SRI SIVASUBRAMANIYA NADAR COLLEGE OF ENGINEERING

(AN AUTONOMOUS INSTITUTION,
AFFILIATED TO ANNA UNIVERSITY)

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

LABORATORY RECORD

NAME : Krithika Swaminathan

Reg. No. : 205001057

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

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 ..... Laboratory by

Name Krithika Swaminathan

Register Number 205001057

Semester IV

Branch CSE

Sri Sivasubramaniya Nadar College of Engineering, Kalavakkam.

During the Academic year 2021 - 22

Faculty

Head of the Department

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

Internal Examiner

External Examiner

INDEX

Name : KRITHIKA SWAMINATHAN Reg. No. 205001057

Sem : IV Sec : A

Ex. No.	Date of Expt.	Title of the Experiment	Page No.	Signature of the Faculty	Remarks
1.	10/03/2022	DDL for Mail Order Database	1	P.Sf 1/4/22	
2.	17/03/2022	DML Fundamentals	9		
3.	07/04/2022	Advanced DML - using Joins, Sub queries, Set Operations	24	P.Sf 28/4	
4.	21/04/2022	Views	38		
5.	28/04/2022	PL/SQL - Control Structures	70	P.Sf 12/05	
6.	12/05/2022	PL/SQL - Stored Procedures and Functions	83		

INDEX

Name : KRITHIKA SWAMINATHAN Reg. No. 205001057

Sem : IV Sec : A

Ex. No.	Date of Expt.	Title of the Experiment	Page No.	Signature of the Faculty	Remarks
7.	19/05/2022	PL/SQL - Triggers	100		
8.	26/05/2022	PL/SQL - Exception	110		
		Handling			
9.	09/06/2022	Database	117		
		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.

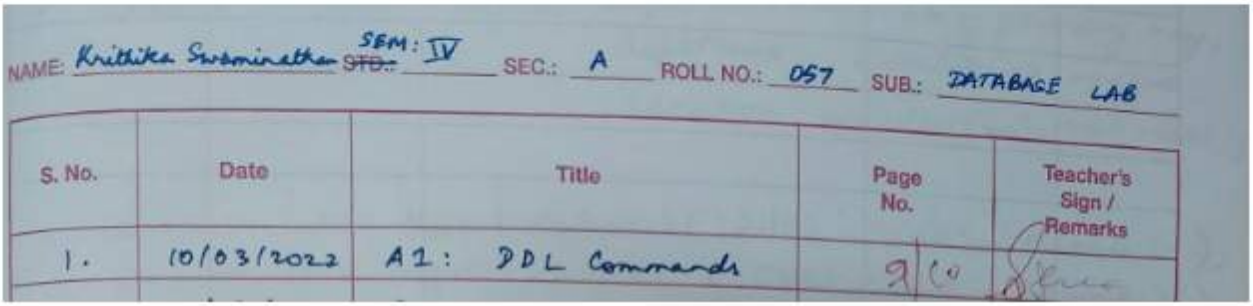
<u>Note:</u>
Populate each relation with relevant row(s) and prepare test cases to demonstrate that the requirements are satisfied.

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

1. Schema Diagram with constraints
2. Demo script file

Assignment 1 – DDL Commands

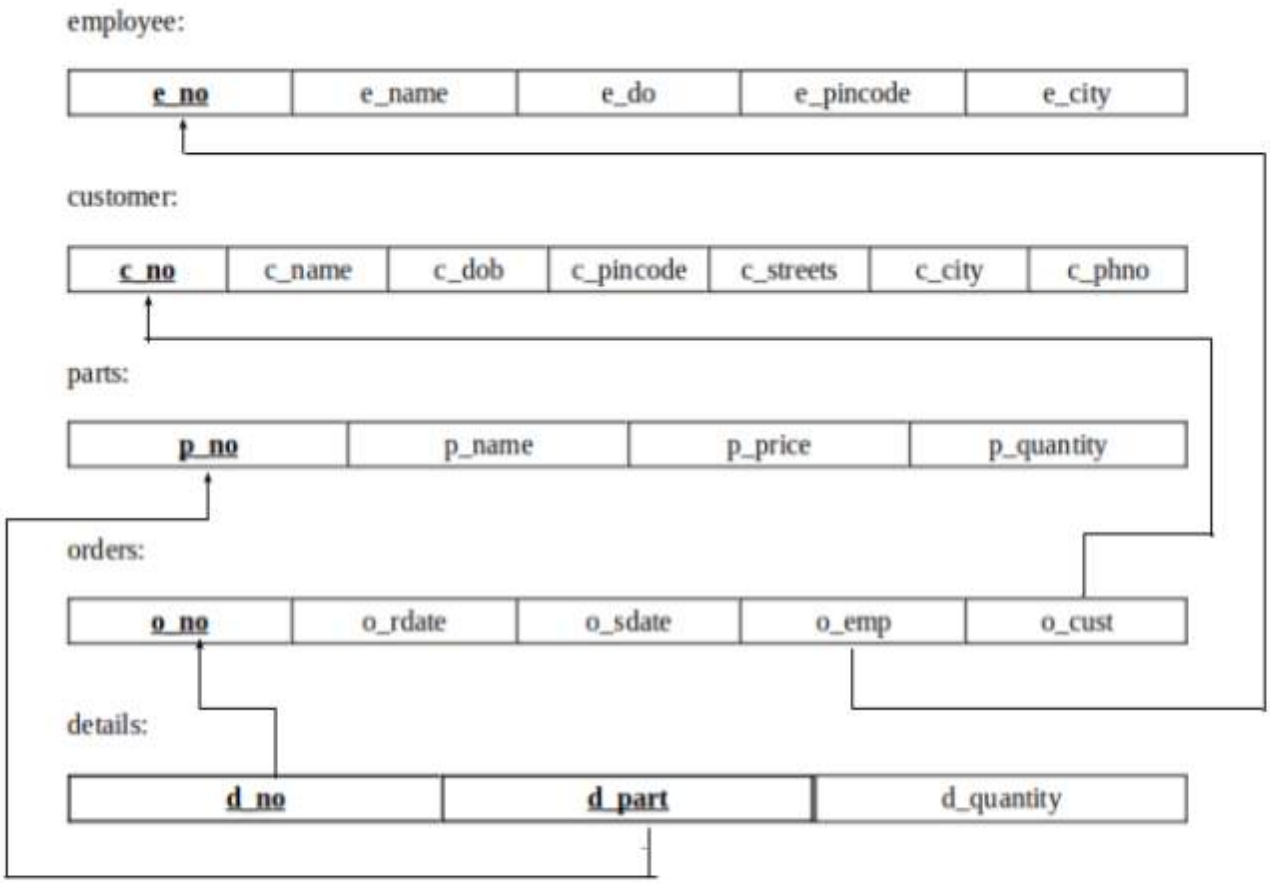
Validation:



NAME: Krithika Swaminathan ~~STD.:~~ SEM: IV SEC.: A ROLL NO.: 057 SUB.: DATABASE LAB

S. No.	Date	Title	Page No.	Teacher's Sign / Remarks
1.	10/03/2022	A1: DDL Commands	9/10	

Schema diagram:



Code:

```
REM ***Database Lab***
REM ***Assignment 1 - DDL commands***

drop table details;
drop table orders;
drop table parts;
drop table customer;
drop table employee;



REM ***Creating employee table***

create table employee(
        e_no varchar2(5) constraint emp_pk primary key,
        e_name char(15),
        e_dob DATE,
        e_pin number(6),
        constraint emp_format check(e_no like 'E%')
        );

REM ***Violating constraints of employee table***

REM Violating pk constraint for employee table
insert into employee values ('E001','Mira',TO_DATE('09/03/1985','DD/MM/YYYY'),600013);
insert into employee values ('E001','Manu',TO_DATE('09/03/1985','DD/MM/YYYY'),600013);

REM Violating fk constraint for employee table
insert into employee values ('E006','Devi',TO_DATE('21/05/1992','DD/MM/YYYY'),600020);

REM Violating check constraint for employee table
insert into employee values ('e006','Sowmya',TO_DATE('18/11/1991','DD/MM/YYYY'),600020);

REM Adding values into employee table
insert into employee values ('E013','Renu',TO_DATE('15/06/1999','DD/MM/YYYY'),600091);

select * from employee;



REM ***Creating customer table***
create table customer(
        c_no varchar2(5) constraint cust_pk primary key,
        c_name char(15),
        c_street char(15),
        c_pin number(6),
        c_dob DATE,
        c_phNo number(10) constraint ph_uniq UNIQUE,
```

```
        constraint cus_format check(c_no like 'C%')
        );

REM ***Violating constraints of customer table***

REM Violating pk constraint for customer table
insert into customer values ('C001','Rani','Street
1',600304,TO_DATE('31/10/1999','DD/MM/YYYY'),8847383878);
insert into customer values ('C001','Raj','Street
2',600304,TO_DATE('09/01/1997','DD/MM/YYYY'),8765676867);

REM Violating fk constraint for customer table
insert into customer values ('C003','Sita','Street
4',600304,TO_DATE('25/08/1999','DD/MM/YYYY'),8847383878);

REM Violating check constraint for customer table
insert into customer values ('c004','Rima','Street
1',600304,TO_DATE('31/10/1999','DD/MM/YYYY'),7764545656);

REM Adding values into customer table
insert into customer values ('C002','Raj','Street
2',600304,TO_DATE('09/01/1997','DD/MM/YYYY'),8765676867);
insert into customer values ('C009','Mani','Street
9',600102,TO_DATE('24/03/2002','DD/MM/YYYY'),7892354260);

select * from customer;



REM ***Creating parts table***
create table parts(
        p_no varchar2(5) constraint part_pk primary key,
        p_name char(15),
        p_price number(10) constraint pr_nn not null,
        p_qty number(10) constraint qty_chk check(p_qty>0),
        constraint part_format check(p_no like 'P%')
        );


REM ***Violating constraints of parts table***

REM Violating pk constraint for parts table
insert into parts values ('P001','Screw',350,56);
insert into parts values ('P001','Nail',200,40);

REM Violating not null constraint for parts table
insert into parts values ('P002','Bolts',null,70);

REM Violating check constraint of quantity for parts table
insert into parts values ('P003','Virtual item',0,-2);
```

```
REM Violating check constraint of part number for parts table
insert into parts values ('p003','Hammer',500,5);

select * from parts;



REM ***Creating orders table***
create table orders(
        o_no varchar2(5) constraint ord_pk primary key,
        e_no varchar2(5) constraint emp_fk2 references employee(e_no),
        c_no varchar2(5) constraint cust_fk2 references customer(c_no),
        o_rdate DATE,
        o_sdate DATE,
        constraint date_chk check(o_rdate<o_sdate)
        );

REM ***Violating constraints of orders table***

REM Violating pk constraint for orders table
insert into orders values
('1010','E001','C001',TO_DATE('03/08/2009','DD/MM/YYYY'),TO_DATE('09/10/2009','DD/MM/
YYYY'));
insert into orders values
('1010','E002','C002',TO_DATE('04/08/2009','DD/MM/YYYY'),TO_DATE('10/10/2009','DD/MM/
YYYY'));

REM Violating fk constraint for orders table
insert into orders values
('1015','E111','C001',TO_DATE('04/08/2009','DD/MM/YYYY'),TO_DATE('11/10/2009','DD/MM/
YYYY'));
insert into orders values
('1015','E001','C111',TO_DATE('04/08/2009','DD/MM/YYYY'),TO_DATE('11/10/2009','DD/MM/
YYYY'));

REM Violating check constraint rd<sd for orders table
insert into orders values
('1200','E002','C002',TO_DATE('04/08/2009','DD/MM/YYYY'),TO_DATE('11/10/2008','DD/MM/
YYYY'));

select * from orders;



REM ***Creating details table***
create table details(
        o_no varchar2(5) constraint ord_fk1 references orders(o_no),
        p_no varchar2(15) constraint ord_fk2 references parts(p_no),
        qty number(10) constraint qty_ch2 check(qty>0),
```

```
        constraint ord_pk_pk primary key(o_no,p_no)
        );

REM ***Violating constraints of details table***

REM Violating pk constraint for details table
insert into details values ('1005','P001',7);
insert into details values ('1005','P001',8);

REM Violating fk constraint for details table
insert into details values ('1050','P001',5);
insert into details values ('1005','P009',5);

select * from details;



REM ***2nd part***

REM 7 It is identified that the following attributes are to be included in respective relations:
Parts (reorder level), Employees (hiredate)

desc parts;

alter table parts
        add reord_lev number(5);

desc employee;

alter table employee
        add hiredate DATE;

REM 8 The width of a customer name is not adequate for most of the customers.

desc customer;

alter table customer
        modify c_name char(30);

desc customer;

REM 9 The dateofbirth of a customer can be addressed later / removed from the schema

alter table customer
        drop column c_dob;

desc customer;

REM 10 An order can not be placed without the receive date.
```

```
alter table orders
        modify o_rdate DATE constraint ord_notnn NOT NULL;

desc orders;

REM 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.

REM inserting some values into the tables

insert into orders values
('2001','E001','C001',TO_DATE('19/12/2021','DD/MM/YYYY'),TO_DATE('23/01/2022','DD/MM/
YYYY'));

insert into orders values
('2002','E001','C001',TO_DATE('05/07/2020','DD/MM/YYYY'),TO_DATE('08/11/2020','DD/MM/
YYYY'));

insert into details values ('2001','P001',3);

REM displaying the tables

select * from orders;

select * from details;

REM Violating integrity constraint - child record found

delete from orders where o_no='2001';

REM ON DELETE CASCADE

alter table details drop constraint ord_fk1;

alter table details add constraint ord_fk3 foreign key(o_no) references orders(o_no) on delete
cascade;

delete from orders where o_no='2001';

REM Check to see that the cancelled order is deleted from all tables

select * from orders;

select * from details;
```

	SSN COLLEGE OF ENGINEERING	***Faculty:***
	Department of Computer Science & Engineering	***P.Mirunalini, Asso. Prof.*** ***N.Sujaudeen, Asst. Prof***
	CS8481 – DBMS Lab	
Assigned: 10-Mar-22		***Assignment – 2***
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

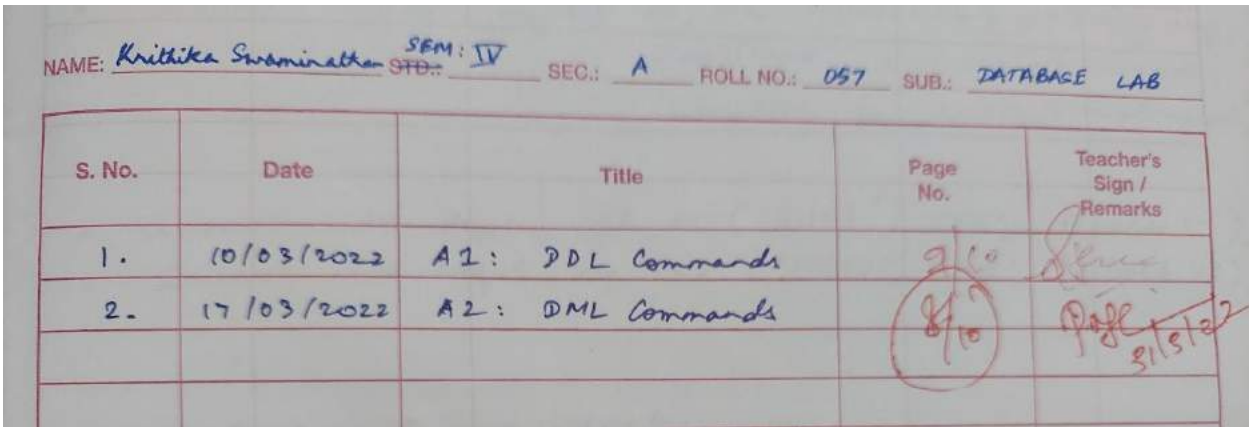
11. Mark an intermediate point in the transaction(savepoint).
12. Insert a new tuple into the nobel relation.
13. Update the aff_role of literature laureates as 'Linguists'.
14. Delete the laureate(s) who was awarded in Enzymes field.
15. Discard the most recent update operations (rollback).
16. Commit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

What you have to submit:

1. Schema Diagram with constraints
2. Demo script file

Assignment 2 – DML Commands

Validation:



NAME: Krithika Swaminathan SEM: IV SEC.: A ROLL NO.: 057 SUB.: DATABASE LAB

S. No.	Date	Title	Page No.	Teacher's Sign / Remarks
1.	10/03/2022	A1: DDL Commands	9/10	
2.	17/03/2022	A2: DML Commands	8/10	31/5/22

Schema diagram:

nobel_id	name	gender	category	field	year_award	aff_role	dob	country

Script file:

```
SQL> set echo on
SQL> @z:\a2dml.sql
SQL> REM ***DATABASE MANAGEMENT SYSTEMS LAB***
SQL> REM ***Assignment 2: DML Fundamentals***
SQL>
SQL> REM ***Drop nobel table***
SQL> drop table nobel;

Table dropped.

SQL>
SQL>
SQL> REM ***Creating nobel relation***
SQL> create table nobel(
 2  laureate_id number(3) constraint laur_pk PRIMARY KEY,
 3  name varchar2(30) constraint name_nn NOT NULL,
 4  gender char(1) constraint gen_ch check (gender in('m','f')),
 5  category char(3) constraint cat_ch check (category in('Phy','Che','Med','Lit','Pea','Eco','Lit')),
 6  field varchar2(25),
 7  year_award number(4),
 8  aff_role varchar2(30),
 9  dob date,
```

```
10  country varchar2(10)
11  );

Table created.

SQL>
SQL>
SQL> REM ***Populate nobel relation***
SQL> insert into nobel values(100,'Robert B. Laughlin','m','Phy','Condensed matter',1998,'Stanford
University','01-nov-1950','USA');

1 row created.

SQL> insert into nobel values(101,'Horst L Stormer','m','Phy','Condensed matter',1998,'Columbia
University','06-apr-1949','Germany');

1 row created.

SQL> insert into nobel values(102,'Daniel C. Tsui','m','Phy','Condensed matter',1998,'Princeton
University','28-feb-1939','China');

1 row created.

SQL> insert into nobel values(103,'Walter Kohn','m','Che','Theoretical Chemistry',1998,'University
of California','09-mar-1923','Austria');

1 row created.

SQL> insert into nobel values(104,'John Pople','m','Che','Theoretical Chemistry',1998,'North
Western University','31-oct-1925','UK');

1 row created.

SQL> insert into nobel values(106,'John Hume','m','Pea','Negotiation',1998,'Labour party
Leader','18-jan-1937','Ireland');

1 row created.

SQL> insert into nobel values(107,'David Trimble','m','Pea','Negotiation',1998,'Ulster Unionist
party Leader','15-oct-1944','Ireland');

1 row created.

SQL> insert into nobel values(108,'Louis J Ignaroo','m','Med','Cardiovascular
system',1998,'University of California','31-may-1941','USA');

1 row created.

SQL> insert into nobel values(109,'Amartya Sen','m','Eco','Welfare Economics',1998,'Trinity
College','03-nov-1933','India');
```

1 row created.

SQL> insert into nobel values(110,'Jose Saramago','m','Lit','Portuguese',1998,null,'16-nov-1922','Portugal');

1 row created.

SQL> insert into nobel values(111,'Eric A Cornell','m','Phy','Atomic physics',2001,'University of Colorado','19-dec-1961','USA');

1 row created.

SQL> insert into nobel values(112,'Carl E Wieman','m','Phy','Atomic physics',2001,'University of Colorado','26-mar-1951','USA');

1 row created.

SQL> insert into nobel values(113,'Ryoji Noyori','m','Che','Organic Chemistry',2001,'Nagoya University','03-sep-1938','Japan');

1 row created.

SQL> insert into nobel values(114,'K Barry Sharpless','m','Che','Organic Chemistry',2001,'Scripps Research Institute','28-apr-1941','USA');

1 row created.

SQL> insert into nobel values(115,'Kofi Annan','m','Pea','World organizing',2001,'UN General','08-apr-1938','Ghana');

1 row created.

SQL> insert into nobel values(116,'Joerge A Akeriof','m','Eco','Economic of Information',2001,'University of California','17-jun-1940','USA');

1 row created.

SQL> insert into nobel values(117,'V S Naipaul','m','Lit','English',2001,null,'17-aug-1932','UK');

1 row created.

SQL> insert into nobel values(118,'Charles A Kao','m','Phy','Fiber technology',2009,'University of Hongkong','04-nov-1933','China');

1 row created.

