SRI SIVASUBRAMANIYA NADAR COLLEGE OF ENGINEERING

(AN AUTONOMOUS INSTITUTION, AFFILIATED TO ANNA UNIVERSITY)

Rajiv Gandhi Salai (OMR), Kalavakkam - 603 110.

LABORATORY RECORD

NAME : P.T. JAYANNTHAN

Reg. No. : 205001049

Dept. : CSE Sem. : IV Sec. : 'A'

SSN

SRI SIVASUBRAMANIYA NADAR COLLEGE OF ENGINEERING, CHENNAI

(AN AUTONOMOUS INSTITUTION, AFFILIATED TO ANNA UNIVERSITY)

BONAFIDE CERTIFICATE

Certified that this is the bonafide record of the practical work done in the

UCS1412 - DATABASE LAB ..... **Laboratory by**

Name ..... P.T. JAYANNTHAN .....

Register Number ..... 205001049 .....

Semester ..... IV .....

Branch ..... CSE .....

Sri Sivasubramaniya Nadar College of Engineering, Kalavakkam.

During the Academic year ..... 2021 - 2022 .....

Faculty **Head of the Department**

Submitted for the..... Practical Examination held at SSNCE on.....

Internal Examiner **External Examiner**

INDEX

Name : P.T. JAYANNTHAN Reg. No. 205001049

Sem : IV Sec : A

Ex. No.	Date of Expt.	Title of the Experiment	Page No.	Signature of the Faculty	Remarks
1	10/3/22	DDL for Mail-Order Database		P.T.J 7/4/22	
2	17/3/22	DML Fundamentals			
3	7/4/22	Advanced DML: Joins Subqueries, Set operation		P.T.J 28/4/22	
4	21/4/22	Views			
5	28/4/22	PL/SQL - Control Structures		P.T.J 12/5/22	
6	12/5/22	PL/SQL - Procedures and Functions		P.T.J 26/5/22	
7	19/5/22	Triggers			
8	26/5/22	Exception Handling		P.T.J 9/6/22	
9	9/6/22	Database Programming using JDBC/ODBC			

	SSN COLLEGE OF ENGINEERING ***Department of Computer Science & Engineering***	***Faculty:*** ***P. Mirunalini, Asso. Prof.*** ***N. Sujaudeen, Asst. Prof***
UCS1412 – DBMS Lab ***Assignment – 1***		***Assigned: 28-02-22*** ***Due: 10-03-22***
Title: DDL for Mail Order Database		

Mail Order Database

Consider a mail order database in which employees take orders for parts from customers. The data requirements are summarized as follows:

a) The mail order company has employees identified by a unique employee number, their name, date-of-birth, pin code and city where they are located.

b) The customers of the company are identified by a unique customer number, their name, street name, pin code, city where they are located, date-of-birth and a phone number.

c) The parts being sold by the company are identified by a unique part number, a part name, their price, and quantity on hand.

d) Orders placed by customers are taken by employees and are given a unique order number. Each order may contain certain quantities of one or more parts and their received date as well as a shipped date is recorded.

Create the relations with the above mentioned specifications and also consider the following constraints:

1. Identify the primary key(s) and foreign key(s) from the schema.
2. Ensure that order number begins with O, similarly customer number with C, employee number with E and part number with P.
3. The phone numbers of the customers should not be identical to each other.
4. The quantity ordered should not be zero.
5. Order received date should always be less than the shipped date.
6. The price of the part should compulsorily contain some value.

The following changes have been identified due to increasing business. As a database designer you must accommodate these changes in your design.

7. It is identified that the following attributes are to be included in respective relations: Parts (reorder level), Employees (hiredate)
8. The width of a customer name is not adequate for most of the customers.
9. The date-of-birth of a customer can be addressed later / removed from the schema.
10. An order can not be placed without the receive date.
11. A customer may cancel an order or ordered part(s) may not be available in a stock. Hence on removing the details of the order, ensure that all the corresponding details are also deleted.

Note:
Populate each relation with relevant row(s) and prepare test cases to demonstrate that the requirements are satisfied.

What you have to submit:

1. Schema Diagram with constraints
2. Demo script file

SSN COLLEGE OF ENGINEERING, KALAVAKKAM
(An Autonomous Institution, Affiliated to Anna University, Chennai)

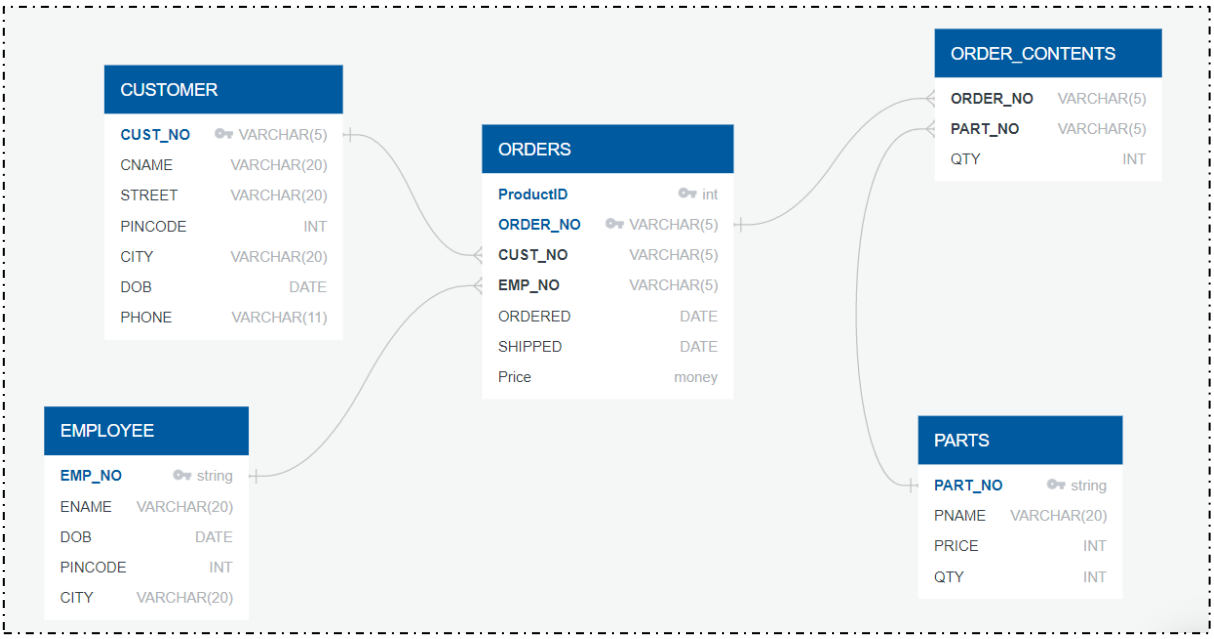
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UCS1412 – DBMS Lab
Assignment – 1

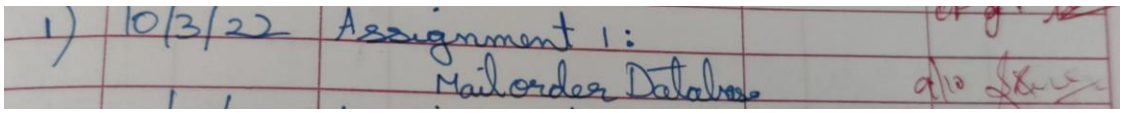
Name: Jayannthan P T *Dept: CSE 'A'* *Roll No.: 205001049*

Title: DDL for Mail Order Database

Schema Diagram:



Observation Index Correction:



Spool File Output:

```
SQL> REM ***DROPPING TABLES***
SQL>
SQL> DROP TABLE ORDER_CONTENTS;

Table dropped.

SQL> DROP TABLE ORDERS;

Table dropped.
```

```
SQL> DROP TABLE PARTS;

Table dropped.

SQL> DROP TABLE CUSTOMER;

Table dropped.

SQL> DROP TABLE EMPLOYEE;

Table dropped.

SQL>
SQL>
SQL> REM ***CREATING TABLE EMPLOYEE***
SQL>
SQL> CREATE TABLE EMPLOYEE(
  2  EMP_NO VARCHAR(5) CONSTRAINT E_PRIMEKEY PRIMARY KEY,
  3  ENAME VARCHAR(20),
  4  DOB DATE,
  5  PINCODE INT,
  6  CITY VARCHAR(20),
  7  CONSTRAINT ESTART_LETTER CHECK(EMP_NO LIKE 'E%')
  8  );

Table created.

SQL> DESC EMPLOYEE;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 EMP_NO                                    NOT NULL VARCHAR2(5)
 ENAME                                              VARCHAR2(20)
 DOB                                                DATE
 PINCODE                                            NUMBER(38)
 CITY                                               VARCHAR2(20)

SQL>
SQL> REM ***INSERTING VALUES INTO EMPLOYEE***
SQL>
SQL> INSERT INTO EMPLOYEE VALUES
  2  ('E01','Tony','05-DEC-1992',601101,'Chennai');

1 row created.

SQL> INSERT INTO EMPLOYEE VALUES
  2  ('E02','Steve','23-MAR-1989',601050,'Madurai');

1 row created.

SQL> INSERT INTO EMPLOYEE VALUES
  2  ('E03','Peter','17-JUN-1995',601302,'Trichy');

1 row created.
```

```
SQL>
SQL> INSERT INTO EMPLOYEE VALUES
  2  ('E04','Chris','07-JAN-1985',601023,'Kovai');

1 row created.

SQL>
SQL>
SQL> REM ***check constraint ESTART_LETTER***
SQL> INSERT INTO Employee VALUES ('03','Roshan','29/MAR/2022',600006,'trichy');
INSERT INTO Employee VALUES ('03','Roshan','29/MAR/2022',600006,'trichy')
*
ERROR at line 1:
ORA-02290: check constraint (SYSTEM.ESTART_LETTER) violated


SQL>
SQL>
SQL> REM ***CREATING TABLE CUSTOMER***
SQL>
SQL> CREATE TABLE CUSTOMER(
  2  CUST_NO VARCHAR(5) CONSTRAINT C_PRIMEKEY PRIMARY KEY,
  3  CNAME VARCHAR(20),
  4  STREET VARCHAR(20),
  5  PINCODE INT,
  6  CITY VARCHAR(20),
  7  DOB DATE,
  8  PHONE VARCHAR(11) CONSTRAINT PHONE_UNIQUE UNIQUE,
  9  CONSTRAINT CSTART_LETTER CHECK(CUST_NO LIKE 'C%'));

Table created.

SQL> DESC CUSTOMER;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CUST_NO                                   NOT NULL VARCHAR2(5)
 CNAME                                              VARCHAR2(20)
 STREET                                             VARCHAR2(20)
 PINCODE                                            NUMBER(38)
 CITY                                               VARCHAR2(20)
 DOB                                                DATE
 PHONE                                              VARCHAR2(11)

SQL>
SQL> REM ***INSERTING VALUES INTO CUSTOMER***
SQL>
SQL> INSERT INTO CUSTOMER VALUES
  2  ('C01','Sam','Street1',501101,'Delhi','20-Jul-1999',9673125647);

1 row created.

SQL> INSERT INTO CUSTOMER VALUES
  2  ('C02','Shyam','Street2',501060,'Mumbai','19-Oct-1979',9670005647);
```

```
1 row created.

SQL> INSERT INTO CUSTOMER VALUES
  2  ('C03','Stark','Street3',501101,'Kolkata','20-Jul-1999',9848949218);

1 row created.

SQL> INSERT INTO CUSTOMER VALUES
  2  ('C04','Gwen','Street4',501101,'Hyderabad','20-Jul-1999',9676489841);

1 row created.

SQL>
SQL>
SQL> REM ***check constraint C_PRIMEKEY***
SQL> INSERT INTO CUSTOMER VALUES
('C03','Te','Street',620265,'Madurai','30/JUL/2022',9998865646);
INSERT INTO CUSTOMER VALUES
('C03','Te','Street',620265,'Madurai','30/JUL/2022',9998865646)
*
ERROR at line 1:
ORA-00001: unique constraint (SYSTEM.C_PRIMEKEY) violated


SQL> REM ***check constraint CSTART_LETTER***
SQL> INSERT INTO CUSTOMER VALUES
('04','Tem','Street',620275,'pondicherry','30/JUL/2022',999866546);
INSERT INTO CUSTOMER VALUES
('04','Tem','Street',620275,'pondicherry','30/JUL/2022',999866546)
*
ERROR at line 1:
ORA-02290: check constraint (SYSTEM.CSTART_LETTER) violated


SQL>
SQL>
SQL>
SQL> REM ***CREATING TABLE PARTS***
SQL>
SQL> CREATE TABLE PARTS(
  2  PART_NO VARCHAR(5) CONSTRAINT P_PRIMEKEY PRIMARY KEY,
  3  PNAME VARCHAR(20),
  4  PRICE INT CONSTRAINT PRICENOTNULL NOT NULL,
  5  QTY INT ,
  6  CONSTRAINT QTY_MORE CHECK(QTY > 0),
  7  CONSTRAINT PSTART_LETTER CHECK(PART_NO LIKE 'P%'));

Table created.

SQL> DESC PARTS;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 PART_NO                                   NOT NULL VARCHAR2(5)
 PNAME                                              VARCHAR2(20)
```

```
 PRICE                                     NOT NULL NUMBER(38)
 QTY                                               NUMBER(38)

SQL>
SQL>
SQL> REM ***INSERTING VALUES INTO PARTS***
SQL>
SQL> INSERT INTO PARTS VALUES
  2  ('P01','Screw',70,1000);

1 row created.

SQL> INSERT INTO PARTS VALUES
  2  ('P02','Pipe',50,1500);

1 row created.

SQL> INSERT INTO PARTS VALUES
  2  ('P03','Wire',10,1200);

1 row created.

SQL> INSERT INTO PARTS VALUES
  2  ('P04','Wood',90,1100);

1 row created.

SQL>
SQL>
SQL> REM ***check constraint PSTART_LETTER***
SQL> INSERT INTO PARTS VALUES ('01', 'ABD', 2500, 25);
INSERT INTO PARTS VALUES ('01', 'ABD', 2500, 25)
*
ERROR at line 1:
ORA-02290: check constraint (SYSTEM.PSTART_LETTER) violated


SQL> REM ***check constraint P_PRIMEKEY***
SQL> INSERT INTO PARTS VALUES ('P04', 'ABD', 2500, 25);
INSERT INTO PARTS VALUES ('P04', 'ABD', 2500, 25)
*
ERROR at line 1:
ORA-00001: unique constraint (SYSTEM.P_PRIMEKEY) violated


SQL>
SQL>
SQL> REM ***CREATING TABLE ORDERS***
SQL>
SQL> CREATE TABLE ORDERS(
  2  ORDER_NO VARCHAR(5) CONSTRAINT O_PRIMEKEY PRIMARY KEY,
  3  CUST_NO VARCHAR(5),
  4  EMP_NO VARCHAR(5),
  5  ORDERED DATE,
```

```
  6  SHIPPED DATE,
  7  CONSTRAINT DATE_CHECK CHECK(SHIPPED > ORDERED),
  8  CONSTRAINT OSTART_LETTER CHECK(ORDER_NO LIKE 'O%'),
  9  CONSTRAINT FORKEYEMP FOREIGN KEY(EMP_NO) REFERENCES EMPLOYEE(EMP_NO),
 10  CONSTRAINT FORKEYCUS FOREIGN KEY(CUST_NO) REFERENCES CUSTOMER(CUST_NO));

Table created.

SQL> DESC ORDERS;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ORDER_NO                                  NOT NULL VARCHAR2(5)
 CUST_NO                                            VARCHAR2(5)
 EMP_NO                                             VARCHAR2(5)
 ORDERED                                            DATE
 SHIPPED                                            DATE

SQL>
SQL>
SQL> REM ***INSERTING VALUES INTO ORDERS***
SQL>
SQL> INSERT INTO ORDERS VALUES
  2  ('O01','C01','E02','26-Jan-2022','3-Feb-2022');

1 row created.

SQL> INSERT INTO ORDERS VALUES
  2  ('O02','C03','E01','29-Jan-2022','8-Feb-2022');

1 row created.

SQL> INSERT INTO ORDERS VALUES
  2  ('O03','C02','E04','8-Feb-2022','19-Feb-2022');

1 row created.

SQL> INSERT INTO ORDERS VALUES
  2  ('O04','C04','E03','13-Jan-2022','24-Jan-2022');

1 row created.

SQL>
SQL>
SQL> REM ***check constraint O_PRIMEKEY***
SQL> INSERT INTO Orders VALUES('O03', 'C02', 'E03', '12/JAN/2022', '24/JAN/2022');
INSERT INTO Orders VALUES('O03', 'C02', 'E03', '12/JAN/2022', '24/JAN/2022')
*
ERROR at line 1:
ORA-00001: unique constraint (SYSTEM.O_PRIMEKEY) violated


SQL> REM ***check constraint DATE_CHECK***
SQL> INSERT INTO Orders VALUES('O04','C04','E03','12/JAN/2022', '10/JAN/2022');
INSERT INTO Orders VALUES('O04','C04','E03','12/JAN/2022', '10/JAN/2022')
```

```
*
ERROR at line 1:
ORA-02290: check constraint (SYSTEM.DATE_CHECK) violated


SQL> REM ***check constraint FOREIGN KEYS***
SQL> INSERT INTO ORDERS VALUES ('O05','C00','E03','13-Jan-2022','24-Jan-2022');
INSERT INTO ORDERS VALUES ('O05','C00','E03','13-Jan-2022','24-Jan-2022')
*
ERROR at line 1:
ORA-02291: integrity constraint (SYSTEM.FORKEYCUS) violated - parent key not
found


SQL> INSERT INTO ORDERS VALUES ('O05','C01','E00','13-Jan-2022','24-Jan-2022');
INSERT INTO ORDERS VALUES ('O05','C01','E00','13-Jan-2022','24-Jan-2022')
*
ERROR at line 1:
ORA-02291: integrity constraint (SYSTEM.FORKEYEMP) violated - parent key not
found


SQL>
SQL> REM ***CREATING TABLE ORDER_CONTENTS***
SQL>
SQL> CREATE TABLE ORDER_CONTENTS(
  2  ORDER_NO VARCHAR(5),
  3  PART_NO VARCHAR(5),
  4  QTY INT,
  5  CONSTRAINT FORKEYORD FOREIGN KEY(ORDER_NO) REFERENCES ORDERS(ORDER_NO),
  6  CONSTRAINT FORKEYPART FOREIGN KEY(PART_NO) REFERENCES PARTS(PART_NO),
  7  CONSTRAINT ORDERCONTENTSPRIME PRIMARY KEY(ORDER_NO ,PART_NO));

Table created.

SQL> DESC ORDER_CONTENTS;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ORDER_NO                                  NOT NULL VARCHAR2(5)
 PART_NO                                   NOT NULL VARCHAR2(5)
 QTY                                                NUMBER(38)

SQL>
SQL> REM ***INSERTING VALUES INTO ORDER_CONTENTS***
SQL>
SQL> INSERT INTO ORDER_CONTENTS VALUES
  2  ('O01','P02', 50);

1 row created.

SQL> INSERT INTO ORDER_CONTENTS VALUES
  2  ('O03','P01', 45);

1 row created.
```

```
SQL> INSERT INTO ORDER_CONTENTS VALUES
  2  ('O02','P04', 55);

1 row created.

SQL> INSERT INTO ORDER_CONTENTS VALUES
  2  ('O01','P03', 78);

1 row created.

SQL> INSERT INTO ORDER_CONTENTS VALUES
  2  ('O04','P02', 25);

1 row created.

SQL>
SQL> REM ***check constraint FOREIGN KEYS***
SQL> INSERT INTO ORDER_CONTENTS VALUES('O03', 'P001',4);
INSERT INTO ORDER_CONTENTS VALUES('O03', 'P001',4)
*
ERROR at line 1:
ORA-02291: integrity constraint (SYSTEM.FORKEYPART) violated - parent key not
found


SQL> INSERT INTO ORDER_CONTENTS VALUES('O00', 'P04',4);
INSERT INTO ORDER_CONTENTS VALUES('O00', 'P04',4)
*
ERROR at line 1:
ORA-02291: integrity constraint (SYSTEM.FORKEYORD) violated - parent key not
found


SQL>
SQL>
SQL> REM ***ADDING REORDER_LEVEL IN PARTS***
SQL>
SQL> ALTER TABLE PARTS
  2  ADD REORDER_LEVEL INT;

Table altered.

SQL>
SQL>
SQL> REM ***ADDING HIRE_DATE IN EMPLOYEE***
SQL>
SQL> ALTER TABLE EMPLOYEE
  2  ADD HIRE_DATE DATE;

Table altered.

SQL>
SQL>
```

```
SQL> REM ***MODIFYING SIZE OF ENAME IN CUSTOMER***
SQL>
SQL> ALTER TABLE CUSTOMER
  2  MODIFY (CNAME  VARCHAR(25));

Table altered.

SQL>
SQL>
SQL> REM ***DELETING DOB IN CUSTOMER***
SQL>
SQL> ALTER TABLE CUSTOMER
  2  DROP COLUMN DOB;

Table altered.

SQL>
SQL>
SQL> REM ***MODIFYING SHIPPED NOT NULL IN ORDERS***
SQL>
SQL> ALTER TABLE ORDERS
  2  MODIFY (SHIPPED DATE CONSTRAINT RECIEVENOTNULL NOT NULL);

Table altered.

SQL>
SQL>
SQL>
SQL>
SQL> REM ***CASCADING***
SQL>
SQL> ALTER TABLE ORDER_CONTENTS
  2  DROP CONSTRAINT FORKEYORD;

Table altered.

SQL>
SQL>
SQL> ALTER TABLE ORDER_CONTENTS
  2  ADD CONSTRAINT FORKEYORD FOREIGN KEY(ORDER_NO) REFERENCES ORDERS(ORDER_NO) ON DELETE
CASCADE;

Table altered.

SQL>
SQL>
SQL>
SQL> REM ***VIEW TABLES***
SQL>
SQL> select * from employee;

EMP_N ENAME                DOB          PINCODE CITY                HIRE_DATE
----- -------------------- --------- ---------- -------------------- ---------
E01   Tony                 05-DEC-92     601101 Chennai
```

```
E02   Steve               23-MAR-89     601050 Madurai
E03   Peter               17-JUN-95     601302 Trichy
E04   Chris               07-JAN-85     601023 Kovai

SQL>
SQL> select * from customer;

CUST_ CNAME                     STREET                  PINCODE
----- ------------------------- -------------------- ----------
CITY                PHONE
-------------------- -----------
C01   Sam                       Street1                  501101
Delhi               9673125647

C02   Shyam                     Street2                  501060
Mumbai              9670005647

C03   Stark                     Street3                  501101
Kolkata             9848949218


CUST_ CNAME                     STREET                  PINCODE
----- ------------------------- -------------------- ----------
CITY                PHONE
-------------------- -----------
C04   Gwen                      Street4                  501101
Hyderabad           9676489841


SQL>
SQL> select * from parts;

PART_ PNAME                     PRICE        QTY REORDER_LEVEL
----- -------------------- ---------- ---------- -------------
P01   Screw                       70       1000
P02   Pipe                        50       1500
P03   Wire                        10       1200
P04   Wood                        90       1100

SQL> select * from orders;

ORDER CUST_ EMP_N ORDERED   SHIPPED
----- ----- ----- --------- ---------
O01   C01   E02   26-JAN-22 03-FEB-22
O02   C03   E01   29-JAN-22 08-FEB-22
O03   C02   E04   08-FEB-22 19-FEB-22
O04   C04   E03   13-JAN-22 24-JAN-22

SQL>
SQL> select * from order_contents;

ORDER PART_        QTY
----- ----- ----------
O01   P02           50
```

```
O03   P01          45
O02   P04          55
O01   P03          78
O04   P02          25

SQL>
SQL>
SQL>
SQL> REM ***CHECKING ON DELETE CASCADE***
SQL> DELETE FROM ORDERS WHERE ORDER_NO='O01';

1 row deleted.

SQL>
SQL>
SQL> select * from orders;

ORDER CUST_ EMP_N ORDERED   SHIPPED
----- ----- ----- --------- ---------
O02   C03   E01   29-JAN-22 08-FEB-22
O03   C02   E04   08-FEB-22 19-FEB-22
O04   C04   E03   13-JAN-22 24-JAN-22

SQL> select * from order_contents;

ORDER PART_        QTY
----- ----- ----------
O03   P01          45
O02   P04          55
O04   P02          25

SQL> spool off
```

	SSN COLLEGE OF ENGINEERING ***Department of Computer Science & Engineering***	***Faculty:*** ***B. Senthil Kumar, Asst. Prof.*** ***P.Mirunalini, Asso. Prof.*** ***N.Sujaudeen, Asst. Prof***
	CS8481 – DBMS Lab ***Assignment – 2***	***Assigned: 29-Dec-19***
Title: DML Fundamentals		

Manipulating Nobel Laureate Information Using DML

Aim: To learn the following:

a) Update operations such as INSERT, UPDATE, DELETE
b) Controlling the transactions using COMMIT, SAVEPOINT, ROLLBACK
c) SELECT Clause
   i) Using arithmetic operators, logical operators
   ii) Using LIKE, BETWEEN, IN keywords
   iii) Using Character, Date, Number and Aggregate functions
   iv) Using GROUP BY, HAVING, ORDER BY

Schema to be used for the following queries:
nobel (*noble_id, name, gender, category, field, year_award, aff_role, dob, country*) where *aff_role* describes the nobel laureates' affiliation towards an institute/organization or his/her role in that field for the award.
Populate the *nobel* relation as given in the script file (*nobel.sql*)

Write DML queries for the following:

1. Display the nobel laureate(s) who born after 1-Jul-1960.
2. Display the Indian laureate (name, category, field, country, year awarded) who was awarded in the Chemistry category.
3. Display the laureates (name, category,field and year of award) who was awarded between 2000 and 2005 for the Physics or Chemistry category.
4. Display the laureates name with their age at the time of award for the Peace category.
5. Display the laureates (name,category,aff_role,country) whose name starts with *A* or ends with *a*, but not from Isreal.
6. Display the name, gender, affiliation, dob and country of laureates who was born in 1950's. Label the dob column as *Born 1950*.
7. Display the laureates (name,gender,category,aff_role,country) whose name starts with A, D or H. Remove the laureate if he/she do not have any affiliation. Sort the results in ascending order of name.
8. Display the university name(s) that has to its credit by having at least 2 nobel laureate with them.
9. List the date of birth of youngest and eldest laureates by country-wise. Label the column as Younger, Elder respectively. Include only the country having more than one laureate. Sort the output in alphabetical order of country.
10. Show the details (year award,category,field) where the award is shared among the laureates in the same category and field. Exclude the laureates from USA.

