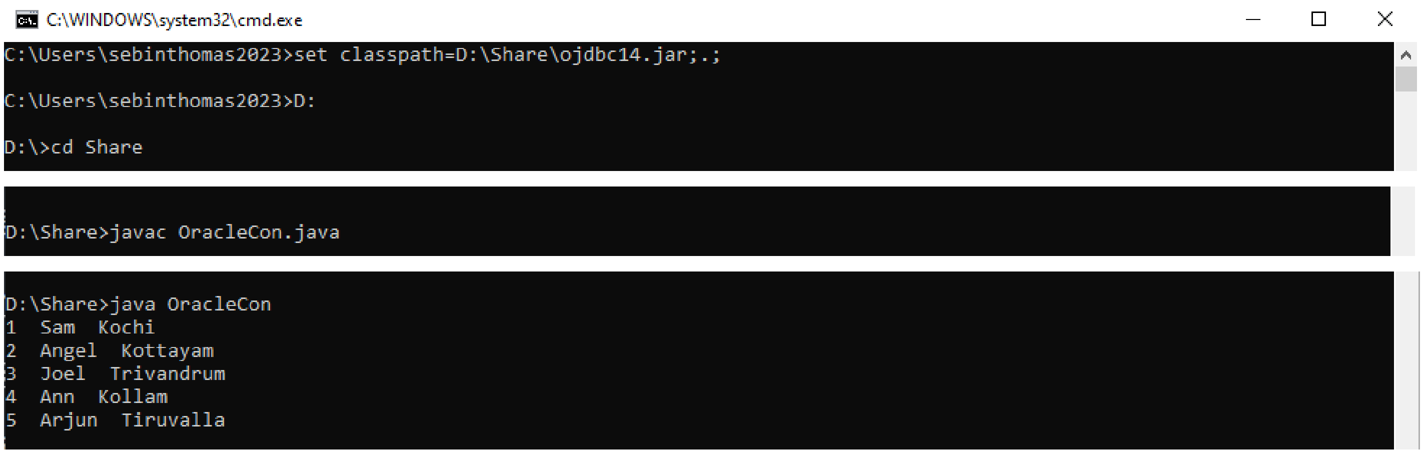
AIM

To connect java application with the oracle database.

PROGRAM

```
import java.sql.*;
class OracleCon{
public static void main(String args[]){
try{
Class.forName("oracle.jdbc.driver.OracleDriver");

Connection con=DriverManager.getConnection(
"jdbc:oracle:thin:@192.168.15.1:1521:xe","sebint","password");

Statement stmt=con.createStatement();

ResultSet rs=stmt.executeQuery("select * from emp");
while(rs.next())
System.out.println(rs.getInt(1)+"  "+rs.getString(2)+"  "+rs.getString(3));

con.close();

}catch(Exception e){ System.out.println(e);}

}
}
```

OUTPUT



```
C:\Users\sebinthomas2023>set classpath=D:\Share\ojdbc14.jar;.;

C:\Users\sebinthomas2023>D:

D:\>cd Share

D:\Share>javac OracleCon.java

D:\Share>java OracleCon
1  Sam  Kochi
2  Angel  Kottayam
3  Joel  Trivandrum
4  Ann  Kollam
5  Arjun  Tiruvalla
```

RESULT

Connection of Java Application with Oracle Database has been successfully executed.