SQL> insert into nobel values(119,'Willard S Boyle','m','Phy','Semiconductor technology',2009,'Bell Laboratories','19-aug-1924','Canada');

```
1 row created.

SQL> insert into nobel values(120,'George E Smith','m','Phy','Semiconductor technology',2009,'Bell
Laboratories','10-may-1930','USA');

1 row created.

SQL> insert into nobel values(121,'Venkatraman
Ramakrishnan','m','Che','Biochemistry',2009,'MRC Laboratory','19-aug-1952','India');

1 row created.

SQL> insert into nobel values(122,'Ada E Yonath','f','Che','Biochemistry',2009,'Weizmann Institute
of Science','22-jun-1939','Israel');

1 row created.

SQL> insert into nobel values(123,'Elizabeth H Blackburn','f','Med','Enzymes',2009,'University of
California','26-nov-1948','Australia');

1 row created.

SQL> insert into nobel values(124,'Carol W Greider','f','Med','Enzymes',2009,'Johns Hopkins
University','15-apr-1961','USA');

1 row created.

SQL> insert into nobel values(125,'Barack H Obama','m','Pea','World organizing',2009,'President of
USA','04-aug-1961','USA');

1 row created.

SQL> insert into nobel values(126,'Oliver E Williamson','m','Eco','Economic
governance',2009,'University of California','27-sep-1932','USA');

1 row created.

SQL> insert into nobel values(127,'Elinor Ostrom','m','Eco','Economic governance',2009,'Indiana
University','07-aug-1933','USA');

1 row created.

SQL> insert into nobel values(128,'Herta
Muller','f','Lit','German',2009,null,'17-aug-1953','Romania');

1 row created.

SQL>
SQL> REM ***************************END OF
INSERT****************************************
```

```
SQL>
SQL>
SQL> REM Displaying the table
SQL> select * 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                          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                          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 Israel

       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                           17-AUG-53 Romania


28 rows selected.

SQL>
SQL>
SQL> REM ***QUESTIONS & ANSWERS***
SQL>
SQL>
SQL> REM ***DML***
SQL>
SQL> REM 1. Display the nobel laureate(s) who born after 1Jul1960.
SQL>
```

```
SQL> select * from nobel where dob>'01-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>
SQL> REM 2. Display the Indian laureate (name, category, field, country, year awarded) who was
awarded in the Chemistry category.
SQL>
SQL> select name, category, field, country, year_award from nobel where category='Che' and
country='India';

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


SQL>
SQL>
SQL> REM 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>
SQL> select name, category, field, year_award from nobel where (year_award between 2000 and
2005) and (category='Phy' or category='Che');

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>
```

```
SQL> REM 4. Display the laureates name with their age at the time of award for the Peace
category.
SQL>
SQL> select name, year_award - extract(year from dob) as age from nobel where category='Pea';

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

SQL>
SQL>
SQL> REM 5. Display the laureates (name,category,aff_role,country) whose name starts with A or
ends with a, but not from Israel.
SQL>
SQL> select name, category, aff_role, country from nobel where (name like 'A%' or name like '%a')
and (country!='Israel');

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

SQL>
SQL>
SQL> REM 6. Display the name, gender, affiliation, dob and country of laureates who was born in
1950s. Label the dob column as Born 1950.
SQL>
SQL> select name, gender, aff_role, country, dob as Born_1950 from nobel where extract(year from
dob) like '%195_';

NAME                           G AFF_ROLE                         COUNTRY
------------------------------ - ------------------------------ ----------
BORN_1950
---------
Robert B. Laughlin              m Stanford University            USA
01-NOV-50

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

Venkatraman Ramakrishnan        m MRC Laboratory                 India
19-AUG-52


NAME                           G AFF_ROLE                         COUNTRY
------------------------------ - ------------------------------ ----------
BORN_1950
```

```
---------
Herta Muller                f                         Romania
17-AUG-53


SQL>
SQL>
SQL> REM 7. Display the laureates (name,gender,category,aff_role,country) whose name starts
with A, D or H. Remove the laureate if he/she does not have any affiliation. Sort the results in
ascending order of name.
SQL>
SQL> REM select name, gender, category aff_role, country from nobel where ((name like 'A%') or
(name like 'D%') or (name like 'H%')) and aff_role is not null order by name;
SQL> select name, gender, category aff_role, country from nobel where substr(name,1,1) in
('A','D','H') and aff_role is not null order by name;

NAME                        G AFF COUNTRY
------------------------------ - --- ----------
Ada E Yonath                 f Che Israel
Amartya Sen                  m Eco India
Daniel C. Tsui               m Phy China
David Trimble                m Pea Ireland
Horst L Stormer              m Phy Germany

SQL>
SQL>
SQL> REM 8. Display the university name(s) that has to its credit by having at least 2 nobel
laureates with them.
SQL>
SQL> select distinct(aff_role) from nobel group by aff_role having count(aff_role)>=2 and
substr(aff_role,1,10)='University';

AFF_ROLE
------------------------------
University of California
University of Colorado

SQL>
SQL>
SQL> REM 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.
SQL>
SQL> select country, min(dob) Younger, max(dob) Elder from nobel group by country having
count(country)>1 order by country;

COUNTRY    YOUNGER   ELDER
---------- --------- ---------
China      04-NOV-33 28-FEB-39
India      03-NOV-33 19-AUG-52
```

```
Ireland    18-JAN-37 15-OCT-44
UK         31-OCT-25 17-AUG-32
USA         10-MAY-30 19-DEC-61

SQL>
SQL>
SQL> REM 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 the USA.
SQL>
SQL> select year_award, category, field from nobel where country!='USA' group by category, field,
year_award having count(laureate_id)>1;

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

SQL>
SQL>
SQL>
SQL> REM ***TCL***
SQL>
SQL> REM 11. Mark an intermediate point in the transaction (savepoint).
SQL>
SQL> select count(*) no_of_entries from nobel;

NO_OF_ENTRIES
-------------
          28

SQL> savepoint inter;

Savepoint created.

SQL>
SQL>
SQL> REM 12. Insert a new tuple into the nobel relation.
SQL>
SQL> insert into nobel values(129,'Akira Suzuki','m','Che','Organic Synthesis',2010,'Hokkaido
University','12-sep-1930','Japan');

1 row created.

SQL> select * from nobel where laureate_id=129;

LAUREATE_ID NAME                           G CAT FIELD
----------- ------------------------------ - --- -------------------------
YEAR_AWARD AFF_ROLE                        DOB       COUNTRY
```

```
---------- ------------------------------ --------- ----------
       129 Akira Suzuki                   m Che Organic Synthesis
      2010 Hokkaido University            12-SEP-30 Japan


SQL>
SQL>
SQL> REM 13. Update the aff_role of literature laureates as 'Linguists'.
SQL>
SQL> update nobel set aff_role='Linguists' where category='Lit';

3 rows updated.

SQL> select * from nobel 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>
SQL>
SQL> REM 14. Delete the laureate(s) who was awarded in Enzymes field.
SQL>
SQL> delete from nobel where field='Enzymes';

2 rows deleted.

SQL> select * from nobel where field='Enzymes';

no rows selected

SQL>
SQL>
SQL> REM 15. Discard the most recent update operations (rollback).
SQL>
SQL> select count(*) no_of_entries from nobel;

NO_OF_ENTRIES
-------------
           27
```

```
SQL>
SQL> rollback to inter;

Rollback complete.

SQL>
SQL> select count(*) no_of_entries from nobel;

NO_OF_ENTRIES
-------------
         28

SQL>
SQL>
SQL> REM 16. Commit the changes.
SQL>
SQL> commit;

Commit complete.

SQL> spool off;
```

	SSN COLLEGE OF ENGINEERING	***Faculty:***
	Department of Computer Science & Engineering	***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:

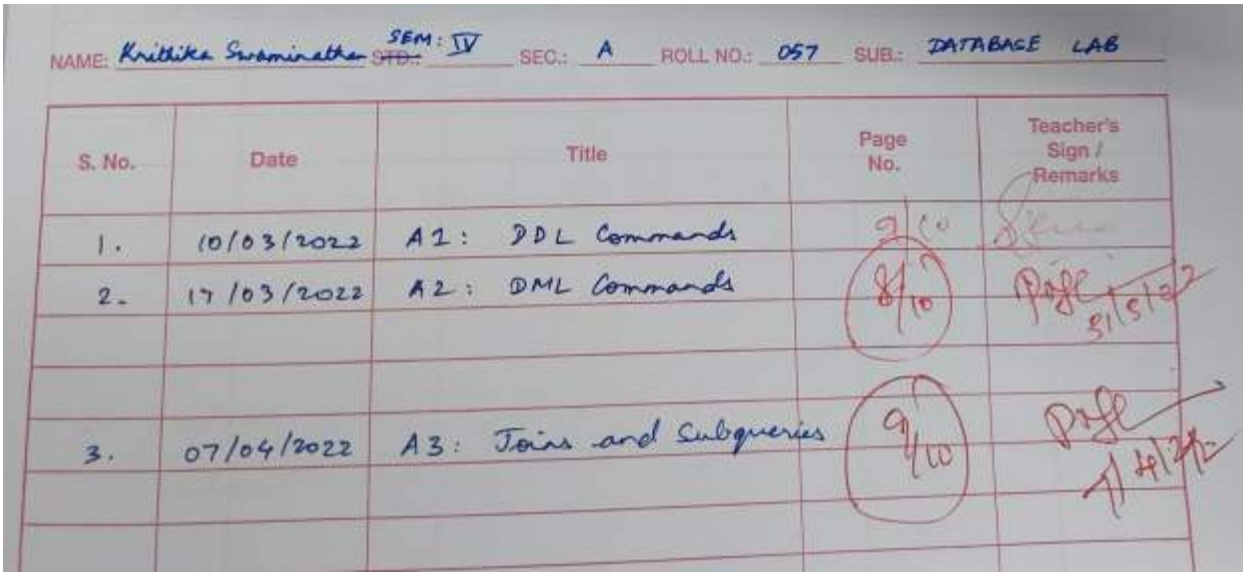
9. Display the product details of both *Pie* and *Bear Claw*.
10. Display the customers details who haven't placed any orders.
11. Display the food that has the same flavor as that of the common flavor between the *Meringue* and *Tart*.

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

1. Schema Diagram with constraints
2. Demo script file

Assignment 3 – Joins and SubQueries

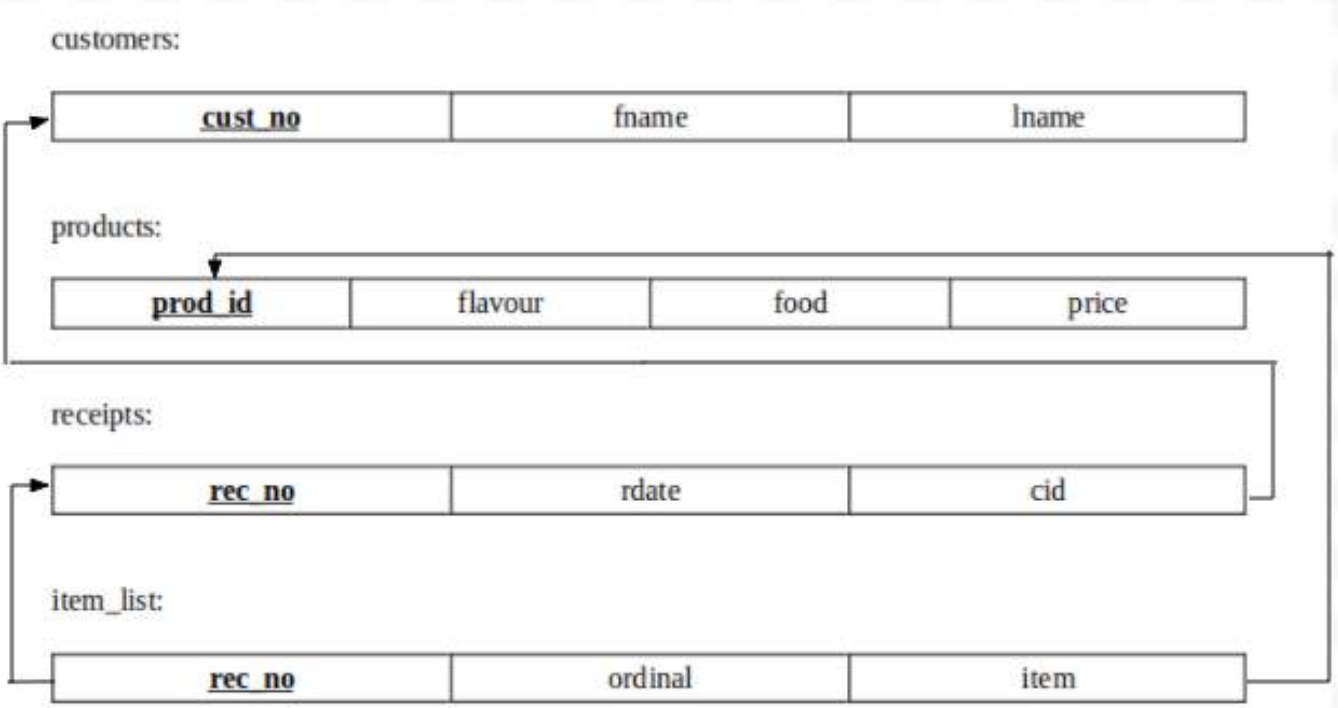
Validation:



NAME: Krithika Swaminathan SEM: IV SEC.: A ROLL NO.: 057 SUB.: DATABASE LAB

S. No.	Date	Title	Page No.	Teacher's Sign / Remarks
1.	10/03/2022	A1: DDL Commands	9/10	
2.	17/03/2022	A2: DML Commands	8/10	Prof 31/5/22
3.	07/04/2022	A3: Joins and Subqueries	9/10	Prof 7/4/22

Schema diagram:



Data file:

```
SQL> @C:/Krithika/DBL/a3data.sql;
SQL> REM Assignment 3
SQL> REM Population of Bakery Database
SQL> REM ------------------------------------------------------------------------
> REM CUSTOMERS ( customer number,  Last name, First name)
SQL> REM ------------------------------------------------------------------------
>
SQL> drop table item_list;

Table dropped.

SQL> drop table receipts;

Table dropped.

SQL> drop table products;

Table dropped.

SQL> drop table customers;

Table dropped.

SQL>
SQL> create table customers(
  2         cust_no number(2) constraint c_pk primary key,
  3         lname varchar2(20),
  4         fname varchar2(20)
  5         );

Table created.

SQL>
SQL> insert into customers values(1, 'LOGAN', 'JULIET');

1 row created.

.
.
.

SQL> insert into customers values(21, 'JOHN', 'DAVID');

1 row created.

SQL>
SQL> REM ------------------------------------------------------------------------
> REM PRODUCTS (product number,  Flavor, Food, Price)
```

```
SQL> REM ----------------------------------------------------------------------
>
SQL> create table products(
  2         prod_id varchar2(20) constraint prod_pk primary key,
  3         flavour varchar2(20),
  4         food varchar2(20),
  5         price number
  6         );

Table created.

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

1 row created.

.
.
.

SQL> insert into products values('51-BLU','Blueberry','Danish',1.15);

1 row created.

SQL>
SQL> REM ----------------------------------------------------------------------
> REM RECEIPTS(receipt number, receipt Date, Customer)
SQL> REM ----------------------------------------------------------------------
>
SQL> create table receipts(
  2         rec_no number(5) constraint rec_pk primary key,
  3         rdate date,
  4         cid number(2) constraint rec_fk references customers(cust_no)
  5         );

Table created.

SQL>
SQL> INSERT INTO Receipts values(18129, '28-Oct-2007', 15);

1 row created.

.
.
.

SQL> INSERT INTO Receipts values(34378, '23-Oct-2007', 6);

1 row created.
```

```
SQL>
SQL> REM ----------------------------------------------------------------------------
> REM ITEM_LIST (receipt number,  Ordinal, Item)
SQL> REM ----------------------------------------------------------------------------
>
SQL> create table item_list(
  2          rec_no number(5) constraint it_fk1 references receipts(rec_no),
  3          ordinal number(2),
  4          item varchar2(20) constraint it_fk2 references products(prod_id),
  5          constraint item_pk primary key(rec_no,ordinal)
  6          );

Table created.

SQL>
SQL> insert into item_list values(18129, 1,  '70-TU');

1 row created.

.
.
.

SQL> insert into item_list values(34378, 2,  '45-VA');

1 row created.

SQL>
SQL> REM *** End of database population ***
SQL>
SQL>
SQL> REM *** Checking tables ***
SQL>
SQL> select * from customers;

   CUST_NO LNAME               FNAME
---------- -------------------- --------------------
         1 LOGAN                JULIET
         2 ARZT                TERRELL
         3 ESPOSITA            TRAVIS
         4 ENGLEY              SIXTA
         5 DUNLOW              OSVALDO
         6 SLINGLAND           JOSETTE
         7 TOUSSAND            SHARRON
         8 HELING              RUPERT
         9 HAFFERKAMP          CUC
        10 DUKELOW             CORETTA
        11 STADICK             MIGDALIA

   CUST_NO LNAME               FNAME
```

```
---------- -------------------- --------------------
        12 MCMAHAN              MELLIE
        13 ARNN                 KIP
        14 SOPKO                RAYFORD
        15 CALLENDAR            DAVID
        16 CRUZEN               ARIANE
        17 MESDAQ               CHARLENE
        18 DOMKOWSKI            ALMETA
        19 STENZ                NATACHA
        20 ZEME                 STEPHEN
        21 JOHN                 DAVID

21 rows selected.

SQL> select * from products;

PROD_ID              FLAVOUR              FOOD                 PRICE
-------------------- -------------------- -------------------- ----------
20-BC-C-10           Chocolate            Cake                   8.95
20-BC-L-10           Lemon                Cake                   8.95
20-CA-7.5            Casino               Cake                  15.95
24-8x10              Opera                Cake                  15.95
25-STR-9             Strawberry           Cake                  11.95
26-8x10              Truffle              Cake                  15.95
45-CH                Chocolate            Eclair                 3.25
45-CO                Coffee               Eclair                 3.5
45-VA                Vanilla              Eclair                3.25
46-11                Napoleon             Cake                  13.49
90-ALM-I             Almond               Tart                   3.75

PROD_ID              FLAVOUR              FOOD                 PRICE
-------------------- -------------------- -------------------- ----------
90-APIE-10           Apple                Pie                    5.25
90-APP-11            Apple                Tart                   3.25
90-APR-PF            Apricot              Tart                   3.25
90-BER-11            Berry                Tart                   3.25
90-BLK-PF            Blackberry           Tart                   3.25
90-BLU-11            Blueberry            Tart                   3.25
90-CH-PF             Chocolate            Tart                   3.75
90-CHR-11            Cherry               Tart                   3.25
90-LEM-11            Lemon                Tart                   3.25
90-PEC-11            Pecan                Tart                   3.75
70-GA                Ganache              Cookie                 1.15

PROD_ID              FLAVOUR              FOOD                 PRICE
-------------------- -------------------- -------------------- ----------
70-GON               Gongolais            Cookie                 1.15
70-R                 Raspberry            Cookie                 1.09
70-LEM               Lemon                Cookie          .79
70-M-CH-DZ           Chocolate            Meringue               1.25
```

```
70-M-VA-SM-DZ       Vanilla            Meringue                1.15
70-MAR              Marzipan           Cookie                  1.25
70-TU               Tuile              Cookie                  1.25
70-W                Walnut             Cookie                  .79
50-ALM              Almond             Croissant               1.45
50-APP              Apple              Croissant               1.45
50-APR              Apricot            Croissant               1.45

PROD_ID             FLAVOUR            FOOD                    PRICE
-------------------- -------------------- -------------------- ----------
50-CHS              Cheese             Croissant               1.75
50-CH               Chocolate          Croissant               1.75
51-APR              Apricot            Danish                  1.15
51-APP              Apple              Danish                  1.15
51-ATW              Almond             Twist                   1.15
51-BC               Almond             Bear Claw               1.95
51-BLU              Blueberry          Danish                  1.15

40 rows selected.

SQL> select * from receipts;

   REC_NO RDATE           CID
---------- --------- ----------
    18129 28-OCT-07        15
    51991 17-OCT-07        14
    83085 12-OCT-07         7
    70723 28-OCT-07        20
    13355 19-OCT-07         7
    52761 27-OCT-07         8
    99002 13-OCT-07        20
    58770 22-OCT-07        18
    84665 10-OCT-07         6
    55944 16-OCT-07        19
    42166 14-OCT-07         8

   REC_NO RDATE           CID
---------- --------- ----------
    16034 10-OCT-07         4
    25906 29-OCT-07        15
    27741 25-OCT-07         8
    64451 10-OCT-07        11
    41028 06-OCT-07        17
    73716 29-OCT-07        18
    76667 14-OCT-07        15
    21040 03-OCT-07         6
    48332 15-OCT-07        20
    35011 10-OCT-07        20
    95962 26-OCT-07         8
```

```
...
...
...

   REC_NO RDATE           CID
---------- --------- ----------
    46674 29-OCT-07         15
    67946 18-OCT-07          7
    31233 20-OCT-07         13
    15904 06-OCT-07         13
    17488 20-OCT-07          6
    97097 23-OCT-07          9
    50512 27-OCT-07          8
    11548 21-OCT-07         13
    29908 14-OCT-07         13
    20127 07-OCT-07         15
    41963 29-OCT-07          8

   REC_NO RDATE           CID
---------- --------- ----------
    16532 21-OCT-07          4
    34378 23-OCT-07          6

200 rows selected.

SQL> select * from item_list;

   REC_NO    ORDINAL ITEM
---------- ---------- --------------------
    18129          1 70-TU
    51991          1 90-APIE-10
    51991          2 90-CH-PF
    51991          3 90-APP-11
    51991          4 26-8x10
    83085          1 25-STR-9
    83085          2 24-8x10
    83085          3 90-APR-PF
    83085          4 51-ATW
    83085          5 26-8x10
    70723          1 45-CO

...
...
...

   REC_NO    ORDINAL ITEM
---------- ---------- --------------------
    41963          2 90-CH-PF
    16532          1 50-APP
    16532          2 70-MAR
```

```
16532        3 70-TU
16532        4 24-8x10
34378        1 90-CHR-11
34378        2 45-VA

557 rows selected.

SQL>
SQL> REM ******************** END OF DATA FILE ************************
```

Script file:

```
SQL> @C:/Krithika/DBL/a3queries.sql;
SQL> REM Assignment 3
SQL>
SQL> REM ---------------------------------------------------------------------------
> REM *** ASSIGNMENT QUESTIONS ***
SQL> REM ---------------------------------------------------------------------------
>
SQL> REM **I**_____Write the following using sub-queries: _____**
SQL>
SQL> REM 1. Display the food details that is not purchased by any of customers.
SQL>
SQL> select * from products where prod_id not in (select item from item_list);

PROD_ID              FLAVOUR              FOOD                 PRICE
-------------------- -------------------- -------------------- ----------
20-BC-C-10           Chocolate            Cake                    8.95


SQL>
SQL>
SQL> REM 2. Show the customer details who had placed more than 2 orders on the same date.
SQL>
SQL> select * from customers where cust_no in (select cid from receipts group by cid,rdate having
count(rec_no)>2) order by cust_no;

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

SQL>
SQL>
SQL> REM 3. Display the products details that has been ordered maximum by the customers. (use
ALL)
SQL>
SQL> select * from products where prod_id in (select item from item_list group by item having
count(item)>= all(select max(count(item)) from item_list group by item));
```

```
PROD_ID            FLAVOUR            FOOD                 PRICE
-------------------- -------------------- -------------------- ----------
90-APP-11          Apple              Tart                 3.25

SQL>
SQL>
SQL> REM 4. Show the number of receipts that contain the product whose price is more than the
average price of its food type.
SQL>
SQL> select count(distinct(rec_no)) as no_of_receipts from item_list where item in (select prod_id
from products p where price> any (select avg(price) from products group by food having p.food =
food));

NO_OF_RECEIPTS
--------------
           137


SQL>
SQL>
SQL>
SQL> REM **II**_____Write the following using JOIN: (Use sub-query if required)_____**
SQL>
SQL>
SQL> REM 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.
SQL>
SQL> select c.cust_no, c.fname, c.lname, r.rec_no, r.rdate from receipts r join customers c on
(c.cust_no = r.cid) where r.rdate = last_day(r.rdate);

   CUST_NO FNAME              LNAME                REC_NO RDATE
---------- -------------------- -------------------- ---------- ---------
         1 JULIET             LOGAN                 85858 31-OCT-07
         3 TRAVIS             ESPOSITA              39829 31-OCT-07
        11 MIGDALIA           STADICK               60270 31-OCT-07
        12 MELLIE             MCMAHAN               70796 31-OCT-07
        19 NATACHA            STENZ                 36343 31-OCT-07
        20 STEPHEN            ZEME                  49845 31-OCT-07

6 rows selected.

SQL>
SQL>
SQL> REM 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.
SQL>
SQL> select rec_no, sum(price) from item_list
  2  join receipts using (rec_no)
  3  join products on (prod_id = item)
```

```
  4  where rec_no in
  5          (select rec_no from item_list join products on (prod_id = item)
  6          where food = 'Twist' group by rec_no)
  7  group by rec_no having sum(price)>25 and count(*)=5;

    REC_NO SUM(PRICE)
---------- ----------
     83085      48.25
     64477      25.35
     17729      25.55

SQL>
SQL>
SQL> REM 7. Display the details (customer details, receipt number, item) for the product that was
purchased by the least number of customers.
SQL>
SQL> select i.item, rec_no, p.flavour, p.food, c.cust_no, c.fname, c.lname
  2  from item_list i
  3  join receipts r using (rec_no)
  4  join customers c on (c.cust_no = r.cid)
  5  join products p on (p.prod_id = i.item)
  6  where i.item in (
  7          select item from item_list group by item having count(item) in (
  8                  select min(count(item)) from item_list group by item
  9                  )
 10          );

ITEM                 REC_NO FLAVOUR             FOOD
-------------------- ---------- -------------------- --------------------
   CUST_NO FNAME               LNAME
---------- -------------------- --------------------
50-CH                 73716 Chocolate          Croissant
        18 ALMETA             DOMKOWSKI

50-CH                 95962 Chocolate          Croissant
         8 RUPERT             HELING

50-CH                 99994 Chocolate          Croissant
         6 JOSETTE            SLINGLAND


ITEM                 REC_NO FLAVOUR             FOOD
-------------------- ---------- -------------------- --------------------
   CUST_NO FNAME               LNAME
---------- -------------------- --------------------
50-CH                 82056 Chocolate          Croissant
        18 ALMETA             DOMKOWSKI

50-CH                 77032 Chocolate          Croissant
        14 RAYFORD            SOPKO
```

```
50-CH                 49845 Chocolate          Croissant
       20 STEPHEN            ZEME


6 rows selected.

SQL>
SQL>
SQL> REM 8. Display the customer details along with the receipt number who ordered all the
flavors of Meringue in the same receipt.
SQL>
SQL> select cust_no, fname, lname, rec_no from customers
  2  join receipts on (cust_no = cid)
  3  where rec_no in (
  4          select rec_no from item_list join products p on (prod_id = item)
  5          where flavour in (select flavour from products where food='Meringue') and
food='Meringue'
  6          group by rec_no having count(distinct(flavour))=(select count(*) from products where
food='Meringue')
  7          );

   CUST_NO FNAME               LNAME                  REC_NO
---------- -------------------- -------------------- ----------
        8 RUPERT               HELING                  61797

SQL>
SQL>
SQL>
SQL> REM **III**_____Write the following using Set Operations: _____**
SQL>
SQL> REM 9. Display the product details of both Pie and BEAR CLAW.
SQL> REM UNION
SQL>
SQL> (select * from products where food='Pie') union (select * from products where food='Bear
Claw');

PROD_ID              FLAVOUR              FOOD                 PRICE
-------------------- -------------------- -------------------- ----------
51-BC                Almond               Bear Claw                1.95
90-APIE-10           Apple                Pie                      5.25

SQL>
SQL>
SQL> REM 10. Display the customers details who have not placed any orders.
SQL> REM DIFF OF SETS
SQL>
SQL> select * from customers where cust_no in (
  2          (select cust_no from customers) minus (select cid from receipts)
  3          );
```

```
  CUST_NO LNAME              FNAME
---------- -------------------- --------------------
        21 JOHN                 DAVID


SQL>
SQL>
SQL> REM 11. Display the food that has the same flavor as that of the common flavor between the
Meringue and Tart.
SQL> REM INTERSECTION
SQL>
SQL> select food from products where flavour in (
  2          (select flavour from products where food='Meringue')
  3          intersect
  4          (select flavour from products where food='Tart')
  5          );

FOOD
--------------------
Cake
Eclair
Tart
Meringue
Croissant


SQL> REM ***************** END OF FILE ********************
```