Use TCL Statements

11.Mark an intermediate point in the transaction (savepoint).
12.Insert a new tuple into the nobel relation.
13.Update the aff_role of literature laureates as 'Linguists'.
14.Delete the laureate(s) who was awarded in Enzymes field.
15.Discard the most recent update operations (rollback).
16.Commit the changes.

What you have to submit:

1. Schema Diagram with constraints
2. Demo script file

SSN COLLEGE OF ENGINEERING, KALAVAKKAM
(An Autonomous Institution, Affiliated to Anna University, Chennai)

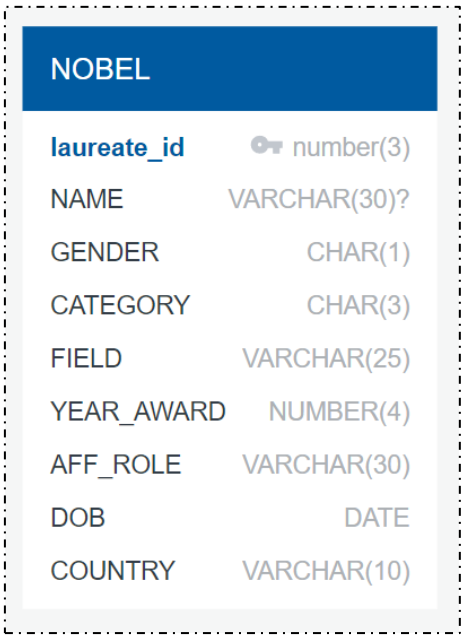
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UCS1412 – DBMS Lab
Assignment – 1

Name: Jayannthan P T *Dept: CSE 'A'* *Roll No.: 205001049*

Title: DDL for Mail Order Database

Schema Diagram:



Spool File Output:

```
SQL> -- Write DML queries for the following:
SQL>
SQL> --1.Display the nobel laureate(s) who born after 1-Jul-1960.
SQL> SELECT *
  2  FROM NOBEL
  3  WHERE DOB >('1-Jul-1960');

LAUREATE_ID NAME                           G CAT FIELD
----------- ------------------------------ - --- -------------------------
YEAR_AWARD AFF_ROLE                       DOB       COUNTRY
---------- ------------------------------ --------- ----------
       111 Eric A Cornell                 m Phy Atomic physics
      2001 University of Colorado         19-DEC-61 USA

       124 Carol W Greider                f Med Enzymes
      2009 Johns Hopkins University       15-APR-61 USA
```

```
       125 Barack H Obama                m Pea World organizing
      2009 President of USA              04-AUG-61 USA


SQL>
SQL> --2. Display the Indian laureate (name, category, field, country, year awarded) who
was awarded in the Chemistry category.
SQL> SELECT NAME, CATEGORY, FIELD, COUNTRY, YEAR_AWARD
  2  FROM NOBEL
  3  WHERE COUNTRY='India' AND CATEGORY='Che';

NAME                           CAT FIELD                     COUNTRY
------------------------------ --- ------------------------- ----------
YEAR_AWARD
----------
Venkatraman Ramakrishnan       Che Biochemistry              India
      2009


SQL>
SQL> -- 3. Display the laureates (name, category,field and year of award) who was awarded
between 2000 and 2005 for the Physics or Chemistry category.
SQL> SELECT NAME, CATEGORY, FIELD, YEAR_AWARD
  2  FROM NOBEL
  3  WHERE (YEAR_AWARD BETWEEN 2001 AND 2005) AND (CATEGORY='Che' OR CATEGORY='Phy');

NAME                           CAT FIELD                     YEAR_AWARD
------------------------------ --- ------------------------- ----------
Eric A Cornell                 Phy Atomic physics                  2001
Carl E Wieman                  Phy Atomic physics                  2001
Ryoji Noyori                   Che Organic Chemistry               2001
K Barry Sharpless              Che Organic Chemistry               2001

SQL>
SQL> -- 4. Display the laureates name with their age at the time of award for the Peace
category.
SQL> SELECT NAME, YEAR_AWARD-EXTRACT(YEAR FROM DOB) AS AGE, YEAR_AWARD
  2  FROM NOBEL
  3  WHERE CATEGORY='Pea';

NAME                                 AGE YEAR_AWARD
------------------------------ ---------- ----------
John Hume                              61       1998
David Trimble                          54       1998
Kofi Annan                             63       2001
Barack H Obama                         48       2009

SQL>
SQL> -- 5. Display the laureates (name,category,aff_role,country) whose name starts with A
or ends with a, but not from Isreal.
SQL> SELECT NAME, CATEGORY, AFF_ROLE, COUNTRY
  2  FROM NOBEL
  3  WHERE (NAME LIKE 'A%' OR NAME LIKE '%a') AND COUNTRY <> 'Isreal';
```

```
NAME                           CAT AFF_ROLE                       COUNTRY
------------------------------ --- ------------------------------ ----------
Amartya Sen                    Eco Trinity College                India
Barack H Obama                 Pea President of USA               USA

SQL>
SQL> -- 6. Display the name, gender, affiliation, dob and country of laureates who was
born in 1950's. Label the dob column as Born 1950.
SQL> SELECT NAME, GENDER, AFF_ROLE, DOB AS BORN1950, COUNTRY
  2  FROM NOBEL
  3  WHERE DOB >('1-Jan-1950') AND DOB < ('31-Dec-1959');

NAME                           G AFF_ROLE                       BORN1950
------------------------------ - ------------------------------ ---------
COUNTRY
----------
Robert B. Laughlin             m Stanford University            01-NOV-50
USA

Carl E Wieman                  m University of Colorado         26-MAR-51
USA

Venkatraman Ramakrishnan       m MRC Laboratory                 19-AUG-52
India


NAME                           G AFF_ROLE                       BORN1950
------------------------------ - ------------------------------ ---------
COUNTRY
----------
Herta Muller                   f                                17-AUG-53
Romania


SQL>
SQL>
SQL> -- 7. Display the laureates (name,gender,category,aff_role,country) whose name starts
with A, D or H. Remove the laureate if he/she do not have any affiliation. Sort the
results in ascending order of name.
SQL> SELECT NAME, GENDER, CATEGORY, AFF_ROLE, COUNTRY
  2  FROM NOBEL
  3  WHERE (NAME LIKE 'A%' OR NAME LIKE 'D%' OR NAME LIKE 'H%') AND AFF_ROLE IS NOT NULL
  4  ORDER BY NAME;

NAME                           G CAT AFF_ROLE                       COUNTRY
------------------------------ - --- ------------------------------ ----------
Ada E Yonath                   f Che Weizmann Institute of Science  Isreal
Amartya Sen                    m Eco Trinity College                India
Daniel C. Tsui                 m Phy Princeton University           China
David Trimble                  m Pea Ulster Unionist party Leader   Ireland
Horst L Stormer                m Phy Columbia University            Germany

SQL>
```

```
SQL> -- 8. Display the university name(s) that has to its credit by having at least 2
nobel laureate with them.
SQL> SELECT AFF_ROLE
  2  FROM NOBEL
  3  GROUP BY AFF_ROLE
  4  HAVING COUNT(AFF_ROLE) > 1;

AFF_ROLE
-----------------------------
Bell Laboratories
University of California
University of Colorado

SQL>
SQL> -- 9. List the date of birth of youngest and eldest laureates by countrywise. Label
the column as Younger, Elder respectively. Include only the country having more than one
laureate. Sort the output in alphabetical order of country.
SQL> SELECT COUNTRY, MAX(DOB) AS YOUNGEST, MIN(DOB) AS ELDEST
  2  FROM NOBEL
  3  GROUP BY COUNTRY
  4  HAVING (COUNT(COUNTRY) >= 2 )
  5  ORDER BY COUNTRY;

COUNTRY    YOUNGEST  ELDEST
---------- --------- ---------
China      28-FEB-39 04-NOV-33
India      19-AUG-52 03-NOV-33
Ireland    15-OCT-44 18-JAN-37
UK         17-AUG-32 31-OCT-25
USA        19-DEC-61 10-MAY-30

SQL>
SQL>
SQL> -- 10. Show the details (year award,category,field) where the award is shared among
the laureates in the same category and field. Exclude the laureates from USA.
SQL> SELECT CATEGORY, FIELD, YEAR_AWARD
  2  FROM NOBEL
  3  WHERE COUNTRY <> 'USA'
  4  GROUP BY FIELD,CATEGORY,YEAR_AWARD
  5  HAVING(COUNT(*) >= 2);

CAT FIELD                     YEAR_AWARD
--- ------------------------- ----------
Phy Condensed matter                1998
Che Theoretical Chemistry           1998
Pea Negotiation                     1998
Che Biochemistry                    2009

SQL>
SQL>
SQL> -- Use TCL Statements
SQL>
SQL> -- 11.Mark an intermediate point in the transaction (savepoint)
SQL> SAVEPOINT
```

```
  2  save;

Savepoint created.

SQL>
SQL> -- 12.Insert a new tuple into the nobel relation
SQL> INSERT INTO NOBEL
  2  VALUES(129, 'Tony Stark', 'm', 'Phy', 'Physics', 2021, '', '17-DEC-1989', 'USA');

1 row created.

SQL> -- printing updated table
SQL> SELECT *
  2  FROM NOBEL
  3  WHERE NAME='Tony Stark';

LAUREATE_ID NAME                           G CAT FIELD
----------- ------------------------------ - --- -------------------------
YEAR_AWARD AFF_ROLE                       DOB       COUNTRY
---------- ------------------------------ --------- ----------
       129 Tony Stark                      m Phy Physics
      2021                                 17-DEC-89 USA


SQL>
SQL> -- 13.Update the aff_role of literature laureates as 'Linguists'.
SQL>
SQL> SELECT *
  2  FROM NOBEL
  3  WHERE CATEGORY='Lit';

LAUREATE_ID NAME                           G CAT FIELD
----------- ------------------------------ - --- -------------------------
YEAR_AWARD AFF_ROLE                       DOB       COUNTRY
---------- ------------------------------ --------- ----------
       110 Jose Saramago                   m Lit Portuguese
      1998                                 16-NOV-22 Portugal

       117 V S Naipaul                     m Lit English
      2001                                 17-AUG-32 UK

       128 Herta Muller                    f Lit German
      2009                                 17-AUG-53 Romania


SQL>
SQL> UPDATE NOBEL SET aff_role='Linguists' WHERE CATEGORY='Lit';

3 rows updated.

SQL>
SQL> -- printing updated table
SQL> SELECT *
  2  FROM NOBEL
```

```
  3  WHERE CATEGORY='Lit';

LAUREATE_ID NAME                           G CAT FIELD
----------- ------------------------------ - --- -------------------------
YEAR_AWARD AFF_ROLE                       DOB       COUNTRY
---------- ------------------------------ --------- ----------
        110 Jose Saramago                  m Lit Portuguese
      1998 Linguists                       16-NOV-22 Portugal

        117 V S Naipaul                    m Lit English
      2001 Linguists                       17-AUG-32 UK

        128 Herta Muller                   f Lit German
      2009 Linguists                       17-AUG-53 Romania


SQL> SELECT *
  2  FROM NOBEL;

LAUREATE_ID NAME                           G CAT FIELD
----------- ------------------------------ - --- -------------------------
YEAR_AWARD AFF_ROLE                       DOB       COUNTRY
---------- ------------------------------ --------- ----------
        100 Robert B. Laughlin             m Phy Condensed matter
      1998 Stanford University             01-NOV-50 USA

        101 Horst L Stormer                m Phy Condensed matter
      1998 Columbia University             06-APR-49 Germany

        102 Daniel C. Tsui                 m Phy Condensed matter
      1998 Princeton University            28-FEB-39 China


LAUREATE_ID NAME                           G CAT FIELD
----------- ------------------------------ - --- -------------------------
YEAR_AWARD AFF_ROLE                       DOB       COUNTRY
---------- ------------------------------ --------- ----------
        103 Walter Kohn                    m Che Theoretical Chemistry
      1998 University of California        09-MAR-23 Austria

        104 John Pople                     m Che Theoretical Chemistry
      1998 North Western University        31-OCT-25 UK

        106 John Hume                      m Pea Negotiation
      1998 Labour party Leader             18-JAN-37 Ireland


LAUREATE_ID NAME                           G CAT FIELD
----------- ------------------------------ - --- -------------------------
YEAR_AWARD AFF_ROLE                       DOB       COUNTRY
---------- ------------------------------ --------- ----------
        107 David Trimble                  m Pea Negotiation
      1998 Ulster Unionist party Leader    15-OCT-44 Ireland
```

```
       108 Louis J Ignaroo                m Med Cardiovascular system
      1998 University of California       31-MAY-41 USA

       109 Amartya Sen                    m Eco Welfare Economics
      1998 Trinity College                03-NOV-33 India


LAUREATE_ID NAME                          G CAT FIELD
----------- ------------------------------ - --- -------------------------
YEAR_AWARD AFF_ROLE                       DOB       COUNTRY
---------- ------------------------------ --------- ----------
       110 Jose Saramago                  m Lit Portuguese
      1998 Linguists                      16-NOV-22 Portugal

       111 Eric A Cornell                 m Phy Atomic physics
      2001 University of Colorado         19-DEC-61 USA

       112 Carl E Wieman                  m Phy Atomic physics
      2001 University of Colorado         26-MAR-51 USA


LAUREATE_ID NAME                          G CAT FIELD
----------- ------------------------------ - --- -------------------------
YEAR_AWARD AFF_ROLE                       DOB       COUNTRY
---------- ------------------------------ --------- ----------
       113 Ryoji Noyori                   m Che Organic Chemistry
      2001 Nagoya University              03-SEP-38 Japan

       114 K Barry Sharpless              m Che Organic Chemistry
      2001 Scripps Research Institute     28-APR-41 USA

       115 Kofi Annan                     m Pea World organizing
      2001 UN General                     08-APR-38 Ghana


LAUREATE_ID NAME                          G CAT FIELD
----------- ------------------------------ - --- -------------------------
YEAR_AWARD AFF_ROLE                       DOB       COUNTRY
---------- ------------------------------ --------- ----------
       116 Joerge A Akeriof               m Eco Economic of Information
      2001 University of California       17-JUN-40 USA

       117 V S Naipaul                    m Lit English
      2001 Linguists                      17-AUG-32 UK

       118 Charles A Kao                  m Phy Fiber technology
      2009 University of Hongkong         04-NOV-33 China


LAUREATE_ID NAME                          G CAT FIELD
----------- ------------------------------ - --- -------------------------
YEAR_AWARD AFF_ROLE                       DOB       COUNTRY
---------- ------------------------------ --------- ----------
       119 Willard S Boyle                m Phy Semiconductor technology
```

```
      2009 Bell Laboratories              19-AUG-24 Canada

        120 George E Smith                 m Phy Semiconductor technology
      2009 Bell Laboratories              10-MAY-30 USA

        121 Venkatraman Ramakrishnan       m Che Biochemistry
      2009 MRC Laboratory                 19-AUG-52 India


LAUREATE_ID NAME                           G CAT FIELD
----------- ------------------------------ - --- -------------------------
YEAR_AWARD AFF_ROLE                        DOB       COUNTRY
---------- ------------------------------ --------- ----------
        122 Ada E Yonath                   f Che Biochemistry
      2009 Weizmann Institute of Science   22-JUN-39 Isreal

        123 Elizabeth H Blackburn          f Med Enzymes
      2009 University of California        26-NOV-48 Australia

        124 Carol W Greider                f Med Enzymes
      2009 Johns Hopkins University        15-APR-61 USA


LAUREATE_ID NAME                           G CAT FIELD
----------- ------------------------------ - --- -------------------------
YEAR_AWARD AFF_ROLE                        DOB       COUNTRY
---------- ------------------------------ --------- ----------
        125 Barack H Obama                 m Pea World organizing
      2009 President of USA                04-AUG-61 USA

        126 Oliver E Williamson            m Eco Economic governance
      2009 University of California        27-SEP-32 USA

        127 Elinor Ostrom                  m Eco Economic governance
      2009 Indiana University              07-AUG-33 USA


LAUREATE_ID NAME                           G CAT FIELD
----------- ------------------------------ - --- -------------------------
YEAR_AWARD AFF_ROLE                        DOB       COUNTRY
---------- ------------------------------ --------- ----------
        128 Herta Muller                   f Lit German
      2009 Linguists                       17-AUG-53 Romania

        129 Tony Stark                     m Phy Physics
      2021                                 17-DEC-89 USA


29 rows selected.

SQL>
SQL> -- 14.Delete the laureate(s) who was awarded in Enzymes field
SQL> SELECT *
  2  FROM NOBEL
```

```
  3  WHERE FIELD='Enzymes';

LAUREATE_ID NAME                           G CAT FIELD
----------- ------------------------------ - --- -------------------------
YEAR_AWARD AFF_ROLE                       DOB       COUNTRY
---------- ------------------------------ --------- ----------
        123 Elizabeth H Blackburn          f Med Enzymes
       2009 University of California       26-NOV-48 Australia

        124 Carol W Greider                f Med Enzymes
       2009 Johns Hopkins University       15-APR-61 USA


SQL>
SQL> DELETE FROM NOBEL
  2  WHERE FIELD='Enzymes';

2 rows deleted.

SQL>
SQL> -- printing updated table
SQL> SELECT *
  2  FROM NOBEL
  3  WHERE FIELD='Enzymes';

no rows selected

SQL>
SQL> -- 15.Discard the most recent update operations (rollback)
SQL>
SQL> ROLLBACK
  2  TO
  3  save;

Rollback complete.

SQL>
SQL> -- printing table after rollback
SQL> SELECT *
  2  FROM NOBEL
  3  WHERE FIELD='Enzymes';

LAUREATE_ID NAME                           G CAT FIELD
----------- ------------------------------ - --- -------------------------
YEAR_AWARD AFF_ROLE                       DOB       COUNTRY
---------- ------------------------------ --------- ----------
        123 Elizabeth H Blackburn          f Med Enzymes
       2009 University of California       26-NOV-48 Australia

        124 Carol W Greider                f Med Enzymes
       2009 Johns Hopkins University       15-APR-61 USA


SQL>
```

```
SQL>
SQL> -- 16.Commit the changes
SQL> Commit;

Commit complete.

SQL> spool off
```

	SSN COLLEGE OF ENGINEERING ***Department of Computer Science & Engineering***	***Faculty:*** ***P.Mirunalini, Asso. Prof.*** ***N.Sujaudeen, Asst. Prof***
	CS8481 – DBMS Lab ***Assignment – 3***	***Assigned: 17-03-22***
Title: Advanced DML – using Joins, Sub queries, Set Operations		

Bakery Database

Consider the following relations for the Bakery database:

CUSTOMERS (*cid*, fname, lname)

PRODUCTS (*pid*, flavor, food, price)

RECEIPTS (*rno*, rdate, cid)

ITEM_LIST (*rno, ordiNAl*, item)

- Understand the database through README_BAKERY.txt file.
- Draw schema diagram for Bakery database.
- Create relations with appropriate data types and integrity constraints.
- Populate the database values using the ***BAkery.sql*** file.

Write the following using Sub-query:

1. Display the food details that is not purchased by any of customers.
2. Show the customer details who had placed more than 2 orders on the same date.
3. Display the products details that has been ordered maximum by the customers. (use ALL)
4. Show the number of receipts that contain the product whose price is more than the average price of its food type.

Write the following using JOIN: (Use sub-query if required)

5. Display the customer details along with receipt number and date for the receipts that are dated on the last day of the receipt month.
6. Display the receipt number(s) and its total price for the receipt(s) that contain Twist as one among five items. Include only the receipts with total price more than $25.
7. Display the details (customer details, receipt number, item) for the product that was

purchased by the least number of customers.

8. Display the customer details along with the receipt number who ordered all the flavors of *Meringue* in the same receipt.

Write the following using Set Operations:

9. Display the product details of both *Pie* and *BEAR CLAW*.
10. Display the customers details who haven't placed any orders.
11. Display the food that has the same flavor as that of the common flavor between the *Meringue* and *TART*.

