Database Systems Project 2 Documentation

By –

Yashas Gopal Jonnada

Gunjan Mishra

Punit Paresh Jagani

"We have done this assignment completely on our own. We have not copied it, nor have we given our solution to anyone else. We understand that if we are involved in plagiarism or cheating we will have to sign an official form that we have cheated and that this form will be stored in our official university records. We also understand that we will receive a grade of 0 for the involved assignment and our grades will be reduced by one level (e.g., from A to A- or from B+ to B) for our first offense, and that we will receive a grade of "F" for the course for any additional offense of any kind."

Description –

All objects used in this project

a) Sequences –

i) seqpur# -

This sequence is used to generate unique values for purchases table in pur# column starting from 100001 and incrementing by 1.

ii) seqlog# -

This sequence is used to generate unique values for logs table in log# column starting from 1001 and incrementing by 1.

b) Packages –

i) check_pkg –

This package holds all the procedures, functions necessary to enable functionalities of this project in the oracle database.

❖ **The functions in this package are –**

i) show_employees –

This function returns a reference to a cursor with all the employees in the employees table of type cursor.

ii) show_customers –

This function returns a reference to a cursor with all the customers in the customers table of type cursor.

iii) show_products –

This function returns a reference to a cursor with all the products in the products table of type cursor.

iv) show_purchases –

This function returns a reference to a cursor with all the purchases in the purchases table of type cursor.

v) no_of_customers –

This function takes product id as an input and returns the number of customers who have a bought the respective product.

❖ **The procedures in this package are –**

i) Purchases_made –
This procedure takes customer id as an input and gives the following information through a out variable of type sys_refcursor –

Name – The name of the customer
Pid – The products this customer has bought
pur_date – The date of a particular purchase
qty – Number of items purchased
unit_price – Unit price of the purchased product
total – Total amount spent by the customer

ii) add_customer –
This procedure takes
P_cid - customer id
P_name – name of the customer
P_telephone# - customer telephone number
as an input and inserts them into the customers table. Additionally, this procedure also inserts number of visits made by the customer and the last visit date of the customer in the customers table.

iii) add_purchase
This procedure takes
p_eid - employee id
p_pid - product id
p_cid - customer id
p_pur_qty - purchase quantity
p_pur_unit_price - purchase unit price
and inserts in the purchases table. Additionally, also inserts
pur# - purchase number generated with the help of the seqpur# sequence
total - the total amount spent by the customer is calculated with the help of the input of purchase quantity and purchase unit price
pur_date - the date of the purchase
saving - The total savings customer has made is calculated with the help of the input of p_pur_qty, p_pur_unit_price and regular_price from the products table into the purchases table.

c) Triggers –

i) customer_update_trigger – fires after an insert on purchases table
- checks if the latest purchase date is newer than the last_visit_date of that particular customer and updates visits_made of the customer only if the check is true.
- Updates the qoh in product table by subtracting the current purchase qty from products.qoh
- Checks if the new qoh in products is less than qoh_threshold and if yes, then adds 10 to the qoh_threshold and assigns the value to the qoh in products.

ii) customer_insert_trigger – fires after an insert on customers table

- inserts a row in logs table as
  log# - unique value generated by the seqlog# sequence
  user_name – the user who has inserted a row in the customers table
  operation – Insert
  op_time – the date of insertion
  table_name – customers
  tuple_pkey – new cid

iii) customer_updt_lvd_trigger – fires after an update on customers table

- inserts a row in logs table wherever the customer's last_visit_date is updated
  log# - unique value generated by the seqlog# sequence
  user_name – the user who has inserted a row in the customers table
  operation – update
  op_time – the date of update
  table_name – customers
  tuple_pkey – the cid in which the change has taken place

iv) customer_updt_vm_trigger – fires after an update on customers table

- inserts a row in logs table wherever the customer's visits_made is updated
  log# - unique value generated by the seqlog# sequence
  user_name – the user who has inserted a row in the customers table
  operation – update
  op_time – the date of update
  table_name – customers
  tuple_pkey – the cid in which the change has taken place

v) purchases_insert_trigger – fires after an insert on purchases table

- inserts a row in logs table whenever a new purchase is added to the purchases table
  log# - unique value generated by the seqlog# sequence
  user_name – the user who has inserted a row in the purchases table
  operation – insert
  op_time – the date of insert
  table_name – purchases
  tuple_pkey – the pur# which has been added in the purchases table

vi) products_update_trigger - fires after an update on the products table

- inserts a row in the logs table wherever the qoh of any product is updated
  log# - unique value generated by the seqlog# sequence
  user_name – the user who has updated the qoh of the product table
  operation – update
  op_time – the date of update
  table_name – products
  tuple_pkey – the qoh which has been updated in the products table

TEAM REPORT

Meetings –

May 1, Saturday @ 1:00 pm - 4:00 pm

- Understanding the project and plotting down the requirements

May 2, Sunday @ 1:00 pm - 6:00 pm

- Division of responsibilities to achieve the required goal within the deadline

May 5, Wednesday @ 9:00 am - 12:00 pm

- Tried to remove errors and make a clean code until the 1st 5 questions

May 7, Friday @ 9:00 am – 12 pm

- Ran a test with the frontend code developed in plsql to call the procedures and functions

May 9, Sunday @ 9:00 am – 12:00 pm

- Completed question 6 testing to make sure all the question 6 requirements are satisfied

May 11, Tuesday @ 12:00 pm – 3:00 pm

- Implemented question 7 triggers and resolved the errors.