	SSN COLLEGE OF ENGINEERING	***Faculty:***
	Department of Computer Science & Engineering	***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:

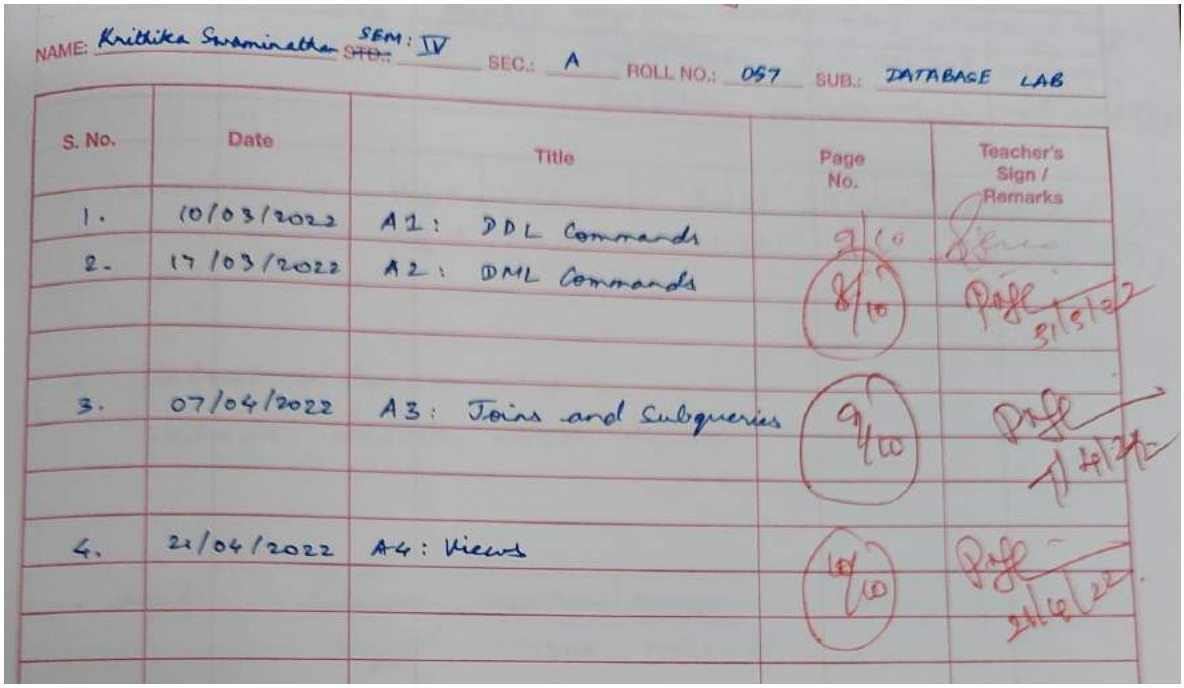
i) updatable /updatable join view – views whose rows can be modified
ii) insertable-into view – views into which only new rows can be inserted
iii) To view the meta-data of views:
USER_VIEWS, USER_UPDATABLE_COLUMNS

What you have to submit:

1. Schema Diagram with constraints
2. Demo script file

Assignment 4 – Views

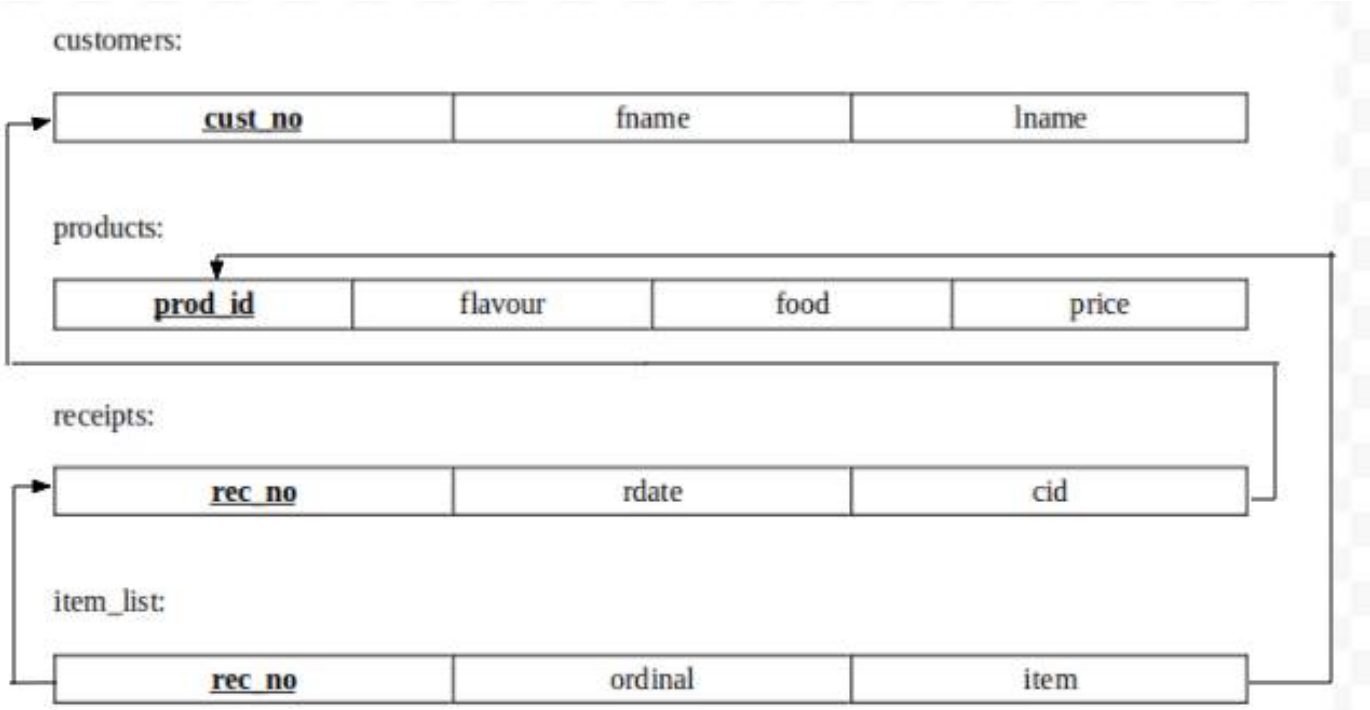
Validation:



NAME: Krithika Swaminathan SEM: IV SEC.: A ROLL NO.: 057 SUB.: DATABASE LAB

S. No.	Date	Title	Page No.	Teacher's Sign / Remarks
1.	10/03/2022	A1: DDL Commands	9/10	
2.	17/03/2022	A2: DML Commands	8/10	31/5/22
3.	07/04/2022	A3: Joins and Subqueries	9/10	14/4/22
4.	21/04/2022	A4: Views	10/10	21/4/22

Schema diagram:



Inferences:

View 1: Blue_Flavor

Operation	*Operation on View*	*Reflection in Base Table*
INSERT	Yes	Yes
UPDATE	prod_id, food, price	prod_id, food, price
DELETE	Yes	Yes

Remark: Since *flavour* is absent from the view and is also the selection criterion for creating the view, insertions cannot be seen in the view.
The view is updatable.

View 2: Cheap_Food

Operation	*Operation on View*	*Reflection in Base Table*
INSERT	Yes	Yes
UPDATE	prod_id, flavour, price	prod_id, flavour, price
DELETE	Yes	Yes

Remark: The price is checked with condition price < 1 before any record is inserted into the view. The view is updated for valid inputs.
The view is updatable.

View 3: Hot_Food

Operation	*Operation on View*	*Reflection in Base Table*
INSERT	No	No
UPDATE	NIL	NIL
DELETE	No	No

Remark:

A virtual column does not allow insertion.
The presence of aggregate functions indicate that a view cannot be updated.
Data manipulation is illegal for virtual columns.

The view is not updatable.

View 4: Pie_Food

Operation	*Operation on View*	*Reflection in Base Table*
INSERT	No	No
UPDATE	rec_no, ordinal	rec_no, ordinal
DELETE	Yes	No

Remark:

Only two attributes – *rec_no* and *ordinal* – are from *item_list*, which is a key-preserved table.
All the other attributes are from non-key-preserved tables and hence, they are not updatable.

The view is not updatable.

View 5: Cheap_View

Operation	*Operation on View*	*Reflection in Table*
INSERT	No	No
UPDATE	prod_id, flavour, food	prod_id, flavour, food
DELETE	Yes	Yes

Remark: Since the check option is present in the view Cheap_Food (from which Cheap_View has been derived), insertion is not allowed.
The view is not updatable.

Data file:

```
SQL> @C:/Krithika/DBL/a4data.sql;
SQL> REM Population of Bakery Database

SQL> drop table item_list;
Table dropped.

SQL> drop table receipts;
Table dropped.

SQL> drop table products;
Table dropped.

SQL> drop table customers;
Table dropped.

SQL>
SQL> create table customers(
  2         cust_no number(2) constraint c_pk primary key,
  3         lname varchar2(20),
  4         fname varchar2(20)
  5         );

Table created.

SQL>
SQL> insert into customers values(1, 'LOGAN', 'JULIET');
1 row created.
.
.
.
SQL> insert into customers values(21, 'JOHN', 'DAVID');
1 row created.

SQL> create table products(
  2         prod_id varchar2(20) constraint prod_pk primary key,
  3         flavour varchar2(20),
  4         food varchar2(20),
```

```
  5         price number
  6         );

Table created.

SQL>
SQL> insert into products values('20-BC-C-10','Chocolate','Cake',8.95);
1 row created.
.
.
.
SQL> insert into products values('51-BLU','Blueberry','Danish',1.15);
1 row created.

SQL> create table receipts(
  2         rec_no number(5) constraint rec_pk primary key,
  3         rdate date,
  4         cid number(2) constraint rec_fk references customers(cust_no)
  5         );

Table created.

SQL>
SQL> INSERT INTO Receipts values(18129, '28-Oct-2007', 15);
1 row created.
.
.
.
SQL> INSERT INTO Receipts values(34378, '23-Oct-2007', 6);
1 row created.

SQL> create table item_list(
  2         rec_no number(5) constraint it_fk1 references receipts(rec_no),
  3         ordinal number(2),
  4         item varchar2(20) constraint it_fk2 references products(prod_id),
  5         constraint item_pk primary key(rec_no,ordinal)
  6         );

Table created.

SQL>
SQL> insert into item_list values(18129, 1,  '70-TU');
1 row created.
.
.
.
SQL> insert into item_list values(34378, 2,  '45-VA');
1 row created.

SQL>
```

```
SQL> REM *** End of database population ***

SQL>
SQL> REM ******************** END OF DATA FILE ************************
```