What you have to submit:

1. Schema Diagram with constraints
2. Demo script file

SSN COLLEGE OF ENGINEERING, KALAVAKKAM
(An Autonomous Institution, Affiliated to Anna University, Chennai)

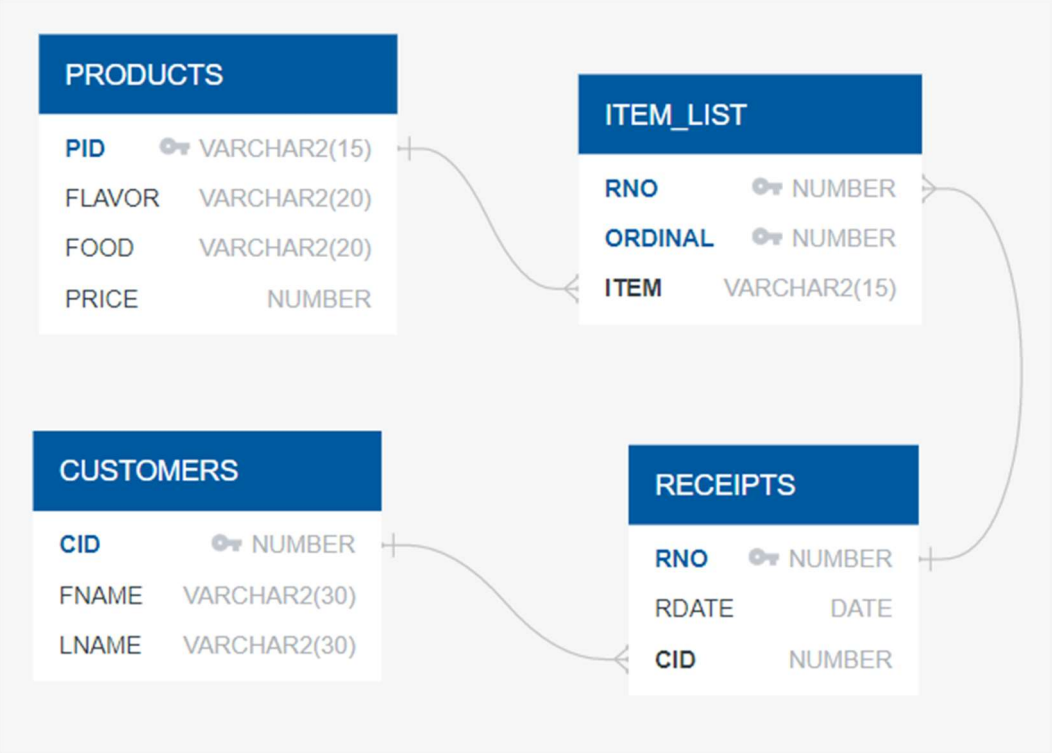
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UCS1412 – DBMS Lab
Assignment – 3

Name: Jayannthan P T *Dept: CSE 'A'* *Roll No.: 205001049*

Title: Bakery Database

Schema Diagram:



Spool File Output for Table Creation:

```
SQL> @D:/assn3create;
SQL> --  DROPPING ALL TABLES
SQL> DROP TABLE ITEM_LIST;

Table ITEM_LIST dropped.

SQL> DROP TABLE RECEIPTS;

Table RECEIPTS dropped.

SQL> DROP TABLE PRODUCTS;

Table PRODUCTS dropped.

SQL> DROP TABLE CUSTOMERS;
```

```
Table CUSTOMERS dropped.

SQL>
SQL> --  CREATING CUSTOMERS TABLE
SQL> CREATE TABLE CUSTOMERS
  2  (
  3      CID NUMBER CONSTRAINT CUSID_PKEY PRIMARY KEY,
  4      FNAME VARCHAR2(30),
  5      LNAME VARCHAR2(30)
  6  );

Table CUSTOMERS created.

SQL>
SQL> --  DISPLAYING THE ATTRIBUTES AND THEIR DATA TYPES OF THE CUSTOMERS TABLE
SQL> DESC CUSTOMERS;
Name  Null?    Type
----- -------- ------------
CID   NOT NULL NUMBER
FNAME          VARCHAR2(30)
LNAME          VARCHAR2(30)
SQL>
SQL> --  CREATING PRODUCTS TABLE
SQL> CREATE TABLE PRODUCTS
  2  (
  3      PID VARCHAR2(15) CONSTRAINT PRODUCTS_PKEY PRIMARY KEY,
  4      FLAVOR VARCHAR2(20),
  5      FOOD VARCHAR2(20),
  6      PRICE NUMBER
  7  );

Table PRODUCTS created.

SQL>
SQL> --  DISPLAYING THE ATTRIBUTES AND THEIR DATA TYPES OF THE PRODUCTS TABLE
SQL> DESC PRODUCTS;
Name   Null?    Type
------ -------- ------------
PID    NOT NULL VARCHAR2(15)
FLAVOR          VARCHAR2(20)
FOOD            VARCHAR2(20)
PRICE           NUMBER
SQL>
SQL> --  CREATING RECEIPTS TABLE
SQL> CREATE TABLE RECEIPTS
  2  (
  3      RNO NUMBER CONSTRAINT RNO_PKEY PRIMARY KEY,
  4      RDATE DATE,
  5      CID NUMBER CONSTRAINT CUSID_FORKEY REFERENCES CUSTOMERS(CID)
  6  );

Table RECEIPTS created.
```

```
SQL>
SQL> --  DISPLAYING THE ATTRIBUTES AND THEIR DATA TYPES OF THE RECEIPTS TABLE
SQL> DESC RECEIPTS;
Name  Null?    Type
----- -------- ------
RNO   NOT NULL NUMBER
RDATE          DATE
CID            NUMBER
SQL>
SQL> --  CREATING TABLE ITEM_LIST
SQL> CREATE TABLE ITEM_LIST
  2  (
  3      RNO NUMBER CONSTRAINT RECEIPTNO_FORKEY REFERENCES RECEIPTS(RNO),
  4      ORDINAL NUMBER CONSTRAINT ITEM_CHK CHECK(ORDINAL!=0),
  5      ITEM VARCHAR2(15) CONSTRAINT PID_FORKEY REFERENCES PRODUCTS(PID),
  6      CONSTRAINT ITEM_LIST_FORKEY PRIMARY KEY(RNO,ORDINAL)
  7  );

Table ITEM_LIST created.

SQL>
SQL> --  DISPLAYING THE ATTRIBUTES AND THEIR DATA TYPES OF THE ITEM_LIST TABLE
SQL> DESC ITEM_LIST;
Name    Null?    Type
------- -------- ------------
RNO     NOT NULL NUMBER
ORDINAL NOT NULL NUMBER
ITEM             VARCHAR2(15)


SQL> spool off;
```

Spool File Output for insertion into Customers table:

```
SQL> -- REM Population of Bakery Database
SQL> -- REM ********************************************************************
SQL> -- REM CUSTOMERS ( customer number,  Last name, First name)
SQL> -- REM ********************************************************************
SQL>
SQL> insert into customers values(1, 'LOGAN', 'JULIET');

1 row inserted.

SQL> insert into customers values(2, 'ARZT', 'TERRELL');

1 row inserted.

SQL> insert into customers values(3, 'ESPOSITA', 'TRAVIS');

1 row inserted.

SQL> insert into customers values(4, 'ENGLEY', 'SIXTA');
```

```
1 row inserted.

SQL> insert into customers values(5, 'DUNLOW', 'OSVALDO');

1 row inserted.
```

Spool File Output for insertion into Products table:

```
SQL>
SQL> -- REM ------------------------------------------------------------------------
SQL> -- REM PRODUCTS (product number,  Flavor, Food, Price)
SQL> -- REM ------------------------------------------------------------------------
SQL>
SQL> insert into products values('20-BC-C-10','Chocolate','Cake',8.95);

1 row inserted.

SQL> insert into products values('20-BC-L-10','Lemon','Cake',8.95);

1 row inserted.

SQL> insert into products values('20-CA-7.5','Casino','Cake',15.95);

1 row inserted.

SQL> insert into products values('24-8x10','Opera','Cake',15.95);

1 row inserted.

SQL> insert into products values('25-STR-9','Strawberry','Cake',11.95);

1 row inserted.
```

Spool File Output for insertion into Receipts table:

```
SQL>
SQL> -- REM ------------------------------------------------------------------------
SQL> -- REM RECEIPTS(receipt number, receipt Date, Customer)
SQL> -- REM ------------------------------------------------------------------------
SQL>
SQL> INSERT INTO Receipts values(18129, '28-Oct-2007', 15);

1 row inserted.

SQL> INSERT INTO Receipts values(51991, '17-Oct-2007', 14);

1 row inserted.

SQL> INSERT INTO Receipts values(83085, '12-Oct-2007', 7);

1 row inserted.

SQL> INSERT INTO Receipts values(70723, '28-Oct-2007', 20);
```

```
1 row inserted.

SQL> INSERT INTO Receipts values(13355, '19-Oct-2007', 7);

1 row inserted.
```

Spool File Output for insertion into ITEM_LIST table:

```
SQL>
SQL> -- REM ------------------------------------------------------------------------
SQL> -- REM ITEM_LIST (receipt number,  Ordinal, Item)
SQL> -- REM ------------------------------------------------------------------------
SQL>
SQL> insert into item_list values(18129, 1,  '70-TU');

1 row inserted.

SQL> insert into item_list values(51991, 1,  '90-APIE-10');

1 row inserted.

SQL> insert into item_list values(51991, 2,  '90-CH-PF');

1 row inserted.

SQL> insert into item_list values(51991, 3,  '90-APP-11');

1 row inserted.
```

Spool File Output for given Questions:

```
SQL>
SQL> @"D:\assn3Qns.sql"
SQL> -- ASSIGNMENT 3
SQL>
SQL> -- WRITE THE FOLLOWING USING SUB-QUERY:
SQL> -- 1. DISPLAY THE FOOD DETAILS THAT IS NOT PURCHASED BY ANY OF CUSTOMERS.
SQL>
SQL> SELECT * FROM PRODUCTS
  2  WHERE PID NOT IN(SELECT ITEM FROM ITEM_LIST);

PID             FLAVOR               FOOD                      PRICE
--------------- -------------------- -------------------- ----------
20-BC-C-10      Chocolate            Cake                        8.95

SQL>
SQL> -- 2. SHOW THE CUSTOMER DETAILS WHO HAD PLACED MORE THAN 2 ORDERS ON THE SAME DATE.
SQL>
SQL> SELECT * FROM CUSTOMERS
  2  WHERE CID IN(SELECT CID FROM RECEIPTS
  3  GROUP BY CID,RDATE
```

```
  4  HAVING COUNT(*)>2);

       CID FNAME                          LNAME
---------- ------------------------------ ------------------------------
        14 SOPKO                          RAYFORD
         8 HELING                         RUPERT

SQL>
SQL> -- 3. DISPLAY THE PRODUCTS DETAILS THAT HAS BEEN ORDERED MAXIMUM BY THE CUSTOMERS.
(USE
SQL> -- ALL)
SQL>
SQL> SELECT * FROM PRODUCTS
  2  WHERE PID IN (SELECT ITEM FROM ITEM_LIST GROUP BY ITEM
  3  HAVING COUNT(*) >= ALL (SELECT COUNT(*) FROM ITEM_LIST GROUP BY ITEM));

PID             FLAVOR               FOOD                      PRICE
--------------- -------------------- -------------------- ----------
90-APP-11       Apple                Tart                        3.25

SQL>
SQL>
SQL> -- 4. SHOW THE NUMBER OF RECEIPTS THAT CONTAIN THE PRODUCT WHOSE PRICE IS MORE THAN
THE
SQL> -- AVERAGE PRICE OF ITS FOOD TYPE.
SQL>
SQL> SELECT COUNT(DISTINCT RNO) AS RECEIPT_COUNT FROM ITEM_LIST
  2  WHERE ITEM IN (SELECT PID  FROM PRODUCTS OUTER
  3  WHERE PRICE > (SELECT AVG(PRICE) FROM PRODUCTS WHERE FOOD=OUTER.FOOD));

RECEIPT_COUNT
-------------
          137

SQL>
SQL>
SQL> -- WRITE THE FOLLOWING USING JOIN: (USE SUB-QUERY IF REQUIRED)
SQL> -- 5. DISPLAY THE CUSTOMER DETAILS ALONG WITH RECEIPT NUMBER AND DATE FOR THE
RECEIPTS THAT
SQL> -- ARE DATED ON THE LAST DAY OF THE RECEIPT MONTH.
SQL>
SQL> SELECT DISTINCT CID,FNAME,LNAME,RNO,RDATE
  2  FROM CUSTOMERS C NATURAL JOIN RECEIPTS R
  3  WHERE R.RDATE=LAST_DAY(R.RDATE);

       CID FNAME                          LNAME                               RNO
RDATE
---------- ------------------------------ ------------------------------ ---------- ------
--
        11 STADICK                        MIGDALIA                            60270 31-10-
07
        20 ZEME                           STEPHEN                             49845 31-10-
07
```

```
         3 ESPOSITA                      TRAVIS                                39829 31-10-
07
        19 STENZ                         NATACHA                               36343 31-10-
07
        12 MCMAHAN                       MELLIE                                70796 31-10-
07
         1 LOGAN                         JULIET                                85858 31-10-
07

6 rows selected.

SQL>
SQL> -- 6. DISPLAY THE RECEIPT NUMBER(S) AND ITS TOTAL PRICE FOR THE RECEIPT(S) THAT
CONTAIN TWIST
SQL> -- AS ONE AMONG FIVE ITEMS. INCLUDE ONLY THE RECEIPTS WITH TOTAL PRICE MORE THAN $25.
SQL>
SQL> SELECT RNO, SUM(PRICE) AS TOTALPRICE FROM ITEM_LIST I JOIN PRODUCTS P ON (I.ITEM =
P.PID)
  2  WHERE RNO IN (SELECT RNO FROM ITEM_LIST I JOIN PRODUCTS P ON (I.ITEM = P.PID) WHERE
FOOD='Twist')
  3  GROUP BY RNO HAVING FLOOR(SUM(PRICE)) >25;

       RNO TOTALPRICE
---------- ----------
     83085      48.25

SQL>
SQL> -- 7. DISPLAY THE DETAILS (CUSTOMER DETAILS, RECEIPT NUMBER, ITEM) FOR THE PRODUCT
THAT WAS PURCHASED BY THE LEAST NUMBER OF CUSTOMERS.
SQL>
SQL> SELECT DISTINCT CUSTOMERS.CID,FNAME,LNAME,RECEIPTS.RNO,ITEM
  2  FROM CUSTOMERS,RECEIPTS,ITEM_LIST
  3  WHERE CUSTOMERS.CID=RECEIPTS.CID AND RECEIPTS.RNO=ITEM_LIST.RNO AND ITEM IN
  4  (SELECT ITEM FROM ITEM_LIST GROUP BY ITEM HAVING COUNT(*)=(SELECT MIN(COUNT(*)) FROM
ITEM_LIST GROUP BY ITEM));

       CID FNAME                           LNAME                                   RNO
ITEM
---------- ------------------------------ ------------------------------ ---------- ------
---------
        20 ZEME                           STEPHEN                               49845 50-
CH
        18 DOMKOWSKI                      ALMETA                                82056 50-
CH
        18 DOMKOWSKI                      ALMETA                                73716 50-
CH
         6 SLINGLAND                      JOSETTE                               99994 50-
CH
        14 SOPKO                          RAYFORD                               77032 50-
CH
         8 HELING                         RUPERT                                95962 50-
CH

6 rows selected.
```

```
SQL>
SQL> -- 8. DISPLAY THE CUSTOMER DETAILS ALONG WITH THE RECEIPT NUMBER WHO ORDERED ALL THE
FLAVORS OF MERINGUE IN THE SAME RECEIPT.
SQL>
SQL> SELECT DISTINCT CID, FNAME,LNAME,RNO FROM CUSTOMERS NATURAL JOIN RECEIPTS
  2  WHERE RNO IN (SELECT RNO FROM (SELECT DISTINCT RNO, FLAVOR FROM PRODUCTS JOIN
ITEM_LIST ON(ITEM=PID)
  3  WHERE FOOD='Meringue' GROUP BY RNO,FLAVOR) GROUP BY RNO HAVING COUNT(*)>1);

       CID FNAME                          LNAME                               RNO
---------- ------------------------------ ------------------------------ ----------
         8 HELING                         RUPERT                              61797

SQL>
SQL> -- WRITE THE FOLLOWING USING SET OPERATIONS:
SQL> -- 9. DISPLAY THE PRODUCT DETAILS OF BOTH PIE AND BEAR CLAW.
SQL>
SQL>
SQL> SELECT * FROM PRODUCTS WHERE FOOD = 'Pie'
  2  UNION
  3  SELECT * FROM PRODUCTS WHERE FOOD = 'Bear Claw';

PID             FLAVOR               FOOD                      PRICE
--------------- -------------------- -------------------- ----------
51-BC           Almond               Bear Claw                   1.95
90-APIE-10      Apple                Pie                         5.25

SQL>
SQL> -- 10.DISPLAY THE CUSTOMERS DETAILS WHO HAVEN'T PLACED ANY ORDERS.
SQL>
SQL> SELECT * FROM CUSTOMERS
  2  MINUS (SELECT CID, FNAME, LNAME
  3  FROM CUSTOMERS NATURAL JOIN RECEIPTS);

       CID FNAME                          LNAME
---------- ------------------------------ ------------------------------
        21 JOHN                           DAVID

SQL>
SQL> -- 11.DISPLAY THE FOOD THAT HAS THE SAME FLAVOR AS THAT OF THE COMMON FLAVOR BETWEEN
THE
SQL> -- MERINGUE AND TART.
SQL>
SQL> SELECT FOOD FROM PRODUCTS
  2  WHERE FLAVOR IN (SELECT FLAVOR FROM PRODUCTS
  3  WHERE FOOD = 'Meringue'
  4  INTERSECT
  5  SELECT FLAVOR FROM PRODUCTS
  6  WHERE FOOD = 'Tart');

FOOD
--------------------
Cake
```

```
Eclair
Tart
Meringue
Croissant

SQL>
SQL> spool off;
```

	SSN COLLEGE OF ENGINEERING ***Department of Computer Science & Engineering***	***Faculty:*** ***P.Mirunalini, Asso. Prof.*** ***N.Sujaudeen, Asst. Prof***
CS8481 – DBMS Lab ***Assignment – 4***		***Assigned: 04-Apr-22*** ***Due: 1 Lab Hour***
Title: Views		

Aim:

a) To create view(s) based on table(s) or view(s) and observe its behavior while performing update operations on it.

Consider the schema used in the Assignment-3.

Create the following ***views*** and perform DML operations on it. Classify whether the view is *updatable or not*.

1. Create a view named **Blue_Flavor,** which display the product details (product id, food, price) of Blueberry flavor.
2. Create a view named **Cheap_Food,** which display the details (product id, flavor, food, price) of products with price lesser than $1. Ensure that, the price of these food(s) should never rise above $1 through view.
3. Create a view called **Hot_Food** that show the product id and its quantity where the same product is ordered more than once in the same receipt.
4. Create a view named **Pie_Food** that will display the details (customer lname, flavor, receipt number and date, ordinal) who had ordered the Pie food with receipt details.
5. Create a view **Cheap_View** from **Cheap_Food** that shows only the product id, flavor and food.
6. Drop the view Cheap_View

Note: Notify the changes reflected in the base tables when you update through the view. Usethe following format to record the view behavior:

View 1 : (testview)

Observation / Operation	Operation on View	Reflection in Base Table
INSERT	√	√
UPDATE	attr1, attr3	attr1, attr3
DELETE	√	√

RESULT : The View 1 is a updatable-join view

General Guidelines

i) Place a tick [✓] mark if the operation is successful through the view/base table OR reflected in view/base table.

ii) Place a cross [X] mark, if the operation is not allowed through view/base table OR is not reflected in view/base table.

iii) For Insert/Update operation, only if some (or subset) of attributes are permitted, mention those attributes in the corresponding column.

Classify the views as follows:

i) updatable /updatable join view – views whose rows can be modified
ii) insertable-into view – views into which only new rows can be inserted
iii) To view the meta-data of views:
USER_VIEWS, USER_UPDATABLE_COLUMNS

What you have to submit:

1. Schema Diagram with constraints
2. Demo script file

SSN COLLEGE OF ENGINEERING, KALAVAKKAM
(An Autonomous Institution, Affiliated to Anna University, Chennai)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UCS1412 – DBMS Lab
Assignment – 4

Name: Jayannthan P T *Dept: CSE 'A'* *Roll No.: 205001049*

Title: Views

Blue_Flavor		
Operation	Operation on View	Reflection in Base Table
Insert	✗	✓
Update	PID, FOOD, PRICE	PID, FOOD, PRICE
Modify	✓	✓

Cheap_Food		
Operation	Operation on View	Reflection in Base Table
Insert	✓	✓
Update	PID, FLAVOR, FOOD, PRICE	PID, FLAVOR, FOOD, PRICE
Modify	✓	✓

Hot_Food		
Operation	Operation on View	Reflection in Base Table
Insert	✗	✗
Update	✗	✗
Modify	✗	✗

Pie_Food

Operation	Operation on View	Reflection in Base Table
Insert	✗ (non-key preserved table)	✗
Update	✓	✓ (On ITEM_LIST)
Modify	✓	✓

Cheap_View		
Operation	Operation on View	Reflection in Base Table
Insert	✗ (cannot insert without price)	✗
Update	PID, FLAVOR, FOOD	PID, FLAVOR, FOOD
Modify	✓	✓