May 12, Wednesday @ 9:00 am – 3:00 pm

- Shared our perspectives on the interface and tried to understand the demo codes with the example data table.

May 13, Thursday @ 12 :00 pm – 6:00 pm

- Completed interface programs until question 5

May 14, Friday @ 9:00 am – 12 pm

- Completed interface program for question 6

May 14, Friday @ 6:00pm – 8:00 pm

- Collated all the interfaces programs into one single code and achieved the menu driven functionality to make our project interactive.

May 15, Saturday @ 9:00 am – 6 :00 pm

- Ran final execution and checked all the project requirements, Completed the documentation for the project and congratulated each other on the completion.

May 16 –

- The date of submission

THE PROJECTED PLAN TO COMPLETE THE PROJECT

- 1st week from the project start date –

  Lay the foundation with the necessary functions, procedures in the database and remove all the errors to make it fully functional.

- 2nd week from the project start date –

  Build on the foundation to implement triggers and do the exception handling to meet the project requirements.

- 3rd week from the project start date –

  Complete the interface and documentation part of the project.

RESPONSIBILITIES –

1) Yashas Gopal Jonnada – Build the foundation and Backend database coding until question 6.
2) Gunjan Mishra – All triggers implementation, error and exception handling and package consolidation.
3) Punit Paresh Jagani – Trial runs to find possible leaks in the entire code and report bugs in the code. Ran multiple custom defined test cases to crack the code and find the vulnerabilities.
4) Three of us have contributed to the interface part of the project.

SELF ASSESMENT OF THE TEAM WORK

We Worked really well together .