Script file:

```
SQL> @z:/a4views.sql;
SQL> REM Assignment 4
SQL>
SQL> REM ---------------------------------------------------------------------------
> REM *** ASSIGNMENT QUESTIONS ***
SQL> REM ---------------------------------------------------------------------------
> REM Consider the schema used in Assignment 3.
SQL>
SQL>
SQL> REM **_____Create view(s) based on table(s) or view(s) and observe their behaviour while
performing update operations on them_____**
SQL>


SQL> REM 1. Create a view named Blue_Flavor, which displays the product details (product id,
food, price) of Blueberry flavour.
SQL>
SQL> REM Creating view:
SQL> create or replace view Blue_Flavor as
  2  select prod_id, food, price from products
  3  where flavour='Blueberry';

View created.

SQL>
SQL> REM Displaying view:
SQL> select * from Blue_Flavor;

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

SQL>
SQL> REM Savepoint:
SQL> savepoint question1;

Savepoint created.

SQL>
SQL> REM Checking if updatable:
SQL> select COLUMN_NAME, UPDATABLE, INSERTABLE, DELETABLE
```

```
  2  from USER_UPDATABLE_COLUMNS
  3  where TABLE_NAME='BLUE_FLAVOR';

COLUMN_NAME                    UPD INS DEL
------------------------------ --- --- ---
PROD_ID                        YES YES YES
FOOD                           YES YES YES
PRICE                          YES YES YES

SQL>
SQL> REM Insertion:
SQL> insert into Blue_Flavor values ('61-GC','Cake',9.20);

1 row created.

SQL>
SQL> REM Insertion verification:
SQL> select * from Blue_Flavor;

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

SQL> REM Flavour is not specified as Blueberry and hence, the row is not present in the view.
SQL>
SQL> select * from products where prod_id='61-GC';

PROD_ID              FLAVOUR              FOOD                    PRICE
-------------------- -------------------- -------------------- ----------
61-GC                                     Cake                      9.2

SQL> REM In the table 'products', the values are inserted leaving 'flavour' empty.
SQL>
SQL> REM Updation:
SQL> update Blue_Flavor set prod_id='61-BLU' where prod_id='61-GC';

0 rows updated.

SQL> REM The row with prod_id='61-BLU' is not in the view and hence, it cannot be updated.
SQL>
SQL> update Blue_Flavor set food='Cake' where prod_id='51-BLU';

1 row updated.

SQL> update Blue_Flavor set price=3.75 where prod_id='90-BLU-11';

1 row updated.

SQL>
```

```
SQL> REM Updation verification:
SQL> select * from Blue_Flavor where prod_id='61-BLU';

no rows selected

SQL> select * from products where prod_id='61-BLU';

no rows selected

SQL>
SQL> select * from Blue_Flavor where prod_id='51-BLU';

PROD_ID              FOOD                 PRICE
-------------------- -------------------- ----------
51-BLU               Cake                    1.15

SQL> select * from products where prod_id='51-BLU';

PROD_ID              FLAVOUR              FOOD                 PRICE
-------------------- -------------------- -------------------- ----------
51-BLU               Blueberry            Cake                    1.15

SQL>
SQL> select * from Blue_Flavor where prod_id='90-BLU-11';

PROD_ID              FOOD                 PRICE
-------------------- -------------------- ----------
90-BLU-11            Tart                    3.75

SQL> select * from products where prod_id='51-BLU';

PROD_ID              FLAVOUR              FOOD                 PRICE
-------------------- -------------------- -------------------- ----------
51-BLU               Blueberry            Cake                    1.15

SQL>
SQL> REM Deletion:
SQL> delete from Blue_Flavor where prod_id='61-BLU';

0 rows deleted.

SQL>
SQL> REM Deletion verification:
SQL> select * from Blue_Flavor;

PROD_ID              FOOD                 PRICE
-------------------- -------------------- ----------
90-BLU-11            Tart                    3.75
51-BLU               Cake                    1.15
```

```
SQL> select * from products where prod_id='61-BLU';

no rows selected

SQL>
SQL> REM Insertion in table:
SQL> insert into products values('88-SS-10','Blueberry','Cone', 2.95);

1 row created.

SQL>
SQL> REM Insertion verification:
SQL> select * from Blue_Flavor;

PROD_ID              FOOD                    PRICE
-------------------- -------------------- ----------
90-BLU-11            Tart                      3.75
51-BLU               Cake                      1.15
88-SS-10             Cone                      2.95

SQL> select * from products where prod_id='88-SS-10';

PROD_ID              FLAVOUR              FOOD                    PRICE
-------------------- -------------------- -------------------- ----------
88-SS-10             Blueberry            Cone                      2.95

SQL>
SQL> REM Updation in table:
SQL> update products set prod_id='61-BLU' where prod_id='88-SS-10';

1 row updated.

SQL> update products set food='Cake' where prod_id='61-BLU';

1 row updated.

SQL> update products set price=4.25 where prod_id='61-BLU';

1 row updated.

SQL>
SQL> REM Updation verification:
SQL> select * from Blue_Flavor where prod_id='61-BLU';

PROD_ID              FOOD                    PRICE
-------------------- -------------------- ----------
61-BLU               Cake                      4.25

SQL> select * from products where prod_id='61-BLU';
```

```
PROD_ID            FLAVOUR            FOOD                 PRICE
-------------------- -------------------- -------------------- ----------
61-BLU             Blueberry          Cake                   4.25

SQL>
SQL> REM Deletion in table:
SQL> delete from Blue_Flavor where prod_id='61-BLU';

1 row deleted.

SQL>
SQL> REM Deletion verification:
SQL> select * from Blue_Flavor;

PROD_ID            FOOD                 PRICE
-------------------- -------------------- ----------
90-BLU-11          Tart                   3.75
51-BLU             Cake                   1.15

SQL> select * from products where prod_id='61-BLU';

no rows selected

SQL>
SQL> rollback to question1;

Rollback complete.

SQL>
SQL> REM *INFERENCE:*
SQL> REM Insertion: Insertion into the view is reflected in the parent table but not in the view as
flavour is the selection criterion but not an attribute of the view. Insertion into the parent table is
reflected in both.
SQL> REM Updation: Key preserved. All the attributes in the view are updatable and updation in
the main table is reflected in both.
SQL> REM Deletion: Deletions in both the view and the parent table are reflected in both.
SQL>


SQL>
SQL> REM 2. Create a view named Cheap_Food, which displays the details (product id, flavour,
food, price) of products with price lesser than $1. Ensure that, the price of these food(s) should
never rise above $1 through view.
SQL>
SQL> REM Creating view:
SQL> create or replace view Cheap_Food as
  2  select prod_id, flavour, food, price from products
  3  where price < 1
  4  with check option;
```

```
View created.

SQL>
SQL> REM Displaying view:
SQL> select * from Cheap_Food;

PROD_ID              FLAVOUR              FOOD                    PRICE
-------------------- -------------------- -------------------- ----------
70-LEM               Lemon                Cookie                     .79
70-W                 Walnut               Cookie                     .79

SQL>
SQL> REM Savepoint:
SQL> savepoint question2;

Savepoint created.

SQL>
SQL> REM Checking if updatable:
SQL> select COLUMN_NAME, UPDATABLE, INSERTABLE, DELETABLE
  2  from USER_UPDATABLE_COLUMNS
  3  where TABLE_NAME='CHEAP_FOOD';

COLUMN_NAME                    UPD INS DEL
------------------------------ --- --- ---
PROD_ID                        YES YES YES
FLAVOUR                        YES YES YES
FOOD                           YES YES YES
PRICE                          YES YES YES

SQL>
SQL> REM Insertion:
SQL> REM invalid
SQL> insert into Cheap_Food values ('89-NE','Blackberry','Tart',7.50);
insert into Cheap_Food values ('89-NE','Blackberry','Tart',7.50)
            *
ERROR at line 1:
ORA-01402: view WITH CHECK OPTION where-clause violation


SQL> REM valid
SQL> insert into Cheap_Food values ('89-NE','Blackberry','Tart',0.50);

1 row created.

SQL>
SQL> REM Insertion verification:
SQL> select * from Cheap_Food;

PROD_ID              FLAVOUR              FOOD                    PRICE
```

```
-------------------- -------------------- -------------------- ----------
70-LEM               Lemon                Cookie                     .79
70-W                 Walnut               Cookie                    .79
89-NE                Blackberry           Tart                       .5

SQL> select * from products where prod_id='89-NE';

PROD_ID              FLAVOUR              FOOD                    PRICE
-------------------- -------------------- -------------------- ----------
89-NE                Blackberry           Tart                       .5

SQL>
SQL> REM Updation:
SQL> update Cheap_Food set prod_id='90-NEW' where prod_id='89-NE';

1 row updated.

SQL> update Cheap_Food set flavour='Tomato' where prod_id='90-NEW';

1 row updated.

SQL> update Cheap_Food set food='Chips' where prod_id='90-NEW';

1 row updated.

SQL> update Cheap_Food set price=0.75 where prod_id='90-NEW';

1 row updated.

SQL>
SQL> REM Updation verification:
SQL> select * from Cheap_Food where prod_id='90-NEW';

PROD_ID              FLAVOUR              FOOD                    PRICE
-------------------- -------------------- -------------------- ----------
90-NEW               Tomato               Chips                     .75

SQL> select * from products where prod_id='90-NEW';

PROD_ID              FLAVOUR              FOOD                    PRICE
-------------------- -------------------- -------------------- ----------
90-NEW               Tomato               Chips                     .75

SQL>
SQL> REM Deletion:
SQL> delete from Cheap_Food where prod_id='90-NEW';

1 row deleted.

SQL>
```

```
SQL> REM Deletion verification:
SQL> select * from Cheap_Food;

PROD_ID              FLAVOUR              FOOD                    PRICE
-------------------- -------------------- -------------------- ----------
70-LEM               Lemon                Cookie                    .79
70-W                 Walnut               Cookie                   .79

SQL> select * from products where prod_id='90-NEW';

no rows selected

SQL>
SQL> REM Insertion in table:
SQL> insert into products values('88-SS-10','Blueberry','Cone', 0.75);

1 row created.

SQL>
SQL> REM Insertion verification:
SQL> select * from Cheap_Food;

PROD_ID              FLAVOUR              FOOD                    PRICE
-------------------- -------------------- -------------------- ----------
70-LEM               Lemon                Cookie                    .79
70-W                 Walnut               Cookie                   .79
88-SS-10             Blueberry            Cone                      .75

SQL> select * from products where prod_id='88-SS-10';

PROD_ID              FLAVOUR              FOOD                    PRICE
-------------------- -------------------- -------------------- ----------
88-SS-10             Blueberry            Cone                      .75

SQL>
SQL> REM Updation in table:
SQL> update products set prod_id='90-NEW-2' where prod_id='88-SS-10';

1 row updated.

SQL> update products set flavour='Tomato' where prod_id='90-NEW-2';

1 row updated.

SQL> update products set food='Chips' where prod_id='90-NEW-2';

1 row updated.

SQL> update products set price=0.75 where prod_id='90-NEW-2';
```

```
1 row updated.

SQL>
SQL> REM Updation verification:
SQL> select * from Cheap_Food where prod_id='90-NEW-2';

PROD_ID              FLAVOUR              FOOD                 PRICE
-------------------- -------------------- -------------------- ----------
90-NEW-2             Tomato               Chips                     .75

SQL> select * from products where prod_id='90-NEW-2';

PROD_ID              FLAVOUR              FOOD                 PRICE
-------------------- -------------------- -------------------- ----------
90-NEW-2             Tomato               Chips                     .75

SQL>
SQL> REM Deletion in table:
SQL> delete from products where prod_id='90-NEW-2';

1 row deleted.

SQL>
SQL> REM Deletion verification:
SQL> select * from Cheap_Food;

PROD_ID              FLAVOUR              FOOD                 PRICE
-------------------- -------------------- -------------------- ----------
70-LEM               Lemon                Cookie                    .79
70-W                 Walnut               Cookie                    .79

SQL> select * from products where prod_id='90-NEW-2';
no rows selected

SQL>
SQL> rollback to question2;

Rollback complete.

SQL>
SQL> REM *INFERENCE:*
SQL> REM Insertion: Insertions into both the view and the parent table are reflected in both.
Products with price > 1 are not allowed due to the 'with check' option in the view.
SQL> REM Updation: Key preserved. All the attributes in the view are updatable and updations in
the main table are reflected in both.
SQL> REM Deletion: Deletions in both the view and the parent table are reflected in both.
SQL>


SQL>
```

```
SQL> REM 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.
SQL>
SQL> REM Creating view:
SQL> create or replace view Hot_Food as
  2  select item, count(*) as quantity from item_list
  3  group by rec_no, item having count(*)>1;

View created.

SQL>
SQL> REM Displaying view:
SQL> select * from Hot_Food;

ITEM                 QUANTITY
-------------------- ----------
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

ITEM                 QUANTITY
-------------------- ----------
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> REM Savepoint:
SQL> savepoint question3;

Savepoint created.

SQL>
SQL> REM Checking if updatable:
```

```
SQL> select COLUMN_NAME, UPDATABLE, INSERTABLE, DELETABLE
  2  from USER_UPDATABLE_COLUMNS
  3  where TABLE_NAME='HOT_FOOD';

COLUMN_NAME                    UPD INS DEL
------------------------------ --- --- ---
ITEM                           NO  NO  NO
QUANTITY                       NO  NO  NO

SQL>
SQL> REM Insertion:
SQL> insert into Hot_Food values ('999-ZA',2);
insert into Hot_Food values ('999-ZA',2)
*
ERROR at line 1:
ORA-01733: virtual column not allowed here


SQL>
SQL> REM Insertion verification:
SQL> select * from Hot_Food;

ITEM                 QUANTITY
-------------------- ----------
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

ITEM                 QUANTITY
-------------------- ----------
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> select * from item_list where item='999-ZA';

no rows selected

SQL>
SQL> REM Updation:
SQL> update Hot_Food set item='99-NEW' where item='999-ZA';
update Hot_Food set item='99-NEW' where item='999-ZA'
       *
ERROR at line 1:
ORA-01732: data manipulation operation not legal on this view


SQL> update Hot_Food set quantity=4 where item='99-NEW';
update Hot_Food set quantity=4 where item='99-NEW'
       *
ERROR at line 1:
ORA-01732: data manipulation operation not legal on this view


SQL>
SQL> REM Updation verification:
SQL> select * from Hot_Food where item='99-NEW';

no rows selected

SQL> select * from item_list where item='99-NEW';

no rows selected

SQL>
SQL> REM Deletion:
SQL> delete from Hot_Food where item='99-NEW';
delete from Hot_Food where item='99-NEW'
           *
ERROR at line 1:
ORA-01732: data manipulation operation not legal on this view


SQL>
SQL> REM Deletion verification:
SQL> select * from Hot_Food;

ITEM                 QUANTITY
-------------------- ----------
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

ITEM                 QUANTITY
-------------------- ----------
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> select * from item_list where item='99-NEW';

no rows selected

SQL>
SQL> REM Insertion in table:
SQL> insert into item_list values(11923,20,'70-MAR');

1 row created.

SQL>
SQL> REM Insertion verification:
SQL> select * from Hot_Food;

ITEM                 QUANTITY
-------------------- ----------
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

ITEM                 QUANTITY
```

```
-------------------- ----------
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> select * from item_list where item='70-MAR';

   REC_NO    ORDINAL ITEM
---------- ---------- --------------------
    59716          2 70-MAR
    66227          3 70-MAR
    38157          1 70-MAR
    31874          1 70-MAR
    31874          2 70-MAR
    72207          1 70-MAR
    77032          4 70-MAR
    15286          1 70-MAR
    17685          4 70-MAR
    70162          4 70-MAR
    74741          3 70-MAR

   REC_NO    ORDINAL ITEM
---------- ---------- --------------------
    61948          1 70-MAR
    95514          3 70-MAR
    97097          1 70-MAR
    16532          2 70-MAR
    11923         20 70-MAR

16 rows selected.

SQL>
SQL> REM Deletion in table:
SQL> delete from Hot_Food where item='70-MAR';
delete from Hot_Food where item='70-MAR'
           *
ERROR at line 1:
ORA-01732: data manipulation operation not legal on this view


SQL>
SQL> REM Deletion verification:
```

```
SQL> select * from Hot_Food;

ITEM                 QUANTITY
-------------------- ----------
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

ITEM                 QUANTITY
-------------------- ----------
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> select * from item_list where item='70-MAR';

   REC_NO    ORDINAL ITEM
---------- ---------- --------------------
    59716          2 70-MAR
    66227          3 70-MAR
    38157          1 70-MAR
    31874          1 70-MAR
    31874          2 70-MAR
    72207          1 70-MAR
    77032          4 70-MAR
    15286          1 70-MAR
    17685          4 70-MAR
    70162          4 70-MAR
    74741          3 70-MAR

   REC_NO    ORDINAL ITEM
---------- ---------- --------------------
    61948          1 70-MAR
    95514          3 70-MAR
```

```
    97097         1 70-MAR
    16532         2 70-MAR
    11923        20 70-MAR

16 rows selected.

SQL>
SQL> rollback to question3;

Rollback complete.

SQL>
SQL> REM *INFERENCE:*
SQL> REM Insertion: Insertion in view is not allowed due to the presence of a virtual column
(group by column). Insertion in the parent table is reflected in both.
SQL> REM Updation: None of the attributes in the view are updatable due to the virtual column.
Updation in the parent table is reflected in both.
SQL> REM Deletion: Deletion in view is not allowed as data manipulation is not allowed with
virtual columns. Deletion in the parent table is reflected in both.
SQL>


SQL>
SQL> REM 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.
SQL>
SQL> REM Creating view:
SQL> create or replace view Pie_Food as
  2  select lname, flavour, rec_no, rdate, ordinal
  3  from customers
  4  join receipts on (cust_no = cid)
  5  join item_list using (rec_no)
  6  join products on (item = prod_id)
  7  where food='Pie';

View created.

SQL>
SQL> REM Displaying view:
SQL> select * from Pie_Food;

LNAME              FLAVOUR              REC_NO RDATE        ORDINAL
-------------------- -------------------- ---------- --------- ----------
LOGAN              Apple                39685 28-OCT-07         4
LOGAN              Apple                66227 10-OCT-07         2
ESPOSITA           Apple                48647 09-OCT-07         2
SLINGLAND          Apple                87454 21-OCT-07         1
SLINGLAND          Apple                47353 12-OCT-07         2
HELING             Apple                53376 30-OCT-07         3
HAFFERKAMP         Apple                50660 18-OCT-07         2
```

```
ARNN                Apple                  11548 21-OCT-07         2
SOPKO               Apple                  29226 26-OCT-07         2
SOPKO               Apple                  51991 17-OCT-07         1
CRUZEN              Apple                  39109 02-OCT-07         1

LNAME               FLAVOUR                REC_NO RDATE        ORDINAL
-------------------- -------------------- ---------- --------- ----------
CRUZEN              Apple                  44798 04-OCT-07         3
MESDAQ              Apple                  98806 15-OCT-07         3

13 rows selected.

SQL>
SQL> REM Savepoint:
SQL> savepoint question4;

Savepoint created.

SQL>
SQL> REM Checking if updatable:
SQL> select COLUMN_NAME, UPDATABLE, INSERTABLE, DELETABLE
  2  from USER_UPDATABLE_COLUMNS
  3  where TABLE_NAME='PIE_FOOD';

COLUMN_NAME                    UPD INS DEL
------------------------------ --- --- ---
LNAME                          NO  NO  NO
FLAVOUR                        NO  NO  NO
REC_NO                         YES YES YES
RDATE                          NO  NO  NO
ORDINAL                        YES YES YES

SQL>
SQL> REM Insertion:
SQL> insert into Pie_Food values ('HOLMES','Lemon',83939,'21-OCT-2007',1);
insert into Pie_Food values ('HOLMES','Lemon',83939,'21-OCT-2007',1)
*
ERROR at line 1:
ORA-01779: cannot modify a column which maps to a non key-preserved table


SQL>
SQL> REM Insertion verification:
SQL> select * from Pie_Food;

LNAME               FLAVOUR                REC_NO RDATE        ORDINAL
-------------------- -------------------- ---------- --------- ----------
LOGAN               Apple                  39685 28-OCT-07         4
LOGAN               Apple                  66227 10-OCT-07         2
ESPOSITA            Apple                  48647 09-OCT-07         2
```

```
SLINGLAND          Apple                  87454 21-OCT-07         1
SLINGLAND          Apple                  47353 12-OCT-07         2
HELING             Apple                  53376 30-OCT-07         3
HAFFERKAMP         Apple                  50660 18-OCT-07         2
ARNN               Apple                  11548 21-OCT-07         2
SOPKO              Apple                  29226 26-OCT-07         2
SOPKO              Apple                  51991 17-OCT-07         1
CRUZEN             Apple                  39109 02-OCT-07         1

LNAME              FLAVOUR                REC_NO RDATE        ORDINAL
-------------------- -------------------- ---------- --------- ----------
CRUZEN             Apple                  44798 04-OCT-07         3
MESDAQ             Apple                  98806 15-OCT-07         3

13 rows selected.

SQL>
SQL> REM Updation:
SQL> update Pie_Food set lname='NEWNAME' where lname='LOGAN';
update Pie_Food set lname='NEWNAME' where lname='LOGAN'
                *
ERROR at line 1:
ORA-01779: cannot modify a column which maps to a non key-preserved table


SQL> update Pie_Food set flavor='NEWFLAVOUR' where lname='LOGAN';
update Pie_Food set flavor='NEWFLAVOUR' where lname='LOGAN'
                *
ERROR at line 1:
ORA-00904: "FLAVOR": invalid identifier


SQL> update Pie_Food set rec_no=56789 where lname='LOGAN';
update Pie_Food set rec_no=56789 where lname='LOGAN'
*
ERROR at line 1:
ORA-02291: integrity constraint (1057.IT_FK1) violated - parent key not found


SQL> update Pie_Food set rdate='20-MAR-2007' where lname='LOGAN';
update Pie_Food set rdate='20-MAR-2007' where lname='LOGAN'
                *
ERROR at line 1:
ORA-01779: cannot modify a column which maps to a non key-preserved table


SQL> update Pie_Food set ordinal=10 where lname='LOGAN';

2 rows updated.
```

```
SQL>
SQL> REM Updation verification:
SQL> select * from Pie_Food;

LNAME                FLAVOUR                REC_NO RDATE        ORDINAL
-------------------- -------------------- ---------- --------- ----------
LOGAN                Apple                   39685 28-OCT-07         10
LOGAN                Apple                   66227 10-OCT-07         10
ESPOSITA             Apple                   48647 09-OCT-07          2
SLINGLAND            Apple                   87454 21-OCT-07          1
SLINGLAND            Apple                   47353 12-OCT-07          2
HELING               Apple                   53376 30-OCT-07          3
HAFFERKAMP           Apple                   50660 18-OCT-07          2
ARNN                 Apple                   11548 21-OCT-07          2
SOPKO                Apple                   29226 26-OCT-07          2
SOPKO                Apple                   51991 17-OCT-07          1
CRUZEN               Apple                   39109 02-OCT-07          1

LNAME                FLAVOUR                REC_NO RDATE        ORDINAL
-------------------- -------------------- ---------- --------- ----------
CRUZEN               Apple                   44798 04-OCT-07          3
MESDAQ               Apple                   98806 15-OCT-07          3

13 rows selected.

SQL>
SQL> REM Deletion:
SQL> delete from Pie_Food where rec_no=56789;

0 rows deleted.

SQL>
SQL> REM Deletion verification:
SQL> select * from Pie_Food;

LNAME                FLAVOUR                REC_NO RDATE        ORDINAL
-------------------- -------------------- ---------- --------- ----------
LOGAN                Apple                   39685 28-OCT-07         10
LOGAN                Apple                   66227 10-OCT-07         10
ESPOSITA             Apple                   48647 09-OCT-07          2
SLINGLAND            Apple                   87454 21-OCT-07          1
SLINGLAND            Apple                   47353 12-OCT-07          2
HELING               Apple                   53376 30-OCT-07          3
HAFFERKAMP           Apple                   50660 18-OCT-07          2
ARNN                 Apple                   11548 21-OCT-07          2
SOPKO                Apple                   29226 26-OCT-07          2
SOPKO                Apple                   51991 17-OCT-07          1
CRUZEN               Apple                   39109 02-OCT-07          1

LNAME                FLAVOUR                REC_NO RDATE        ORDINAL
```

```
-------------------- -------------------- ---------- --------- ----------
CRUZEN               Apple                 44798 04-OCT-07          3
MESDAQ               Apple                 98806 15-OCT-07          3

13 rows selected.

SQL> select * from receipts where rec_no=56789;

no rows selected

SQL>
SQL> REM Insertion in table:
SQL> insert into item_list values(11923,20,'70-MAR');

1 row created.

SQL>
SQL> REM Insertion verification:
SQL> select * from Pie_Food;

LNAME                FLAVOUR                REC_NO RDATE       ORDINAL
-------------------- -------------------- ---------- --------- ----------
LOGAN                Apple                 39685 28-OCT-07         10
LOGAN                Apple                 66227 10-OCT-07         10
ESPOSITA             Apple                 48647 09-OCT-07          2
SLINGLAND            Apple                 87454 21-OCT-07          1
SLINGLAND            Apple                 47353 12-OCT-07          2
HELING               Apple                 53376 30-OCT-07          3
HAFFERKAMP           Apple                 50660 18-OCT-07          2
ARNN                 Apple                 11548 21-OCT-07          2
SOPKO                Apple                 29226 26-OCT-07          2
SOPKO                Apple                 51991 17-OCT-07          1
CRUZEN               Apple                 39109 02-OCT-07          1

LNAME                FLAVOUR                REC_NO RDATE       ORDINAL
-------------------- -------------------- ---------- --------- ----------
CRUZEN               Apple                 44798 04-OCT-07          3
MESDAQ               Apple                 98806 15-OCT-07          3

13 rows selected.

SQL> select * from item_list where item='70-MAR';

    REC_NO    ORDINAL ITEM
---------- ---------- --------------------
     59716          2 70-MAR
     66227          3 70-MAR
     38157          1 70-MAR
     31874          1 70-MAR
     31874          2 70-MAR
```

```
    72207         1 70-MAR
    77032         4 70-MAR
    15286         1 70-MAR
    17685         4 70-MAR
    70162         4 70-MAR
    74741         3 70-MAR

   REC_NO    ORDINAL ITEM
---------- ---------- --------------------
    61948         1 70-MAR
    95514         3 70-MAR
    97097         1 70-MAR
    16532         2 70-MAR
    11923        20 70-MAR

16 rows selected.

SQL>
SQL> REM Deletion in table:
SQL> delete from Pie_Food where item='70-MAR';
delete from Pie_Food where item='70-MAR'
                       *
ERROR at line 1:
ORA-00904: "ITEM": invalid identifier


SQL>
SQL> REM Deletion verification:
SQL> select * from Pie_Food;

LNAME                FLAVOUR                REC_NO RDATE        ORDINAL
-------------------- -------------------- ---------- --------- ----------
LOGAN                Apple                   39685 28-OCT-07         10
LOGAN                Apple                   66227 10-OCT-07         10
ESPOSITA             Apple                   48647 09-OCT-07          2
SLINGLAND            Apple                   87454 21-OCT-07          1
SLINGLAND            Apple                   47353 12-OCT-07          2
HELING               Apple                   53376 30-OCT-07          3
HAFFERKAMP           Apple                   50660 18-OCT-07          2
ARNN                 Apple                   11548 21-OCT-07          2
SOPKO                Apple                   29226 26-OCT-07          2
SOPKO                Apple                   51991 17-OCT-07          1
CRUZEN               Apple                   39109 02-OCT-07          1

LNAME                FLAVOUR                REC_NO RDATE        ORDINAL
-------------------- -------------------- ---------- --------- ----------
CRUZEN               Apple                   44798 04-OCT-07          3
MESDAQ               Apple                   98806 15-OCT-07          3

13 rows selected.
```

```
SQL> select * from item_list where item='70-MAR';

   REC_NO    ORDINAL ITEM
---------- ---------- --------------------
    59716          2 70-MAR
    66227          3 70-MAR
    38157          1 70-MAR
    31874          1 70-MAR
    31874          2 70-MAR
    72207          1 70-MAR
    77032          4 70-MAR
    15286          1 70-MAR
    17685          4 70-MAR
    70162          4 70-MAR
    74741          3 70-MAR

   REC_NO    ORDINAL ITEM
---------- ---------- --------------------
    61948          1 70-MAR
    95514          3 70-MAR
    97097          1 70-MAR
    16532          2 70-MAR
    11923         20 70-MAR

16 rows selected.

SQL>
SQL> rollback to question4;

Rollback complete.

SQL>
SQL> REM *INFERENCE:*
SQL> REM Insertion: Insertion in the view is only allowed for the key preserved columns. (Note that rec_no with ordinal is a composite key.) Insertion in the parent table is reflected in both.
SQL> REM Updation: Updation in the view is only allowed in key preserved columns. Updation in the main table is reflected in both.
SQL> REM Deletion: Deletion in the view is updated in the view alone. Deletion in the parent table is not reflected in the view.
SQL>


SQL>
SQL> REM 5. Create a view Cheap_View from Cheap_Food that shows only the product id, flavor and food.
SQL>
SQL> REM Creating view:
SQL> create or replace view Cheap_View as
  2  select prod_id, flavour, food from Cheap_Food;
```

```
View created.

SQL>
SQL> REM Displaying view:
SQL> select * from Cheap_View;

PROD_ID              FLAVOUR              FOOD
-------------------- -------------------- --------------------
70-LEM               Lemon                Cookie
70-W                 Walnut               Cookie

SQL>
SQL> REM Savepoint:
SQL> savepoint question5;

Savepoint created.

SQL>
SQL> REM Checking if updatable:
SQL> select COLUMN_NAME, UPDATABLE, INSERTABLE, DELETABLE
  2  from USER_UPDATABLE_COLUMNS
  3  where TABLE_NAME='CHEAP_VIEW';

COLUMN_NAME                    UPD INS DEL
------------------------------ --- --- ---
PROD_ID                        YES YES YES
FLAVOUR                        YES YES YES
FOOD                           YES YES YES

SQL>
SQL> REM Insertion:
SQL> insert into Cheap_View values ('61-GC','Strawberry','Cake');
insert into Cheap_View values ('61-GC','Strawberry','Cake')
            *
ERROR at line 1:
ORA-01402: view WITH CHECK OPTION where-clause violation


SQL>
SQL> REM Insertion verification:
SQL> select * from Cheap_View;

PROD_ID              FLAVOUR              FOOD
-------------------- -------------------- --------------------
70-LEM               Lemon                Cookie
70-W                 Walnut               Cookie

SQL> select * from products where prod_id='61-GC';
```

```
no rows selected

SQL>
SQL> REM Updation:
SQL> update Cheap_View set prod_id='61-BLU' where prod_id='61-GC';

0 rows updated.

SQL> update Cheap_View set flavour='Banana' where prod_id='61-BLU';

0 rows updated.

SQL> update Cheap_View set food='Cake' where prod_id='61-BLU';

0 rows updated.

SQL>
SQL> REM Updation verification:
SQL> select * from Cheap_View where prod_id='61-BLU';

no rows selected

SQL> select * from products where prod_id='61-BLU';

no rows selected

SQL>
SQL> REM Deletion:
SQL> delete from Cheap_View where prod_id='61-BLU';

0 rows deleted.

SQL>
SQL> REM Deletion verification:
SQL> select * from Cheap_View;

PROD_ID              FLAVOUR              FOOD
-------------------- -------------------- --------------------
70-LEM               Lemon                Cookie
70-W                 Walnut               Cookie

SQL> select * from products where prod_id='61-BLU';

no rows selected

SQL>
SQL> REM Insertion in table:
SQL> insert into products values('88-SS-10','Blueberry','Cone', 2.95);

1 row created.
```

```
SQL>
SQL> REM Insertion verification:
SQL> select * from Cheap_View;

PROD_ID              FLAVOUR              FOOD
-------------------- -------------------- --------------------
70-LEM               Lemon                Cookie
70-W                 Walnut               Cookie

SQL> select * from products where prod_id='88-SS-10';

PROD_ID              FLAVOUR              FOOD                 PRICE
-------------------- -------------------- -------------------- ----------
88-SS-10             Blueberry            Cone                    2.95

SQL>
SQL> REM Updation in table:
SQL> update products set prod_id='61-BLU' where prod_id='88-SS-10';

1 row updated.

SQL> update products set flavour='Chocolate' where prod_id='61-BLU';

1 row updated.

SQL> update products set price=4.25 where prod_id='61-BLU';

1 row updated.

SQL>
SQL> REM Updation verification:
SQL> select * from Cheap_View where prod_id='61-BLU';

no rows selected

SQL> select * from products where prod_id='61-BLU';

PROD_ID              FLAVOUR              FOOD                 PRICE
-------------------- -------------------- -------------------- ----------
61-BLU               Chocolate            Cone                    4.25

SQL>
SQL> REM Deletion in table:
SQL> delete from Cheap_View where prod_id='61-BLU';

0 rows deleted.

SQL>
SQL> REM Deletion verification:
```

```
SQL> select * from Cheap_View;

PROD_ID              FLAVOUR              FOOD
-------------------- -------------------- --------------------
70-LEM               Lemon                Cookie
70-W                 Walnut               Cookie

SQL> select * from products where prod_id='61-BLU';

PROD_ID              FLAVOUR              FOOD                 PRICE
-------------------- -------------------- -------------------- ----------
61-BLU               Chocolate            Cone                   4.25

SQL>
SQL> rollback to question5;

Rollback complete.

SQL>
SQL> REM *INFERENCE:*
SQL> REM Insertion: Insertions into both the view and the parent table are reflected in both.
Products with price > 1 are not allowed due to the 'with check' option in the view.
SQL> REM Updation: Key preserved. All the attributes in the view are updatable and updations in
the main table are reflected in both.
SQL> REM Deletion: Deletions in both the view and the parent table are reflected in both.
SQL>


SQL>
SQL> REM 6. Drop the view Cheap_View.
SQL>
SQL> drop view Cheap_View;

View dropped.

SQL> select * from Cheap_View;
select * from Cheap_View
          *
ERROR at line 1:
ORA-00942: table or view does not exist


SQL>
SQL>
SQL> REM ********************** END OF FILE ******************************
```

	SSN COLLEGE OF ENGINEERING	***Faculty:***
	Department of Computer Science & Engineering	***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:

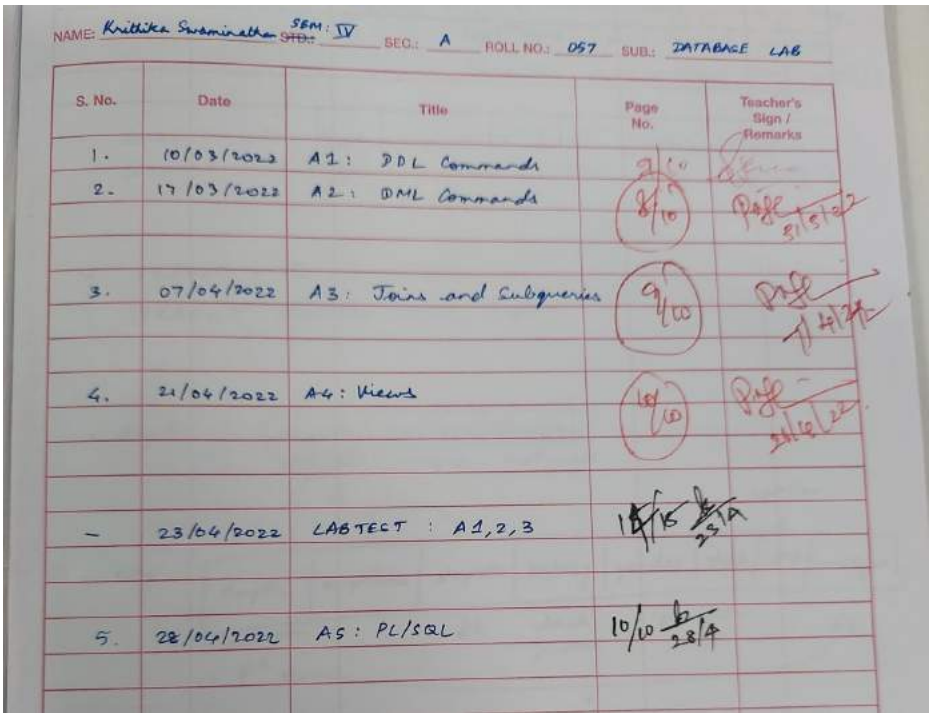
```
SQL> /
Enter value for rid: 51991
old  11:       rid:=&rid;
new  11:       rid:=51991;

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
```

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

1. Schema Diagram with constraints
2. Demo script file

Assignment 5 – PL/SQL: Structures

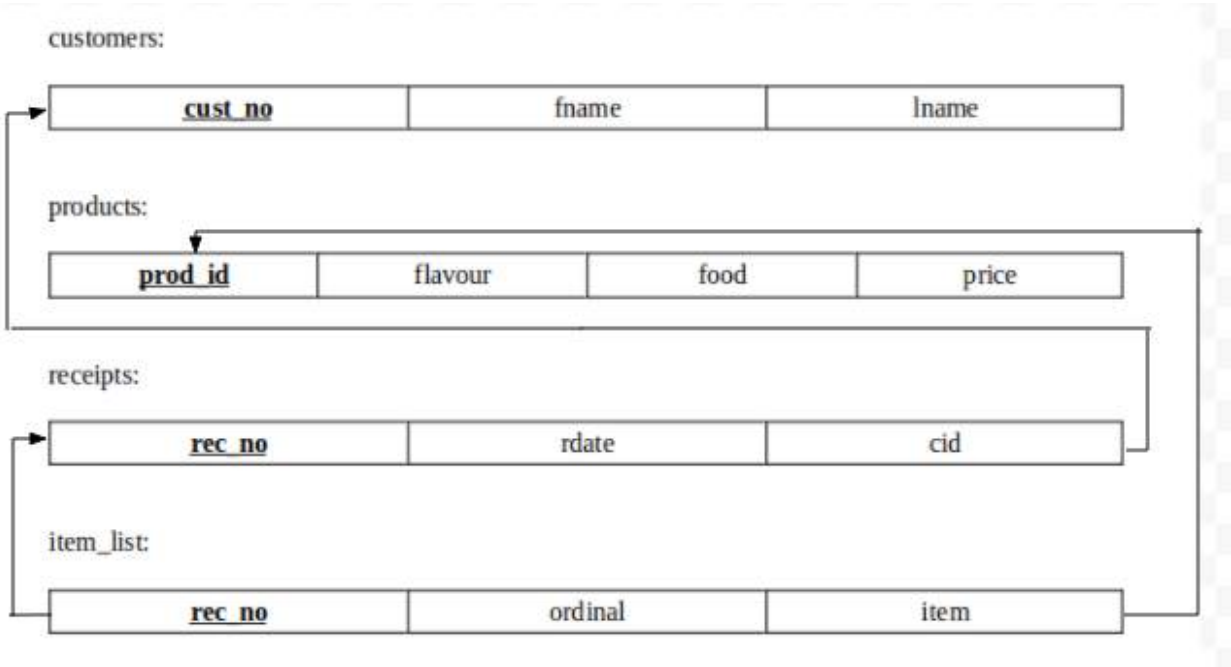
Validation:



NAME: Krithika Swaminathan SEM: IV STD: SEC.: A ROLL NO.: 057 SUB.: DATABASE LAB

S. No.	Date	Title	Page No.	Teacher's Sign / Remarks
1.	10/03/2022	A1: DDL Commands	9/10	
2.	17/03/2022	A2: DML Commands	8/10	31/5/22
3.	07/04/2022	A3: Joins and Subqueries	9/10	7/4/22
4.	21/04/2022	A4: Views	10/10	21/4/22
–	23/04/2022	LABTEST : A1,2,3	15/15 23/4	
5.	28/04/2022	A5: PL/SQL	10/10 28/4	

Schema diagram:



Data file:

```
SQL> @C:/Krithika/DBL/a5data.sql;
SQL> REM Population of Bakery Database

SQL> drop table item_list;
Table dropped.

SQL> drop table receipts;
Table dropped.

SQL> drop table products;
Table dropped.

SQL> drop table customers;
Table dropped.

SQL>
SQL> create table customers(
  2         cust_no number(2) constraint c_pk primary key,
  3         lname varchar2(20),
  4         fname varchar2(20)
  5         );

Table created.

SQL>
SQL> insert into customers values(1, 'LOGAN', 'JULIET');
1 row created.
.
.
.
SQL> insert into customers values(21, 'JOHN', 'DAVID');
1 row created.

SQL> create table products(
  2         prod_id varchar2(20) constraint prod_pk primary key,
  3         flavour varchar2(20),
  4         food varchar2(20),
  5         price number
  6         );

Table created.

SQL>
SQL> insert into products values('20-BC-C-10','Chocolate','Cake',8.95);
1 row created.
.
.
```

```
.
SQL> insert into products values('51-BLU','Blueberry','Danish',1.15);
1 row created.

SQL> create table receipts(
  2         rec_no number(5) constraint rec_pk primary key,
  3         rdate date,
  4         cid number(2) constraint rec_fk references customers(cust_no)
  5         );

Table created.

SQL>
SQL> INSERT INTO Receipts values(18129, '28-Oct-2007', 15);
1 row created.
.
.
.
SQL> INSERT INTO Receipts values(34378, '23-Oct-2007', 6);
1 row created.

SQL> create table item_list(
  2         rec_no number(5) constraint it_fk1 references receipts(rec_no),
  3         ordinal number(2),
  4         item varchar2(20) constraint it_fk2 references products(prod_id),
  5         constraint item_pk primary key(rec_no,ordinal)
  6         );

Table created.

SQL>
SQL> insert into item_list values(18129, 1,  '70-TU');
1 row created.
.
.
.
SQL> insert into item_list values(34378, 2,  '45-VA');
1 row created.

SQL>
SQL> REM *** End of database population ***

SQL>
SQL> REM ******************** END OF DATA FILE ************************
```

Script file:

```
SQL> @z:/a5plsql.sql;
SQL> REM Assignment 5
SQL>
SQL> REM ------------------------------------------------------------------------
> REM *** ASSIGNMENT QUESTIONS ***
SQL> REM ------------------------------------------------------------------------
> REM Consider the schema used in Assignment 3.
SQL>
SQL>
SQL> REM **_____Write a PL/SQL block for the following:_____**
SQL>
SQL> REM 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>
SQL> create or replace function prod_det(food1 products.food%type, flav1 products.flavour%type)
return int is
  2  case1 varchar2(15);
  3  case2 varchar2(15);
  4  case3 varchar2(15);
  5  cursor c1 is select prod_id from products where food=food1 and flavour=flav1;
  6  cursor c2 is select prod_id from products where food=food1;
  7  cursor c3 is select prod_id from products where flavour=flav1;
  8  begin
  9          open c1;
 10          loop
 11                  fetch c1 into case1;
 12                  if c1%found then
 13                          return 1;
 14                  else
 15                          open c2;
 16                          loop
 17                                  fetch c2 into case2;
 18                                  if c2%found then
 19                                          return 2;
 20                                  else
 21                                          return 5;
 22                                  end if;
 23                          end loop;
 24                          close c2;
 25                          open c3;
 26                          loop
 27                                  fetch c3 into case3;
 28                                  if c3%found then
 29                                          return 3;
 30                                  else
 31                                          return 4;
 32                                  end if;
 33                          end loop;
```

```
34                              close c3;
35                      end if;
36          end loop;
37          close c1;
38  EXCEPTION
39  when no_data_found then
40          return 0;
41  end;
42  /
```

Function created.

```
SQL>
SQL> declare
  2  foods products.food%type;
  3  flavours products.flavour%type;
  4  prod products.prod_id%type;
  5  begin
  6  foods:='&foods';
  7  flavours:='&flavours';
  8  prod:=prod_det(foods,flavours);
  9
 10  if prod = 1 then
 11          dbms_output.put_line('The combination of food '||foods||' and flavour '||flavours||' is
available.');
 12  elsif prod = 2 then
 13          dbms_output.put_line('The food '||foods||' is available, but without the flavour
'||flavours||'.');
 14  elsif prod = 3 then
 15          dbms_output.put_line('The flavour '||flavours||' is available, but without the food
'||foods||'.');
 16  elsif prod = 4 then
 17          dbms_output.put_line('Neither the food '||foods||' nor the flavour '||flavours||' is
available.');
 18  elsif prod = 5 then
 19          dbms_output.put_line('Neither the food '||foods||' nor the flavour '||flavours||' is
available.');
 20  end if;
 21
 22  end;
 23  /
Enter value for foods: Cake
old   6: foods:='&foods';
new   6: foods:='Cake';
Enter value for flavours: Chocolate
old   7: flavours:='&flavours';
new   7: flavours:='Chocolate';
The combination of food Cake and flavour Chocolate is available.

PL/SQL procedure successfully completed.
```

```
SQL> /
Enter value for foods: Cake
old   6: foods:='&foods';
new   6: foods:='Cake';
Enter value for flavours: Cheese
old   7: flavours:='&flavours';
new   7: flavours:='Cheese';
The food Cake is available, but without the flavour Cheese.

PL/SQL procedure successfully completed.

SQL> /
Enter value for foods: Random
old   6: foods:='&foods';
new   6: foods:='Random';
Enter value for flavours: Variable
old   7: flavours:='&flavours';
new   7: flavours:='Variable';
Neither the food Random nor the flavour Variable is available.

PL/SQL procedure successfully completed.

SQL>
SQL> REM Validations:
SQL>
SQL> select * from products where food='Cake' and flavour='Chocolate';

PROD_ID              FLAVOUR              FOOD                 PRICE
-------------------- -------------------- -------------------- ----------
20-BC-C-10           Chocolate            Cake                   8.95

SQL> select * from products where food='Cake';

PROD_ID              FLAVOUR              FOOD                 PRICE
-------------------- -------------------- -------------------- ----------
20-BC-C-10           Chocolate            Cake                   8.95
20-BC-L-10           Lemon                Cake                   8.95
20-CA-7.5            Casino               Cake                  15.95
24-8x10              Opera                Cake                  15.95
25-STR-9             Strawberry           Cake                  11.95
26-8x10              Truffle              Cake                  15.95
46-11                Napoleon             Cake                  13.49

7 rows selected.

SQL> select * from products where food='Random';

no rows selected
```

```
SQL>
SQL>
SQL> REM 2. On a given date, find the number of items sold (Use Implicit cursor).
SQL>
SQL> create or replace procedure dateitems (dt in date) is
  2  numitems item_list.item%type;
  3  begin
  4          select count(item) into numitems
  5                  from receipts join item_list using(rec_no) where rdate=dt;
  6          if sql%notfound then
  7                  dbms_output.put_line('No items sold');
  8          elsif sql%found then
  9                  dbms_output.put_line('No. of items sold: '||numitems);
 10          end if;
 11  end;
 12  /

Procedure created.

SQL>
SQL> declare
  2  rdate date:=&rdate;
  3  begin
  4          dateitems(rdate);
  5  end;
  6  /
Enter value for rdate: '20-OCT-07'
old   2: rdate date:=&rdate;
new   2: rdate date:='20-OCT-07';
No. of items sold: 25

PL/SQL procedure successfully completed.

SQL> /
Enter value for rdate: 20-MAR-07'
old   2: rdate date:=&rdate;
new   2: rdate date:=20-MAR-07';
ERROR:
ORA-01756: quoted string not properly terminated


SQL>
SQL> REM Validations:
SQL>
SQL> select count(*) from receipts join item_list using(rec_no) where rdate='20-OCT-07';

  COUNT(*)
----------
        25
```

```
SQL> select count(*) from receipts join item_list using(rec_no) where rdate='20-MAR-07';

  COUNT(*)
----------
         0

SQL>
SQL>
SQL> REM 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.
SQL> REM 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>
SQL> create or replace procedure price_det (pr in products.price%type) is
  2  flav1 products.flavour%type;
  3  food1 products.food%type;
  4  price1 products.price%type;
  5  pid1 products.prod_id%type;
  6  cursor c1 is select * from products where price in
  7          (select max(price) from products where price<=pr);
  8  begin
  9          open c1;
 10          dbms_output.put_line('PROD_ID      FOOD          FLAVOUR      QTY');
 11          loop
 12                  fetch c1 into pid1,food1,flav1,price1;
 13                  exit when c1%notfound;
 14                  dbms_output.put_line(pid1||'            '||food1||'      '||flav1||'            '||price1);
 15          end loop;
 16          dbms_output.put_line(c1%rowcount||' product(s) were found equal to or closest to the
given price.');
 17          close c1;
 18  end;
 19  /

Procedure created.

SQL>
SQL> declare
  2  price number:=&price;
  3  begin
  4          price_det(price);
  5  end;
  6  /
Enter value for price: 15.95
old   2: price number:=&price;
new   2: price number:=15.95;
PROD_ID          FOOD         FLAVOUR      QTY
20-CA-7.5          Casino         Cake       15.95
24-8x10          Opera         Cake       15.95
26-8x10          Truffle         Cake       15.95
```

```
3 product(s) were found equal to or closest to the given price.

PL/SQL procedure successfully completed.

SQL> /
Enter value for price: 16
old   2: price number:=&price;
new   2: price number:=16;
PROD_ID          FOOD        FLAVOUR      QTY
20-CA-7.5        Casino        Cake       15.95
24-8x10          Opera        Cake       15.95
26-8x10          Truffle        Cake       15.95
3 product(s) were found equal to or closest to the given price.

PL/SQL procedure successfully completed.

SQL>
SQL> REM Validations:
SQL>
SQL> select * from products where price=15.95;

PROD_ID              FLAVOUR              FOOD                    PRICE
-------------------- -------------------- -------------------- ----------
20-CA-7.5            Casino               Cake                    15.95
24-8x10              Opera                Cake                    15.95
26-8x10              Truffle              Cake                    15.95

SQL>
SQL>
SQL> REM 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.
SQL>
SQL> create or replace procedure ord_det (rnum in number) is
  2  cursor c1 is select food,flavour,count(*) from item_list,products
  3          where rec_no=rnum and item=prod_id
  4          group by food,flavour;
  5  namef customers.fname%type;
  6  namel customers.lname%type;
  7  flav1 products.flavour%type;
  8  food1 products.food%type;
  9  qty1 number;
 10  total number:=0;
 11  begin
 12          select fname,lname into namef,namel from customers join receipts on (cust_no=cid)
where rec_no=rnum;
 13          dbms_output.put_line('Customer name: '||namef||'  '||namel);
 14          dbms_output.put_line('Items: ');
 15          open c1;
 16          dbms_output.put_line('FOOD       FLAVOUR      QTY');
 17          loop
```

```
18                      fetch c1 into food1,flav1,qty1;
19                      if c1%found then
20                              total:= total+qty1;
21                              dbms_output.put_line(food1||'        '||flav1||'             '||qty1);
22                      else
23                              exit;
24                      end if;
25              end loop;
26              dbms_output.put_line('------------------------------------');
27              dbms_output.put_line('Total quantity: '||total);
28  EXCEPTION
29  when no_data_found then
30              dbms_output.put_line('Order number does not exist!');
31  end;
32  /

Procedure created.

SQL>
SQL> declare
  2  rec_no number:=&rec_no;
  3  begin
  4              ord_det(rec_no);
  5  end;
  6  /
Enter value for rec_no: 64574
old   2: rec_no number:=&rec_no;
new   2: rec_no number:=64574;
Customer name: JOSETTE  SLINGLAND
Items:
FOOD        FLAVOUR         QTY
Twist       Almond          1
Cookie      Tuile           1
Cookie      Walnut          1
------------------------------------
Total quantity: 3

PL/SQL procedure successfully completed.

SQL> /
Enter value for rec_no: 12345
old   2: rec_no number:=&rec_no;
new   2: rec_no number:=12345;
Order number does not exist!

PL/SQL procedure successfully completed.

SQL>
SQL> REM Validations:
SQL>
```

```
SQL> select rec_no from customers join receipts on (cust_no=cid) where rec_no=64574;

   REC_NO
----------
    64574

SQL> select rec_no from receipts where rec_no=12345;

no rows selected

SQL>
SQL>
SQL> REM ********************** END OF FILE ******************************
```

	SSN COLLEGE OF ENGINEERING ***Department of Computer Science & Engineering***	***Faculty:*** ***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 !!!
*********************************************************
```

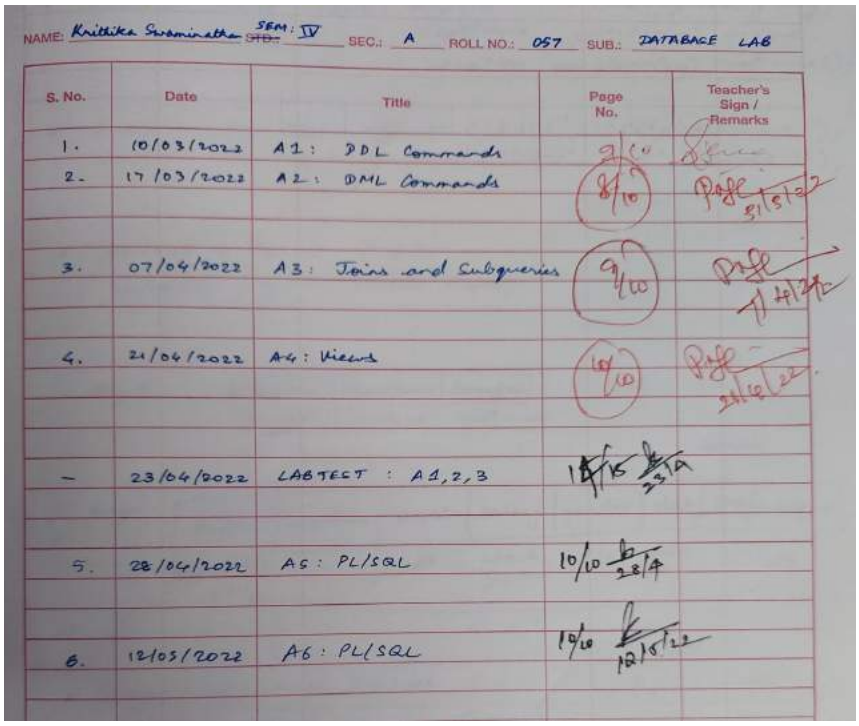
3. Take a receipt number and item as arguments, and insert this information into the Item list. However, if there is already a receipt with that receipt number, then keep adding 1 to the maximum ordinal number. Else before inserting into the Item list with ordinal as 1, ask the user to give the customer 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

Assignment 6 – PL/SQL: Procedures and Functions

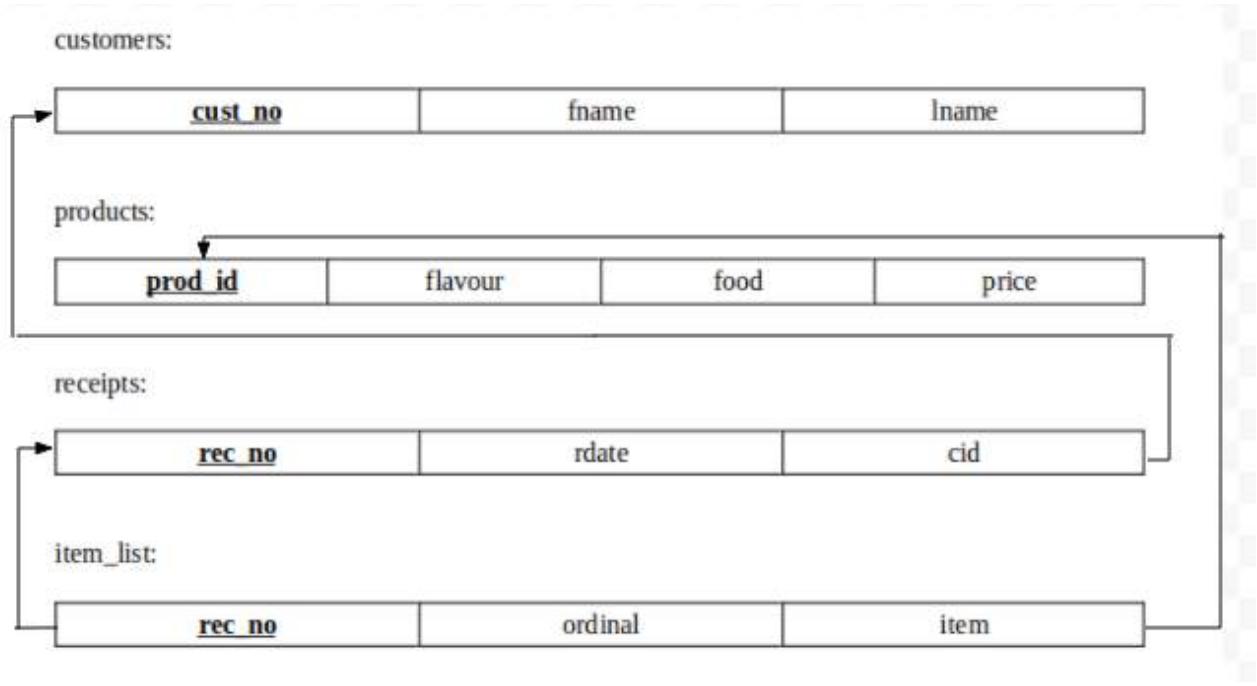
Validation:



NAME: Krithika Swaminatha SEM: IV SEC.: A ROLL NO.: 057 SUB.: DATABASE LAB

S. No.	Date	Title	Page No.	Teacher's Sign / Remarks
1.	10/03/2022	A1: DDL Commands	9/10	
2.	17/03/2022	A2: DML Commands	8/10	Prof 31/5/22
3.	07/04/2022	A3: Joins and Subqueries	9/10	Prof 7/4/22
4.	21/04/2022	A4: Views	10/10	Prof 21/4/22
–	23/04/2022	LABTEST : A1,2,3	15/15 23/4	
5.	28/04/2022	A5: PL/SQL	10/10 28/4	
6.	12/05/2022	A6: PL/SQL	10/10 12/5/22	

Schema diagram:



Data file:

```
SQL> @C:/Krithika/DBL/a6data.sql;
SQL> REM Population of Bakery Database

SQL> drop table item_list;
Table dropped.

SQL> drop table receipts;
Table dropped.