Spool File:

```
SQL> @Z:assn4_views.sql
SQL> -- Consider the schema used in the Assignment3.
SQL>
SQL> -- Create the following views and perform DML operations on it. Classify whether the
view is updatable or not.
SQL>
SQL> -- 1. Create a view named Blue_Flavor, which display the product details (product id,
SQL> -- food, price) of Blueberry flavor.
SQL>
SQL> DROP VIEW Blue_Flavor;

View dropped.

SQL> -- Creating view
SQL> CREATE VIEW Blue_Flavor
  2  AS
  3    SELECT PID, FOOD, PRICE
  4    FROM PRODUCTS
  5    WHERE FLAVOR='Blueberry';

View created.

SQL>
SQL> -- DESC Blue_Flavor;
SQL>
SQL> --displaying items in Blue_Flavor
SQL> SELECT *
```

```
  2  FROM Blue_Flavor;

PID             FOOD                      PRICE
--------------- -------------------- ----------
90-BLU-11       Tart                        3.25
51-BLU          Danish                      1.15

SQL>
SQL> --viewing the updatable columns in Blue_Flavor
SQL> SELECT COLUMN_NAME, UPDATABLE
  2  FROM USER_UPDATABLE_COLUMNS
  3  WHERE
  4  TABLE_NAME='BLUE_FLAVOR';

COLUMN_NAME                    UPD
------------------------------ ---
PID                            YES
FOOD                           YES
PRICE                          YES

SQL>
SQL> --  CREATING A SAVEPOINT
SQL> SAVEPOINT VIEW1;

Savepoint created.

SQL>
SQL> --inserting a tuple in products with flavour="Blueberry"
SQL> INSERT INTO PRODUCTS
  2  VALUES
  3    ('20-Ch', 'Blueberry', 'Chocolate', 15);

1 row created.

SQL>
SQL>
SQL> --displaying items in Blue_Flavor after adding tuple in products
SQL> SELECT *
  2  FROM Blue_Flavor;

PID             FOOD                      PRICE
--------------- -------------------- ----------
90-BLU-11       Tart                        3.25
51-BLU          Danish                      1.15
20-Ch           Chocolate                     15

SQL>
SQL> --updating a tuple in Blue_Flavor
SQL> UPDATE Blue_Flavor
  2  SET PRICE=PRICE*1.15
  3  WHERE PID='20-Ch';

1 row updated.
```

```
SQL>
SQL> --displaying items in Blue_Flavor after updating
SQL> SELECT *
  2  FROM Blue_Flavor
  3  WHERE PID ='20-Ch';

PID             FOOD                      PRICE
--------------- ------------------------- ----------
20-Ch           Chocolate                 17.25

SQL>
SQL> --displaying items in base table after updating
SQL> SELECT *
  2  FROM PRODUCTS
  3  WHERE PID ='20-Ch';

PID             FLAVOR                    FOOD                      PRICE
--------------- ------------------------- ------------------------- ----------
20-Ch           Blueberry                 Chocolate                 17.25

SQL>
SQL>
SQL> --deleting a tuple in Blue_Flavor
SQL> DELETE FROM Blue_Flavor WHERE PID='20-Ch';

1 row deleted.

SQL>
SQL> --displaying items in Blue_Flavor after deleting
SQL> SELECT *
  2  FROM Blue_Flavor
  3  WHERE PID ='20-Ch';

no rows selected

SQL>
SQL> --displaying items in base table after deleting
SQL> SELECT *
  2  FROM PRODUCTS
  3  WHERE PID ='20-Ch';

no rows selected

SQL>
SQL> --ROLLINGBACK TO SAVEPOINT
SQL> ROLLBACK
  2  TO VIEW1;

Rollback complete.

SQL>
SQL> --SAVING PROGRESS
SQL> COMMIT;
```

```
Commit complete.

SQL>
SQL> -- 2. Create a view named Cheap_Food, which display the details (product id, flavor,
SQL> -- food, price) of products with price lesser than $1. Ensure that, the price of
these
SQL> -- food(s) should never rise above $1 through view.
SQL>
SQL> DROP VIEW Cheap_Food;

View dropped.

SQL> -- Creating view
SQL> CREATE VIEW Cheap_Food
  2  AS
  3    SELECT PID, FLAVOR, FOOD, PRICE
  4    FROM PRODUCTS
  5    WHERE PRICE < 1
  6  WITH CHECK OPTION;

View created.

SQL>
SQL> -- DESC Cheap_Food;
SQL>
SQL> -- Displaying the contents of the Cheap_Food
SQL> SELECT *
  2  FROM Cheap_Food;

PID             FLAVOR               FOOD                      PRICE
--------------- -------------------- -------------------- ----------
70-LEM          Lemon                Cookie                      .79
70-W            Walnut               Cookie                      .79

SQL>
SQL> --viewing the updatable columns in Cheap_Food
SQL> SELECT COLUMN_NAME, UPDATABLE
  2  FROM USER_UPDATABLE_COLUMNS
  3  WHERE
  4  TABLE_NAME='CHEAP_FOOD';

COLUMN_NAME                    UPD
------------------------------ ---
PID                            YES
FLAVOR                         YES
FOOD                           YES
PRICE                          YES

SQL>
SQL> --  CREATING A SAVEPOINT
SQL> SAVEPOINT VIEW2;

Savepoint created.
```

```
SQL>
SQL>
SQL> --inserting a tuple in Cheap_Food
SQL>
SQL> insert INTO Cheap_Food
  2  values
  3    ('20-Ch1', 'Mango', 'Chocolate', 0.75);

1 row created.

SQL>
SQL>
SQL> --inserting a tuple in Cheap_Food with price>1
SQL>
SQL> insert INTO Cheap_Food
  2  values
  3    ('20-Ch2', 'Man', 'Choco', 1.75);
insert INTO Cheap_Food
           *
ERROR at line 1:
ORA-01402: view WITH CHECK OPTION where-clause violation


SQL>
SQL>
SQL> --displaying items in Cheap_Food after insertion
SQL> SELECT *
  2  FROM Cheap_Food
  3  WHERE PID='20-Ch1';

PID             FLAVOR               FOOD                      PRICE
--------------- -------------------- -------------------- ----------
20-Ch1          Mango                Chocolate                   .75

SQL>
SQL> --displaying items in base table after insertion
SQL> SELECT *
  2  FROM PRODUCTS
  3  WHERE PID='20-Ch1';

PID             FLAVOR               FOOD                      PRICE
--------------- -------------------- -------------------- ----------
20-Ch1          Mango                Chocolate                   .75

SQL>
SQL> --updating a tuple in Cheap_Food
SQL> UPDATE Cheap_Food
  2  SET PRICE=PRICE+0.1
  3  WHERE PID='20-Ch1';

1 row updated.

SQL>
SQL> --displaying items in Cheap_Food after updating
```

```
SQL> SELECT *
  2  FROM Cheap_Food
  3  WHERE PID ='20-Ch1';

PID             FLAVOR               FOOD                      PRICE
--------------- -------------------- -------------------- ----------
20-Ch1          Mango                Chocolate                   .85

SQL>
SQL> --displaying items in base table after updating
SQL> SELECT *
  2  FROM PRODUCTS
  3  WHERE PID='20-Ch1';

PID             FLAVOR               FOOD                      PRICE
--------------- -------------------- -------------------- ----------
20-Ch1          Mango                Chocolate                   .85

SQL>
SQL>
SQL> --deleting a tuple in Cheap_Food
SQL> DELETE FROM Cheap_Food WHERE PID='20-Ch1';

1 row deleted.

SQL>
SQL> --displaying items in Cheap_Food after deleting
SQL> SELECT *
  2  FROM Cheap_Food
  3  WHERE PID ='20-Ch1';

no rows selected

SQL>
SQL>
SQL> --displaying items in base table after deleting
SQL> SELECT *
  2  FROM PRODUCTS
  3  WHERE PID='20-Ch1';

no rows selected

SQL>
SQL>
SQL>
SQL>
SQL> --ROLLINGBACK TO SAVEPOINT
SQL> ROLLBACK
  2  TO VIEW2;

Rollback complete.

SQL>
SQL> --SAVING PROGRESS
```

```
SQL> COMMIT;

Commit complete.

SQL>
SQL> -- 3. Create a view called Hot_Food that show the product id and its quantity WHERE
the
SQL> -- same product is ordered more than once in the same receipt.
SQL>
SQL> DROP VIEW Hot_Food;

View dropped.

SQL>
SQL> -- Creating view
SQL> CREATE VIEW Hot_Food
  2  (
  3    pid,
  4    count
  5  )
  6  AS
  7    SELECT ITEM, COUNT(ITEM)
  8    FROM ITEM_LIST
  9    GROUP BY RNO,ITEM
 10    HAVING COUNT(*)>1;

View created.

SQL>
SQL>
SQL> -- DESC Hot_Food;
SQL>
SQL> --displaying items in Hot_Food
SQL> SELECT *
  2  FROM Hot_Food;

PID                 COUNT
--------------- ----------
70-R                     2
90-APR-PF                2
50-APP                   2
51-ATW                   2
90-ALM-I                 2
90-BER-11                2
90-PEC-11                2
70-M-CH-DZ               2
46-11                    2
70-M-CH-DZ               2
90-CHR-11                2

PID                 COUNT
--------------- ----------
90-BLU-11                2
50-CHS                   2
```

```
70-M-CH-DZ                2
70-R                      2
90-APP-11                 2
70-MAR                    2
50-APR                    2
51-BC                     2
50-ALM                    2

20 rows selected.

SQL>
SQL> --viewing the updatable columns in Hot_Food
SQL> SELECT COLUMN_NAME, UPDATABLE
  2  FROM USER_UPDATABLE_COLUMNS
  3  WHERE
  4  TABLE_NAME='HOT_FOOD';

COLUMN_NAME                    UPD
------------------------------ ---
PID                            NO
COUNT                          NO

SQL> -- This table isnt updatable because we have used aggregate functions (group by
class)
SQL>
SQL>
SQL> --  CREATING A SAVEPOINT
SQL> SAVEPOINT VIEW3;

Savepoint created.

SQL>
SQL> --inserting a tuple in Cheap_Food
SQL> INSERT INTO Hot_Food
  2  VALUES('50-AB', 3);
INSERT INTO Hot_Food
*
ERROR at line 1:
ORA-01733: virtual column not allowed here


SQL>
SQL>
SQL> -- updating item in Hot_Food
SQL> UPDATE Hot_Food
  2  SET ITEM='54-SD9-34JS'
  3  WHERE ITEM='46-11';
WHERE ITEM='46-11'
      *
ERROR at line 3:
ORA-00904: "ITEM": invalid identifier


SQL>
```

```
SQL> --updating quantity in Hot_Food
SQL> UPDATE Hot_Food
  2  SET QUANTITY=4
  3  WHERE ITEM='46-11';
WHERE ITEM='46-11'
      *
ERROR at line 3:
ORA-00904: "ITEM": invalid identifier


SQL>
SQL> --deleting a row in Hot_Food
SQL> DELETE FROM Hot_Food WHERE ITEM='46-11';
DELETE FROM Hot_Food WHERE ITEM='46-11'
                            *
ERROR at line 1:
ORA-00904: "ITEM": invalid identifier


SQL>
SQL>
SQL> -- making changes in base table to see reflection
SQL> INSERT INTO PRODUCTS
  2  VALUES('9-B', 'ORANGE', 'JUICE', 9);

1 row created.

SQL> INSERT INTO RECEIPTS
  2  VALUES(12345, '9-JUN-22', 21);

1 row created.

SQL> INSERT INTO ITEM_LIST
  2  VALUES(12345, 98, '9-B');

1 row created.

SQL> INSERT INTO ITEM_LIST
  2  VALUES(12345, 99, '9-B');

1 row created.

SQL>
SQL> SELECT *
  2  from Hot_food;

PID                 COUNT
--------------- ----------
70-R                     2
90-APR-PF                2
50-APP                   2
51-ATW                   2
90-ALM-I                 2
90-BER-11                2
```

```
90-PEC-11                2
70-M-CH-DZ               2
46-11                    2
70-M-CH-DZ               2
90-CHR-11                2

PID                   COUNT
--------------- ----------
90-BLU-11                2
9-B                      2
50-CHS                   2
70-M-CH-DZ               2
70-R                     2
90-APP-11                2
70-MAR                   2
50-APR                   2
51-BC                    2
50-ALM                   2

21 rows selected.

SQL> -- note the changes in counts
SQL> -- making changes in the base table reflects in the view but not vice versa
SQL>
SQL> --ROLLINGBACK TO SAVEPOINT
SQL> ROLLBACK
  2  TO VIEW3;

Rollback complete.

SQL>
SQL> --SAVING PROGRESS
SQL> COMMIT;

Commit complete.

SQL>
SQL>
SQL> -- 4. Create a view named Pie_Food that will display the details (customer lname,
flavor,
SQL> -- receipt number and date, ordinal) who had ordered the Pie food with receipt
details.
SQL>
SQL> DROP VIEW Pie_Food;

View dropped.

SQL>
SQL> -- Creating view
SQL> CREATE VIEW Pie_Food
  2  AS
  3    SELECT LNAME, FLAVOR, I.RNO, RDATE, ORDINAL
  4    FROM CUSTOMERS C, RECEIPTS R, ITEM_LIST I, PRODUCTS P
  5    WHERE C.CID=R.CID AND R.RNO=I.RNO AND I.ITEM=P.PID AND FOOD='Pie';
```

```
View created.

SQL>
SQL> --displaying items in Pie_Food
SQL> SELECT *
  2  FROM Pie_Food;

LNAME                          FLAVOR                    RNO RDATE
------------------------------ -------------------- ---------- ---------
   ORDINAL
----------
JULIET                         Apple                   39685 28-OCT-07
         4

JULIET                         Apple                   66227 10-OCT-07
         2

TRAVIS                         Apple                   48647 09-OCT-07
         2


LNAME                          FLAVOR                    RNO RDATE
------------------------------ -------------------- ---------- ---------
   ORDINAL
----------
JOSETTE                        Apple                   87454 21-OCT-07
         1

JOSETTE                        Apple                   47353 12-OCT-07
         2

RUPERT                         Apple                   53376 30-OCT-07
         3


LNAME                          FLAVOR                    RNO RDATE
------------------------------ -------------------- ---------- ---------
   ORDINAL
----------
CUC                            Apple                   50660 18-OCT-07
         2

KIP                            Apple                   11548 21-OCT-07
         2

RAYFORD                        Apple                   29226 26-OCT-07
         2


LNAME                          FLAVOR                    RNO RDATE
------------------------------ -------------------- ---------- ---------
   ORDINAL
----------
```

```
RAYFORD                        Apple                      51991 17-OCT-07
         1

ARIANE                         Apple                      39109 02-OCT-07
         1

ARIANE                         Apple                      44798 04-OCT-07
         3


LNAME                          FLAVOR                       RNO RDATE
------------------------------ -------------------- ---------- ---------
   ORDINAL
----------
CHARLENE                       Apple                      98806 15-OCT-07
         3


13 rows selected.

SQL>
SQL> --viewing the updatable columns in Pie_Food
SQL> SELECT COLUMN_NAME, UPDATABLE
  2  FROM USER_UPDATABLE_COLUMNS
  3  WHERE
  4  TABLE_NAME='PIE_FOOD';

COLUMN_NAME                    UPD
------------------------------ ---
LNAME                          NO
FLAVOR                         NO
RNO                            YES
RDATE                          NO
ORDINAL                        YES

SQL>
SQL> -- THIS VIEW ISNT UPDATABLE BECAUSE WE HAVE JOINED MULTIPLE TABLE AND THE NOT NULL
COLUMNS FROM PRODUCTS , CUSTOMERS (PK) ARENT A PART OF THE VIEW
SQL>
SQL> --  CREATING A SAVEPOINT
SQL> SAVEPOINT VIEW4;

Savepoint created.

SQL>
SQL>
SQL> --inserting a tuple in Pie_Food
SQL> INSERT INTO Pie_Food
  2  VALUES('Tony', 'Blueberry', 51991, '9-Jun-21', 98);
INSERT INTO Pie_Food
*
ERROR at line 1:
ORA-01779: cannot modify a column which maps to a non key-preserved table
```

```
SQL>
SQL>
SQL> --displaying items in Pie_Food after inserting
SQL> SELECT *
  2  FROM Pie_Food
  3  WHERE LNAME='Steve';

no rows selected

SQL>
SQL>
SQL> --displaying items in base table after inserting
SQL> SELECT LNAME, FLAVOR, R.RNO, RDATE, ORDINAL
  2  FROM CUSTOMERS C, RECEIPTS R, ITEM_LIST I, PRODUCTS P
  3  WHERE C.CID=R.CID AND R.RNO=I.RNO AND I.ITEM=P.PID AND FOOD='Pie' AND LNAME='Tony';

no rows selected

SQL>
SQL>
SQL> --updating a tuple in Pie_Food
SQL> UPDATE Pie_Food SET LNAME='Steve' WHERE RNO=51991;
UPDATE Pie_Food SET LNAME='Steve' WHERE RNO=51991
                    *
ERROR at line 1:
ORA-01779: cannot modify a column which maps to a non key-preserved table


SQL>
SQL> -- updating RNO
SQL> UPDATE Pie_Food SET RNO=51991 WHERE LNAME='Steve';
UPDATE Pie_Food SET RNO=51991 WHERE LNAME='Steve'
                    *
ERROR at line 1:
ORA-01779: cannot modify a column which maps to a non key-preserved table


SQL>
SQL> -- updating ORDINAL
SQL> UPDATE Pie_Food SET ORDINAL=99 WHERE LNAME='Steve';

0 rows updated.

SQL>
SQL> -- updating FLAVOUR
SQL> UPDATE Pie_Food SET FLAVOR='NEW_FLAVOUR' WHERE LNAME='Steve';
UPDATE Pie_Food SET FLAVOR='NEW_FLAVOUR' WHERE LNAME='Steve'
                    *
ERROR at line 1:
ORA-01779: cannot modify a column which maps to a non key-preserved table


SQL>
SQL>
```

```
SQL> --displaying items in Pie_Food after updating
SQL>
SQL> SELECT *
  2  from Pie_Food
  3  WHERE LNAME='Steve';

no rows selected

SQL>
SQL> --displaying items in base table after updating
SQL> SELECT LNAME, FLAVOR, R.RNO, RDATE, ORDINAL
  2  FROM CUSTOMERS C, RECEIPTS R, ITEM_LIST I, PRODUCTS P
  3  WHERE C.CID=R.CID AND R.RNO=I.RNO AND I.ITEM=P.PID AND FOOD='Pie' AND LNAME='Steve';

no rows selected

SQL>
SQL> -- deleting a tuple in view
SQL> SELECT *
  2  FROM Pie_Food
  3  WHERE LNAME='Steve';

no rows selected

SQL>
SQL> --displaying items in Pie_Food after deleting
SQL>
SQL> SELECT *
  2  from Pie_Food
  3  WHERE LNAME='Steve';

no rows selected

SQL>
SQL> --displaying items in base table after deleting
SQL> SELECT LNAME, FLAVOR, R.RNO, RDATE, ORDINAL
  2  FROM CUSTOMERS C, RECEIPTS R, ITEM_LIST I, PRODUCTS P
  3  WHERE C.CID=R.CID AND R.RNO=I.RNO AND I.ITEM=P.PID AND FOOD='Pie' AND LNAME='Steve';

no rows selected

SQL>
SQL>
SQL> --ROLLINGBACK TO SAVEPOINT
SQL> ROLLBACK
  2  TO VIEW4;

Rollback complete.

SQL>
SQL> --SAVING PROGRESS
SQL> COMMIT;

Commit complete.
```

```
SQL>
SQL> -- 5. Create a view Cheap_View from Cheap_Food that shows only the product id, flavor
SQL> -- and food.
SQL>
SQL> -- Creating view
SQL> CREATE VIEW Cheap_View
  2  AS
  3    SELECT PID, FLAVOR, FOOD
  4    FROM CHEAP_FOOD;

View created.

SQL>
SQL> --displaying items in Cheap_View
SQL> SELECT *
  2  FROM Cheap_View;

PID             FLAVOR               FOOD
--------------- -------------------- --------------------
70-LEM          Lemon                Cookie
70-W            Walnut               Cookie

SQL>
SQL> --viewing the updatable columns in Cheap_View
SQL> SELECT COLUMN_NAME, UPDATABLE
  2  FROM USER_UPDATABLE_COLUMNS
  3  WHERE
  4  TABLE_NAME='CHEAP_VIEW';

COLUMN_NAME                    UPD
------------------------------ ---
PID                            YES
FLAVOR                         YES
FOOD                           YES

SQL>
SQL>
SQL> --  CREATING A SAVEPOINT
SQL> SAVEPOINT NEW_VIEW;

Savepoint created.

SQL>
SQL> --inserting a tuple in Cheap_View
SQL> insert INTO Cheap_food
  2  VALUES('20-Ch', 'Mango', 'Chocolate');
insert INTO Cheap_food
           *
ERROR at line 1:
ORA-00947: not enough values


SQL>
```

```
SQL> --displaying items in Cheap_View after inserting
SQL> SELECT *
  2  from Cheap_View
  3  WHERE PID='20-Ch';

no rows selected

SQL>
SQL> --displaying items in Cheap_Food after inserting
SQL> SELECT *
  2  from Cheap_Food
  3  WHERE PID='20-Ch';

no rows selected

SQL>
SQL> --updating a tuple in Pie_Food
SQL>
SQL> UPDATE Cheap_View SET PID='20-Ch1' WHERE PID='20-Ch';

0 rows updated.

SQL> UPDATE Cheap_View SET FLAVOR='NEW_FLAVOUR' WHERE PID='20-Ch1';

0 rows updated.

SQL> UPDATE Cheap_View SET FOOD='New_Food' WHERE PID='20-Ch1';

0 rows updated.

SQL>
SQL> --displaying items in Cheap_View after updating
SQL> SELECT *
  2  from Cheap_View
  3  WHERE PID='20-Ch1';

no rows selected

SQL>
SQL> --displaying items in Cheap_Food after updating
SQL> SELECT *
  2  from Cheap_Food
  3  WHERE PID='20-Ch1';

no rows selected

SQL>
SQL>
SQL> DELETE FROM Cheap_View WHERE PID='20-Ch1';

0 rows deleted.

SQL> DELETE FROM Cheap_View WHERE FOOD='New_Food';
```

```
0 rows deleted.

SQL>
SQL>
SQL> --displaying items in Cheap_View after deleting
SQL> SELECT *
  2  from Cheap_View
  3  WHERE PID='20-Ch1';

no rows selected

SQL>
SQL> --displaying items in Cheap_Food after deleting
SQL> SELECT *
  2  from Cheap_Food
  3  WHERE PID='20-Ch1';

no rows selected

SQL>
SQL>
SQL> --ROLLINGBACK TO SAVEPOINT
SQL> ROLLBACK
  2  TO NEW_VIEW;

Rollback complete.

SQL>
SQL> --SAVING PROGRESS
SQL> COMMIT;

Commit complete.

SQL>
SQL>
SQL>
SQL> -- 6. Drop the view Cheap_View
SQL>
SQL> DROP VIEW Cheap_View;

View dropped.

SQL>
SQL> spool off;
```

	SSN COLLEGE OF ENGINEERING ***Department of Computer Science & Engineering***	***Faculty:*** ***Dr. P.Mirunalini, Asso. Prof.*** ***N. Suajudeen, Asst. Prof***
	UCS1412 – Database Lab ***Assignment – 5***	***Assigned: 18-Apr-22*** ***Due: 1 Lab Hour***
Title: PL/SQL – Control Structures		

Consider the following relations for the Bakery database:

CUSTOMERS (*cid*, fname, lname)

PRODUCTS (*pid*, flavor, food, price)

RECEIPTS (*rno*, rdate, cid)

ITEM_LIST (*rno, ordinal*, item)

Write a PL/SQL block for the following:

Note:

a. Use implicit/explicit cursor wherever required.
b. Handle the **error** and **display appropriate message** if the data is **non-available**.

1. Check whether the given combination of food and flavor is available. If any one or both are not available, display the relevant message.
2. On a given date, find the number of items sold (Use Implicit cursor).
3. An user desired to buy the product with the specific price. Ask the user for a price, find the food item(s) that is equal or closest to the desired price. Print the product number, food type, flavor and price. Also print the number of items that is equal or closest to the desired price.

   Enter value for dprice: 0.8 old 13:
   price:=&dprice;
   new 13: price:=0.8;

ProductID	Food	Flavor	Price
70-LEM	Lemon	Cookie	0.79
70-W	Walnut	Cookie	0.79

   2 product(s) found EQUAL/CLOSEST to given price PL/SQL procedure successfully completed.
4. Display the customer name along with the details of item and its quantity ordered for the given order number. Also calculate the total quantity ordered as shown below:

```
SQL> /
Enter value for rid: 51991
old  11:       rid:=&rid;
new  11:       rid:=51991;

Customer name: SOPKO RAYFORD

Ordered Following Items:

FOOD FLAVOR           QTY
Apple Pie             1
Chocolate Tart        1
Apple Tart            1
Truffle Cake          1
-------------------------
Total Qty:            4
```

What you have to submit:

1. Schema Diagram with constraints
2. Demo script file

SSN COLLEGE OF ENGINEERING, KALAVAKKAM
(An Autonomous Institution, Affiliated to Anna University, Chennai)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UCS1412 – DBMS Lab
Assignment – 5