** KINDLY REFER TO THE NEXT PAGE**

JAVA CODE

```
package database;

import java.sql.*;

import oracle.jdbc.*;

import java.math.*;

import java.io.*;

import java.awt.*;

import oracle.jdbc.pool.OracleDataSource;

import java.util.Scanner;


public class menudriven {


  public static void main(String args[]) throws SQLException {

    try {


        //connecting to database

        OracleDataSource ds = new oracle.jdbc.pool.OracleDataSource();

        ds.setURL("jdbc:oracle:thin:@castor.cc.binghamton.edu:1521:acad111");


        //read input from user for username and password

        BufferedReader input;

        String username;

        String password;


        //prompt for username

        input = new BufferedReader(new InputStreamReader(System.in));

        System.out.print("Please enter your pods Username for oracle sqlplus connection:");

        username = input.readLine();


        //prompt for password

        System.out.print("Please enter your sqlplus Password for oracle sqlplus connection:");

        password = input.readLine();

        Connection conn = ds.getConnection(username, password);


        //flag for continuous loop to ask the user on entering options

        int option = 0;
        while (option != 6) {
          System.out.print("\nEnter '1' to show all the tables\n");
          System.out.print("Enter '2' to check the purchases made by a specific customer\n");
          System.out.print("Enter '3' to check the number of customers who have purchased a specific product\n");
          System.out.print("Enter '4' to add a new customer into customer table\n");
          System.out.print("Enter '5' to add a new purchases into purchase table  :: Warning - this action will result changes in
multiple tables\n");
          System.out.print("Enter '6' to exit\n");

          Scanner action = new Scanner(System.in);
          option = action.nextInt();

          if (option == 1) {

            //Prepare to call stored procedure show_employees:
            CallableStatement em = conn.prepareCall("begin ? := check_pkg.show_employees(); end;");

            //register the out parameter (the first parameter)
            em.registerOutParameter(1, OracleTypes.CURSOR);
            // execute and retrieve the result set
            em.execute();
            ResultSet emrs = (ResultSet) em.getObject(1);
            // print the results
            System.out.println();
            System.out.println(" ALL EMPLOYEES  \n");

            int testEmployeeCount = 0;

            while (emrs.next()) {
              System.out.println(emrs.getString(1) + "\t" + emrs.getString(2) + "\t" + emrs.getString(3) + emrs.getString(4));
              testEmployeeCount++;
            }

            if (testEmployeeCount == 0) {
              System.out.println("No data found in employees table");
            }
```

```
//Prepare to call stored procedure show_customers:
CallableStatement cu = conn.prepareCall("begin ? := check_pkg.show_customers(); end;");

//register the out parameter (the first parameter)
cu.registerOutParameter(1, OracleTypes.CURSOR);
// execute and retrieve the result set
cu.execute();
ResultSet curs = (ResultSet) cu.getObject(1);
// print the results
System.out.println();
System.out.println(" ALL CUSTOMERS  \n");

int testCustomerCount = 0;
while (curs.next()) {
  System.out.println(curs.getString(1) + "\t" + curs.getString(2) + "\t" + curs.getString(3) + "\t" + curs.getInt(4) +
    curs.getDate(5));
  testCustomerCount++;
}

if (testCustomerCount == 0) {
  System.out.println("No data found in customers table");
}
//Prepare to call stored procedure show_products:
CallableStatement pr = conn.prepareCall("begin ? := check_pkg.show_products(); end;");

//register the out parameter (the first parameter)
pr.registerOutParameter(1, OracleTypes.CURSOR);
// execute and retrieve the result set
pr.execute();
ResultSet prrs = (ResultSet) pr.getObject(1);
// print the results
System.out.println();
System.out.println(" ALL PRODUCTS  \n");
int testProductCount = 0;
while (prrs.next()) {
  System.out.println(prrs.getString(1) + "\t" + prrs.getString(2) + "\t" + prrs.getInt(3) + "\t" + prrs.getInt(4) +
    "\t" + prrs.getDouble(5) + "\t" + prrs.getDouble(6));
```

```
                testProductCount++;
            }
            if (testProductCount == 0) {
                System.out.println("No data found in products table");
            }

            //Prepare to call stored procedure show_purchases:
            CallableStatement pu = conn.prepareCall("begin ? := check_pkg.show_purchases(); end;");

            //register the out parameter (the first parameter)
            pu.registerOutParameter(1, OracleTypes.CURSOR);
            // execute and retrieve the result set
            pu.execute();
            ResultSet purs = (ResultSet) pu.getObject(1);
            // print the results
            System.out.println();
            System.out.println(" ALL PURCHASES  \n");
            int testPurchasesCount = 0;
            while (purs.next()) {
                System.out.println(purs.getString(1) + "\t" + purs.getString(2) + "\t" + purs.getString(3) + "\t" + purs.getString(4) +
                    "\t" + purs.getDate(5) + "\t" + purs.getInt(6) + "\t" + purs.getDouble(7) + "\t" + purs.getDouble(8) + "\t" +
purs.getDouble(9));