SQL> drop table products;
Table dropped.

SQL> drop table customers;
Table dropped.

SQL>
SQL> create table customers(
  2         cust_no number(2) constraint c_pk primary key,
  3         lname varchar2(20),
  4         fname varchar2(20)
  5         );

Table created.

SQL>
SQL> insert into customers values(1, 'LOGAN', 'JULIET');
1 row created.
.
.
.
SQL> insert into customers values(21, 'JOHN', 'DAVID');
1 row created.

SQL> create table products(
  2         prod_id varchar2(20) constraint prod_pk primary key,
  3         flavour varchar2(20),
  4         food varchar2(20),
  5         price number
  6         );

Table created.

SQL>
SQL> insert into products values('20-BC-C-10','Chocolate','Cake',8.95);
1 row created.
.
.
```

```
.
SQL> insert into products values('51-BLU','Blueberry','Danish',1.15);
1 row created.

SQL> create table receipts(
  2         rec_no number(5) constraint rec_pk primary key,
  3         rdate date,
  4         cid number(2) constraint rec_fk references customers(cust_no)
  5         );

Table created.

SQL>
SQL> INSERT INTO Receipts values(18129, '28-Oct-2007', 15);
1 row created.
.
.
.
SQL> INSERT INTO Receipts values(34378, '23-Oct-2007', 6);
1 row created.

SQL> create table item_list(
  2         rec_no number(5) constraint it_fk1 references receipts(rec_no),
  3         ordinal number(2),
  4         item varchar2(20) constraint it_fk2 references products(prod_id),
  5         constraint item_pk primary key(rec_no,ordinal)
  6         );

Table created.

SQL>
SQL> insert into item_list values(18129, 1,  '70-TU');
1 row created.
.
.
.
SQL> insert into item_list values(34378, 2,  '45-VA');
1 row created.

SQL>
SQL> REM *** End of database population ***

SQL>
SQL> REM ******************** END OF DATA FILE ************************
```

Script file:

```
SQL> @z:/a6plsql.sql;
SQL> REM Assignment 6
SQL>
SQL> REM ------------------------------------------------------------------------
> REM *** ASSIGNMENT QUESTIONS ***
SQL> REM ------------------------------------------------------------------------
> REM Consider the schema used in Assignment 3.
SQL>
SQL>
SQL> REM **_____Write a PL/SQL stored procedure for the following:_____**
SQL>
SQL> REM 1. For the given receipt number, calculate the Discount as follows:
SQL> REM For total amount > $10 and total amount < $25: Discount=5%
SQL> REM For total amount > $25 and total amount < $50: Discount=10%
SQL> REM For total amount > $50: Discount=20%
SQL> REM Calculate the amount (after the discount) and update the same in Receipts table.
SQL> REM Print the receipt as shown below:
SQL> REM ************************************************************
SQL> REM Receipt Number:13355
SQL> REM Customer Name: TOUSSAND SHARRON
SQL> REM Receipt Date :19Oct2007
SQL> REM ************************************************************
SQL> REM Sno Flavor Food Price
SQL> REM 1. Opera Cake 15.95
SQL> REM 2. Lemon Cookie 0.79
SQL> REM 3. Napoleon Cake 13.49
SQL> REM
SQL> REM Total = $ 30.23
SQL> REM Discount(10%) :$ 3.02
SQL> REM
SQL> REM Amount to be paid :$ 27.21
SQL> REM ************************************************************
SQL> REM Great Offers! Discount up to 25% on DIWALI Festival Day...
SQL> REM ************************************************************
SQL>
SQL>
SQL> create or replace procedure discount
  2  (amt in products.price%type, dis out products.price%type, dispct out products.price%type, sp
out products.price%type) is
  3  begin
  4          dis := 0;
  5          dispct := 0;
  6          if amt>10 and amt<25 then
  7                  dis := (5*amt)/100.00;
  8                  dispct := 5;
  9          elsif amt>25 and amt<50 then
 10                  dis := (10*amt)/100.00;
 11                  dispct := 10;
```

```
 12          elsif amt>50 then
 13                  dis := (20*amt)/100.00;
 14                  dispct := 20;
 15          end if;
 16          sp := amt - dis;
 17  end;
 18  /

Procedure created.

SQL>
SQL> declare
  2  rid receipts.rec_no%type;
  3  billdate receipts.rdate%type;
  4  custfname customers.fname%type;
  5  custlname customers.lname%type;
  6  rtotal products.price%type;
  7  d products.price%type;
  8  dp products.price%type;
  9  finalamt products.price%type;
 10  cursor c1 is select flavour, food, price
 11          from products join item_list on (prod_id = item)
 12          where rec_no = rid;
 13  countnum integer;
 14  itemfood products.food%type;
 15  itemfl products.flavour%type;
 16  itemprice products.price%type;
 17
 18  begin
 19
 20  rid := &receipt;
 21
 22  select rdate, fname, lname, count(item), sum(price) as total
 23  into billdate, custfname, custlname, countnum, rtotal
 24  from receipts join item_list using (rec_no) join products on (prod_id = item) join customers on
(cid = cust_no)
 25  group by rec_no, rdate, fname, lname having rec_no = rid;
 26
 27  discount(rtotal,d,dp,finalamt);
 28
 29  dbms_output.put_line('->');
 30  dbms_output.put_line('**************************************************');
 31  dbms_output.put_line('Receipt no.: '||rid);
 32  dbms_output.put_line('Customer name: '||custfname||' '||custlname);
 33  dbms_output.put_line('Receipt date: '||billdate);
 34  dbms_output.put_line('**************************************************');
 35  dbms_output.put_line('S.No.  Flavour     Food       Price');
 36  open c1;
 37  for sno in 1..countnum loop
 38          fetch c1 into itemfl, itemfood, itemprice;
```

```
 39          dbms_output.put_line(sno||'     '||itemfl||'     '||itemfood||'    '||itemprice);
 40          end loop;
 41  close c1;
 42  dbms_output.put_line('_____________________________________________________');
 43  dbms_output.put_line('Total = $'||rtotal);
 44  dbms_output.put_line('Discount('||dp||'%): $'||d);
 45  dbms_output.put_line('_____________________________________________________');
 46  dbms_output.put_line('Grand Total = $'||finalamt);
 47  dbms_output.put_line('**************************************************');
 48  dbms_output.put_line('Great Offers! Discounts up to 25% on DIWALI Day...');
 49  dbms_output.put_line('**************************************************');
 50
 51  end;
 52  /
Enter value for receipt: 13355
old  20: rid := &receipt;
new  20: rid := 13355;
->
**************************************************
Receipt no.: 13355
Customer name: SHARRON TOUSSAND
Receipt date: 19-OCT-07
**************************************************
S.No.  Flavour      Food      Price
1     Opera       Cake        15.95
2     Lemon       Cookie        .79
3     Napoleon       Cake        13.49
_____________________________________________________
Total = $30.23
Discount(10%): $3.023
_____________________________________________________
Grand Total = $27.207
**************************************************
Great Offers! Discounts up to 25% on DIWALI Day...
**************************************************

PL/SQL procedure successfully completed.

SQL>
SQL>
SQL> REM Validations:
SQL>
SQL> select rdate, fname, lname, count(item) as count, sum(price) as total
  2  from receipts join item_list using (rec_no) join products on (prod_id = item) join customers on
(cid = cust_no)
  3  group by rec_no, rdate, fname, lname having rec_no = 13355;

RDATE     FNAME                LNAME                  COUNT      TOTAL
--------- -------------------- -------------------- ---------- ----------
19-OCT-07 SHARRON              TOUSSAND                    3      30.23
```

```
SQL>
SQL>
SQL> REM 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.
SQL> REM The best item is determined by the maximum ordered product among many customers
for the given food type.
SQL>
SQL> create or replace procedure calcount
  2  (budget in products.price%type, val in products.price%type, qty out integer) is
  3  begin
  4          if val <= budget then
  5                  qty := budget/val;
  6          else
  7                  qty := 0;
  8          end if;
  9  end;
 10  /

Procedure created.

SQL>
SQL> declare
  2  budget products.price%type;
  3  val products.price%type;
  4  pfood products.food%type;
  5  qty integer(3);
  6  pid products.prod_id%type;
  7  psample products%rowtype;
  8  cursor c1 is select distinct(p.prod_id), p.food, p.flavour, p.price
  9          from products p join item_list i on (p.prod_id = i.item)
 10          where p.price <= budget and p.food = pfood
 11          group by p.prod_id, p.food, p.flavour, p.price
 12          order by count(*) desc;
 13  countnum integer;
 14  pfl products.flavour%type;
 15
 16  begin
 17  budget := '&budget';
 18  pfood := '&food';
 19
 20  begin
 21  select p1.prod_id, p1.price, p1.flavour into pid, val, pfl
 22          from products p1 join item_list i on (p1.prod_id = i.item)
 23          where p1.price <= budget and p1.food = pfood
 24          group by p1.prod_id, p1.food, p1.flavour, p1.price
 25          having count(*) >= all (
 26                  select count(*)
 27                  from products p2 join item_list i on p2.prod_id = i.item
 28                  where p2.price <= budget and p2.food = pfood
```

```
                    group by p2.prod_id, p2.food, p2.flavour, p2.price);
EXCEPTION
when no_data_found then
        dbms_output.put_line('No recommendations found');
        return;
end;

select count(count(*)) into countnum
        from products p join item_list i on (p.prod_id = i.item)
        where p.price <= budget and p.food = pfood
        group by p.prod_id, p.food, p.flavour, p.price;


dbms_output.put_line('*************************************************************');
dbms_output.put_line('S.No.  Prod_ID   Food       Flavour         Price');
dbms_output.put_line('*************************************************************');

open c1;
for sno in 1..countnum loop
        fetch c1 into psample;
        dbms_output.put_line(sno||' '||psample.prod_id||'    '||psample.food||'   '||psample.flavour||'  '||psample.price);
        end loop;
close c1;
dbms_output.put_line('*************************************************************');

calcount(budget,val,qty);
dbms_output.put_line(pid||' with '||pfl||' flavour is the best item in '||pfood||' type!');
dbms_output.put_line('You are entitled to purchase '||qty||' '||pfood||' '||pfl||'s for the given budget!!!');
dbms_output.put_line('*************************************************************');

end;
/
Enter value for budget: 10
old  17: budget := '&budget';
new  17: budget := '10';
Enter value for food: Meringue
old  18: pfood := '&food';
new  18: pfood := 'Meringue';
*************************************************************
S.No.  Prod_ID       Food        Flavour       Price
*************************************************************
```

```
1 70-M-CH-DZ    Chocolate   Meringue   1.25
2 70-M-VA-SM-DZ    Vanilla   Meringue   1.15
************************************************************
70-M-CH-DZ with Chocolate flavour is the best item in Meringue type!
You are entitled to purchase 8 Meringue Chocolates for the given budget!!!
************************************************************

PL/SQL procedure successfully completed.

SQL>
SQL>
SQL> REM 3. Take a receipt number and item as arguments, and insert this information into the
Item list.
SQL> REM However, if there is already a receipt with that receipt number, then keep adding 1 to
the maximum ordinal number.
SQL> REM 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> create or replace procedure insertitem
  2  (rid in receipts.rec_no%type, ord in item_list.ordinal%type, pid in products.prod_id%type) is
  3  begin
  4          insert into item_list values (rid,ord,pid);
  5  end;
  6  /

Procedure created.

SQL>
SQL> create or replace procedure insertreceipt
  2  (rid in receipts.rec_no%type, rdt in receipts.rdate%type, rcid in customers.cust_no%type) is
  3  begin
  4          insert into receipts values (rid,rdt,rcid);
  5  end;
  6  /

Procedure created.

SQL>
SQL> create or replace procedure findcust
  2  (cfname in customers.fname%type, clname in customers.lname%type, foundcid out
customers.cust_no%type) is
  3  begin
  4          begin
  5                  select c.cust_no into foundcid
  6                  from customers c
  7                  where c.fname = cfname and c.lname = clname;
  8          EXCEPTION
  9          when no_data_found then
 10                  dbms_output.put_line('Customer ID not found!');
 11                  foundcid := 0;
```

```
12          end;
13  end;
14  /
```

Procedure created.

```
SQL>
SQL> declare
  2  cfname customers.fname%type;
  3  clname customers.lname%type;
  4  fcid customers.cust_no%type;
  5  rid receipts.rec_no%type;
  6  ord item_list.ordinal%type;
  7  pid products.prod_id%type;
  8  rdt receipts.rdate%type;
  9  item_row item_list%rowtype;
 10  cursor c1 is select * from item_list i
 11          where i.rec_no = rid
 12          order by i.ordinal desc;
 13  maxord item_list.ordinal%type;
 14  flag number(1) := 0;
 15
 16  begin
 17          rid := '&receipt';
 18          pid := '&product';
 19          open c1;
 20          fetch c1 into item_row;
 21          if c1%rowcount > 0 then
 22                  flag := 1;
 23          end if;
 24          close c1;
 25
 26          if (flag = 1) then
 27                  ord := item_row.ordinal+1;
 28
 29          else
 30                  cfname := '&firstname';
 31                  clname := '&lastname';
 32                  rdt := '&date';
 33                  findcust(cfname,clname,fcid);
 34                  if (fcid = 0) then
 35                          dbms_output.put_line('Cannot add a new customer. Exiting...');
 36                          return;
 37                  end if;
 38                  insertreceipt(rid,rdt,fcid);
 39                  ord := 1;
 40          end if;
 41
 42          insertitem(rid,ord,pid);
 43          dbms_output.put_line('Successfully inserted record!');
```

```
 44
 45  EXCEPTION
 46          when no_data_found then
 47                  dbms_output.put_line('Could not insert record!');
 48  end;
 49  /
Enter value for receipt: 8888
old  17:        rid := '&receipt';
new  17:        rid := '8888';
Enter value for product: 51-BC
old  18:        pid := '&product';
new  18:        pid := '51-BC';
Enter value for firstname: DAVID
old  30:                cfname := '&firstname';
new  30:                cfname := 'DAVID';
Enter value for lastname: JOHN
old  31:                clname := '&lastname';
new  31:                clname := 'JOHN';
Enter value for date: 11-OCT-07
old  32:                rdt := '&date';
new  32:                rdt := '11-OCT-07';
Successfully inserted record!

PL/SQL procedure successfully completed.

SQL> /
Enter value for receipt: 34378
old  17:        rid := '&receipt';
new  17:        rid := '34378';
Enter value for product: 51-BLU
old  18:        pid := '&product';
new  18:        pid := '51-BLU';
Enter value for firstname: JULIET
old  30:                cfname := '&firstname';
new  30:                cfname := 'JULIET';
Enter value for lastname: LOGAN
old  31:                clname := '&lastname';
new  31:                clname := 'LOGAN';
Enter value for date: 12-0CT-07
old  32:                rdt := '&date';
new  32:                rdt := '12-0CT-07';
Successfully inserted record!

PL/SQL procedure successfully completed.

SQL>
SQL> REM Validations:
SQL>
SQL> select * from item_list where rec_no = 8888;
```

```
   REC_NO    ORDINAL ITEM
---------- ---------- --------------------
     8888          1 51-BC

SQL> select * from item_list where rec_no = 34378;

   REC_NO    ORDINAL ITEM
---------- ---------- --------------------
    34378          1 90-CHR-11
    34378          2 45-VA
    34378          3 51-BLU

SQL>
SQL>
SQL> REM 4. Write a stored function to display the customer name who ordered maximum for the
SQL> given food and flavor.
SP2-0734: unknown command beginning "given food..." - rest of line ignored.
SQL>
SQL> create or replace function maxorders
  2  (pid in products.prod_id%type) return varchar2 as
  3  cid customers.cust_no%type;
  4  max int;
  5  name1 customers.fname%type;
  6  name2 customers.lname%type;
  7  name varchar2(40);
  8
  9  begin
 10          select max(count(*)) into max
 11                  from receipts r join item_list i on (i.rec_no = r.rec_no)
 12                  where i.item = pid
 13                  group by r.cid;
 14          select r.cid into cid from receipts r join item_list i on (i.rec_no = r.rec_no)
 15                  where i.item = pid
 16                  group by r.cid having count(*)=max;
 17          select c1.fname, c1.lname into name1, name2 from customers c1 where c1.cust_no = cid;
 18
 19          name := name1||' '||name2;
 20          return name;
 21  EXCEPTION
 22          when no_data_found then
 23                  dbms_output.put_line('The specified product does not exist!');
 24  end maxorders;
 25  /

Function created.

SQL>
SQL> declare
  2  name varchar2(40);
  3  pid products.prod_id%type;
```

```
  4  pfood products.food%type;
  5  pfl products.flavour%type;
  6
  7  begin
  8          pfood := '&food';
  9          pfl := '&flavour';
 10          select p1.prod_id into pid from products p1
 11                  where p1.food = pfood and p1.flavour = pfl;
 12          name := maxorders(pid);
 13          dbms_output.put_line('Name: '||name);
 14  EXCEPTION
 15          when no_data_found then
 16                  dbms_output.put_line('The specified product does not exist!');
 17  end;
 18  /
Enter value for food: Danish
old   8:          pfood := '&food';
new   8:          pfood := 'Danish';
Enter value for flavour: Blueberry
old   9:          pfl := '&flavour';
new   9:          pfl := 'Blueberry';
Name: RAYFORD SOPKO

PL/SQL procedure successfully completed.

SQL>
SQL>
SQL> REM 5. Implement Question (2) using stored function to return the amount to be paid and
SQL> update the same, for the given receipt number.
  2
SQL> create or replace function discountfn
  2  (amt in products.price%type, dis out products.price%type, dispct out products.price%type)
  3  return products.price%type is
  4  sp products.price%type;
  5
  6  begin
  7          dis := 0;
  8          dispct := 0;
  9          if amt>10 and amt<25 then
 10                  dis := (5*amt)/100.00;
 11                  dispct := 5;
 12          elsif amt>25 and amt<50 then
 13                  dis := (10*amt)/100.00;
 14                  dispct := 10;
 15          elsif amt>50 then
 16                  dis := (20*amt)/100.00;
 17                  dispct := 20;
 18          end if;
 19          sp := amt - dis;
 20          return sp;
```

```
 21  end;
 22  /

Function created.

SQL>
SQL> declare
  2  rid receipts.rec_no%type;
  3  billdate receipts.rdate%type;
  4  custfname customers.fname%type;
  5  custlname customers.lname%type;
  6  rtotal products.price%type;
  7  d products.price%type;
  8  dp products.price%type;
  9  finalamt products.price%type;
 10  cursor c1 is select flavour, food, price
 11          from products join item_list on (prod_id = item)
 12          where rec_no = rid;
 13  countnum integer;
 14  itemfood products.food%type;
 15  itemfl products.flavour%type;
 16  itemprice products.price%type;
 17
 18  begin
 19
 20  rid := '&receipt';
 21
 22  select rdate, fname, lname, count(item), sum(price) as total
 23  into billdate, custfname, custlname, countnum, rtotal
 24  from receipts join item_list using (rec_no) join products on (prod_id = item) join customers on
(cid = cust_no)
 25  group by rec_no, rdate, fname, lname having rec_no = rid;
 26
 27  finalamt := discountfn(rtotal,d,dp);
 28
 29  dbms_output.put_line('->');
 30  dbms_output.put_line('**************************************************');
 31  dbms_output.put_line('Receipt no.: '||rid);
 32  dbms_output.put_line('Customer name: '||custfname||' '||custlname);
 33  dbms_output.put_line('Receipt date: '||billdate);
 34  dbms_output.put_line('**************************************************');
 35  dbms_output.put_line('S.No.  Flavour     Food       Price');
 36  open c1;
 37  for sno in 1..countnum loop
 38          fetch c1 into itemfl, itemfood, itemprice;
 39          dbms_output.put_line(sno||'     '||itemfl||'     '||itemfood||'    '||itemprice);
 40          end loop;
 41  close c1;
 42  dbms_output.put_line('_____________________________________________________');
 43  dbms_output.put_line('Total = $'||rtotal);
```

```
 44  dbms_output.put_line('Discount('||dp||'%): $'||d);
 45  dbms_output.put_line('_____________________________________________');
 46  dbms_output.put_line('Grand Total = $'||finalamt);
 47  dbms_output.put_line('**************************************************');
 48  dbms_output.put_line('Great Offers! Discounts up to 25% on DIWALI Day...');
 49  dbms_output.put_line('**************************************************');
 50
 51  end;
 52  /
Enter value for receipt: 13355
old  20: rid := '&receipt';
new  20: rid := '13355';
->
**************************************************
Receipt no.: 13355
Customer name: SHARRON TOUSSAND
Receipt date: 19-OCT-07
**************************************************
S.No.  Flavour      Food      Price
1     Opera       Cake        15.95
2     Lemon       Cookie        .79
3     Napoleon       Cake        13.49
_____________________________________________
Total = $30.23
Discount(10%): $3.023
_____________________________________________
Grand Total = $27.207
**************************************************
Great Offers! Discounts up to 25% on DIWALI Day...
**************************************************

PL/SQL procedure successfully completed.