Name: Jayannthan P T *Dept: CSE 'A'* *Roll No.: 205001049*

Title: Control Structures

Spool File Output:

```
SQL>
-- 1. Check whether the given combination of food and flavor is available. If any one or
both are not available, display the relevant message.
SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
  2      PRODUCTROWS PRODUCTS%ROWTYPE;
  3      COUNT_PRODUCTS NUMBER;
  4      SEARCHFOOD PRODUCTS.FOOD%TYPE;
  5      SEARCHFLAVOR PRODUCTS.FLAVOR%TYPE;
  6  BEGIN
  7      SEARCHFOOD:='&FOOD_TO_SEARCH';
  8      SEARCHFLAVOR:='&FLAVOR_TO_SEARCH';
  9      SELECT COUNT(*) INTO COUNT_PRODUCTS FROM PRODUCTS WHERE
LOWER(FOOD)=LOWER(SEARCHFOOD) AND LOWER(FLAVOR)=LOWER(SEARCHFLAVOR);
 10      IF COUNT_PRODUCTS!=0 THEN
 11          DBMS_OUTPUT.PUT_LINE('THE COMBINATION YOU ASKED FOR IS AVAILABLE');
 12          DBMS_OUTPUT.PUT_LINE('THERE IS '|| SEARCHFOOD ||' WITH ' || SEARCHFLAVOR ||'
FLAVOR');
 13          SELECT * INTO PRODUCTROWS FROM PRODUCTS WHERE LOWER(FOOD)=LOWER(SEARCHFOOD)
AND LOWER(FLAVOR)=LOWER(SEARCHFLAVOR);
 14          DBMS_OUTPUT.PUT_LINE(PRODUCTROWS.PID||'   '||PRODUCTROWS.FLAVOR||'
'||PRODUCTROWS.FOOD||'   '||PRODUCTROWS.PRICE);
 15      END IF;
 16      IF COUNT_PRODUCTS=0 THEN
 17          SELECT COUNT(*) INTO COUNT_PRODUCTS FROM PRODUCTS WHERE
LOWER(FOOD)=LOWER(SEARCHFOOD);
 18      IF COUNT_PRODUCTS!=0 THEN
 19          DBMS_OUTPUT.PUT_LINE('THE COMBINATION YOU ASKED FOR IS NOT AVAILABLE');
 20          DBMS_OUTPUT.PUT_LINE('THE FOOD YOU ASKED FOR IS AVAILABLE IN COMNIATION WITH
OTHER FLAVORS');
 21      END IF;
 22      END IF;
 23      IF COUNT_PRODUCTS=0 THEN
 24          SELECT COUNT(*) INTO COUNT_PRODUCTS FROM PRODUCTS WHERE
LOWER(FLAVOR)=LOWER(SEARCHFLAVOR);
 25      IF COUNT_PRODUCTS!=0 THEN
 26          DBMS_OUTPUT.PUT_LINE('THE COMBINATION YOU ASKED FOR IS NOT AVAILABLE');
```

```
 27          DBMS_OUTPUT.PUT_LINE('THE FLAVOR YOU ASKED FOR IS AVAILABLE IN COMNIATION
WITH OTHER FOODS');
 28      END IF;
 29      END IF;
 30      IF COUNT_PRODUCTS=0 THEN
 31          SELECT COUNT(*) INTO COUNT_PRODUCTS FROM PRODUCTS WHERE
LOWER(FOOD)<>LOWER(SEARCHFOOD) AND LOWER(FLAVOR)<>LOWER(SEARCHFLAVOR);
 32      IF COUNT_PRODUCTS!=0 THEN
 33          DBMS_OUTPUT.PUT_LINE('THE COMBINATION YOU ASKED FOR IS NOT AVAILABLE');
 34          DBMS_OUTPUT.PUT_LINE('BOTH FOOD AND FLAVOR IS NOT AVAILABLE');
 35      END IF;
 36      END IF;
 37  END;
 38  /
Enter value for food_to_search: cake
old   7:     SEARCHFOOD:='&FOOD_TO_SEARCH';
new   7:     SEARCHFOOD:='cake';
Enter value for flavor_to_search: lemon
old   8:     SEARCHFLAVOR:='&FLAVOR_TO_SEARCH';
new   8:     SEARCHFLAVOR:='lemon';
THE COMBINATION YOU ASKED FOR IS AVAILABLE
THERE IS cake WITH lemon FLAVOR
20-BC-L-10   Lemon   Cake   8.95

PL/SQL procedure successfully completed.

SQL> /
Enter value for food_to_search: cake
old   7:     SEARCHFOOD:='&FOOD_TO_SEARCH';
new   7:     SEARCHFOOD:='cake';
Enter value for flavor_to_search: coco
old   8:     SEARCHFLAVOR:='&FLAVOR_TO_SEARCH';
new   8:     SEARCHFLAVOR:='coco';
THE COMBINATION YOU ASKED FOR IS NOT AVAILABLE
THE FOOD YOU ASKED FOR IS AVAILABLE IN COMNIATION WITH OTHER FLAVORS

PL/SQL procedure successfully completed.

SQL> /
Enter value for food_to_search: cake
old   7:     SEARCHFOOD:='&FOOD_TO_SEARCH';
new   7:     SEARCHFOOD:='cake';
Enter value for flavor_to_search: old   8:     SEARCHFLAVOR:='&FLAVOR_TO_SEARCH';
new   8:

PL/SQL procedure successfully completed.


SQL> /
Enter value for food_to_search: coco
old   7:     SEARCHFOOD:='&FOOD_TO_SEARCH';
new   7:     SEARCHFOOD:='coco';
Enter value for flavor_to_search: lemon
old   8:     SEARCHFLAVOR:='&FLAVOR_TO_SEARCH';
```

```
new   8:     SEARCHFLAVOR:='lemon';
THE COMBINATION YOU ASKED FOR IS NOT AVAILABLE
THE FOOD YOU ASKED FOR IS AVAILABLE IN COMNIATION WITH OTHER FLAVORS
THE COMBINATION YOU ASKED FOR IS NOT AVAILABLE
THE FLAVOR YOU ASKED FOR IS AVAILABLE IN COMNIATION WITH OTHER FOODS

PL/SQL procedure successfully completed.


-- 2. On a given date, find the number of items sold (Use Implicit cursor).
SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
  2      COUNTITEMS NUMBER;
  3      GIVENDATE DATE;
  4  BEGIN
  5      GIVENDATE:=TO_DATE('&SEARCHDATE','DD-MM-YY');
  6      SELECT COUNT(ITEM) INTO COUNTITEMS FROM ITEM_LIST JOIN RECEIPTS USING(RNO) GROUP
BY RDATE HAVING RDATE=GIVENDATE;
  7      DBMS_OUTPUT.PUT_LINE('THE NUMBER OF ITEM SOLD ON '||GIVENDATE||' =
'||COUNTITEMS);
  8      EXCEPTION
  9      WHEN NO_DATA_FOUND THEN
 10          DBMS_OUTPUT.PUT_LINE('NO ITEMS SOLD ON '||GIVENDATE);
 11  END;
 12  /
Enter value for searchdate: 12-10-07
old   5:     GIVENDATE:=TO_DATE('&SEARCHDATE','DD-MM-YY');
new   5:     GIVENDATE:=TO_DATE('12-10-07','DD-MM-YY');
THE NUMBER OF ITEM SOLD ON 12-OCT-07 = 37

PL/SQL procedure successfully completed.

SQL> /
Enter value for searchdate: 12-12-07
old   5:     GIVENDATE:=TO_DATE('&SEARCHDATE','DD-MM-YY');
new   5:     GIVENDATE:=TO_DATE('12-12-07','DD-MM-YY');
NO ITEMS SOLD ON 12-DEC-07

PL/SQL procedure successfully completed.


-- 3. An user desired to buy the product with the specific price. Ask the user for a
price,
-- find the food item(s) that is equal or closest to the desired price. Print the product
-- number, food type, flavor and price. Also print the number of items that is equal or
-- closest to the desired price.
SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
  2      SPECIFICPRICE PRODUCTS.PRICE%TYPE;
  3      CURSOR SPECIFIC_PRODUCTS IS SELECT * FROM PRODUCTS WHERE PRICE BETWEEN
FLOOR(SPECIFICPRICE-0.5) AND CEIL(SPECIFICPRICE+0.5);
  4      PRODUCTROWS SPECIFIC_PRODUCTS%ROWTYPE;
  5  BEGIN
  6      SPECIFICPRICE:='&PRICE_TO_SEARCH';
```

```
  7      OPEN SPECIFIC_PRODUCTS;
  8   DBMS_OUTPUT.PUT_LINE('PRODUCT LIST:');
  9   LOOP
 10    FETCH SPECIFIC_PRODUCTS INTO PRODUCTROWS;
 11    IF SPECIFIC_PRODUCTS%NOTFOUND THEN
 12     EXIT;
 13    ELSE
 14     DBMS_OUTPUT.PUT_LINE(PRODUCTROWS.PID||'  '||PRODUCTROWS.FOOD||'
'||PRODUCTROWS.FLAVOR||'  '||PRODUCTROWS.PRICE);
 15    END IF;
 16   END LOOP;
 17   DBMS_OUTPUT.PUT_LINE(SPECIFIC_PRODUCTS%ROWCOUNT||' PRODUCTS FOUND');
 18   EXCEPTION
 19      WHEN NO_DATA_FOUND THEN
 20    DBMS_OUTPUT.PUT_LINE('NO PRODUCTS FOUND');
 21      CLOSE SPECIFIC_PRODUCTS;
 22  END;
 23  /
Enter value for price_to_search: 2
old   6:     SPECIFICPRICE:='&PRICE_TO_SEARCH';
new   6:     SPECIFICPRICE:='2';
PRODUCT LIST:
70-GA  Cookie  Ganache  1.15
70-GON  Cookie  Gongolais  1.15
70-R  Cookie  Raspberry  1.09
70-M-CH-DZ  Meringue  Chocolate  1.25
70-M-VA-SM-DZ  Meringue  Vanilla  1.15
70-MAR  Cookie  Marzipan  1.25
70-TU  Cookie  Tuile  1.25
50-ALM  Croissant  Almond  1.45
50-APP  Croissant  Apple  1.45
50-APR  Croissant  Apricot  1.45
50-CHS  Croissant  Cheese  1.75
50-CH  Croissant  Chocolate  1.75
51-APR  Danish  Apricot  1.15
51-APP  Danish  Apple  1.15
51-ATW  Twist  Almond  1.15
51-BC  Bear Claw  Almond  1.95
51-BLU  Danish  Blueberry  1.15
17 PRODUCTS FOUND

PL/SQL procedure successfully completed.


-- 4. Display the customer name along with the details of item and its quantity ordered
for
-- the given order number. Also calculate the total quantity ordered as shown below:
SQL>
SQL> DECLARE
  2      ORDER_NUMBER ITEM_LIST.RNO%TYPE;
  3      CFNAME CUSTOMERS.FNAME%TYPE;
  4      CLNAME CUSTOMERS.LNAME%TYPE;
  5      TOTAL_ITEM NUMBER:=0;
```

```
  6      CURSOR DETAILS IS SELECT FOOD,FLAVOR,COUNT(*) AS QUANTITY FROM ITEM_LIST,PRODUCTS
WHERE RNO=ORDER_NUMBER AND PID=ITEM GROUP BY FOOD,FLAVOR;
  7      DETAILROW DETAILS%ROWTYPE;
  8  BEGIN
  9      ORDER_NUMBER:='&ORDER_NUMBER_TO_SEARCH';
 10      SELECT FNAME,LNAME INTO CFNAME,CLNAME FROM CUSTOMERS INNER JOIN RECEIPTS
USING(CID) WHERE RNO=ORDER_NUMBER;
 11      DBMS_OUTPUT.PUT_LINE('CUSTOMER NAME: '||CFNAME||' '||CLNAME);
 12      OPEN DETAILS;
 13      DBMS_OUTPUT.PUT_LINE('FOOD            FLAVOR             QUANTITY');
 14      LOOP
 15          FETCH DETAILS INTO DETAILROW;
 16          IF DETAILS%NOTFOUND THEN
 17              EXIT;
 18          ELSE
 19     DBMS_OUTPUT.PUT_LINE(DETAILROW.FOOD||'             '||DETAILROW.FLAVOR||'
'||DETAILROW.QUANTITY);
 20              TOTAL_ITEM:=TOTAL_ITEM+DETAILROW.QUANTITY;
 21          END IF;
 22      END LOOP;
 23      DBMS_OUTPUT.PUT_LINE('TOTAL QUANTITY = '||TOTAL_ITEM);
 24      CLOSE DETAILS;
 25   EXCEPTION
 26      WHEN NO_DATA_FOUND THEN
 27          DBMS_OUTPUT.PUT_LINE('INVALID ORDER NUMBER!');
 28  END;
 29  /
Enter value for order_number_to_search: 51991
old   9:     ORDER_NUMBER:='&ORDER_NUMBER_TO_SEARCH';
new   9:     ORDER_NUMBER:='51991';
CUSTOMER NAME: SOPKO RAYFORD
FOOD            FLAVOR             QUANTITY
Pie             Apple             1
Cake             Truffle             1
Tart             Apple             1
Tart             Chocolate             1
TOTAL QUANTITY = 4

PL/SQL procedure successfully completed.

SQL> spool off;


SQL> /
Enter value for order_number_to_search: 1234
old   9:     ORDER_NUMBER:='&ORDER_NUMBER_TO_SEARCH';
new   9:     ORDER_NUMBER:='1234';
INVALID ORDER NUMBER!

PL/SQL procedure successfully completed.

SQL> spool off;
```

	SSN COLLEGE OF ENGINEERING	***Faculty:***
	Department of Computer Science & Engineering	***Dr. P.Mirunalini, Asso. Prof.*** ***N. Sujaudeen, Asst. Prof***
	CS8481 – DBMS Lab ***Assignment – 6***	***Assigned: 28-Apr-22*** ***Due: 1 Lab Hour***
Title: PL/SQL – Stored Procedures & Functions		

Consider the following relations for the Bakery database:

CUSTOMERS (*cid*, fname, lname)

PRODUCTS (*pid*, flavor, food, price)

RECEIPTS (*rno*, rdate, cid)

ITEM_LIST (*rno, ordinal*, item)

Note:
a. Use implicit/explicit cursor wherever required.
b. Use IN, OUT, INOUT as parameter type wherever needed.

Write a PL/SQL stored procedure for the following:

1. For the given receipt number, calculate the Discount as follows:

   For total amount > $10 and total amount < $25: Discount=5%

   For total amount > $25 and total amount < $50: Discount=10%

   For total amount > $50: Discount=20%

   Calculate the amount (after the discount) and update the same in Receipts table.

   Print the receipt as shown below:

```
**********************************************************
Receipt Number:13355              Customer Name: TOUSSAND
SHARRON Receipt Date :19-Oct-2007
**********************************************************
Sno       Flavor      Food          Price
1.        Opera       Cake          15.95
2.        Lemon       Cookie         0.79
3.        Napoleon    Cake          13.49
----------------------------------------------------------
                      Total =     $ 30.23
----------------------------------------------------------
Total Amount          :$ 30.23
Discount(10%)         :$    3.02
----------------------------------------------------------
Amount to be paid     :$ 27.21
**********************************************************
Great Offers! Discount up to 25% on DIWALI Festival Day...
**********************************************************
```

2. Ask the user for the *budget* and his/her preferred *food type*. You recommend the best item(s) within the planned budget for the given food type. The best item is determined by the maximum ordered product among many customers for the given food type.

   Print the recommended product that suits your budget as below:

```
**********************************************************
*** Budget: $10                          Food type: Meringue
**********************************************************
*** Item ID      Flavor      Food        Price
70-M-CH-DZ      Chocolate   Meringue    1.25
70-M-VA-SM-DZ Vanilla       Meringue    1.15
----------------------------------------------------------
70-M-CH-DZ with Chocolate flavor is the best item in Meringuetype! You are
entitled to purchase 8 Meringue chocolates for the given budget !!!
**********************************************************
```

3. Take a receipt number and item as arguments, and insert this information into the Item list. However, if there is already a receipt with that receipt number, then keep adding 1 to the maximum ordinal number. Else before inserting into the Item list with ordinal as 1, ask the user to give the customer name who placed the order and insert this information into the Receipts.
4. Write a stored function to display the customer name who ordered maximum for the given food and flavor.
5. Implement Question (2) using stored function to return the amount to be paid and update the same, for the given receipt number.

<u>What you have to submit:</u>

1. Schema Diagram with constraints
2. Demo script file

SSN COLLEGE OF ENGINEERING, KALAVAKKAM
(An Autonomous Institution, Affiliated to Anna University, Chennai)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UCS1412 – DBMS Lab
Assignment – 6

Name: Jayannthan P T *Dept: CSE 'A'* *Roll No.: 205001049*

Title: Stored Procedures and Functions

Spool File Output:

```
SQL>
-- Write a PL/SQL stored procedure for the following:

-- 1. For the given receipt number, calculate the Discount as follows:
-- For total amount > $10 and total amount < $25: Discount=5%
-- For total amount > $25 and total amount < $50: Discount=10%
-- For total amount > $50: Discount=20%
-- Calculate the amount (after the discount) and update the same in Receipts table.

SQL> CREATE OR REPLACE PROCEDURE PRINTRECEIPT
  2  (ORDER_NUMBER IN ITEM_LIST.RNO%TYPE)
  3  IS
  4      CFNAME CUSTOMERS.FNAME%TYPE;
  5      CLNAME CUSTOMERS.LNAME%TYPE;
  6   RECDATE RECEIPTS.RDATE%TYPE;
  7      -- TOTAL_ITEM NUMBER:=0;
  8      TOTAL_PRICE NUMBER:=0;
  9      DISCOUNT NUMBER:=0;
 10      CURSOR DETAILS IS SELECT FOOD,FLAVOR,PRICE,COUNT(*) AS QUANTITY FROM
ITEM_LIST,PRODUCTS WHERE RNO=ORDER_NUMBER AND PID=ITEM GROUP BY FOOD,FLAVOR,PRICE;
 11      DETAILROW DETAILS%ROWTYPE;
 12  BEGIN
 13      SELECT FNAME,LNAME,RDATE INTO CFNAME,CLNAME,RECDATE FROM CUSTOMERS JOIN RECEIPTS
USING(CID) WHERE RNO=ORDER_NUMBER;
 14      DBMS_OUTPUT.PUT_LINE('RECEIPT NUMBER: '||ORDER_NUMBER);
 15      DBMS_OUTPUT.PUT_LINE('CUSTOMER NAME: '||CFNAME||' '||CLNAME);
 16      DBMS_OUTPUT.PUT_LINE('RECEIPT DATE: '||RECDATE);
 17      OPEN DETAILS;
 18      DBMS_OUTPUT.PUT_LINE('FOOD            FLAVOR            QUANTITY');
 19      LOOP
 20          FETCH DETAILS INTO DETAILROW;
 21          IF DETAILS%NOTFOUND THEN
 22              EXIT;
 23          ELSE
 24     DBMS_OUTPUT.PUT_LINE(DETAILROW.FOOD||'            '||DETAILROW.FLAVOR||'
    '||DETAILROW.QUANTITY);
 25              -- TOTAL_ITEM:=TOTAL_ITEM+DETAILROW.QUANTITY;
```

```
 26              TOTAL_PRICE:=TOTAL_PRICE+DETAILROW.QUANTITY*DETAILROW.PRICE;
 27          END IF;
 28      END LOOP;
 29      DBMS_OUTPUT.PUT_LINE('TOTAL AMOUNT :$ '||TOTAL_PRICE);
 30   IF TOTAL_PRICE<=10 THEN
 31       DBMS_OUTPUT.PUT_LINE('NO DISCOUNT');
 32   END IF;
 33   IF TOTAL_PRICE>10 AND TOTAL_PRICE<25 THEN
 34    DISCOUNT:=0.05*TOTAL_PRICE;
 35       DBMS_OUTPUT.PUT_LINE('DISCOUNT (5%) :$ '||DISCOUNT);
 36   END IF;
 37   IF TOTAL_PRICE<50 THEN
 38    DISCOUNT:=0.1*TOTAL_PRICE;
 39    DBMS_OUTPUT.PUT_LINE('DISCOUNT (10%) :$ '||DISCOUNT);
 40   END IF;
 41   IF TOTAL_PRICE>=50 THEN
 42    DISCOUNT:=0.2*TOTAL_PRICE;
 43    DBMS_OUTPUT.PUT_LINE('DISCOUNT (20%) :$ '||DISCOUNT);
 44   END IF;
 45      CLOSE DETAILS;
 46   TOTAL_PRICE:=TOTAL_PRICE-DISCOUNT;
 47   DBMS_OUTPUT.PUT_LINE('AMOUNT TO BE PAID $ '||TOTAL_PRICE);
 48   EXCEPTION
 49      WHEN NO_DATA_FOUND THEN
 50          DBMS_OUTPUT.PUT_LINE('INVALID ORDER NUMBER!');
 51  END;
 52  /

Procedure created.

SQL>
SQL> CALL PRINTRECEIPT(13355);
RECEIPT NUMBER: 13355
CUSTOMER NAME: TOUSSAND SHARRON
RECEIPT DATE: 19-OCT-07
FOOD            FLAVOR             QUANTITY
Cake             Napoleon             1
Cookie             Lemon              1
Cake             Opera              1
TOTAL AMOUNT :$ 30.23
DISCOUNT (10%) :$ 3.023
AMOUNT TO BE PAID $ 27.207

Call completed.

-- 2. Ask the user for the budget and his/her preferred food type. You recommend the best
-- item(s) within the planned budget for the given food type. The best item is
-- determined by the maximum ordered product among many customers for the given
-- food type.
SQL>
SQL> CREATE OR REPLACE PROCEDURE BUDGETFOOD
  2  (BUDGET IN NUMBER,FOODTYPE IN PRODUCTS.FOOD%TYPE)
  3  IS
```

```
  4      CURSOR ITEMLIST_CURSOR IS SELECT PID,FOOD,FLAVOR,PRICE FROM PRODUCTS WHERE
FOOD=FOODTYPE AND PRICE<BUDGET;
  5      PRODUCTROW ITEMLIST_CURSOR%ROWTYPE;
  6      CNT NUMBER:=0;
  7      FLAG NUMBER:=0;
  8      PIID ITEM_LIST.ITEM%TYPE;
  9      FLAVOR PRODUCTS.FLAVOR%TYPE;
 10      productprice PRODUCTS.PRICE%TYPE;
 11      ITEMSINBAG NUMBER;
 12
 13  BEGIN
 14      DBMS_OUTPUT.PUT_LINE('BUDGET:'||BUDGET||'    '||'FOOD:'||FOODTYPE);
 15      DBMS_OUTPUT.PUT_LINE('');
 16      DBMS_OUTPUT.PUT_LINE('ITEM ID    FOOD    FLAVOR    PRICE');
 17      OPEN ITEMLIST_CURSOR;
 18      LOOP
 19          FETCH ITEMLIST_CURSOR INTO PRODUCTROW;
 20          IF ITEMLIST_CURSOR%NOTFOUND THEN
 21              EXIT;
 22          END IF;
 23          SELECT COUNT(*) INTO FLAG FROM ITEM_LIST WHERE ITEM=PRODUCTROW.PID;
 24          IF FLAG>CNT THEN
 25              CNT:=FLAG;
 26              PIID:=PRODUCTROW.PID;
 27
 28          END IF;
 29          DBMS_OUTPUT.PUT_LINE(PRODUCTROW.PID||'    '||PRODUCTROW.FOOD||'    '||PRODUCT
ROW.FLAVOR||'    '||PRODUCTROW.PRICE);
 30      END LOOP;
 31      CLOSE ITEMLIST_CURSOR;
 32      SELECT FLAVOR INTO FLAVOR FROM PRODUCTS WHERE PID=PIID;
 33      SELECT PRICE INTO productprice FROM PRODUCTS WHERE PID=PIID;
 34      DBMS_OUTPUT.PUT_LINE('');
 35      DBMS_OUTPUT.PUT_LINE('THE ITEM WITH ID '||PIID||' AND '||FLAVOR||' IS THE BEST
CHOICE FOR YOU');
 36      ITEMSINBAG:=BUDGET/productprice;
 37      DBMS_OUTPUT.PUT_LINE('YOU CAN BUY A MAXIMUM OF '||ITEMSINBAG||' ITEM OF THIS
PRODUCT');
 38      DBMS_OUTPUT.PUT_LINE('');
 39
 40  END;
 41  /

Procedure created.

SQL>
SQL> call BUDGETFOOD(10,'Meringue');
BUDGET:10    FOOD:Meringue
ITEM ID    FOOD    FLAVOR    PRICE
70-M-CH-DZ    Meringue    Chocolate    1.25
70-M-VA-SM-DZ    Meringue    Vanilla    1.15
THE ITEM WITH ID 70-M-CH-DZ AND Chocolate IS THE BEST CHOICE FOR YOU
YOU CAN BUY A MAXIMUM OF 8 ITEM OF THIS PRODUCT
```

```
Call completed.


-- 3. Take a receipt number and item as arguments, and insert this information into the
-- Item list. However, if there is already a receipt with that receipt number, then keep
-- adding 1 to the maximum ordinal number. Else before inserting into the Item list
-- with ordinal as 1, ask the user to give the customer name who placed the order and
-- insert this information into the Receipts.
SQL>
SQL>
SQL> --PROCEDURE TO INSERT INTO ITEM_LIST
SQL> CREATE OR REPLACE PROCEDURE INSERT_INTO_ITEM_LIST
  2  (RECEIPTNO_TO_INS IN RECEIPTS.RNO%TYPE, ORDINAL_TO_INS IN ITEM_LIST.ORDINAL%TYPE,
PRODID_TO_INS IN PRODUCTS.PID%TYPE) IS
  3
  4  BEGIN
  5   INSERT INTO ITEM_LIST VALUES(RECEIPTNO_TO_INS,ORDINAL_TO_INS,PRODID_TO_INS);
  6  END;
  7  /

Procedure created.

SQL>
SQL> --PROCEDURE TO UPDATE INTO ITEM_LIST
SQL> CREATE OR REPLACE PROCEDURE UPDATE_INTO_ITEM_LIST
  2  (RECEIPTNO_TO_INS IN ITEM_LIST.RNO%TYPE, PRODID_TO_INS IN ITEM_LIST.ITEM%TYPE) IS
  3
  4  BEGIN
  5   UPDATE ITEM_LIST
  6      SET ORDINAL=ORDINAL+1
  7    WHERE RNO=RECEIPTNO_TO_INS;
  8  END;
  9  /

Procedure created.

SQL>
SQL> --PROCEDURE TO INSERT INTO RECEIPTS
SQL> CREATE OR REPLACE PROCEDURE INSERT_INTO_RECEIPT
  2  (RECEIPTNO_TO_INS IN RECEIPTS.RNO%TYPE, DATE_TO_INS IN
RECEIPTS.RDATE%TYPE,CUSTID_TO_INS IN RECEIPTS.CID%TYPE) IS
  3  BEGIN
  4   INSERT INTO RECEIPTS VALUES(RECEIPTNO_TO_INS,DATE_TO_INS,CUSTID_TO_INS);
  5  END;
  6  /

Procedure created.

SQL>
SQL> --PROCEDURE TO FIND CUSTOMER WITH GIVEN NAME AND RETURN CUSTOMER ID
SQL> CREATE OR REPLACE PROCEDURE SEARCH_CUSTOMER
  2  (CUST_FNAME IN CUSTOMERS.FNAME%TYPE, CUST_LNAME IN CUSTOMERS.LNAME%TYPE,CUSTID_SAVE
OUT CUSTOMERS.CID%TYPE) IS
  3  BEGIN
```

```
  4   BEGIN
  5    SELECT CID INTO CUSTID_SAVE FROM CUSTOMERS
  6    WHERE FNAME=CUST_FNAME AND LNAME=CUST_LNAME;
  7   EXCEPTION
  8    WHEN NO_DATA_FOUND THEN
  9    DBMS_OUTPUT.PUT_LINE('CUSTOMER ID NOT IN CUSTOMER TABLE');
 10    CUSTID_SAVE:=0;
 11   END;
 12  END;
 13  /

Procedure created.

SQL>  select * from item_list where rno=11548;

       RNO    ORDINAL ITEM
---------- ---------- ---------------
     11548          2 45-CO
     11548          3 90-APIE-10
SQL>
SQL> DECLARE
  2   CUSTFNAME CUSTOMERS.FNAME%TYPE;
  3   CUSTLNAME CUSTOMERS.LNAME%TYPE;
  4   CUSTID CUSTOMERS.CID%TYPE;
  5   RECEIPTNO_TO_INS RECEIPTS.RNO%TYPE;
  6   ORDINAL_TO_INS ITEM_LIST.ORDINAL%TYPE;
  7   PRODID_TO_INS PRODUCTS.PID%TYPE;
  8   DATE_TO_INS RECEIPTS.RDATE%TYPE;
  9   ITEM_ROW ITEM_LIST%ROWTYPE;
 10   CURSOR ITEMLIST_CURSOR IS
 11   SELECT * FROM ITEM_LIST WHERE RNO=RECEIPTNO_TO_INS AND ITEM=PRODID_TO_INS;
 12   MAXORDI ITEM_LIST.ORDINAL%TYPE;
 13
 14  BEGIN
 15
 16   RECEIPTNO_TO_INS:='&RECEIPT_NO';
 17   PRODID_TO_INS:='&PRODUCT_ID';
 18   OPEN ITEMLIST_CURSOR;
 19   FETCH ITEMLIST_CURSOR INTO item_row;
 20   IF ITEMLIST_CURSOR%ROWCOUNT>0 THEN
 21    BEGIN
 22    ORDINAL_TO_INS:=item_row.ORDINAL+1;
 23    UPDATE_INTO_ITEM_LIST(RECEIPTNO_TO_INS,PRODID_TO_INS);
 24    RETURN;
 25    END;
 26   ELSE
 27    BEGIN
 28    DBMS_OUTPUT.PUT_LINE('RECEIPT NOT FOUND');
 29    DBMS_OUTPUT.PUT_LINE('CREATING NEW RECEIPT');
 30    CUSTFNAME:='&FIRST_NAME';
 31    CUSTLNAME:='&LAST_NAME';
 32    DATE_TO_INS:='&DATE';
 33    SEARCH_CUSTOMER(CUSTFNAME,CUSTLNAME,CUSTID);
 34    INSERT_INTO_RECEIPT(RECEIPTNO_TO_INS,DATE_TO_INS,CUSTID);
```

```
 35    ORDINAL_TO_INS:=1;
 36    INSERT_INTO_ITEM_LIST(RECEIPTNO_TO_INS,ORDINAL_TO_INS,PRODID_TO_INS);
 37    RETURN;
 38    END;
 39   END IF;
 40  END;
 41  /
Enter value for receipt_no: 11548
old  16:  RECEIPTNO_TO_INS:='&RECEIPT_NO';
new  16:  RECEIPTNO_TO_INS:='11548';
Enter value for product_id: 45-CO
old  17:  PRODID_TO_INS:='&PRODUCT_ID';
new  17:  PRODID_TO_INS:='45-CO';
Enter value for first_name: nil
old  30:   CUSTFNAME:='&FIRST_NAME';
new  30:   CUSTFNAME:='nil';
Enter value for last_name: nil
old  31:   CUSTLNAME:='&LAST_NAME';
new  31:   CUSTLNAME:='nil';
Enter value for date: nil
old  32:   DATE_TO_INS:='&DATE';
new  32:   DATE_TO_INS:='nil';

PL/SQL procedure successfully completed.

SQL>  select * from item_list where rno=11548;

       RNO    ORDINAL ITEM
---------- ---------- ---------------
     11548          3 45-CO
     11548          4 90-APIE-10

SQL> rollback;

Rollback complete.

-- 4. Write a stored function to display the customer name who ordered maximum for the
-- given food and flavor.
SQL> CREATE OR REPLACE FUNCTION MOST_CUST
  2  (GIVENFOOD PRODUCTS.FOOD%TYPE, GIVENFLAVOR PRODUCTS.FLAVOR%TYPE)
  3  RETURN CUSTOMER.CNAME%TYPE
  4   AS
  5
  6      CUST_NAME CUSTOMER.CNAME%TYPE;
  7
  8  BEGIN
  9      SELECT FNAME ||' '|| LNAME
 10      INTO CUST_NAME
 11      FROM CUSTOMERS
 12      WHERE CID = (
 13          SELECT CID FROM (
 14           SELECT A.CID, COUNT(*)
 15           FROM CUSTOMERS A, PRODUCTS B, ITEM_LIST C, RECEIPTS D
```

```
 16           WHERE A.CID = D.CID AND C.RNO = D.RNO AND C.ITEM = B.PID AND LOWER(B.FOOD) =
LOWER(GIVENFOOD) AND LOWER(FLAVOR) = LOWER(GIVENFLAVOR)
 17           GROUP BY A.CID ORDER BY 2 DESC)
 18          WHERE ROWNUM = 1);
 19
 20  RETURN CUST_NAME;
 21  END;
 22  /

Function created.

SQL>
SQL> ---CALLING FUNCTION FROM ANONYMOUS BLOCK
SQL>
SQL> DECLARE
  2
  3   CUSTNAME varchar(50);
  4   prodid_to_ins PRODUCTS.PID%TYPE;
  5   GIVENFOOD PRODUCTS.FOOD%TYPE;
  6   GIVENFLAVOR PRODUCTS.FLAVOR%TYPE;
  7
  8  BEGIN
  9
 10   GIVENFOOD:='&FOOD';
 11   GIVENFLAVOR:='&FLAVOR';
 12   CUSTNAME:=MOST_CUST(GIVENFOOD,GIVENFLAVOR);
 13   DBMS_OUTPUT.PUT_LINE(CUSTNAME||' BOUGHT MAXIMUM OF '||UPPER(GIVENFOOD)||' IN
'||UPPER(GIVENFLAVOR)||' FLAVOR');
 14  END;
 15  /
Enter value for food: cake
old  10:  GIVENFOOD:='&FOOD';
new  10:  GIVENFOOD:='cake';
Enter value for flavor: lemon
old  11:  GIVENFLAVOR:='&FLAVOR';
new  11:  GIVENFLAVOR:='lemon';
ESPOSITA TRAVIS BOUGHT MAXIMUM OF CAKE IN LEMON FLAVOR

PL/SQL procedure successfully completed.

-- 5. Implement Question (1) using stored function to return the amount to be paid and
-- update the same, for the given receipt number.
SQL>
SQL> ---CREATING FUNCTION
SQL> CREATE OR REPLACE FUNCTION CALC_DISCOUNT_FUNC
  2  (PRODUCT_PRICE IN PRODUCTS.PRICE%TYPE,DISC_PER OUT PRODUCTS.PRICE%TYPE,DISCOUNT_PRICE
OUT PRODUCTS.PRICE%TYPE)
  3  RETURN PRODUCTS.PRICE%TYPE IS
  4
  5  FINAL_PRICE PRODUCTS.PRICE%TYPE;
  6
  7  BEGIN
  8   IF PRODUCT_PRICE>10 AND PRODUCT_PRICE<25 THEN
  9    DISCOUNT_PRICE:=0.05*PRODUCT_PRICE;
```

```
 10    DISC_PER:=5;
 11   END IF;
 12   IF PRODUCT_PRICE>25 AND PRODUCT_PRICE<50 THEN
 13    DISCOUNT_PRICE:=0.1*PRODUCT_PRICE;
 14    DISC_PER:=10;
 15   END IF;
 16   IF PRODUCT_PRICE>50 THEN
 17    DISCOUNT_PRICE:=.2*PRODUCT_PRICE;
 18    DISC_PER:=20;
 19   END IF;
 20   FINAL_PRICE:=PRODUCT_PRICE-DISCOUNT_PRICE;
 21   RETURN FINAL_PRICE;
 22  END;
 23  /

Function created.

SQL>
SQL> DECLARE
  2      TOTAL PRODUCTS.PRICE%TYPE;
  3      PERCENT INT;
  4      DISC PRODUCTS.PRICE%TYPE;
  5      DOP DATE;
  6      SNO INT := 0;
  7      CUST_FNAME CUSTOMERS.FNAME%TYPE;
  8      CUST_LNAME CUSTOMERS.LNAME%TYPE;
  9      RECEIPT_NO_SEARCH RECEIPTS.RNO%TYPE;
 10      CURSOR ITEM_CURSOR IS
 11      SELECT FLAVOR, FOOD, PRICE FROM PRODUCTS, ITEM_LIST
 12      WHERE PRODUCTS.PID = ITEM_LIST.ITEM AND ITEM_LIST.RNO = RECEIPT_NO_SEARCH;
 13      TEMP_ITEM ITEM_CURSOR%ROWTYPE;
 14      FINAL_PRICE PRODUCTS.PRICE%TYPE;
 15
 16
 17  BEGIN
 18      RECEIPT_NO_SEARCH:='&RECEIPT_NUM';
 19      SELECT SUM(PRICE) INTO TOTAL FROM ITEM_LIST, PRODUCTS
 20   WHERE ITEM_LIST.ITEM = PRODUCTS.PID and ITEM_LIST.RNO = RECEIPT_NO_SEARCH;
 21   DBMS_OUTPUT.PUT_LINE('RECEIPT NUMBER : ' || RECEIPT_NO_SEARCH);
 22
 23   SELECT RDATE INTO DOP FROM RECEIPTS
 24   WHERE RNO = RECEIPT_NO_SEARCH;
 25
 26   DBMS_OUTPUT.PUT_LINE('RECEIPT DATE : ' || DOP);
 27
 28   SELECT FNAME, LNAME INTO CUST_FNAME, CUST_LNAME FROM CUSTOMERS, RECEIPTS
 29   WHERE CUSTOMERS.CID = RECEIPTS.CID AND RECEIPTS.RNO = RECEIPT_NO_SEARCH;
 30
 31   DBMS_OUTPUT.PUT_LINE('CUSTOMER NAME : ' || CUST_FNAME || ' ' || CUST_LNAME);
 32   DBMS_OUTPUT.PUT_LINE('FLAVOR     FOOD     PRICE');
 33      OPEN ITEM_CURSOR;
 34   LOOP
 35    FETCH ITEM_CURSOR INTO TEMP_ITEM;
 36    IF ITEM_CURSOR%NOTFOUND THEN
```

```
 37     EXIT;
 38    END IF;
 39    DBMS_OUTPUT.PUT_LINE(TEMP_ITEM.FLAVOR || '     ' || TEMP_ITEM.FOOD || '     ' ||
TEMP_ITEM.PRICE);
 40   END LOOP;
 41      CLOSE ITEM_CURSOR;
 42
 43      FINAL_PRICE:=CALC_DISCOUNT_FUNC(TOTAL,PERCENT,DISC);
 44   DBMS_OUTPUT.PUT_LINE('TOTAL AMOUNT : $' || TOTAL);
 45   DBMS_OUTPUT.PUT_LINE('DISCOUNT ' || PERCENT ||'% : $' || DISC);
 46   DBMS_OUTPUT.PUT_LINE('AMOUNT TO BE PAID : $' ||FINAL_PRICE );
 47  END;
 48  /
Enter value for receipt_num: 13355
old  18:     RECEIPT_NO_SEARCH:='&RECEIPT_NUM';
new  18:     RECEIPT_NO_SEARCH:='13355';
RECEIPT NUMBER : 13355
RECEIPT DATE : 19-OCT-07
CUSTOMER NAME : TOUSSAND SHARRON
FLAVOR     FOOD     PRICE
Opera     Cake     15.95
Lemon     Cookie     .79
Napoleon     Cake     13.49
TOTAL AMOUNT : $30.23
DISCOUNT 10% : $3.023
AMOUNT TO BE PAID : $27.207

PL/SQL procedure successfully completed.

SQL> spool off;
```

	SSN COLLEGE OF ENGINEERING	***Faculty:***
	Department of Computer Science & Engineering	***Dr. P.Mirunalini, Asso. Prof.*** ***Dr. N. Sujaudeen, Asst. Prof***
	CS8481 – DBMS Lab	***Assigned: 12-May-22***
	Assignment – 7	***Due: 1 Lab Hour***
Title: PL/SQL – Triggers		

Consider the following relations for the Bakery database:

CUSTOMERS (*cid*, fname, lname)

PRODUCTS (*pid*, flavor, food, price)

RECEIPTS (*rno*, rdate, cid, amt)

ITEM_LIST (*rno, ordinal*, item)

Note:

a. Use implicit/explicit cursor wherever required.

b. Handle the error and display appropriate message if required.

c. Add a column **amt** to Receipts relation.

Write a PL/SQL Trigger for the following:

1. The combination of Flavor and Food determines the product id. Hence, while inserting a new instance into the *Products* relation, ensure that the same combination of Flavor and Food is not already available.
2. While entering an item into the *item_list* relation, update the amount in *Receipts* with the total amount for that receipt number.
3. Implement the following constraints for *Item_list* relation:
   a. A receipt can contain a maximum of five items only.
   b. A receipt should not allow an item to be purchased more than thrice.

What you have to submit:

1. Schema Diagram with constraints
2. Demo script file

SSN COLLEGE OF ENGINEERING, KALAVAKKAM
(An Autonomous Institution, Affiliated to Anna University, Chennai)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UCS1412 – DBMS Lab
Assignment – 7

Name: Jayannthan P T *Dept: CSE 'A'* *Roll No.: 205001049*

Title: Triggers

Spool File:

```
-- Write a PL/SQL Trigger for the following:
-- 1. The combination of Flavor and Food determines the product id. Hence, while
-- inserting a new instance into the Products relation, ensure that the same combination
-- of Flavor and Food is not already available.

SQL>
SQL> SET SERVEROUTPUT ON;
SQL>
SQL> CREATE OR REPLACE TRIGGER BEFORE_PRODUCT_INSERT
  2  BEFORE INSERT ON PRODUCTS FOR EACH ROW
  3  DECLARE
  4      CT NUMBER:=0;
  5      CURSOR PROCURSOR IS SELECT * FROM PRODUCTS
  6      WHERE FOOD=:NEW.FOOD AND FLAVOR=:NEW.FLAVOR;
  7      RECORD PROCURSOR%ROWTYPE;
  8  BEGIN
  9      OPEN PROCURSOR;
 10      FETCH PROCURSOR INTO RECORD;
 11      IF PROCURSOR%NOTFOUND THEN
 12          CT:=1;
 13      END IF;
 14      CLOSE PROCURSOR;
 15      IF CT=0 THEN
 16          RAISE_APPLICATION_ERROR(-20000,'COMBINATION ALREADY AVAILABLE');
 17      END IF;
 18  END;
 19  /

Trigger created.

SQL>
SQL> INSERT INTO PRODUCTS
  2  VALUES ('20-BC','CHOCOLATE','CAKE',8.95);

1 row created.
```

```
SQL>
SQL> INSERT INTO PRODUCTS
  2  VALUES ('20-BC','CHOCOLATE','CAKE',8.95);
INSERT INTO PRODUCTS
            *
ERROR at line 1:
ORA-20000: COMBINATION ALREADY AVAILABLE
ORA-06512: at "1049.BEFORE_PRODUCT_INSERT", line 14
ORA-04088: error during execution of trigger '1049.BEFORE_PRODUCT_INSERT'


-- 2. While entering an item into the item_list relation, update the amount in Receipts
with
-- the total amount for that receipt number.


SQL>
SQL> SAVEPOINT SAVE1;

Savepoint created.

SQL> ALTER TABLE RECEIPTS
  2  ADD AMT NUMBER;

Table altered.

SQL>
SQL> SELECT * FROM RECEIPTS
  2  WHERE RNO=51991;

       RNO RDATE            CID        AMT
---------- --------- ---------- ----------
     51991 17-OCT-07         14

SQL>
SQL> CREATE OR REPLACE TRIGGER ITEM_LIST_ITEMS_AMOUNT
  2  BEFORE INSERT ON ITEM_LIST FOR EACH ROW
  3  BEGIN
  4      UPDATE RECEIPTS
  5      SET AMT=(SELECT SUM(PRICE) FROM ITEM_LIST,PRODUCTS
  6      WHERE RNO=:NEW.RNO AND PID=ITEM) WHERE RNO=:NEW.RNO;
  7      UPDATE RECEIPTS
  8      SET AMT=AMT+(SELECT PRICE FROM PRODUCTS
  9      WHERE PID=:NEW.ITEM) WHERE RNO=:NEW.RNO;
 10  END;
 11  /

Trigger created.

SQL>
SQL> INSERT INTO ITEM_LIST VALUES(51991,5,'45-VA');

1 row created.
```

```
SQL> SELECT * FROM RECEIPTS
  2  WHERE RNO=51991;

       RNO RDATE            CID        AMT
---------- --------- ---------- ----------
     51991 17-OCT-07         14      31.45



-- 3. Implement the following constraints for Item_list relation:
-- a. A receipt can contain a maximum of five items only.
-- b. A receipt should not allow an item to be purchased more than thrice.

SQL>
SQL> SAVEPOINT SAVE1;

Savepoint created.

SQL>
SQL> CREATE OR REPLACE TRIGGER MAX_CONST_ITEMS
  2  BEFORE INSERT ON ITEM_LIST FOR EACH ROW
  3  DECLARE
  4      CURSOR RNOCOUNT IS SELECT COUNT(*) AS COUNT1 FROM ITEM_LIST
  5      WHERE RNO=:NEW.RNO;
  6      CURSOR RNO_ITEM_COUNT IS SELECT COUNT(*) AS COUNT2 FROM ITEM_LIST
  7      WHERE RNO=:NEW.RNO AND ITEM=:NEW.ITEM;
  8      RNO_ROW RNOCOUNT%ROWTYPE;
  9      RNO_ITEM_ROW RNO_ITEM_COUNT%ROWTYPE;
 10  BEGIN
 11      OPEN RNOCOUNT;
 12      OPEN RNO_ITEM_COUNT;
 13      FETCH RNOCOUNT INTO RNO_ROW;
 14      FETCH RNO_ITEM_COUNT INTO RNO_ITEM_ROW;
 15      IF RNO_ROW.COUNT1>=5 AND RNO_ITEM_ROW.COUNT2>=3 THEN
 16          RAISE_APPLICATION_ERROR(-20001,'THE RECEIPT ALREADY HAS 5 ITEMS AND THE ITEM
IS PURCHASED THRICE');
 17      ELSIF RNO_ROW.COUNT1>=5 THEN
 18          RAISE_APPLICATION_ERROR(-20002,'THE RECEIPT ALREADY HAS 5 ITEMS');
 19      ELSIF RNO_ITEM_ROW.COUNT2>=3 THEN
 20          RAISE_APPLICATION_ERROR(-20003,'THE ITEM IS PURCHASED THRICE');
 21      END IF;
 22      CLOSE RNO_ITEM_COUNT;
 23      CLOSE RNOCOUNT;
 24  END;
 25  /

Trigger created.

SQL>
SQL> INSERT INTO ITEM_LIST VALUES(52761,6,'90-ALM-I');
INSERT INTO ITEM_LIST VALUES(52761,6,'90-ALM-I')
            *
ERROR at line 1:
ORA-20002: THE RECEIPT ALREADY HAS 5 ITEMS
```

```
ORA-06512: at "1049.MAX_CONST_ITEMS", line 16
ORA-04088: error during execution of trigger '1049.MAX_CONST_ITEMS'


SQL> INSERT INTO ITEM_LIST VALUES(83085,6,'45-CH');
INSERT INTO ITEM_LIST VALUES(83085,6,'45-CH')
            *
ERROR at line 1:
ORA-20002: THE RECEIPT ALREADY HAS 5 ITEMS
ORA-06512: at "1049.MAX_CONST_ITEMS", line 16
ORA-04088: error during execution of trigger '1049.MAX_CONST_ITEMS'


SQL> INSERT INTO ITEM_LIST VALUES(32565,4,'50-APP');

1 row created.

SQL> INSERT INTO ITEM_LIST VALUES(32565,4,'50-APP');
INSERT INTO ITEM_LIST VALUES(32565,4,'50-APP')
               *
ERROR at line 1:
ORA-20003: THE ITEM IS PURCHASED THRICE
ORA-06512: at "1049.MAX_CONST_ITEMS", line 18
ORA-04088: error during execution of trigger '1049.MAX_CONST_ITEMS'


SQL> rollback;

Rollback complete.
```

	SSN COLLEGE OF ENGINEERING ***Department of Computer Science & Engineering***	***Faculty:*** ***Dr. P.Mirunalini, Asso. Prof.*** ***N. Sujaudeen, Asst. Prof***
	CS8481 – DBMS Lab ***Assignment – 8***	***Assigned: 21-05-22*** ***Due: 1 Lab Hours***
Title: PL/SQL – Exception Handling		

Consider the following relations for the Bakery database:

CUSTOMERS (*cid*, fname, lname)

PRODUCTS (*pid*, flavor, food, price)

RECEIPTS (*rno*, rdate, cid, amt)

ITEM_LIST (*rno, ordinal*, item)

Note:

a. Use predefined/user-defined exception wherever required.

b. Handle the error(s) and display appropriate messages.

Write a PL/SQL block to handle the following exceptions:

1. For the given receipt number, if there are no rows then display as "No order with the given receipt <number>". If the receipt contains more than one item, display as "The given receipt <number> contains more than one item". If the receipt contains single item, display as "The given receipt <number> contains exactly one item". Use predefined exception handling.
2. While inserting the receipt details, raise an exception when the receipt date is greater than the current date.

What you have to submit:

1. Schema Diagram with constraints
2. Demo script file

SSN COLLEGE OF ENGINEERING, KALAVAKKAM
(An Autonomous Institution, Affiliated to Anna University, Chennai)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UCS1412 – DBMS Lab
Assignment – 8

Name: Jayannthan P T *Dept: CSE 'A'* *Roll No.: 205001049*

Title: Exceptions

Spool File:

```
SET SERVEROUTPUT ON;


-- Write a PL/SQL block to handle the following exceptions:
-- 1. For the given receipt number, if there are no rows then display as “No order with
the
-- given receipt <number>”. If the receipt contains more than one item, display as
-- “The given receipt <number> contains more than one item”. If the receipt contains
-- single item, display as “The given receipt <number> contains exactly one item”. Use
-- predefined exception handling.

SELECT * FROM ITEM_LIST WHERE RNO=18129;


       RNO    ORDINAL ITEM
---------- ---------- ---------------
     18129          1 70-TU


DECLARE
    RNO_INPUT ITEM_LIST.RNO%TYPE;
    CURSOR ITEM_LIST_CURSOR IS SELECT * FROM ITEM_LIST WHERE RNO=RNO_INPUT;
    ITEM_LIST_ROW ITEM_LIST_CURSOR%ROWTYPE;

BEGIN
    RNO_INPUT:='&RECEIPT_NO_INPUT';
    OPEN ITEM_LIST_CURSOR;
    FETCH ITEM_LIST_CURSOR INTO ITEM_LIST_ROW;
    CLOSE ITEM_LIST_CURSOR;
    SELECT * INTO ITEM_LIST_ROW FROM ITEM_LIST WHERE RNO=RNO_INPUT;
    DBMS_OUTPUT.PUT_LINE('THE GIVEN RECEIPT '||RNO_INPUT||' CONTAINS EXACTLY ONE ITEM');
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('NO ORDER WITH THE GIVEN RECEIPT '||RNO_INPUT);
    WHEN TOO_MANY_ROWS THEN
        DBMS_OUTPUT.PUT_LINE('THE GIVEN RECEIPT '||RNO_INPUT||' CONTAINS MORE THAN ONE
ITEM');
END;
/
```

```
old:DECLARE
    RNO_INPUT ITEM_LIST.RNO%TYPE;
    CURSOR ITEM_LIST_CURSOR IS SELECT * FROM ITEM_LIST WHERE RNO=RNO_INPUT;
    ITEM_LIST_ROW ITEM_LIST_CURSOR%ROWTYPE;

BEGIN
    RNO_INPUT:='&RECEIPT_NO_INPUT';
    OPEN ITEM_LIST_CURSOR;
    FETCH ITEM_LIST_CURSOR INTO ITEM_LIST_ROW;
    CLOSE ITEM_LIST_CURSOR;
    SELECT * INTO ITEM_LIST_ROW FROM ITEM_LIST WHERE RNO=RNO_INPUT;
    DBMS_OUTPUT.PUT_LINE('THE GIVEN RECEIPT '||RNO_INPUT||' CONTAINS EXACTLY ONE ITEM');
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('NO ORDER WITH THE GIVEN RECEIPT '||RNO_INPUT);
    WHEN TOO_MANY_ROWS THEN
        DBMS_OUTPUT.PUT_LINE('THE GIVEN RECEIPT '||RNO_INPUT||' CONTAINS MORE THAN ONE
ITEM');
END;

new:DECLARE
    RNO_INPUT ITEM_LIST.RNO%TYPE;
    CURSOR ITEM_LIST_CURSOR IS SELECT * FROM ITEM_LIST WHERE RNO=RNO_INPUT;
    ITEM_LIST_ROW ITEM_LIST_CURSOR%ROWTYPE;

BEGIN
    RNO_INPUT:='18129';
    OPEN ITEM_LIST_CURSOR;
    FETCH ITEM_LIST_CURSOR INTO ITEM_LIST_ROW;
    CLOSE ITEM_LIST_CURSOR;
    SELECT * INTO ITEM_LIST_ROW FROM ITEM_LIST WHERE RNO=RNO_INPUT;
    DBMS_OUTPUT.PUT_LINE('THE GIVEN RECEIPT '||RNO_INPUT||' CONTAINS EXACTLY ONE ITEM');
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('NO ORDER WITH THE GIVEN RECEIPT '||RNO_INPUT);
    WHEN TOO_MANY_ROWS THEN
        DBMS_OUTPUT.PUT_LINE('THE GIVEN RECEIPT '||RNO_INPUT||' CONTAINS MORE THAN ONE
ITEM');
END;
THE GIVEN RECEIPT 18129 CONTAINS EXACTLY ONE ITEM


PL/SQL procedure successfully completed.


SELECT * FROM ITEM_LIST WHERE RNO=85858;

       RNO    ORDINAL ITEM
---------- ---------- ---------------
     85858          1 20-CA-7.5
     85858          2 70-M-VA-SM-DZ
     85858          3 51-BLU
```

```
DECLARE
    RNO_INPUT ITEM_LIST.RNO%TYPE;
    CURSOR ITEM_LIST_CURSOR IS SELECT * FROM ITEM_LIST WHERE RNO=RNO_INPUT;
    ITEM_LIST_ROW ITEM_LIST_CURSOR%ROWTYPE;

BEGIN
    RNO_INPUT:='&RECEIPT_NO_INPUT';
    OPEN ITEM_LIST_CURSOR;
    FETCH ITEM_LIST_CURSOR INTO ITEM_LIST_ROW;
    CLOSE ITEM_LIST_CURSOR;
    SELECT * INTO ITEM_LIST_ROW FROM ITEM_LIST WHERE RNO=RNO_INPUT;
    DBMS_OUTPUT.PUT_LINE('THE GIVEN RECEIPT '||RNO_INPUT||' CONTAINS EXACTLY ONE ITEM');
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('NO ORDER WITH THE GIVEN RECEIPT '||RNO_INPUT);
    WHEN TOO_MANY_ROWS THEN
        DBMS_OUTPUT.PUT_LINE('THE GIVEN RECEIPT '||RNO_INPUT||' CONTAINS MORE THAN ONE
ITEM');
END;
/


old:DECLARE
    RNO_INPUT ITEM_LIST.RNO%TYPE;
    CURSOR ITEM_LIST_CURSOR IS SELECT * FROM ITEM_LIST WHERE RNO=RNO_INPUT;
    ITEM_LIST_ROW ITEM_LIST_CURSOR%ROWTYPE;

BEGIN
    RNO_INPUT:='&RECEIPT_NO_INPUT';
    OPEN ITEM_LIST_CURSOR;
    FETCH ITEM_LIST_CURSOR INTO ITEM_LIST_ROW;
    CLOSE ITEM_LIST_CURSOR;
    SELECT * INTO ITEM_LIST_ROW FROM ITEM_LIST WHERE RNO=RNO_INPUT;
    DBMS_OUTPUT.PUT_LINE('THE GIVEN RECEIPT '||RNO_INPUT||' CONTAINS EXACTLY ONE ITEM');
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('NO ORDER WITH THE GIVEN RECEIPT '||RNO_INPUT);
    WHEN TOO_MANY_ROWS THEN
        DBMS_OUTPUT.PUT_LINE('THE GIVEN RECEIPT '||RNO_INPUT||' CONTAINS MORE THAN ONE
ITEM');
END;

new:DECLARE
    RNO_INPUT ITEM_LIST.RNO%TYPE;
    CURSOR ITEM_LIST_CURSOR IS SELECT * FROM ITEM_LIST WHERE RNO=RNO_INPUT;
    ITEM_LIST_ROW ITEM_LIST_CURSOR%ROWTYPE;

BEGIN
    RNO_INPUT:='85858';
    OPEN ITEM_LIST_CURSOR;
    FETCH ITEM_LIST_CURSOR INTO ITEM_LIST_ROW;
    CLOSE ITEM_LIST_CURSOR;
    SELECT * INTO ITEM_LIST_ROW FROM ITEM_LIST WHERE RNO=RNO_INPUT;
    DBMS_OUTPUT.PUT_LINE('THE GIVEN RECEIPT '||RNO_INPUT||' CONTAINS EXACTLY ONE ITEM');
```

```
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('NO ORDER WITH THE GIVEN RECEIPT '||RNO_INPUT);
    WHEN TOO_MANY_ROWS THEN
        DBMS_OUTPUT.PUT_LINE('THE GIVEN RECEIPT '||RNO_INPUT||' CONTAINS MORE THAN ONE
ITEM');
END;
THE GIVEN RECEIPT 85858 CONTAINS MORE THAN ONE ITEM


PL/SQL procedure successfully completed.

SELECT * FROM ITEM_LIST WHERE RNO=11111;

no rows selected

DECLARE
    RNO_INPUT ITEM_LIST.RNO%TYPE;
    CURSOR ITEM_LIST_CURSOR IS SELECT * FROM ITEM_LIST WHERE RNO=RNO_INPUT;
    ITEM_LIST_ROW ITEM_LIST_CURSOR%ROWTYPE;

BEGIN
    RNO_INPUT:='&RECEIPT_NO_INPUT';
    OPEN ITEM_LIST_CURSOR;
    FETCH ITEM_LIST_CURSOR INTO ITEM_LIST_ROW;
    CLOSE ITEM_LIST_CURSOR;
    SELECT * INTO ITEM_LIST_ROW FROM ITEM_LIST WHERE RNO=RNO_INPUT;
    DBMS_OUTPUT.PUT_LINE('THE GIVEN RECEIPT '||RNO_INPUT||' CONTAINS EXACTLY ONE ITEM');
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('NO ORDER WITH THE GIVEN RECEIPT '||RNO_INPUT);
    WHEN TOO_MANY_ROWS THEN
        DBMS_OUTPUT.PUT_LINE('THE GIVEN RECEIPT '||RNO_INPUT||' CONTAINS MORE THAN ONE
ITEM');
END;
/

old:DECLARE
    RNO_INPUT ITEM_LIST.RNO%TYPE;
    CURSOR ITEM_LIST_CURSOR IS SELECT * FROM ITEM_LIST WHERE RNO=RNO_INPUT;
    ITEM_LIST_ROW ITEM_LIST_CURSOR%ROWTYPE;

BEGIN
    RNO_INPUT:='&RECEIPT_NO_INPUT';
    OPEN ITEM_LIST_CURSOR;
    FETCH ITEM_LIST_CURSOR INTO ITEM_LIST_ROW;
    CLOSE ITEM_LIST_CURSOR;
    SELECT * INTO ITEM_LIST_ROW FROM ITEM_LIST WHERE RNO=RNO_INPUT;
    DBMS_OUTPUT.PUT_LINE('THE GIVEN RECEIPT '||RNO_INPUT||' CONTAINS EXACTLY ONE ITEM');
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('NO ORDER WITH THE GIVEN RECEIPT '||RNO_INPUT);
    WHEN TOO_MANY_ROWS THEN
```

```
        DBMS_OUTPUT.PUT_LINE('THE GIVEN RECEIPT '||RNO_INPUT||' CONTAINS MORE THAN ONE
ITEM');
END;

new:DECLARE
    RNO_INPUT ITEM_LIST.RNO%TYPE;
    CURSOR ITEM_LIST_CURSOR IS SELECT * FROM ITEM_LIST WHERE RNO=RNO_INPUT;
    ITEM_LIST_ROW ITEM_LIST_CURSOR%ROWTYPE;

BEGIN
    RNO_INPUT:='11111.';
    OPEN ITEM_LIST_CURSOR;
    FETCH ITEM_LIST_CURSOR INTO ITEM_LIST_ROW;
    CLOSE ITEM_LIST_CURSOR;
    SELECT * INTO ITEM_LIST_ROW FROM ITEM_LIST WHERE RNO=RNO_INPUT;
    DBMS_OUTPUT.PUT_LINE('THE GIVEN RECEIPT '||RNO_INPUT||' CONTAINS EXACTLY ONE ITEM');
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('NO ORDER WITH THE GIVEN RECEIPT '||RNO_INPUT);
    WHEN TOO_MANY_ROWS THEN
        DBMS_OUTPUT.PUT_LINE('THE GIVEN RECEIPT '||RNO_INPUT||' CONTAINS MORE THAN ONE
ITEM');
END;
NO ORDER WITH THE GIVEN RECEIPT 11111


PL/SQL procedure successfully completed.




-- 2. While inserting the receipt details, raise an exception when the receipt date is
greater
-- than the current date.

SET SERVEROUTPUT ON;
desc receipts;

Name  Null?    Type
----- -------- ------
RNO   NOT NULL NUMBER
RDATE          DATE
CID            NUMBER


CREATE OR REPLACE TRIGGER CHECKINSDATE
BEFORE INSERT ON RECEIPTS FOR EACH ROW

DECLARE
    THROWEXCEPTION EXCEPTION;
    TODAYDATE DATE;
BEGIN
```

```
    SELECT SYSDATE INTO TODAYDATE FROM DUAL;
    IF(:NEW.RDATE>TODAYDATE) THEN
        RAISE THROWEXCEPTION;
    END IF;

EXCEPTION
    WHEN THROWEXCEPTION THEN
        RAISE_APPLICATION_ERROR(-20009,'RECEIPT DATE GREATER THAN TODAYDATE');
END;
/

Trigger CHECKINSDATE compiled

INSERT INTO RECEIPTS VALUES(11111,'22-May-2023',10);


Error starting at line : 21 in command -
INSERT INTO RECEIPTS VALUES(11111,'22-May-2023',10)
Error report -
ORA-20009: RECEIPT DATE GREATER THAN TODAYDATE
ORA-06512: at "SYSTEM.CHECKINSDATE", line 12
ORA-04088: error during execution of trigger 'SYSTEM.CHECKINSDATE'

INSERT INTO RECEIPTS VALUES(11111,'25-May-2022',10);


Error starting at line : 22 in command -
INSERT INTO RECEIPTS VALUES(11111,'25-May-2022',10)
Error report -
ORA-00001: unique constraint (SYSTEM.RNO_PKEY) violated
```

	SSN COLLEGE OF ENGINEERING	***Faculty:***
	Department of Computer Science & Engineering	***Dr. P.Mirunalini, Asso. Prof.*** ***N. Sujaudeen, Asst. Prof***
	UCS1412 – DBMS Lab ***Assignment – 9***	***Assigned: 30-May-22*** ***Due: 2 Lab Hours***
Title: Database Programming using JDBC/ODBC		

Application Development to Database

The JDBC API is the industry standard for database-independent connectivity between the Java programming language and a wide range of databases. The JDBC API provides a call-level API for SQL-based database access. JDBC technology allows you to use the Java programming language to exploit "Write Once, Run Anywhere" capabilities for applications that require access to enterprise data.

The JDBC API makes it possible to do three things:

- Establish a connection with a database or access any tabular data source
- Send SQL statements
- Process the results

JDBC Drive Types:

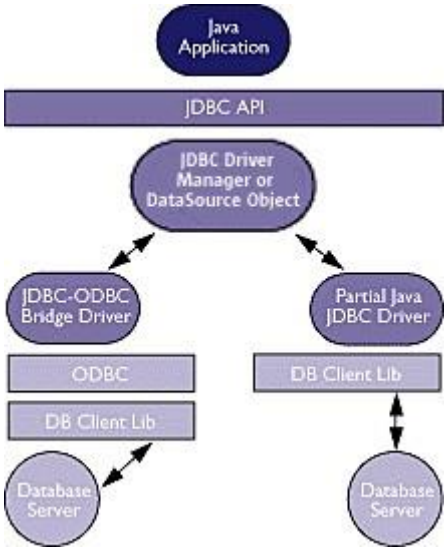
JDBC technology drivers fit into one of four categories:

Left side, Type 1: JDBC-ODBC Bridge plus ODBC Driver

This combination provides JDBC access via ODBC drivers. ODBC binary code -- and in many cases, database client code -- must be loaded on each client machine that uses a JDBC-ODBC Bridge. Sun provides a JDBC-ODBC Bridge driver, which is appropriate for experimental use and for situations in which no other driver is available.

Right side, Type 2: A native API partly Java technology-enabled driver

This type of driver converts JDBC calls into calls on the client API for Oracle, Sybase, Informix, DB2, or other DBMS. Note that, like the bridge driver, this style of driver requires that some binary code be loaded on each client machine.

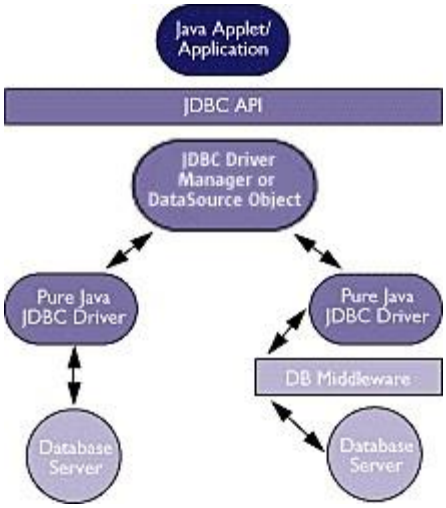


Right side, Type 3: Pure Java Driver for Database Middleware

This style of driver translates JDBC calls into the middleware vendor's protocol, which is then translated to a DBMS protocol by a middleware server. The middleware provides connectivity to many different databases.

Left side, Type 4: Direct-to-Database Pure Java Driver

This style of driver converts JDBC calls into the network protocol used directly by DBMSs, allowing a direct call from the client machine to the DBMS server and providing a practical solution for intranet access.



Requirements

- Software: The Java 2 Platform (either the Java 2 SDK, Standard Edition, or the Java 2 SDK, Enterprise Edition), an SQL database, and a JDBC technology-based driver for that database.
- Hardware: Same as for the Java 2 Platform.

Reference:

http://www.oracle.com/technetwork/java/javase/jdbc/index.html

Problem Specification:

Front-end: NetBeans IDE 6.x (Java)

Back-end: Oracle10g

Schema:

Emp_Payroll (*eid, ename, dob, sex, designation, basic, da, hra, pf, mc, gross, tot_deduc, net_pay*)

Design an interface for the above schema and perform the following operations through the application:

1. Insert (*eid, ename, dob, sex, designation, basic)*

Calculate da, hra, pf ,mc, gross, total deductions and net pay as described below and update the same in the database for the inserted employee.

2. Update
3. Delete
4. Search the record
5. Exit

To calculate the net pay of an employee, develop a PL/SQL procedure/function that accepts only the eid and basic and calculates as per the following:

Dearness Allowance [DA] = 60%

House Rent Allowance [HRA]=11%

Provident Fund [PF] = 4%

Mediclaim [MC] = 3%

Gross = Basic + DA + HRA

Total Deduction = PF + MC

Net Pay = Gross – Total Deduction

Call the procedure/function from the application by passing appropriate parameter(s) and update the corresponding record.

What you have to submit:

1. Schema Diagram with constraints
2. Interface Design, DB Connectivity and Database Updates

SSN COLLEGE OF ENGINEERING, KALAVAKKAM
(An Autonomous Institution, Affiliated to Anna University, Chennai)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UCS1412 – DBMS Lab
Assignment – 9

Name: Jayannthan P T *Dept: CSE 'A'* *Roll No.: 205001049*

Title: Database Connectivity

Spool File for Table Creation and PL/SQL Block:

```
SQL> CREATE TABLE EMP_PAYROLL(
  2       EID VARCHAR(5) PRIMARY KEY,
  3   ENAME VARCHAR(10),
  4   DOB DATE,
  5   SEX VARCHAR(1),
  6   DESIGNATION VARCHAR(10),
  7   BASIC NUMBER,
  8   DA NUMBER,
  9   HRA NUMBER,
 10   PF NUMBER,
 11   MC NUMBER,
 12   GROSS NUMBER,
 13   TOT_DEDUC NUMBER,
 14   NET_PAY NUMBER
 15   );

Table created.


SQL> CREATE OR REPLACE PROCEDURE CALC
  2   (ID IN EMP_PAYROLL.EID%TYPE, BASIC IN EMP_PAYROLL.BASIC%TYPE)
  3   IS
  4   DA_TEMP NUMBER;
  5   HRA_TEMP NUMBER;
  6   PF_TEMP NUMBER;
  7   MC_TEMP NUMBER;
  8   GROSS_TEMP NUMBER;
  9   TOT_DEDUC_TEMP NUMBER;
 10   NET_PAY_TEMP NUMBER;
 11   BEGIN
 12   DA_TEMP:=0.6*BASIC;
 13   HRA_TEMP:=0.11*BASIC;
 14   PF_TEMP:=0.04*BASIC;
 15   MC_TEMP:=0.03*BASIC;
 16   GROSS_TEMP:=BASIC+DA_TEMP+HRA_TEMP;
 17   TOT_DEDUC_TEMP:=PF_TEMP+MC_TEMP;
 18   NET_PAY_TEMP:=GROSS_TEMP-TOT_DEDUC_TEMP;
```

```
 19  UPDATE EMP_PAYROLL
 20  SET DA=DA_TEMP, HRA=HRA_TEMP, PF=PF_TEMP, MC=MC_TEMP, GROSS=GROSS_TEMP,
 21  TOT_DEDUC=TOT_DEDUC_TEMP, NET_PAY=NET_PAY_TEMP
 22  WHERE EID=ID;
 23  END;
 24  /

Procedure created.
```

Java Code of JDBC Block:

```
]
import javax.swing.*;
import java.sql.*;
import java.sql.Statement;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author 1049
 */
public class demo1 extends javax.swing.JFrame {

    /**
     * Creates new form demo1
     */
    Connection con;
    Statement st;
    PreparedStatement ps;
    ResultSet rs;

    public demo1() {
        initComponents();
        try{
            Class.forName("oracle.jdbc.OracleDriver");
            JOptionPane.showMessageDialog(this,"Driver Loaded!");
        try {
            con =
DriverManager.getConnection("jdbc:oracle:thin:@10.6.4.33:1521:orcl","1049","1049");
            JOptionPane.showMessageDialog(this,"Connected to Oracle database!");
        }
        catch (SQLException ex) {
            Logger.getLogger(demo1.class.getName()).log(Level.SEVERE, null, ex);
            JOptionPane.showMessageDialog(this,ex.getMessage());
        }
        }
        catch(ClassNotFoundException ex){
        Logger.getLogger(demo1.class.getName()).log(Level.SEVERE, null, ex);
        JOptionPane.showMessageDialog(this,ex.getMessage());
        }

    }
```

```
    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated
Code">
    private void initComponents() {

        SEX1 = new javax.swing.JLabel();
        EID = new javax.swing.JLabel();
        NAME = new javax.swing.JLabel();
        DOB = new javax.swing.JLabel();
        SEX = new javax.swing.JLabel();
        DESIGNATION = new javax.swing.JLabel();
        BASICPAY = new javax.swing.JLabel();
        name = new javax.swing.JTextField();
        eid = new javax.swing.JTextField();
        sex = new javax.swing.JTextField();
        dob = new javax.swing.JTextField();
        basic = new javax.swing.JTextField();
        designation = new javax.swing.JTextField();
        insert = new javax.swing.JButton();
        update = new javax.swing.JButton();
        delete = new javax.swing.JButton();
        refresh = new javax.swing.JButton();
        search = new javax.swing.JButton();
        jLabel1 = new javax.swing.JLabel();
        exit = new javax.swing.JButton();
        DA = new javax.swing.JLabel();
        HRA = new javax.swing.JLabel();
        PF = new javax.swing.JLabel();
        GROSS = new javax.swing.JLabel();
        TOTAL_DEDUCTION = new javax.swing.JLabel();
        NETPAY = new javax.swing.JLabel();
        da = new javax.swing.JTextField();
        hra = new javax.swing.JTextField();
        pf = new javax.swing.JTextField();
        gross = new javax.swing.JTextField();
        tot_deduc = new javax.swing.JTextField();
        net_pay = new javax.swing.JTextField();

        SEX1.setText("SEX");

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        EID.setText("EID");

        NAME.setText("NAME");

        DOB.setText("DATE OF BIRTH");

        SEX.setText("SEX");
```

```
        DESIGNATION.setText("DESIGNATION");

        BASICPAY.setText("BASIC PAY");

        eid.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                eidActionPerformed(evt);
            }
        });

        sex.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                sexActionPerformed(evt);
            }
        });

        insert.setText("INSERT");
        insert.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                insertActionPerformed(evt);
            }
        });

        update.setText("UPDATE");
        update.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                updateActionPerformed(evt);
            }
        });

        delete.setText("DELETE");
        delete.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                deleteActionPerformed(evt);
            }
        });

        refresh.setText("REFRESH");
        refresh.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                refreshActionPerformed(evt);
            }
        });

        search.setText("SEARCH");
        search.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                searchActionPerformed(evt);
            }
        });

        jLabel1.setText("EMPLOYEE  DATABASE");
```

```
        exit.setText("EXIT");
        exit.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                exitActionPerformed(evt);
            }
        });

        DA.setText("DA");

        HRA.setText("HRA");

        PF.setText("PF");

        GROSS.setText("GROSS");

        TOTAL_DEDUCTION.setText("TOTAL DEDUCTION");

        NETPAY.setText("NET PAY");

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(61, 61, 61)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEA
DING)
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Align
ment.LEADING)
                            .addComponent(search, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                            .addComponent(insert, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                        .addGap(30, 30, 30)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Align
ment.LEADING, false)
                            .addComponent(refresh, javax.swing.GroupLayout.DEFAULT_SIZE,
152, Short.MAX_VALUE)
                            .addComponent(update, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                        .addGap(27, 27, 27)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Align
ment.LEADING, false)
                            .addComponent(delete, javax.swing.GroupLayout.DEFAULT_SIZE,
139, Short.MAX_VALUE)
                            .addComponent(exit, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)))
                    .addGroup(layout.createSequentialGroup()
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Align
ment.LEADING)
                            .addGroup(layout.createSequentialGroup()
```

```
                                .addGroup(layout.createParallelGroup(javax.swing.GroupLayo
ut.Alignment.TRAILING)
                                    .addComponent(BASICPAY,
javax.swing.GroupLayout.PREFERRED_SIZE, 93, javax.swing.GroupLayout.PREFERRED_SIZE)
                                    .addComponent(DESIGNATION,
javax.swing.GroupLayout.PREFERRED_SIZE, 93, javax.swing.GroupLayout.PREFERRED_SIZE)
                                    .addComponent(SEX,
javax.swing.GroupLayout.PREFERRED_SIZE, 93, javax.swing.GroupLayout.PREFERRED_SIZE)
                                    .addComponent(DOB,
javax.swing.GroupLayout.PREFERRED_SIZE, 93, javax.swing.GroupLayout.PREFERRED_SIZE)
                                    .addComponent(NAME,
javax.swing.GroupLayout.PREFERRED_SIZE, 93, javax.swing.GroupLayout.PREFERRED_SIZE)
                                    .addComponent(EID,
javax.swing.GroupLayout.PREFERRED_SIZE, 93, javax.swing.GroupLayout.PREFERRED_SIZE))
                                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacemen
t.RELATED)
                                .addGroup(layout.createParallelGroup(javax.swing.GroupLayo
ut.Alignment.LEADING)
                                    .addGroup(layout.createSequentialGroup()
                                        .addGroup(layout.createParallelGroup(javax.swing.G
roupLayout.Alignment.LEADING)
                                            .addComponent(dob,
javax.swing.GroupLayout.PREFERRED_SIZE, 111, javax.swing.GroupLayout.PREFERRED_SIZE)
                                            .addComponent(name,
javax.swing.GroupLayout.PREFERRED_SIZE, 111, javax.swing.GroupLayout.PREFERRED_SIZE)
                                            .addComponent(eid,
javax.swing.GroupLayout.PREFERRED_SIZE, 111, javax.swing.GroupLayout.PREFERRED_SIZE))
                                        .addGap(39, 39, 39)
                                        .addGroup(layout.createParallelGroup(javax.swing.G
roupLayout.Alignment.LEADING)
                                            .addComponent(PF,
javax.swing.GroupLayout.PREFERRED_SIZE, 84, javax.swing.GroupLayout.PREFERRED_SIZE)
                                            .addGroup(layout.createParallelGroup(javax.swi
ng.GroupLayout.Alignment.LEADING, false)
                                                .addComponent(HRA,
javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT_SIZE, 84,
Short.MAX_VALUE)
                                                .addComponent(DA,
javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))))
                                    .addGroup(layout.createSequentialGroup()
                                        .addGroup(layout.createParallelGroup(javax.swing.G
roupLayout.Alignment.LEADING)
                                            .addComponent(sex,
javax.swing.GroupLayout.PREFERRED_SIZE, 111, javax.swing.GroupLayout.PREFERRED_SIZE)
                                            .addComponent(designation,
javax.swing.GroupLayout.PREFERRED_SIZE, 111, javax.swing.GroupLayout.PREFERRED_SIZE)
                                            .addComponent(basic,
javax.swing.GroupLayout.PREFERRED_SIZE, 111, javax.swing.GroupLayout.PREFERRED_SIZE))
                                        .addGap(39, 39, 39)
                                        .addGroup(layout.createParallelGroup(javax.swing.G
roupLayout.Alignment.LEADING)
                                            .addComponent(NETPAY,
javax.swing.GroupLayout.PREFERRED_SIZE, 84, javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
                                            .addGroup(layout.createParallelGroup(javax.swi
ng.GroupLayout.Alignment.LEADING, false)
                                                .addComponent(TOTAL_DEDUCTION,
javax.swing.GroupLayout.PREFERRED_SIZE, 1, Short.MAX_VALUE)
                                                .addComponent(GROSS,
javax.swing.GroupLayout.DEFAULT_SIZE, 84, Short.MAX_VALUE))))))
                            .addGroup(layout.createSequentialGroup()
                                .addGap(176, 176, 176)
                                .addComponent(jLabel1,
javax.swing.GroupLayout.PREFERRED_SIZE, 124, javax.swing.GroupLayout.PREFERRED_SIZE)))
                        .addGap(31, 31, 31)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Align
ment.TRAILING)
                            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.A
lignment.LEADING, false)
                                .addComponent(hra)
                                .addComponent(pf)
                                .addComponent(gross)
                                .addComponent(tot_deduc)
                                .addComponent(net_pay,
javax.swing.GroupLayout.DEFAULT_SIZE, 121, Short.MAX_VALUE))
                            .addComponent(da, javax.swing.GroupLayout.PREFERRED_SIZE, 121,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGap(0, 0, Short.MAX_VALUE)))
                .addGap(57, 57, 57))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 24,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(18, 18, 18)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BAS
ELINE)
                    .addComponent(EID)
                    .addComponent(eid, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(DA)
                    .addComponent(da, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(12, 12, 12)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BAS
ELINE)
                    .addComponent(NAME)
                    .addComponent(name, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(HRA)
                    .addComponent(hra, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BAS
ELINE)
```

```
                    .addComponent(DOB, javax.swing.GroupLayout.PREFERRED_SIZE, 20,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(dob)
                    .addComponent(PF)
                    .addComponent(pf, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BAS
ELINE)
                    .addComponent(SEX)
                    .addComponent(sex, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(GROSS)
                    .addComponent(gross, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(11, 11, 11)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BAS
ELINE)
                    .addComponent(DESIGNATION)
                    .addComponent(designation)
                    .addComponent(TOTAL_DEDUCTION)
                    .addComponent(tot_deduc, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BAS
ELINE)
                    .addComponent(BASICPAY)
                    .addComponent(basic, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(NETPAY)
                    .addComponent(net_pay, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(54, 54, 54)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BAS
ELINE)
                    .addComponent(insert)
                    .addComponent(update)
                    .addComponent(delete))
                .addGap(18, 18, 18)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BAS
ELINE)
                    .addComponent(refresh)
                    .addComponent(search)
                    .addComponent(exit))
                .addGap(19, 19, 19))
        );

        pack();
    }// </editor-fold>

    private void eidActionPerformed(java.awt.event.ActionEvent evt)
{
        // TODO add your handling code here:
```

```
    }

    private void refresh(){
        eid.setText("");
        name.setText("");
        designation.setText("");
        basic.setText("");
        dob.setText("");
        sex.setText("");
    }
    private void calcf(String eid, String basic){
        try{
            String sql="call calc(?,?)";
            ps=con.prepareStatement(sql);
            ps.setString(1,eid);
            ps.setString(2, basic);
            ps.execute();
        }
        catch(SQLException ex){
            Logger.getLogger(demo1.class.getName()).log(Level.SEVERE, null, ex);
            JOptionPane.showMessageDialog(this, ex.getMessage());
        }
    }

    private void refreshActionPerformed(java.awt.event.ActionEvent evt)
{
        // TODO add your handling code here:

            refresh();

    }

    private void sexActionPerformed(java.awt.event.ActionEvent evt)
{
        // TODO add your handling code here:
    }

    private void insertActionPerformed(java.awt.event.ActionEvent evt)
{
        // TODO add your handling code here:
        try {
            String sql = "insert into emp_payroll
values(?,?,?,?,?,?,NULL,NULL,NULL,NULL,NULL,NULL,NULL)";
            ps = con.prepareStatement(sql);
            ps.setString(1, eid.getText());
            ps.setString(2, name.getText());
            ps.setString(3, dob.getText());
            ps.setString(4, sex.getText());
            ps.setString(5, designation.getText());
            ps.setString(6, basic.getText());
            ps.execute();
            calcf(eid.getText(),basic.getText());
            JOptionPane.showMessageDialog(this,"Inserted!");
            refresh();
```

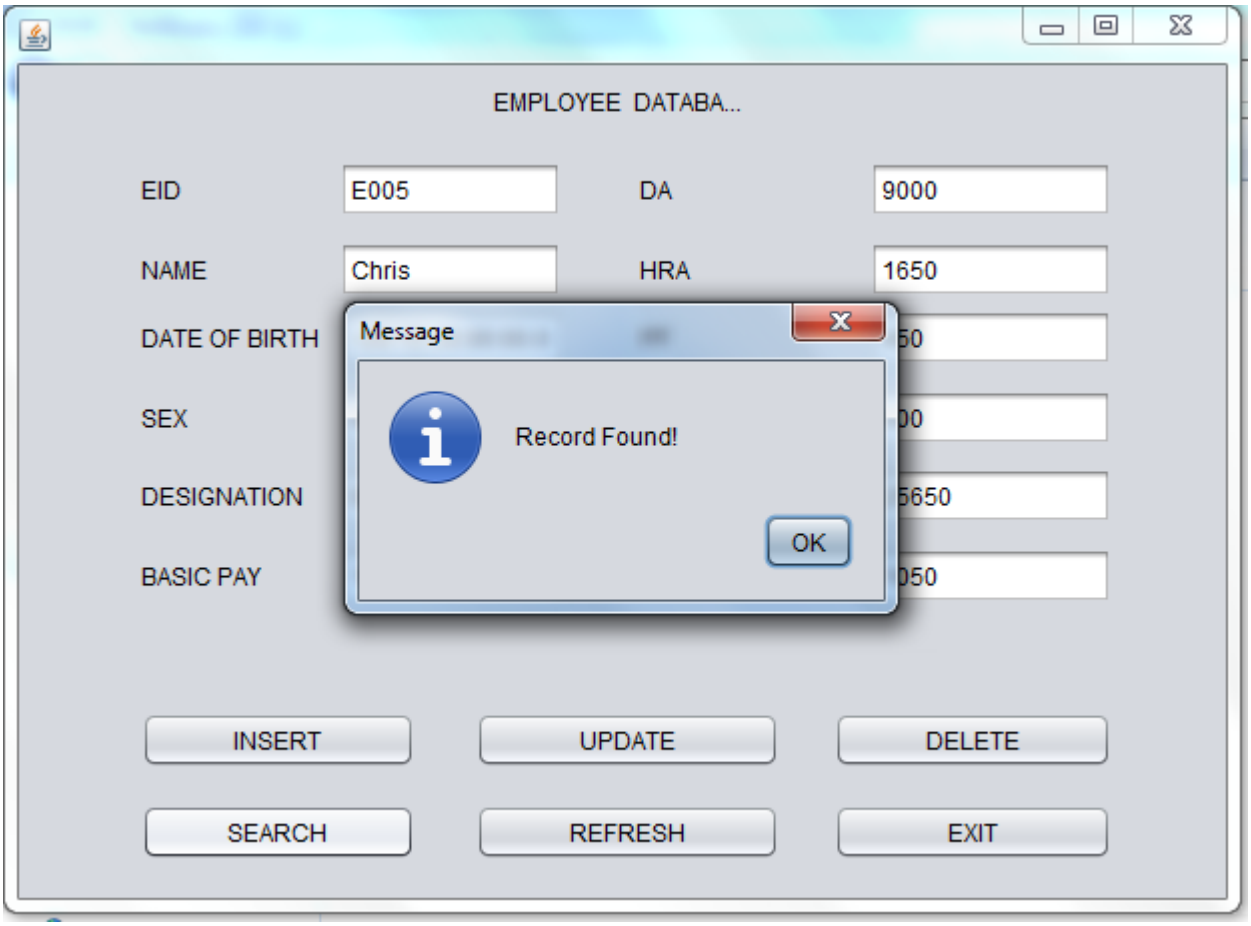
```
            }
        catch (SQLException ex) {
            Logger.getLogger(demo1.class.getName()).log(Level.SEVERE, null, ex);
            JOptionPane.showMessageDialog(this,ex.getMessage());
        }
    }

    private void deleteActionPerformed(java.awt.event.ActionEvent evt)
{
        // TODO add your handling code here:
        try{
            String sql = "delete from emp_payroll where eid=?";
            ps = con.prepareStatement(sql);
            ps.setString(1, eid.getText());
            ps.execute();
            JOptionPane.showMessageDialog(this, "Deleted!");
            refresh();
            }
            catch (SQLException ex) {
            Logger.getLogger(demo1.class.getName()).log(Level.SEVERE, null, ex);
            JOptionPane.showMessageDialog(this,ex.getMessage());
            }
    }

    private void exitActionPerformed(java.awt.event.ActionEvent evt)
{
        // TODO add your handling code here:
        System.exit(0);
    }


    private void searchActionPerformed(java.awt.event.ActionEvent evt)
{
        // TODO add your handling code here:
        try {
            String sql = "select * from emp_payroll where eid = '"+eid.getText()+"'";
            st =
con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,ResultSet.CONCUR_UPDATABLE);
            rs = st.executeQuery(sql);
            if(rs.next()){
                eid.setText(rs.getString(1));
                name.setText(rs.getString(2));
                dob.setText(rs.getString(3));
                sex.setText(rs.getString(4));
                designation.setText(rs.getString(5));
                basic.setText(rs.getString(6));
                da.setText(rs.getString(7));
                hra.setText(rs.getString(8));
                gross.setText(rs.getString(9));
                pf.setText(rs.getString(10));
                tot_deduc.setText(rs.getString(11));
                net_pay.setText(rs.getString(12));
                //String s=rs.getString(13);
                JOptionPane.showMessageDialog(this, "Record Found!");
```

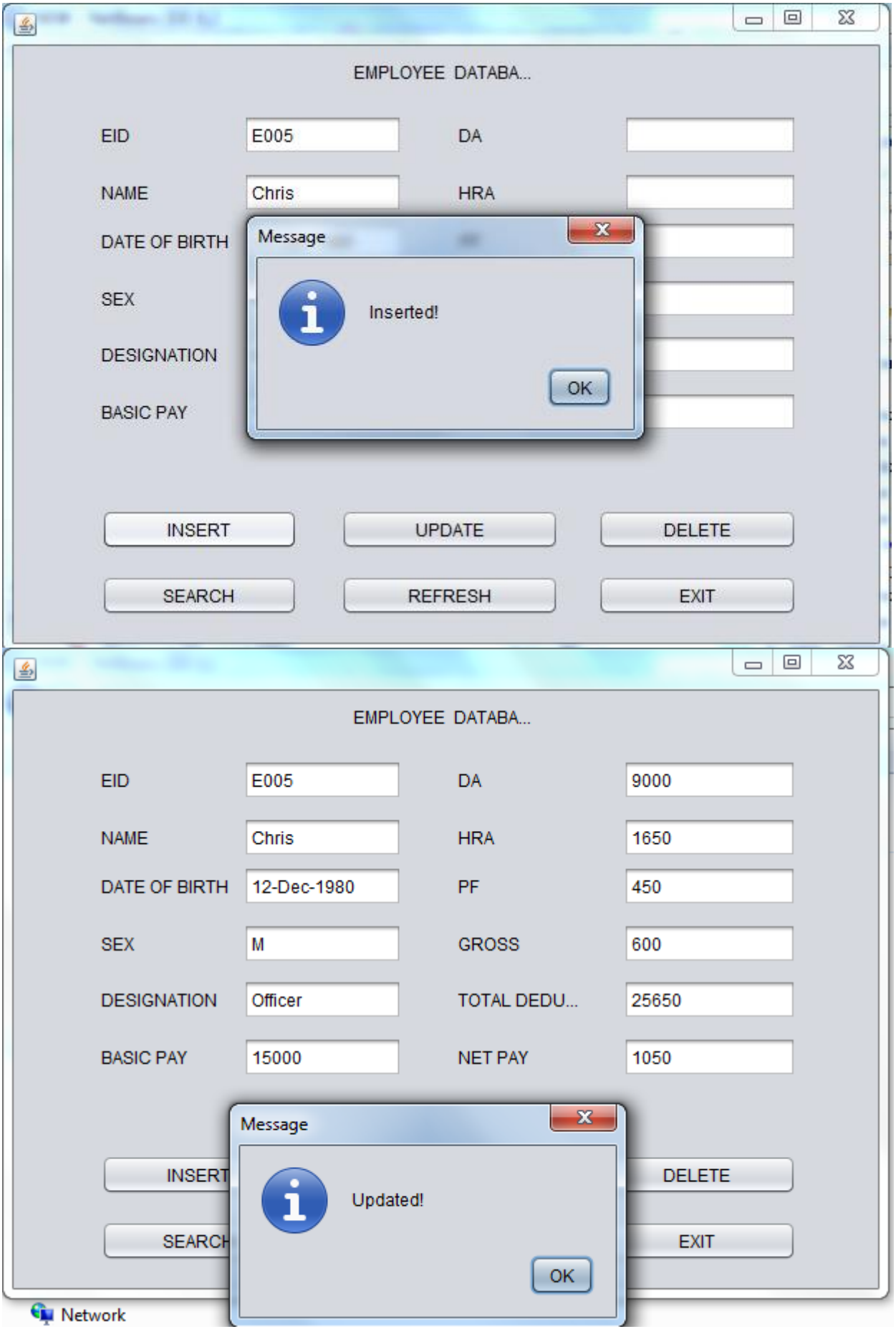
```
            }
            else
                JOptionPane.showMessageDialog(this, "Record Not Found!");
            }
            catch (SQLException ex) {
                Logger.getLogger(demo1.class.getName()).log(Level.SEVERE, null, ex);
                JOptionPane.showMessageDialog(this, ex.getMessage());
            }
    }

    private void updateActionPerformed(java.awt.event.ActionEvent evt)
{
        // TODO add your handling code here:
        try{
            String sql = "update emp_payroll set ename=?,dob=?,sex=?,designation=?,basic=?
where eid=?";
            ps = con.prepareStatement(sql);
            ps.setString(6, eid.getText());
            ps.setString(1, name.getText());
            ps.setString(2, dob.getText());
            ps.setString(3, sex.getText());
            ps.setString(4, designation.getText());
            ps.setString(5, basic.getText());
            ps.execute();
            calcf(eid.getText(),basic.getText());
            JOptionPane.showMessageDialog(this, "Updated!");
            refresh();
            }
        catch (SQLException ex) {
            Logger.getLogger(demo1.class.getName()).log(Level.SEVERE, null, ex);
            JOptionPane.showMessageDialog(this,ex.getMessage());
        }
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code
(optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default
look and feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
```

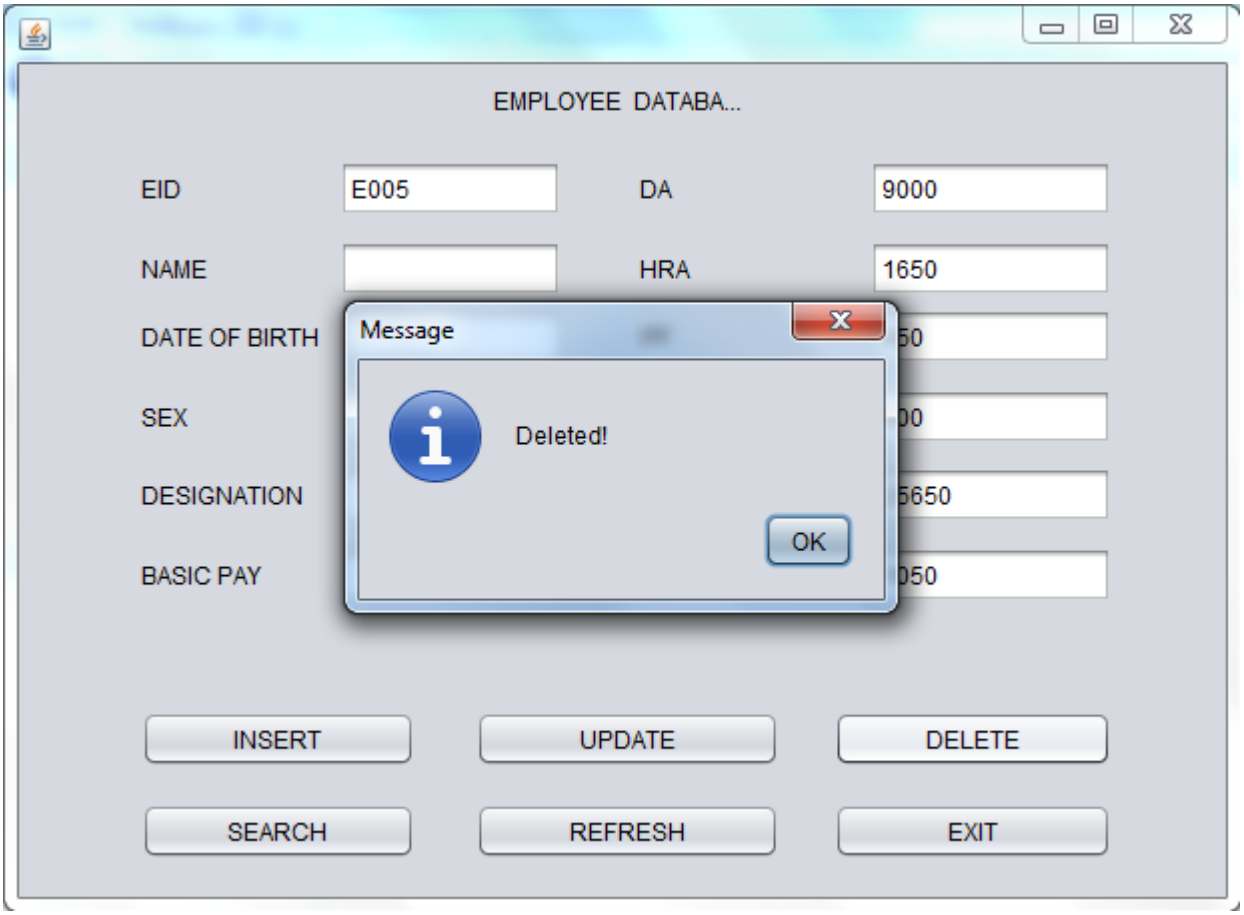
```
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(demo1.class.getName()).log(java.util.loggin
g.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(demo1.class.getName()).log(java.util.loggin
g.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(demo1.class.getName()).log(java.util.loggin
g.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(demo1.class.getName()).log(java.util.loggin
g.Level.SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new demo1().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JLabel BASICPAY;
    private javax.swing.JLabel DA;
    private javax.swing.JLabel DESIGNATION;
    private javax.swing.JLabel DOB;
    private javax.swing.JLabel EID;
    private javax.swing.JLabel GROSS;
    private javax.swing.JLabel HRA;
    private javax.swing.JLabel NAME;
    private javax.swing.JLabel NETPAY;
    private javax.swing.JLabel PF;
    private javax.swing.JLabel SEX;
    private javax.swing.JLabel SEX1;
    private javax.swing.JLabel TOTAL_DEDUCTION;
    private javax.swing.JTextField basic;
    private javax.swing.JTextField da;
    private javax.swing.JButton delete;
    private javax.swing.JTextField designation;
    private javax.swing.JTextField dob;
    private javax.swing.JTextField eid;
    private javax.swing.JButton exit;
    private javax.swing.JTextField gross;
    private javax.swing.JTextField hra;
    private javax.swing.JButton insert;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JTextField name;
    private javax.swing.JTextField net_pay;
    private javax.swing.JTextField pf;
    private javax.swing.JButton refresh;
    private javax.swing.JButton search;
    private javax.swing.JTextField sex;
```

```
    private javax.swing.JTextField tot_deduc;
    private javax.swing.JButton update;
    // End of variables declaration

    private void log(Level SEVERE, Object object, SQLException ex) {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }
}
```

Screenshots:




EMPLOYEE DATABA...
EID
E005
DA
NAME
Chris
HRA
DATE OF BIRTH
SEX
DESIGNATION
BASIC PAY
Message
Inserted!
OK
INSERT
UPDATE
DELETE
SEARCH
REFRESH
EXIT
EMPLOYEE DATABA...
EID
E005
DA
9000
NAME
Chris
HRA
1650
DATE OF BIRTH
12-Dec-1980
PF
450
SEX
M
GROSS
600
DESIGNATION
Officer
TOTAL DEDU...
25650
BASIC PAY
15000
NET PAY
1050
Message
Updated!
OK
INSERT
DELETE
SEARCH
EXIT
Network


EMPLOYEE DATABA...
EID
E005
DA
9000
NAME
HRA
1650
DATE OF BIRTH
SEX
DESIGNATION
BASIC PAY
Message
Deleted!
OK
INSERT
UPDATE
DELETE
SEARCH
REFRESH
EXIT