                testPurchasesCount++;
            }
            if (testPurchasesCount == 0) {
                System.out.println("No data found in products table");
            }
            //close the result set, statement, and the connection
            em.close();
            emrs.close();
            cu.close();
            curs.close();
            pr.close();
            prrs.close();
            pu.close();
            purs.close();
        }
```

```
if (option == 2) {

   // Input cid from keyboard
   BufferedReader input_cid;
   String cid;
   input_cid = new BufferedReader(new InputStreamReader(System.in));
   System.out.print("Please Enter customer CID:");
   cid = input_cid.readLine();

   //Prepare to call stored procedure purchases_made:
   CallableStatement cs = conn.prepareCall("begin check_pkg.purchases_made(:1, :2); end;");

   //set the in parameter (the first parameter)
   cs.setString(1, cid);

   //register the out parameter (the first parameter)
   cs.registerOutParameter(2, OracleTypes.CURSOR);

   // execute and retrieve the result set
   cs.execute();
   ResultSet rs = (ResultSet) cs.getObject(2);
   String nameCheck = "unavailable";
   // print the results
   while (rs.next()) {
      nameCheck = rs.getString(1);
      System.out.println(rs.getString(1) + "\t" + rs.getString(2) + "\t" + rs.getDate(3) + "\t" +
         rs.getInt(4) + "\t" + rs.getDouble(5) + "\t" + rs.getDouble(6));
   }
   if (nameCheck == "unavailable") {
      System.out.println("\nThe customer Id you have entered does not exist in the customers table");
   }
   //close the result set, statement, and the connection
   rs.close();
   cs.close();

}
```

```
if (option == 3) {

    //Prepare to call stored procedure:
    CallableStatement cs = conn.prepareCall("begin ? := check_pkg.no_of_customers(?); end;");

    //read pid input from the keyboard
    BufferedReader input_pid;
    String pid;
    input_pid = new BufferedReader(new InputStreamReader(System.in));
    System.out.print("Please Enter Product ID:");
    pid = input_pid.readLine();

    //set the in parameter (the first parameter)
    cs.setString(2, pid);

    //register the out parameter (the first parameter)
    cs.registerOutParameter(1, java.sql.Types.INTEGER);

    // execute and retrieve the result set
    cs.execute();
    int purchase_count;
    //ResultSet rs = (ResultSet)cs.getObject(1);
    purchase_count = cs.getInt(1);

    //print the number of customers who have purchased a specific product
    if (purchase_count == 0) {
        System.out.println("\nThe product Id you have entered does not exist in the products table");
    } else {
        System.out.println("\nThe number of customers who have puchased product " + pid + " are:" + purchase_count +
"\n");
    }

    //close the result set, statement, and the connection
    cs.close();

}
```

```
if (option == 4) {
   // Query
   Statement stmt = conn.createStatement();

   // Save result
   ResultSet rset;
   rset = stmt.executeQuery("SELECT * FROM customers");

   // Print the customers table before insertion
   while (rset.next()) {

      System.out.print(rset.getString(1) + "  " + "\t");
      System.out.print(rset.getString(2) + "  " + "\t");
      System.out.print(rset.getString(3) + "  " + "\t");
      System.out.print(rset.getInt(4) + "  " + "\t");
      System.out.print(rset.getString(5) + "  " + "\n");
   }



   //Input customer_id from the keyboard
   BufferedReader inputCustId;
   inputCustId = new BufferedReader(new InputStreamReader(System.in));
   String customer_id;
   System.out.print("Please enter Customer ID:");
   customer_id = inputCustId.readLine();

   //Input customer name from the keyboard
   BufferedReader inputCustName;
   inputCustName = new BufferedReader(new InputStreamReader(System.in));
   String name;
   System.out.print("Please enter Customer Name:");
   name = inputCustName.readLine();

   //Input customer telephone# from the keyboard
   BufferedReader inputCustTelephone;
   inputCustTelephone = new BufferedReader(new InputStreamReader(System.in));
```

```
    String telephone;
    System.out.print("Please enter telephone:");
    telephone = inputCustTelephone.readLine();

    //Prepare to call stored procedure add_customer:
    CallableStatement insert = conn.prepareCall("Begin check_pkg.add_customer(:1,:2,:3); end;");

    // set the input for the procedure
    insert.setString(1, customer_id);
    insert.setString(2, name);
    insert.setString(3, telephone);

    insert.executeQuery();

    //Query again.
    rset = stmt.executeQuery("SELECT * FROM customers");

    // Print the customers table after insertion of a new row
    while (rset.next()) {

      System.out.print(rset.getString(1) + "  " + "\t");
      System.out.print(rset.getString(2) + "  " + "\t");
      System.out.print(rset.getString(3) + "  " + "\t");
      System.out.print(rset.getInt(4) + "  " + "\t");
      System.out.print(rset.getString(5) + "  " + "\n");
    }

    //close the result set, statement, and the connection
    rset.close();
    stmt.close();
  }

  if (option == 5) {

    // Query
    Statement stmt = conn.createStatement(ResultSet.TYPE_FORWARD_ONLY, ResultSet.CONCUR_UPDATABLE);
```

```
// Save result
//for products table
ResultSet rset;
//for purchases table
ResultSet rs;
//for customers table
ResultSet rs1;