SQL>
SQL>
SQL> REM ************************* END OF FILE ******************************
```

	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:

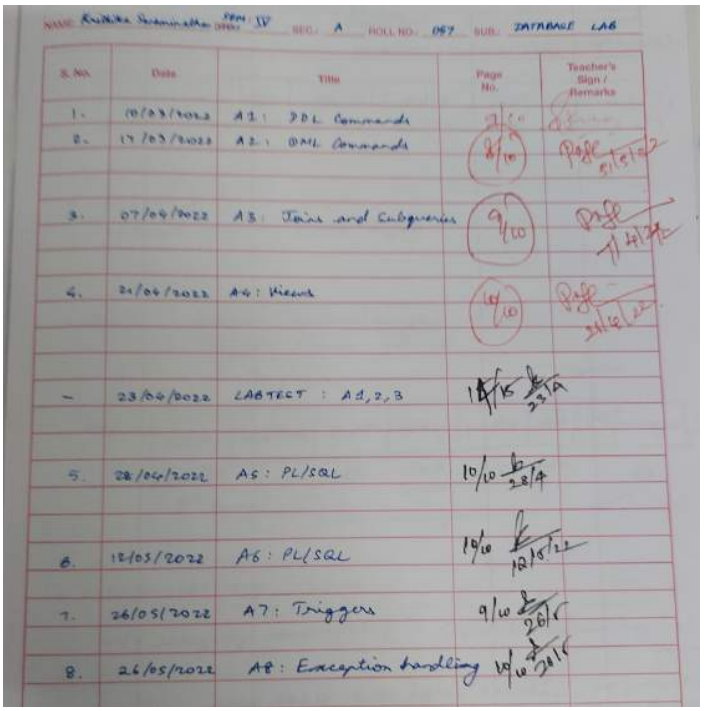
1. The combination of Flavor and Food determines the product id. Hence, while inserting a new instance into the *Products* relation, ensure that the same combination of Flavor and Food is not already available.
2. While entering an item into the *item_list* relation, update the amount in *Receipts* with 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

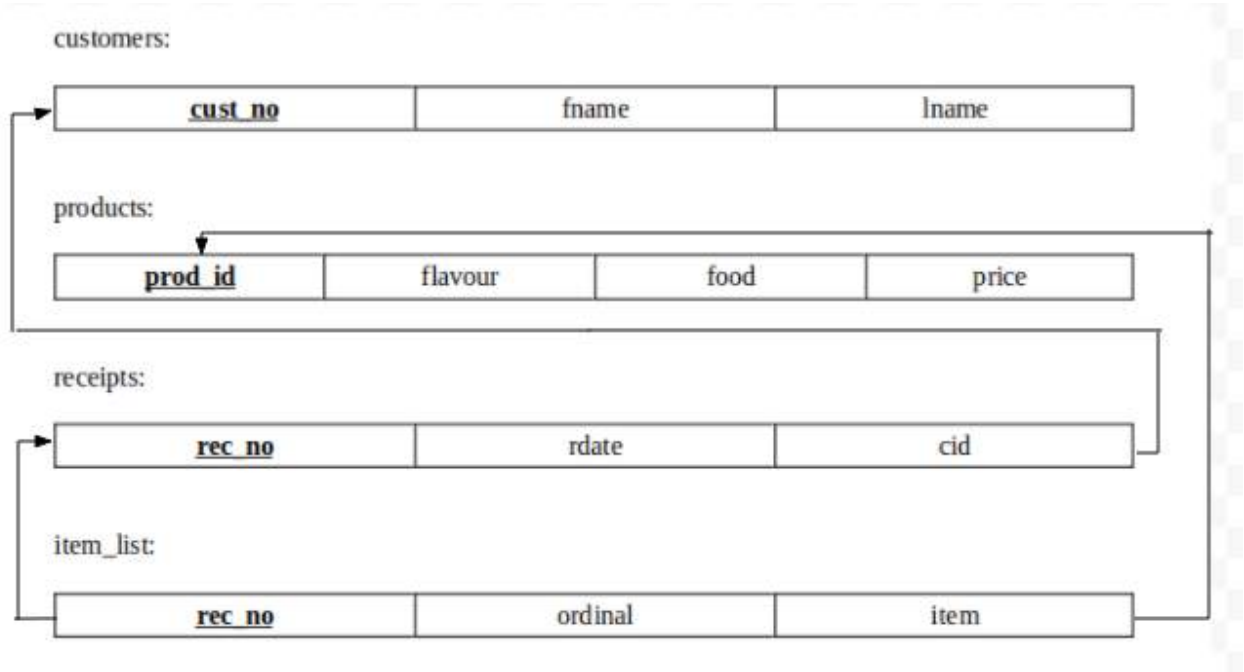
Assignment 7 – Triggers

Validation:



S. No.	Date	Title	Page No.	Teacher's Sign / Remarks
1.	10/03/2022	A1: DDL Commands	7/10	
2.	17/03/2022	A2: DML Commands	8/10	Page 31/5/22
3.	07/04/2022	A3: Joins and Subqueries	9/10	Page 1/4/22
4.	21/04/2022	A4: Views	10/10	Page 21/4/22
–	23/04/2022	LABTEST : A1,2,3	15/15 23/4	
5.	28/04/2022	A5: PL/SQL	10/10 28/4	
6.	12/05/2022	A6: PL/SQL	10/10 12/5/22	
7.	26/05/2022	A7: Triggers	9/10 26/5	
8.	26/05/2022	A8: Exception handling	10/10 26/5	

Schema diagram:



Data file:

```
SQL> @C:/Krithika/DBL/a7data.sql;
SQL> REM Population of Bakery Database

SQL> drop table item_list;
Table dropped.

SQL> drop table receipts;
Table dropped.

SQL> drop table products;
Table dropped.

SQL> drop table customers;
Table dropped.

SQL>
SQL> create table customers(
  2         cust_no number(2) constraint c_pk primary key,
  3         lname varchar2(20),
  4         fname varchar2(20)
  5         );

Table created.

SQL>
SQL> insert into customers values(1, 'LOGAN', 'JULIET');
1 row created.
.
.
.
SQL> insert into customers values(21, 'JOHN', 'DAVID');
1 row created.

SQL> create table products(
  2         prod_id varchar2(20) constraint prod_pk primary key,
  3         flavour varchar2(20),
  4         food varchar2(20),
  5         price number
  6         );

Table created.

SQL>
SQL> insert into products values('20-BC-C-10','Chocolate','Cake',8.95);
1 row created.
.
.
.
```

```
SQL> insert into products values('51-BLU','Blueberry','Danish',1.15);
1 row created.

SQL> create table receipts(
  2         rec_no number(5) constraint rec_pk primary key,
  3         rdate date,
  4         cid number(2) constraint rec_fk references customers(cust_no)
  5         );

Table created.

SQL>
SQL> INSERT INTO Receipts values(18129, '28-Oct-2007', 15);
1 row created.
.
.
.
SQL> INSERT INTO Receipts values(34378, '23-Oct-2007', 6);
1 row created.

SQL> create table item_list(
  2         rec_no number(5) constraint it_fk1 references receipts(rec_no),
  3         ordinal number(2),
  4         item varchar2(20) constraint it_fk2 references products(prod_id),
  5         constraint item_pk primary key(rec_no,ordinal)
  6         );

Table created.

SQL>
SQL> insert into item_list values(18129, 1,  '70-TU');
1 row created.
.
.
.
SQL> insert into item_list values(34378, 2,  '45-VA');
1 row created.

SQL>
SQL> REM *** End of database population ***

SQL>
SQL> REM ******************* END OF DATA FILE ************************
```

Script file:

```
SQL> @z:/a7triggers.sql;
SQL> REM Assignment 7
SQL>
SQL> REM ------------------------------------------------------------------------
> REM *** ASSIGNMENT QUESTIONS ***
SQL> REM ------------------------------------------------------------------------
> REM Consider the schema used in Assignment 3.
SQL>
SQL> REM **Updating the Bakery Database**
SQL>
SQL> alter table receipts add amount number;

Table altered.

SQL>
SQL> update receipts r set amount = (
  2           select sum(price)
  3           from receipts join item_list using (rec_no) join products on (prod_id = item)
  4           group by rec_no having rec_no = r.rec_no);

200 rows updated.

SQL>
SQL>
SQL> REM **_____Write a PL/SQL Trigger for the following:_____**
SQL>
SQL> REM 1. The combination of Flavor and Food determines the product id. Hence, while
SQL> REM inserting a new instance into the Products relation, ensure that the same combination
SQL> REM of Flavor and Food is not already available.
SQL>
SQL> create or replace trigger check_combo
  2  BEFORE INSERT ON Products FOR EACH ROW
  3  declare
  4    flag number:=0;
  5    cursor c1 is select * from products where food=:NEW.food and flavour=:NEW.flavour;
  6    record c1%rowtype;
  7  begin
  8    open c1;
  9    fetch c1 into record;
 10    if c1%NOTFOUND then
 11        flag:=1;
 12    end if;
 13    close c1;
 14    if (flag=0) then
 15        raise_application_error(-20000,'The combination already exists.');
 16    end if;
 17  end;
 18  /
```

```
Trigger created.

SQL>
SQL> savepoint q1;

Savepoint created.

SQL>
SQL> REM Validations:
SQL>
SQL> insert into products values ('11-1','Vanilla','Tart',4);

1 row created.

SQL> insert into products values ('11-2','Vanilla','Tart',2);
insert into products values ('11-2','Vanilla','Tart',2)
            *
ERROR at line 1:
ORA-20000: The combination already exists.
ORA-06512: at "1057.CHECK_COMBO", line 13
ORA-04088: error during execution of trigger '1057.CHECK_COMBO'


SQL> select * from products where prod_id='11-1';

PROD_ID              FLAVOUR              FOOD                    PRICE
-------------------- -------------------- -------------------- ----------
11-1                 Vanilla              Tart                         4

SQL> select * from products where flavour='Lemon' and food='Meringue';

no rows selected

SQL>
SQL> rollback to q1;

Rollback complete.

SQL>
SQL>
SQL> REM 2. While entering an item into the item_list relation, update the amount in Receipts
with
SQL> REM the total amount for that receipt number.
SQL>
SQL> create or replace trigger update_amt
  2  BEFORE INSERT ON item_list FOR EACH ROW
  3  declare
  4    total number;
  5    pr products.price%type;
```

```
  6  begin
  7    select price into pr from products where prod_id=:NEW.item;
  8    select sum(price) into total from item_list, products where item=prod_id and
rec_no=:NEW.rec_no;
  9    total:=total+pr;
 10    update receipts set amount = total where rec_no=:NEW.rec_no;
 11  end;
 12  /

Trigger created.

SQL>
SQL> REM Checking items for receipt number 13355
SQL>
SQL> select * from receipts where rec_no = 13355;

    REC_NO RDATE           CID     AMOUNT
---------- --------- ---------- ----------
     13355 19-OCT-07         7      30.23

SQL> select * from item_list where rec_no = 13355 order by ordinal asc;

    REC_NO    ORDINAL ITEM
---------- ---------- --------------------
     13355          1 24-8x10
     13355          2 70-LEM
     13355          3 46-11

SQL>
SQL> insert into item_list values(13355,4,'51-BLU');

1 row created.

SQL>
SQL> select * from receipts where rec_no = 13355;

    REC_NO RDATE           CID     AMOUNT
---------- --------- ---------- ----------
     13355 19-OCT-07         7      31.38

SQL> select * from item_list where rec_no = 13355 order by ordinal asc;

    REC_NO    ORDINAL ITEM
---------- ---------- --------------------
     13355          1 24-8x10
     13355          2 70-LEM
     13355          3 46-11
     13355          4 51-BLU

SQL>
```

```
SQL>
SQL> REM 3. Implement the following constraints for Item_list relation:
SQL> REM a. A receipt can contain a maximum of five items only.
SQL> REM b. A receipt should not allow an item to be purchased more than thrice.
SQL>
SQL> create or replace trigger check_receipts
  2  BEFORE INSERT ON item_list FOR EACH ROW
  3  declare
  4    cursor c1 is  select count(*) as cnt1 from item_list where rec_no=:NEW.rec_no;
  5    cursor c2 is  select count(*) as cnt2 from item_list where rec_no=:NEW.rec_no and
item=:NEW.item;
  6    record1 c1%rowtype;
  7    record2 c2%rowtype;
  8  begin
  9    open c1;
 10    open c2;
 11    fetch c1 into record1;
 12    fetch c2 into record2;
 13    if record1.cnt1>=5 and record2.cnt2>=3 then
 14        raise_application_error(-20000,'The receipt has five items. The item has been purchased
thrice.');
 15    elsif record1.cnt1>=5 then
 16        raise_application_error(-20001,'The receipt already has five items.');
 17    elsif record2.cnt2>=3 then
 18        raise_application_error(-20002,'The item to be purchased has already been purchased
thrice.');
 19    end if;
 20    close c2;
 21    close c1;
 22  end;
 23  /

Trigger created.

SQL>
SQL> REM Validations:
SQL>
SQL> savepoint q3;

Savepoint created.

SQL>
SQL> select * from item_list where rec_no = 44798 order by ordinal asc;

    REC_NO    ORDINAL ITEM
---------- ---------- --------------------
     44798          1 90-APR-PF
     44798          2 90-CH-PF
     44798          3 90-APIE-10
     44798          4 90-APP-11
```

```
    44798         5 25-STR-9

SQL> insert into item_list values (44798,6,'51-BC');
insert into item_list values (44798,6,'51-BC')
           *
ERROR at line 1:
ORA-20001: The receipt already has five items.
ORA-06512: at "1057.CHECK_RECEIPTS", line 14
ORA-04088: error during execution of trigger '1057.CHECK_RECEIPTS'


SQL> select * from item_list where rec_no = 44798 order by ordinal asc;

   REC_NO    ORDINAL ITEM
---------- ---------- --------------------
    44798         1 90-APR-PF
    44798         2 90-CH-PF
    44798         3 90-APIE-10
    44798         4 90-APP-11
    44798         5 25-STR-9

SQL>
SQL> select * from item_list where rec_no = 53240 order by ordinal asc;

   REC_NO    ORDINAL ITEM
---------- ---------- --------------------
    53240         1 25-STR-9
    53240         2 51-ATW

SQL> insert into item_list values (53240,3,'51-ATW');

1 row created.

SQL> insert into item_list values (53240,4,'51-ATW');

1 row created.

SQL> insert into item_list values (53240,5,'51-ATW');
insert into item_list values (53240,5,'51-ATW')
           *
ERROR at line 1:
ORA-20002: The item to be purchased has already been purchased thrice.
ORA-06512: at "1057.CHECK_RECEIPTS", line 16
ORA-04088: error during execution of trigger '1057.CHECK_RECEIPTS'


SQL> select * from item_list where rec_no = 53240 order by ordinal asc;

   REC_NO    ORDINAL ITEM
---------- ---------- --------------------
```

```
    53240        1 25-STR-9
    53240        2 51-ATW
    53240        3 51-ATW
    53240        4 51-ATW

SQL>
SQL> rollback to q3;

Rollback complete.

SQL>
SQL>
SQL> REM ************************* END OF FILE******************************
```

	SSN COLLEGE OF ENGINEERING	***Faculty:***
	Department of Computer Science & Engineering	***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:

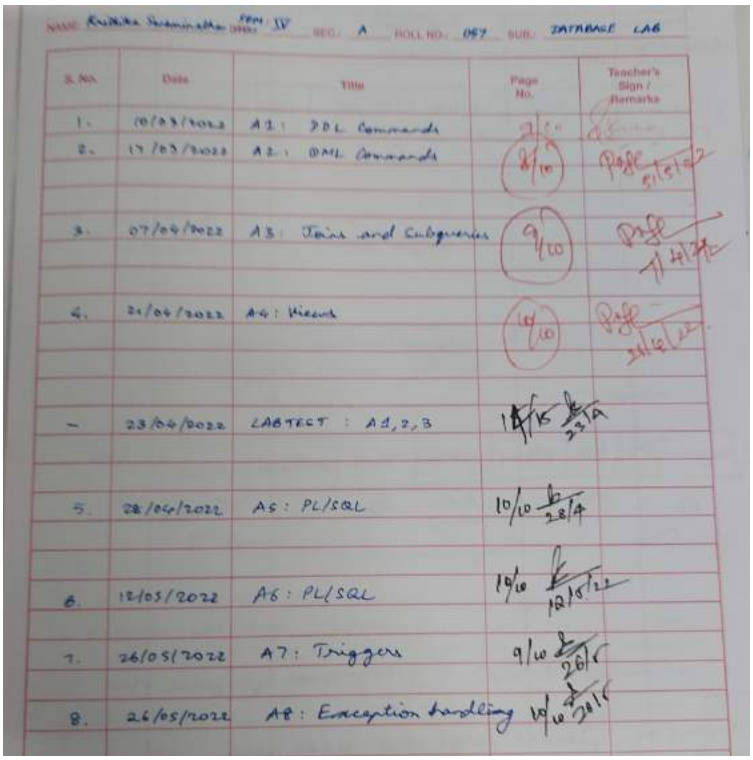
1. For the given receipt number, if there are no rows then display as "No order with the given receipt <number>". If the receipt contains more than one item, display as "The given receipt <number> contains more than one item". If the receipt contains single item, display as "The given receipt <number> 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

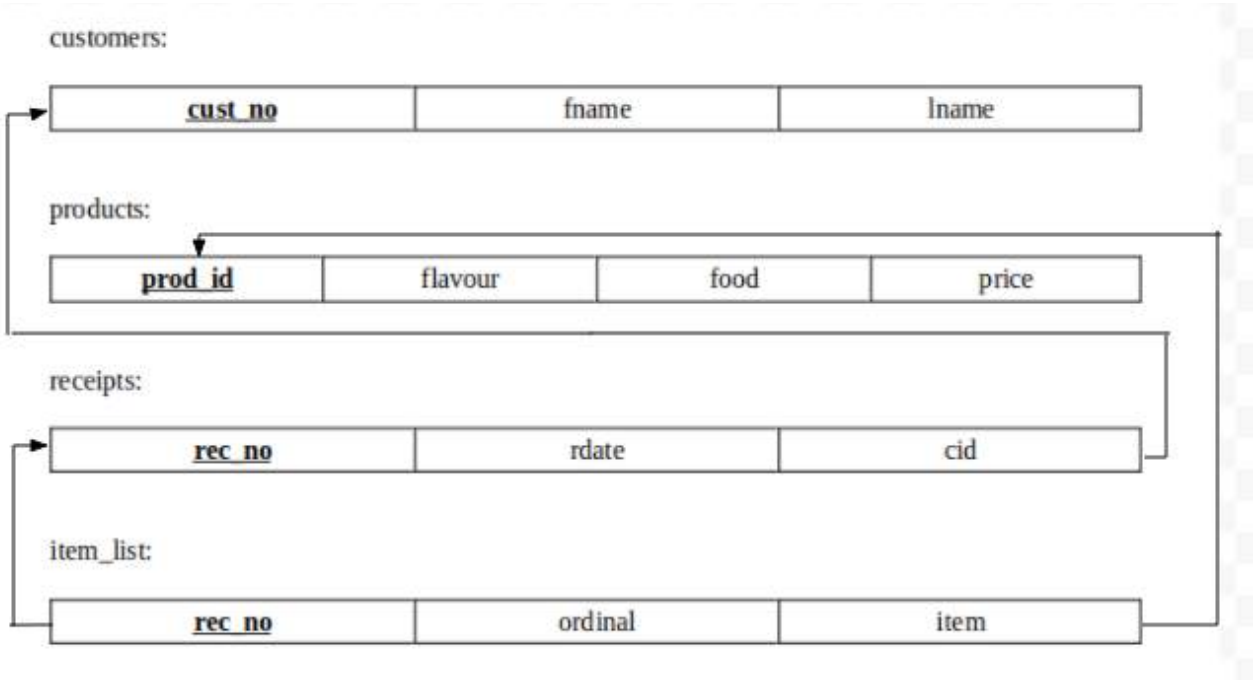
Assignment 8 – Exception Handling

Validation:



S. No.	Date	Title	Page No.	Teacher's Sign / Remarks
1.	10/03/2022	A1: DDL Commands	7/10	
2.	17/03/2022	A2: DML Commands	8/10	
3.	07/04/2022	A3: Joins and Subqueries	9/10	
4.	21/04/2022	A4: Views	10/10	
–	23/04/2022	LABTEST : A1,2,3	15/15	
5.	28/04/2022	A5: PL/SQL	10/10	
6.	12/05/2022	A6: PL/SQL	10/10	
7.	26/05/2022	A7: Triggers	9/10	
8.	26/05/2022	A8: Exception handling	10/10	

Schema diagram:



Data file:

```
SQL> @C:/Krithika/DBL/a8data.sql;
SQL> REM Population of Bakery Database

SQL> drop table item_list;
Table dropped.

SQL> drop table receipts;
Table dropped.

SQL> drop table products;
Table dropped.

SQL> drop table customers;
Table dropped.

SQL>
SQL> create table customers(
  2         cust_no number(2) constraint c_pk primary key,
  3         lname varchar2(20),
  4         fname varchar2(20)
  5         );

Table created.

SQL>
SQL> insert into customers values(1, 'LOGAN', 'JULIET');
1 row created.
.
.
.
SQL> insert into customers values(21, 'JOHN', 'DAVID');
1 row created.

SQL> create table products(
  2         prod_id varchar2(20) constraint prod_pk primary key,
  3         flavour varchar2(20),
  4         food varchar2(20),
  5         price number
  6         );

Table created.

SQL>
SQL> insert into products values('20-BC-C-10','Chocolate','Cake',8.95);
1 row created.
.
.
.
```

```
SQL> insert into products values('51-BLU','Blueberry','Danish',1.15);
1 row created.

SQL> create table receipts(
  2         rec_no number(5) constraint rec_pk primary key,
  3         rdate date,
  4         cid number(2) constraint rec_fk references customers(cust_no)
  5         );

Table created.

SQL>
SQL> INSERT INTO Receipts values(18129, '28-Oct-2007', 15);
1 row created.
.
.
.
SQL> INSERT INTO Receipts values(34378, '23-Oct-2007', 6);
1 row created.

SQL> create table item_list(
  2         rec_no number(5) constraint it_fk1 references receipts(rec_no),
  3         ordinal number(2),
  4         item varchar2(20) constraint it_fk2 references products(prod_id),
  5         constraint item_pk primary key(rec_no,ordinal)
  6         );

Table created.

SQL>
SQL> insert into item_list values(18129, 1,  '70-TU');
1 row created.
.
.
.
SQL> insert into item_list values(34378, 2,  '45-VA');
1 row created.

SQL>
SQL> REM *** End of database population ***

SQL>
SQL> REM ******************* END OF DATA FILE ************************
```

Script file:

```
SQL> @z:/a8exception.sql;
SQL> REM Assignment 8
SQL>
SQL> REM ------------------------------------------------------------------------
> REM *** ASSIGNMENT QUESTIONS ***
SQL> REM ------------------------------------------------------------------------
> REM Consider the schema used in Assignment 3.
SQL>
SQL>
SQL> REM **_____Write a PL/SQL block to handle the following exceptions:_____**
SQL>
SQL> REM 1. For the given receipt number, if there are no rows then display as "No order with the
SQL> REM given receipt <number>". If the receipt contains more than one item, display as
SQL> REM "The given receipt <number> contains more than one item". If the receipt contains
SQL> REM single item, display as "The given receipt <number> contains exactly one item". Use
SQL> REM predefined exception handling.
SQL>
SQL> create or replace procedure numitems (rno in receipts.rec_no%type) is
  2  numrows number;
  3  begin
  4          select count(item) into numrows from item_list where rec_no = rno;
  5          if (numrows = 0) then
  6                  dbms_output.put_line('No order with the given receipt number '||rno);
  7          elsif (numrows = 1) then
  8                  dbms_output.put_line('The given receipt '||rno||' contains exactly one item.');
  9          elsif (numrows > 1) then
 10                  dbms_output.put_line('The given receipt '||rno||' contains more than one item');
 11          end if;
 12  EXCEPTION
 13  when no_data_found then
 14          dbms_output.put_line('Invalid receipt number');
 15  end;
 16  /

Procedure created.

SQL>
SQL> declare
  2  rid receipts.rec_no%type;
  3
  4  begin
  5
  6  rid := &receipt;
  7  numitems(rid);
  8
  9  end;
 10  /
Enter value for receipt: 34378
```

```
old   6: rid := &receipt;
new   6: rid := 34378;
The given receipt 34378 contains more than one item

PL/SQL procedure successfully completed.

SQL> /
Enter value for receipt: 17947
old   6: rid := &receipt;
new   6: rid := 17947;
The given receipt 17947 contains exactly one item.

PL/SQL procedure successfully completed.

SQL> /
Enter value for receipt: 88888
old   6: rid := &receipt;
new   6: rid := 88888;
No order with the given receipt number 88888

PL/SQL procedure successfully completed.

SQL>
SQL> REM Validations:
SQL>
SQL> select rec_no, count(item) as numrows from item_list group by rec_no having rec_no =
34378 or rec_no = 17947 or rec_no = 88888;

    REC_NO    NUMROWS
---------- ----------
     34378          2
     17947          1

SQL>
SQL>
SQL> REM 2. While inserting the receipt details, raise an exception when the receipt date is greater
than the current date.
SQL>
SQL> create or replace trigger check_rdate
  2  BEFORE INSERT ON receipts FOR EACH ROW
  3  declare
  4          cursor c1 is select * from receipts where rdate > (select sysdate from dual);
  5          record c1%rowtype;
  6  begin
  7          open c1;
  8          fetch c1 into record;
  9          if c1%found then
 10                  raise_application_error(-20000,'Receipt date cannot be after current date!');
 11          end if;
 12  end;
```

```
 13  /

Trigger created.