rset = stmt.executeQuery("SELECT * FROM products");
System.out.println("\nPrevious details in Products table\n");
// Print previous products table
while (rset.next()) {

   System.out.print(rset.getString(1) + "  " + "\t");
   System.out.print(rset.getString(2) + "  " + "\t");
   System.out.print(rset.getInt(3) + "  " + "\t");
   System.out.print(rset.getInt(4) + "  " + "\t");
   System.out.print(rset.getDouble(5) + "  " + "\t");
   System.out.print(rset.getDouble(6) + "  " + "\n");
}

rs1 = stmt.executeQuery("SELECT * FROM customers");
System.out.println("\nPrevious details in customers table\n");
// Print previous customers table
while (rs1.next()) {
   System.out.print(rs1.getString(1) + "  " + "\t");
   System.out.print(rs1.getString(2) + "  " + "\t");
   System.out.print(rs1.getString(3) + "  " + "\t");
   System.out.print(rs1.getInt(4) + "  " + "\t");
   System.out.print(rs1.getDate(5) + "  " + "\n");
}

rs = stmt.executeQuery("SELECT * FROM purchases");
System.out.println("\nPrevious details in Purchases table\n");
// Print previous purchases table
while (rs.next()) {
   System.out.print(rs.getString(1) + "  " + "\t");
```

```
    System.out.print(rs.getString(2) + "  " + "\t");
    System.out.print(rs.getString(3) + "  " + "\t");
    System.out.print(rs.getString(4) + "  " + "\t");
    System.out.print(rs.getDate(5) + "  " + "\t");
    System.out.print(rs.getInt(6) + "  " + "\t");
    System.out.print(rs.getDouble(7) + "  " + "\t");
    System.out.print(rs.getDouble(8) + "  " + "\t");
    System.out.print(rs.getDouble(9) + "  " + "\n");
}

//Input employee id from keyboard
BufferedReader input_eid;
input_eid = new BufferedReader(new InputStreamReader(System.in));
String eid;
System.out.print("Please enter Employee ID: ");
eid = input_eid.readLine();

//Input product ID from keyboard
BufferedReader input_pid;
input_pid = new BufferedReader(new InputStreamReader(System.in));
String pid;
System.out.print("Please enter Product ID: ");
pid = input_pid.readLine();

//Input customer ID from keyboard
BufferedReader input_cid;
input_cid = new BufferedReader(new InputStreamReader(System.in));
String cid;
System.out.print("Please enter Customer ID: ");
cid = input_cid.readLine();

//Input Purchase Qty from keyboard
BufferedReader input_pur_qty;
input_pur_qty = new BufferedReader(new InputStreamReader(System.in));
String pur_qty;
System.out.print("Please enter Purchase Quantity: ");
pur_qty = input_pur_qty.readLine();
```

```
//Input Purchase unit price from keyboard
BufferedReader input_pur_price;
input_pur_price = new BufferedReader(new InputStreamReader(System.in));
String pur_price;
System.out.print("Please enter Purchase Unit Price: ");
pur_price = input_pur_price.readLine();

//Prepare to call stored procedure add_purchase:
CallableStatement insert = conn.prepareCall("Begin check_pkg.add_purchase(:1,:2,:3,:4,:5); end;");

insert.setString(1, eid);
insert.setString(2, pid);
insert.setString(3, cid);
insert.setString(4, pur_qty);
insert.setString(5, pur_price);



// execute and retrieve the result set
insert.executeQuery();

//Query again.
System.out.println("\nModified details of Products table\n");
rset = stmt.executeQuery("SELECT * FROM products");

// Print products table after new changes
while (rset.next()) {
   System.out.print(rset.getString(1) + "  " + "\t");
   System.out.print(rset.getString(2) + "  " + "\t");
   System.out.print(rset.getInt(3) + "  " + "\t");
   System.out.print(rset.getInt(4) + "  " + "\t");
   System.out.print(rset.getDouble(5) + "  " + "\t");
   System.out.print(rset.getDouble(6) + "  " + "\n");
}

System.out.println("\nModified details of Customers table\n");
rs1 = stmt.executeQuery("SELECT * FROM customers");
```

```
    // Print customers table after new changes
    while (rs1.next()) {
        System.out.print(rs1.getString(1) + "  " + "\t");
        System.out.print(rs1.getString(2) + "  " + "\t");
        System.out.print(rs1.getString(3) + "  " + "\t");
        System.out.print(rs1.getInt(4) + "  " + "\t");
        System.out.print(rs1.getDate(5) + "  " + "\n");
    }

    System.out.println("\nModified details in Purchases table\n");
    rs = stmt.executeQuery("SELECT * FROM purchases");
    // Print purchases table after new changes
    while (rs.next()) {
        System.out.print(rs.getString(1) + "  " + "\t");
        System.out.print(rs.getString(2) + "  " + "\t");
        System.out.print(rs.getString(3) + "  " + "\t");
        System.out.print(rs.getString(4) + "  " + "\t");
        System.out.print(rs.getDate(5) + "  " + "\t");
        System.out.print(rs.getInt(6) + "  " + "\t");
        System.out.print(rs.getDouble(7) + "  " + "\t");
        System.out.print(rs.getDouble(8) + "  " + "\t");
        System.out.print(rs.getDouble(9) + "  " + "\n");
    }

    //close the result set, statement, and the connection
    rset.close();
    rs.close();
    rs1.close();
    stmt.close();



}
//if 6 is entered by the user then quit
if (option == 6) {
    System.out.println("\n Bye !");
    //close the connection to sqlplus
    conn.close();
```

```
            }
          }
       } catch (SQLException ex) {
          System.out.println("\n*** SQLException caught ***\n" + ex.getMessage());
       } catch (Exception e) {
          System.out.println("\n*** other Exception caught ***\n");
       }
   }
}
```

PL/SQL CODE

STEP 1:

SAVE THE BELOW CODE AS REFRESH.SQL IN YOUR HARVEY ACCOUNT

```
set serveroutput on
drop trigger customer_update_trigger;
drop trigger customer_insert_trigger;
drop trigger customer_updt_lvd_trigger;
drop trigger customer_updt_vm_trigger;
drop trigger purchases_insert_trigger;
drop trigger products_update_trigger;

drop function show_employees ;
drop function show_customers ;
drop function show_products ;
drop function show_purchases ;
drop function no_of_customers;

drop procedure purchases_made;
drop procedure add_customer;
drop procedure add_purchase;

drop table purchases;
drop table employees;
drop table customers;

drop table products;

drop table logs;


drop sequence seqpur#;

drop sequence seqlog#;


create table logs

(log# number(4) primary key,

user_name varchar2(12) not null,

operation varchar2(6) not null,

op_time date not null,

table_name varchar2(20) not null,

tuple_pkey varchar2(6));


create table employees

(eid char(3) primary key,

name varchar2(15) not null,

telephone# char(12),

email varchar2(20) unique);


create table customers

(cid char(4) primary key,

name varchar2(15),

telephone# char(12),

visits_made number(4) check (visits_made >= 1),

last_visit_date date);


create table products

(pid char(4) primary key,

name varchar2(15),
qoh number(4),

qoh_threshold number(4),

regular_price number(6,2),

discnt_rate number(3,2) check (discnt_rate in (0.0, 0.05, 0.1, 0.15, 0.2, 0.25)));


create table purchases

(pur# number(6) primary key,

eid char(3) references employees(eid),

pid char(4) references products(pid),

cid char(4) references customers(cid),

pur_date date,

qty number(5),

unit_price number(6,2),

total number(7,2),

saving number(6,2),

unique(eid, pid, cid, pur_date));


insert into employees values ('e01', 'David', '666-555-1234', 'david@rb.com');

insert into employees values ('e02', 'Peter', '777-555-2341', 'peter@rb.com');

insert into employees values ('e03', 'Susan', '888-555-3412', 'susan@rb.com');

insert into employees values ('e04', 'Anne', '666-555-4123', 'anne@rb.com');

insert into employees values ('e05', 'Mike', '444-555-4231', 'mike@rb.com');


insert into customers values ('c001', 'Kathy', '666-555-4567', 3, '30-MAR-21');

insert into customers values ('c002', 'John', '888-555-7456', 1, '08-OCT-20');

insert into customers values ('c003', 'Chris', '666-555-6745', 3, '18-FEB-21');

insert into customers values ('c004', 'Mike', '999-555-5674', 1, '15-JAN-21');

insert into customers values ('c005', 'Mike', '777-555-4657', 2, '30-AUG-20');

insert into customers values ('c006', 'Connie', '777-555-7654', 2, '16-MAR-21');

insert into customers values ('c007', 'Katie', '888-555-6574', 1, '12-DEC-20');

insert into customers values ('c008', 'Joe', '666-555-5746', 1, '14-NOV-20');
insert into products values ('p001', 'stapler', 60, 20, 9.99, 0.1);

insert into products values ('p002', 'TV', 6, 5, 249, 0.15);

insert into products values ('p003', 'camera', 15, 3, 148, 0.2);

insert into products values ('p004', 'pencil', 100, 10, 0.99, 0.0);

insert into products values ('p005', 'chair', 10, 8, 49.98, 0.2);

insert into products values ('p006', 'lamp', 10, 6, 19.95, 0.1);

insert into products values ('p007', 'tablet', 50, 10, 199, 0.1);

insert into products values ('p008', 'computer', 5, 3, 499, 0.25);

insert into products values ('p009', 'facemask', 25, 20, 18.50, 0.1);

insert into products values ('p010', 'powerbank', 20, 5, 30, 0.1);


-------------------------------

--pro2que1

create sequence seqpur#

start with 100001

increment by 1;


create sequence seqlog#

start with 1001

increment by 1;


insert into purchases values (seqpur#.nextval, 'e01', 'p002', 'c001', to_date('12-AUG-2020', 'DD-MON-YYYY'), 1, 211.65,
211.65, 37.35);

insert into purchases values (seqpur#.nextval, 'e01', 'p003', 'c001', to_date('20-DEC-2020', 'DD-MON-YYYY'), 1, 118.40,
118.40, 29.60);

insert into purchases values (seqpur#.nextval, 'e02', 'p004', 'c002', to_date('08-OCT-2020', 'DD-MON-YYYY'), 5, 0.99, 4.95,
0);

insert into purchases values (seqpur#.nextval, 'e01', 'p005', 'c003', to_date('23-NOV-2020', 'DD-MON-YYYY'), 2, 39.98,
79.96, 20);

insert into purchases values (seqpur#.nextval, 'e04', 'p007', 'c004', to_date('15-JAN-2021', 'DD-MON-YYYY'), 1, 179.10,
179.10, 19.90);
```

```
insert into purchases values (seqpur#.nextval, 'e03', 'p009', 'c001', to_date('30-MAR-2021', 'DD-MON-YYYY'), 2, 16.65,
33.30, 3.70);

insert into purchases values (seqpur#.nextval, 'e03', 'p009', 'c003', to_date('10-JAN-2021', 'DD-MON-YYYY'), 3, 16.65,
49.95, 5.55);

insert into purchases values (seqpur#.nextval, 'e03', 'p006', 'c005', to_date('16-AUG-2020', 'DD-MON-YYYY'), 1, 17.96,
17.96, 1.99);

insert into purchases values (seqpur#.nextval, 'e03', 'p001', 'c007', to_date('12-DEC-2020', 'DD-MON-YYYY'), 1, 8.99, 8.99,
1.0);

insert into purchases values (seqpur#.nextval, 'e04', 'p002', 'c006', to_date('19-OCT-2020', 'DD-MON-YYYY'), 1, 211.65,
211.65, 37.35);

insert into purchases values (seqpur#.nextval, 'e02', 'p004', 'c006', to_date('16-MAR-2021', 'DD-MON-YYYY'), 10, 0.99,
9.90, 0);

insert into purchases values (seqpur#.nextval, 'e02', 'p008', 'c003', to_date('18-FEB-2021', 'DD-MON-YYYY'), 2, 374.25,
748.50, 249.50);

insert into purchases values (seqpur#.nextval, 'e04', 'p009', 'c005', to_date('30-AUG-2020', 'DD-MON-YYYY'), 2, 16.65,
33.30, 3.70);

insert into purchases values (seqpur#.nextval, 'e03', 'p010', 'c008', to_date('14-NOV-2020', 'DD-MON-YYYY'), 3, 27, 81, 9);

--------------------------------

--pro2que6triggers

create or replace trigger customer_update_trigger

after

insert on purchases

for each row

declare

    new_cid customers.cid%type;

    new_pur_qty purchases.qty%type;

    new_pid products.pid%type;

    new_pur_date purchases.pur_date%type;

    date_temp customers.last_visit_date%type;

    qoh_temp products.qoh%type;

    qoh_threshold_temp products.qoh_threshold%type;

begin

    new_cid := :new.cid;

    new_pur_qty := :new.qty;
```

```
	new_pid := :new.pid;

	new_pur_date := :new.pur_date;


	select last_visit_date into date_temp from customers where new_cid = customers.cid;


	if trunc(date_temp) < trunc(new_pur_date) then

	update customers set customers.visits_made = visits_made + 1, customers.last_visit_date = sysdate where
customers.cid = new_cid;

	end if;



	update products set products.qoh = qoh - new_pur_qty where products.pid = new_pid;


	select qoh into qoh_temp from products where products.pid = new_pid;

	select qoh_threshold into qoh_threshold_temp from products where products.pid = new_pid;


	if qoh_temp < qoh_threshold_temp then

		dbms_output.put_line('The current qoh of the product is below the required threshold and new supply is
required.');

		update products set qoh = qoh_threshold + 10 where products.pid = new_pid;

		select qoh into qoh_temp from products where products.pid = new_pid;

		dbms_output.put_line('The new value of the qoh of the product after new supply is: '||qoh_temp);

	end if;

end;

/

show errors

-------------------------------------

--pro2que7triggers

--1

create or replace trigger customer_insert_trigger

after insert on customers

for each row
```

```
declare
    u_cid customers.cid%type;
begin
    u_cid := :new.cid;
    insert into logs(log#,user_name,operation,op_time, table_name, tuple_pkey) values
        (seqlog#.nextval, USER, 'INSERT', sysdate, 'customers', u_cid );
end;
/
show errors


--2


create or replace trigger customer_updt_lvd_trigger
after update on customers
for each row
when (new.last_visit_date > old.last_visit_date)
declare
    v_cid customers.cid%type;
begin
    v_cid := :new.cid;
    insert into logs(log#,user_name,operation,op_time, table_name, tuple_pkey) values
        (seqlog#.nextval, USER, 'UPDATE', sysdate, 'customers', v_cid );
end;
/
show errors


--3


create or replace trigger customer_updt_vm_trigger
after update on customers
for each row
```

```
when (new.visits_made > old.visits_made)
declare
    w_cid customers.cid%type;
begin
    w_cid := :new.cid;
    insert into logs(log#,user_name,operation,op_time, table_name, tuple_pkey) values
        (seqlog#.nextval, USER, 'UPDATE', sysdate, 'customers', w_cid );
end;
/
show errors


--4


create or replace trigger purchases_insert_trigger
after insert on purchases
for each row
when(new.pur# > old.pur#)
declare
    x_pur# purchases.pur#%type;
begin
    x_pur# := :new.pur#;
    insert into logs(log#,user_name,operation,op_time, table_name, tuple_pkey) values
        (seqlog#.nextval, USER, 'INSERT', sysdate, 'purchases', x_pur#);
end;
/
show errors


--5


create or replace trigger products_update_trigger
after update on products
```

```
for each row
when(new.qoh > old.qoh)
declare
      y_qoh products.qoh%type;
begin
      y_qoh := :new.qoh;
      insert into logs(log#,user_name,operation,op_time, table_name, tuple_pkey) values
            (seqlog#.nextval, USER, 'UPDATE', sysdate, 'products', y_qoh);
end;
/
show errors
```

STEP 2:

SAVE THE BELOW CODE AS PACKAGE.SQL IN YOUR HARVEY ACCOUNT

```
set serveroutput on
create or replace package check_pkg as
type ref_cursor is ref cursor;
function show_employees return ref_cursor;
function show_customers return ref_cursor;
function show_products return ref_cursor;
function show_purchases return ref_cursor;

procedure purchases_made(p_cid in customers.cid%type, p_refcursor out sys_refcursor);
function no_of_customers(f_pid in products.pid%type) return number;
procedure add_customer(p_cid in customers.cid%type,p_name in customers.name%type,p_telephone# in customers.telephone#%type);
procedure add_purchase(p_eid in employees.eid%type, p_pid in products.pid%type, p_cid in customers.cid%type, p_pur_qty in purchases.qty%type, p_pur_unit_price in purchases.unit_price%type);
end check_pkg; --end of package specification
/
```

```
show errors


--body of check_pkg

 create or replace package body check_pkg as

-----------------------------------------------------------------------------------

function show_employees

return ref_cursor

is

a ref_cursor;

begin

open a for

select * from employees;

return a;

EXCEPTION

WHEN NO_DATA_FOUND THEN

       dbms_output.put_line('No data found in employees table');

WHEN OTHERS THEN

       dbms_output.put_line('Could not fetch the records. Please try again!');

end;



function show_customers

return ref_cursor is

b ref_cursor;

begin

open b for

select * from customers;

return b;

EXCEPTION

WHEN NO_DATA_FOUND THEN

       dbms_output.put_line('No data found in customers table');
```

```
WHEN OTHERS THEN
	dbms_output.put_line('Could not fetch the records. Please try again!');
end;




function show_products
return ref_cursor is
c ref_cursor;
begin
open c for
select * from products;
return c;
EXCEPTION
WHEN NO_DATA_FOUND THEN
	dbms_output.put_line('No data found in products table');
WHEN OTHERS THEN
	dbms_output.put_line('Could not fetch the records. Please try again!');
end;




function show_purchases
return ref_cursor is
d ref_cursor;
begin
open d for
select * from purchases;
return d;
EXCEPTION
WHEN NO_DATA_FOUND THEN
```

```
        dbms_output.put_line('No data found in purchases table');
WHEN OTHERS THEN
        dbms_output.put_line('Could not fetch the records. Please try again!');
end;


----------------------------------------------------------------------------------------------
  procedure purchases_made
        (p_cid in customers.cid%type,
        p_refcursor out sys_refcursor)
        is
        exceptions varchar(100);
        begin
                open p_refcursor for
                select name, pid, pur_date, qty, unit_price, total
                from customers, purchases where customers.cid = p_cid
                and customers.cid = purchases.cid;
EXCEPTION
        WHEN NO_DATA_FOUND THEN
                dbms_output.put_line('The customer Id you have entered does not exist in the customers table');
        WHEN OTHERS THEN
                dbms_output.put_line('Could not fetch the records. Please try again!');


 end;


---------------------------------------------------------------------------------------------
function no_of_customers(f_pid in products.pid%type)
        return number is
        num_of_customers number;
        begin
                select count(cid) into num_of_customers from
                purchases where pid = f_pid;
```

```
            return (num_of_customers);
 EXCEPTION
       WHEN NO_DATA_FOUND THEN
       dbms_output.put_line('The product Id you have entered does not exist in the products table');


       WHEN OTHERS THEN
       dbms_output.put_line('Could not fetch the number of customers who have bought the specific product. Please try
again!');
       end;
------------------------------------------------------------------------------------------------


procedure add_customer
       (p_cid in customers.cid%type,
       p_name in customers.name%type,
       p_telephone# in customers.telephone#%type)
       is
       incorrect_telephone# EXCEPTION;
       begin
          if LENGTH(p_telephone#) <>  12 then
             raise incorrect_telephone#;
       else
          insert into customers(cid, name, telephone#, visits_made, last_visit_date) values
             (p_cid, p_name, p_telephone#, 1, sysdate);
       end if;


       EXCEPTION
       WHEN DUP_VAL_ON_INDEX THEN
       dbms_output.put_line('The customer id you have entered already exists ! Please enter a unique customer id');


       WHEN incorrect_telephone# THEN
       dbms_output.put_line('Incorrect format of telephone number entered. Please enter in (XXX-XXX-XXXX) format!');
```

```
        WHEN OTHERS THEN
        dbms_output.put_line('Could not perform insert operation. Please try again!');
end;

------------------------------------------------------------------------------------------------

procedure add_purchase
        (p_eid in employees.eid%type,
        p_pid in products.pid%type,
        p_cid in customers.cid%type,
        p_pur_qty in purchases.qty%type,
        p_pur_unit_price in purchases.unit_price%type)
        is
        v_eid employees.eid%type;
        v_pid products.pid%type;
        v_cid customers.cid%type;
        v_pur_qty purchases.qty%type;
        v_pur_unit_price purchases.unit_price%type;
        v_saving purchases.saving%type;
        v_regular_price products.regular_price%type;
        v_qoh products.qoh%type;
        qoh_exception exception;

        eid_exception exception;
        pid_exception exception;
        cid_exception exception;
        eid_count number(2);
        pid_count number(2);
        cid_count number(2);
```

```
    begin

        select count(eid) into eid_count from employees where eid = p_eid;
        select count(pid) into pid_count from products where pid = p_pid;
        select count(cid) into cid_count from customers where cid = p_cid;

        if eid_count < 1 then
            raise eid_exception;
        END IF;
        if pid_count < 1 then
            raise pid_exception;
        END IF;
        if cid_count < 1 then
            raise cid_exception;
        END IF;

        select products.qoh into v_qoh from products where products.pid = p_pid;

        if p_pur_qty > v_qoh then
            raise qoh_exception;
        end if;

        select regular_price into v_regular_price from products where products.pid = p_pid;
        v_saving := p_pur_qty*(v_regular_price - p_pur_unit_price);

        insert into purchases(pur#, eid, pid, cid, pur_date, qty, unit_price, total, saving) values
        (seqpur#.nextval, p_eid, p_pid, p_cid, sysdate, p_pur_qty, p_pur_unit_price, p_pur_qty * p_pur_unit_price,
v_saving);

        EXCEPTION
        WHEN eid_exception THEN
        dbms_output.put_line('The eid you have entered does not exist in the employees table. Please try again');

        WHEN pid_exception THEN
        dbms_output.put_line('The pid you have entered does not exist in the products table. Please try again');

        WHEN cid_exception THEN
        dbms_output.put_line('The cid you have entered does not exist in the customers table. Please try again');

        WHEN qoh_exception THEN
        dbms_output.put_line('Insufficient quantity in stock');

        WHEN OTHERS THEN
        dbms_output.put_line('Could not perform insert action. Please check values entered!');
    end;
------------------------------------------------------------------------------------------
end; --packg body end

/
show errors
```