SQL>
SQL> REM Validations:
SQL>
SQL> insert into receipts values (12299,'01-JUL-2022',11);
insert into receipts values (12299,'01-JUL-2022',11)
          *
ERROR at line 1:
ORA-20000: Receipt date cannot be after current date!
ORA-06512: at "1057.CHECK_RDATE", line 8
ORA-04088: error during execution of trigger '1057.CHECK_RDATE'


SQL>
SQL>
SQL>
SQL> REM ************************* END OF FILE ******************************
```

	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:

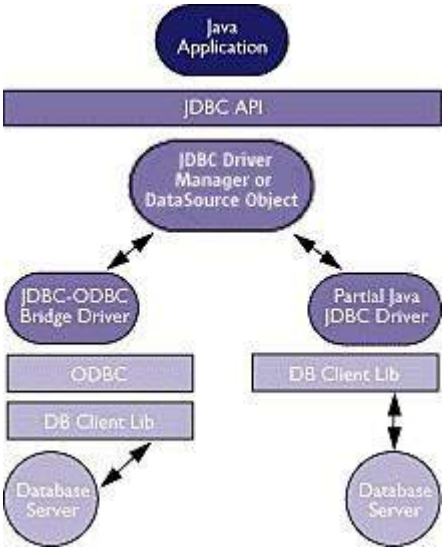
JDBC technology drivers fit into one of four categories:

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.

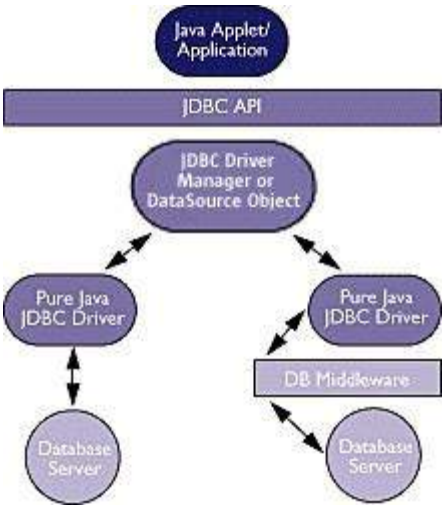


Right side, Type 3: Pure Java Driver for Database Middleware

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

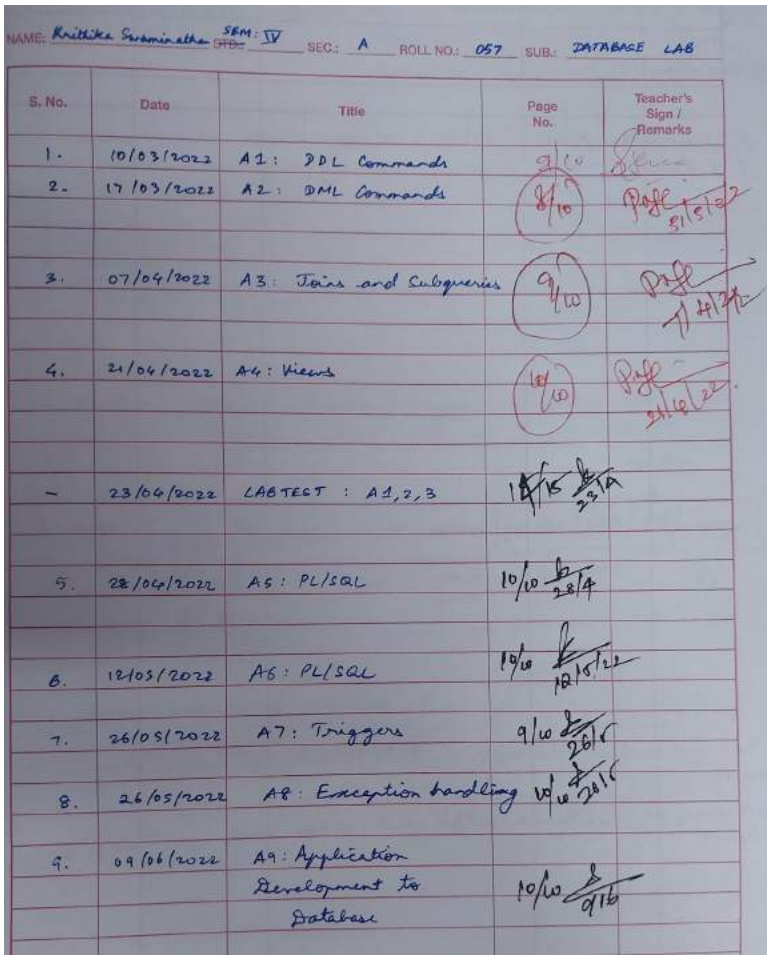
Call the procedure/function from the application by passing appropriate parameter(s) and update the corresponding record.

What you have to submit:

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

Assignment 9 – Database Programming using JDBC

Validation:



NAME: Krithika Swaminathan SEM: IV SEC.: A ROLL NO.: 057 SUB.: DATABASE LAB

S. No.	Date	Title	Page No.	Teacher's Sign / Remarks
1.	10/03/2022	A1: DDL Commands	9/10	
2.	17/03/2022	A2: DML Commands	8/10	21/5/22
3.	07/04/2022	A3: Joins and Subqueries	9/10	21/4/22
4.	21/04/2022	A4: Views	10/10	21/4/22
–	23/04/2022	LABTEST : A1,2,3	18/15	23/4
5.	28/04/2022	A5: PL/SQL	10/10	28/4
6.	12/05/2022	A6: PL/SQL	10/10	12/5/22
7.	26/05/2022	A7: Triggers	9/10	26/5
8.	26/05/2022	A8: Exception handling	10/10	26/5
9.	09/06/2022	A9: Application Development to Database	10/10	9/6

Schema diagram:

Emp_Payroll:

eid	ename	dob	sex	desg	basic	da	hra	pf	mc	gross	tot_deduc	net_pay

Script file:

```
SQL> @z:/a9conn.sql;
SQL> REM Assignment 9
SQL>
SQL> REM ------------------------------------------------------------------------
> REM *** ASSIGNMENT QUESTIONS ***
SQL> REM ------------------------------------------------------------------------
>
SQL> REM 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:
SQL> REM Dearness Allowance [DA] = 60%
SQL> REM House Rent Allowance [HRA]=11%
SQL> REM Provident Fund [PF] = 4%
SQL> REM Mediclaim [MC] = 3%
SQL> REM Gross = Basic + DA + HRA
SQL> REM Total Deduction = PF + MC
SQL> REM Net Pay = Gross – Total Deduction
SQL> REM Call the procedure/function from the application by passing appropriate parameter(s)
and update the corresponding record.

SQL>
SQL> drop table emp_payroll;

Table dropped.

SQL>
SQL> create table emp_payroll(
  2         eid number(5) constraint e_pk primary key,
  3         ename varchar2(20),
  4         dob date,
  5         sex varchar2(10),
  6         desg varchar2(30),
  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>
SQL> desc emp_payroll;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 EID                                       NOT NULL NUMBER(5)
```

```
 ENAME                                     VARCHAR2(20)
 DOB                                     DATE
 SEX                                     VARCHAR2(10)
 DESG                                      VARCHAR2(30)
 BASIC                                     NUMBER
 DA                                     NUMBER
 HRA                                     NUMBER
 PF                                     NUMBER
 MC                                     NUMBER
 GROSS                                     NUMBER
 TOT_DEDUC                                     NUMBER
 NET_PAY                                     NUMBER

SQL>
SQL> select * from emp_payroll;

no rows selected

SQL>
SQL> create or replace procedure calcpay
  2  (eidip in emp_payroll.eid%type) as
  3          b emp_payroll.basic%type;
  4          daip emp_payroll.da%type;
  5          hraip emp_payroll.hra%type;
  6          pfip emp_payroll.pf%type;
  7          mcip emp_payroll.mc%type;
  8          grossip emp_payroll.gross%type;
  9          totded emp_payroll.tot_deduc%type;
 10          netpay emp_payroll.net_pay%type;
 11  begin
 12          select basic into b from emp_payroll where eid = eidip;
 13          update emp_payroll set da = 0.6*b, hra = 0.11*b, pf = 0.04*b, mc = 0.03*b where eid  =
eidip;
 14          select da, hra, pf, mc into daip, hraip, pfip, mcip from emp_payroll where eid = eidip;
 15          totded:= pfip+mcip;
 16          grossip:=b+daip+hraip;
 17          netpay:=grossip-totded;
 18          update emp_payroll set tot_deduc = totded, gross = grossip, net_pay = netpay where eid
= eidip;
 19  end;
 20  /

Procedure created.

SQL>
SQL>
SQL>
SQL> REM ************************END OF SQL FILE ****************************
```

Connectivity - Script file (Java):

```
package jdbc;

/*@author 1057*/

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

public class JDBC extends javax.swing.JFrame {

    /**
     * Creates new form JDBC
     */
    Connection con;
    Statement st;
    PreparedStatement ps;
    ResultSet rs;
    public JDBC() {
        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","1057","1057");
                JOptionPane.showMessageDialog(this,"Connected to Oracle database!");
            }
            catch (SQLException ex) {
                Logger.getLogger(JDBC.class.getName()).log(Level.SEVERE, null, ex);
                JOptionPane.showMessageDialog(this,ex.getMessage());
            }
        }
        catch(ClassNotFoundException ex){
            Logger.getLogger(JDBC.class.getName()).log(Level.SEVERE, null, ex);
            JOptionPane.showMessageDialog(this,ex.getMessage());
        }
    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jLabel1 = new javax.swing.JLabel();
        jTextField2 = new javax.swing.JTextField();
        jButton1 = new javax.swing.JButton();
```

```
jLabel2 = new javax.swing.JLabel();
jLabel3 = new javax.swing.JLabel();
jLabel4 = new javax.swing.JLabel();
jLabel5 = new javax.swing.JLabel();
jLabel6 = new javax.swing.JLabel();
jLabel7 = new javax.swing.JLabel();
empid = new javax.swing.JTextField();
name = new javax.swing.JTextField();
dob = new javax.swing.JTextField();
sex = new javax.swing.JTextField();
sal = new javax.swing.JTextField();
update = new javax.swing.JButton();
insert = new javax.swing.JButton();
delete = new javax.swing.JButton();
search = new javax.swing.JButton();
clear = new javax.swing.JButton();
exit = new javax.swing.JButton();
jLabel8 = new javax.swing.JLabel();
desg = new javax.swing.JTextField();
calc_pay = new javax.swing.JButton();
jLabel9 = new javax.swing.JLabel();
netpay = new javax.swing.JTextField();

jLabel1.setText("jLabel1");

jTextField2.setText("jTextField1");

jButton1.setText("jButton1");

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

jLabel2.setText("Employee ID: ");

jLabel3.setText("EMPLOYEE DATABASE");

jLabel4.setText("Name: ");

jLabel5.setText("Date of birth: ");
jLabel5.setToolTipText("");

jLabel6.setText("Sex: ");

jLabel7.setText("Basic salary: ");

empid.setCursor(new java.awt.Cursor(java.awt.Cursor.TEXT_CURSOR));
empid.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        empidActionPerformed(evt);
    }
});
```

```
name.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        nameActionPerformed(evt);
    }
});

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

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

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

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

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

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

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

```
        }
    });

    clear.setText("Clear");
    clear.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            clearActionPerformed(evt);
        }
    });

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

    jLabel8.setText("Designation: ");

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

    calc_pay.setText("Calculate Net Pay");
    calc_pay.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            calc_payActionPerformed(evt);
        }
    });

    jLabel9.setText("Net pay: ");

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

    javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
    getContentPane().setLayout(layout);
    layout.setHorizontalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addComponent(jLabel3)
            .addGap(140, 140, 140))
        .addGroup(layout.createSequentialGroup()
```

```
            .addGap(96, 96, 96)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jLabel4)
                .addComponent(jLabel2)
                .addComponent(jLabel5)
                .addComponent(jLabel6)
                .addComponent(jLabel7)
                .addComponent(jLabel8))
            .addGap(80, 80, 80)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(sex)
                .addComponent(sal)
                .addComponent(empid, javax.swing.GroupLayout.PREFERRED_SIZE, 122,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(name)
                .addComponent(dob)
                .addComponent(desg))
            .addGap(45, 45, 45))
        .addGroup(layout.createSequentialGroup()
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(layout.createSequentialGroup()
                    .addGap(48, 48, 48)
                    .addComponent(insert)
                    .addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                        .addGroup(layout.createSequentialGroup()
                            .addComponent(update)
                            .addGap(26, 26, 26)
                            .addComponent(delete)
                            .addGap(28, 28, 28)
                            .addComponent(search))
                        .addGroup(layout.createSequentialGroup()
                            .addGap(10, 10, 10)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                                .addComponent(calc_pay, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                                .addGroup(layout.createSequentialGroup()
                                    .addComponent(clear)
                                    .addGap(18, 18, 18)
                                    .addComponent(exit))))))
                .addGroup(layout.createSequentialGroup()
                    .addGap(111, 111, 111)
                    .addComponent(jLabel9)
                    .addGap(18, 18, 18)
                    .addComponent(netpay, javax.swing.GroupLayout.PREFERRED_SIZE, 109,
javax.swing.GroupLayout.PREFERRED_SIZE)))
            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        );
```

```
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(18, 18, 18)
                .addComponent(jLabel3)
                .addGap(22, 22, 22)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel2)
                    .addComponent(empid, 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.BASELINE)
                    .addComponent(jLabel4)
                    .addComponent(name, 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.BASELINE)
                    .addComponent(dob, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel5))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel6)
                    .addComponent(sex, 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.BASELINE)
                    .addComponent(jLabel8)
                    .addComponent(desg, 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.BASELINE)
                    .addComponent(sal, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel7))
                .addGap(18, 18, 18)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(update)
                    .addComponent(delete)
                    .addComponent(search)
                    .addComponent(insert))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(clear)
                    .addComponent(exit))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(calc_pay)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
```

```
                .addComponent(netpay, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jLabel9))
            .addContainerGap(16, Short.MAX_VALUE))
        );

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

    private void cleartext(){
        empid.setText("");
        name.setText("");
        dob.setText("");
        sex.setText("");
        desg.setText("");
        sal.setText("");
        netpay.setText("");
        }

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

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

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

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

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

    private void deleteActionPerformed(java.awt.event.ActionEvent evt) {
        try{
            String sql = "delete from emp_payroll where eid=?";
            ps = con.prepareStatement(sql);
            ps.setString(1, empid.getText());
            ps.execute();
            JOptionPane.showMessageDialog(this, "Deleted!");
            cleartext();
            }
            catch (SQLException ex) {
            Logger.getLogger(JDBC.class.getName()).log(Level.SEVERE, null, ex);
```

```
        JOptionPane.showMessageDialog(this,ex.getMessage());
        }
}

private void insertActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        String sql = "insert into emp_payroll values(?,?,?,?,?,?,null,null,null,null,null,null,null)";
        ps = con.prepareStatement(sql);
        ps.setString(1, empid.getText());
        ps.setString(2, name.getText());
        ps.setString(3, dob.getText());
        ps.setString(4, sex.getText());
        ps.setString(5, desg.getText());
        ps.setString(6, sal.getText());
        ps.execute();
        JOptionPane.showMessageDialog(this,"Inserted!");
        cleartext();
        }
        catch (SQLException ex) {
        Logger.getLogger(JDBC.class.getName()).log(Level.SEVERE, null, ex);
        JOptionPane.showMessageDialog(this,ex.getMessage());
        }
}

private void clearActionPerformed(java.awt.event.ActionEvent evt) {
    cleartext();
}

private void updateActionPerformed(java.awt.event.ActionEvent evt) {
    try{
        String sql = "update emp_payroll set ename=?,dob=?,sex=?,desg=?,basic=? where eid=?";
        ps = con.prepareStatement(sql);
        ps.setString(6, empid.getText());
        ps.setString(1, name.getText());
        ps.setString(2, dob.getText());
        ps.setString(3, sex.getText());
        ps.setString(4, desg.getText());
        ps.setString(5, sal.getText());
        ps.execute();
        JOptionPane.showMessageDialog(this, "Updated!");
        cleartext();
        }
        catch (SQLException ex) {
        Logger.getLogger(JDBC.class.getName()).log(Level.SEVERE, null, ex);
        JOptionPane.showMessageDialog(this,ex.getMessage());
        }
}

private void exitActionPerformed(java.awt.event.ActionEvent evt) {
    System.exit(0);
```

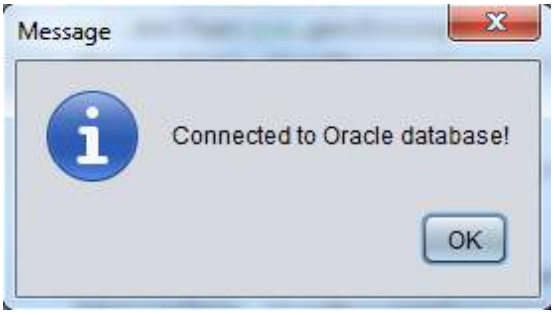
```
    }

    private void searchActionPerformed(java.awt.event.ActionEvent evt) {
        try {
            String sql = "select * from emp_payroll where eid = '"+empid.getText()+"'";
            st =
con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,ResultSet.CONCUR_UPDATABLE)
;
            rs = st.executeQuery(sql);
            if(rs.next()){
            empid.setText(rs.getString(1));
            name.setText(rs.getString(2));
            dob.setText(rs.getString(3));
            sex.setText(rs.getString(4));
            desg.setText(rs.getString(5));
            sal.setText(rs.getString(6));
            JOptionPane.showMessageDialog(this, "Record Found!");
            }
            else
            JOptionPane.showMessageDialog(this, "Record Not Found!");
            }
            catch (SQLException ex) {
            Logger.getLogger(JDBC.class.getName()).log(Level.SEVERE, null, ex);
            JOptionPane.showMessageDialog(this, ex.getMessage());
            }
    }

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

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

    private void calc_payActionPerformed(java.awt.event.ActionEvent evt) {
        try {
            // TODO add your handling code here:
            CallableStatement stmt = con.prepareCall("{call calcpay(?)}");
            stmt.setString(1, empid.getText());
            stmt.execute();
            String sql = "select net_pay from emp_payroll where eid = '"+empid.getText()+"'";
            st =
con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,ResultSet.CONCUR_UPDATABLE)
;
            rs = st.executeQuery(sql);
            if(rs.next()){
                netpay.setText(rs.getString(1));
                JOptionPane.showMessageDialog(this, "Net pay calculated!");
            }
```

```
        }
        catch (SQLException ex) {
            Logger.getLogger(JDBC.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) ">

        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(JDBC.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(JDBC.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(JDBC.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(JDBC.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        }
        //</editor-fold>

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

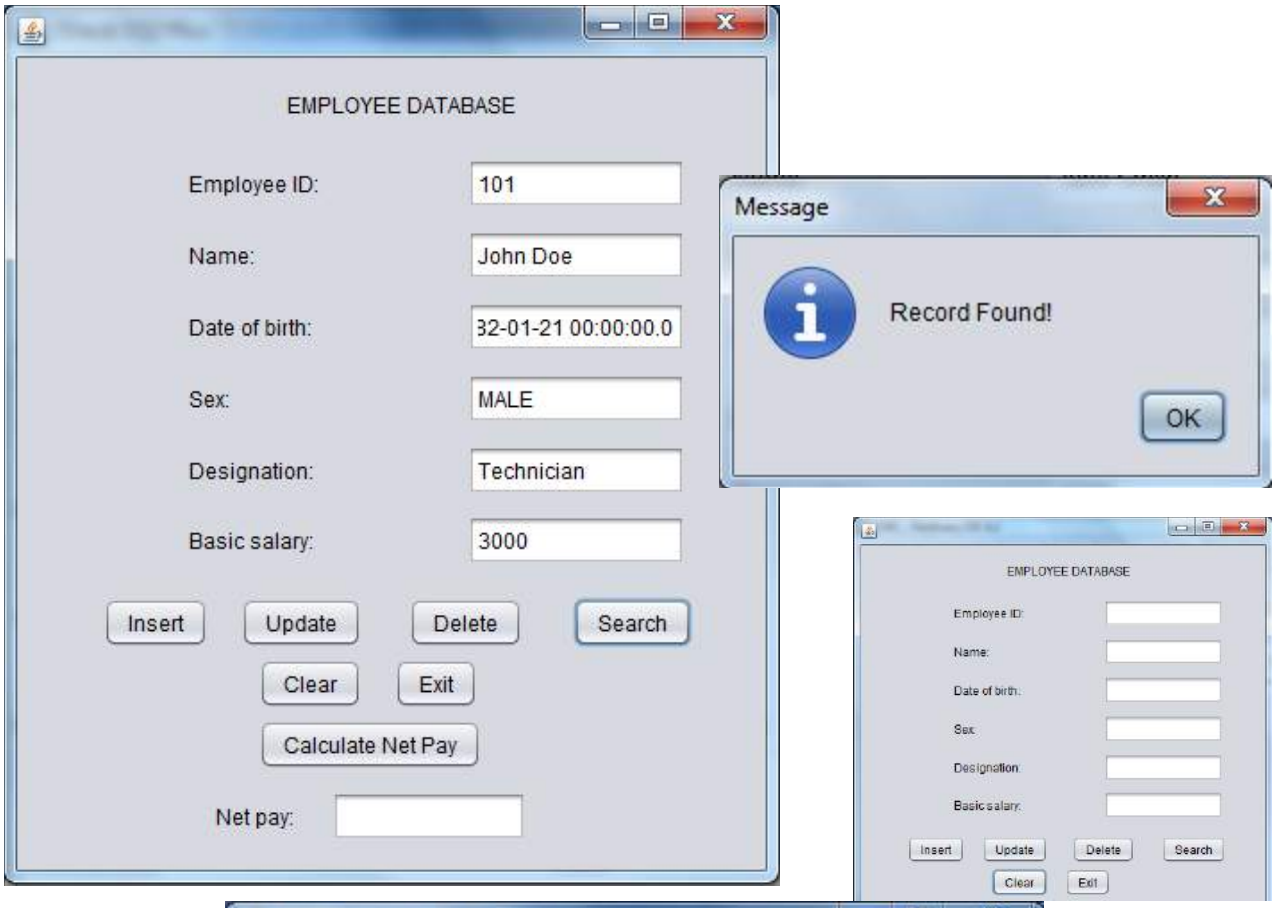
```
    // Variables declaration - do not modify
    private javax.swing.JButton calc_pay;
    private javax.swing.JButton clear;
    private javax.swing.JButton delete;
    private javax.swing.JTextField desg;
    private javax.swing.JTextField dob;
    private javax.swing.JTextField empid;
    private javax.swing.JButton exit;
    private javax.swing.JButton insert;
    private javax.swing.JButton jButton1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JLabel jLabel8;
    private javax.swing.JLabel jLabel9;
    private javax.swing.JTextField jTextField2;
    private javax.swing.JTextField name;
    private javax.swing.JTextField netpay;
    private javax.swing.JTextField sal;
    private javax.swing.JButton search;
    private javax.swing.JTextField sex;
    private javax.swing.JButton update;
    // End of variables declaration
}

//END OF JAVA FILE
```

Output:


Message
Driver Loaded!
OK


Message
Connected to Oracle database!
OK


EMPLOYEE DATABASE
Employee ID:
101
Name:
John Doe
Message
Inserted!
OK
Insert
Update
Delete
Search
Clear
Exit
Calculate Net Pay
Net pay:


EMPLOYEE DATABASE
Employee ID:
101
Name:
John Doe
Date of birth:
32-01-21 00:00:00.0
Sex:
MALE
Designation:
Technician
Basic salary:
3000
Insert
Update
Delete
Search
Clear
Exit
Calculate Net Pay
Net pay:
Message
Record Found!
OK


EMPLOYEE DATABASE
Employee ID:
101
Name:
Message
Net pay calculated!
OK
Insert
Update
Delete
Search
Clear
Exit
Calculate Net Pay
Net pay:
1640


EMPLOYEE DATABASE
Employee ID:
101
Name:
Message
Deleted!
OK
Insert
Update
Delete
Search
Clear
Exit
Calculate Net Pay
Net pay:


EMPLOYEE DATABASE
Employee ID:
101
Name:
Message
Record Not Found!
OK
Insert
Update
Delete
Search
Clear
Exit
Calculate Net Pay
Net pay: