Btech CSE Semester-IV

CSE250 Database Management Systems

Stationery retail management

Group Number-33

Group Members:

1. AU1940146-Nimisha Patel B.tech CSE
2. AU1940272-Dhruvi Desai B.tech CSE

Description of Project:

In this project we use java for the frontend and for the database we have used oracle SQL developerIt tracks daily store operations like shop's employees, customers, discount for products, products Information, purchases, suppliers and supplies.

A detail explanation of all functionalities:-

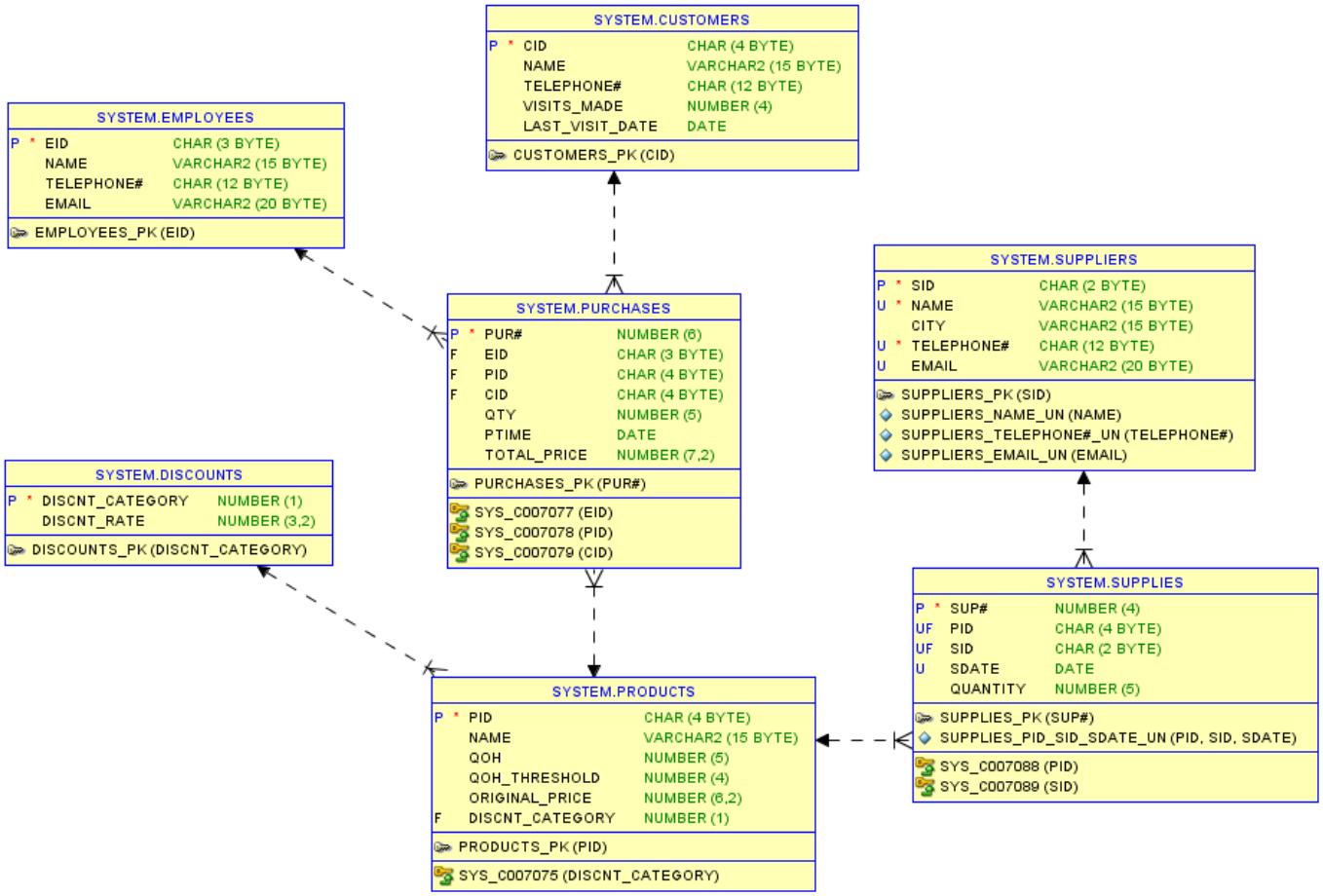
1) Adding functionalities:-
    - Add Customer: We are adding details of new customers(id, name, telephone, number of visits, last visit date)
    - Add employee: If a new employee has joined the shop then we are adding the details of a new employee(id, name, telephone, email)
    - Add suppliers: If a new supplier has came then we are adding the details of a new suppliers(id, name, city, telephone, email)
    - Add purchase: If a product has been sold then we are adding the id of an employee who sold the product, id of a customer who has purchased, id of a product being sold and it's quantity.

2) Delete functionalities:-
    - Delete purchase
    - Delete Employee
    - Delete Customer

3) Other functionalities:-
    - Purchase saving: Using the purchase id we are calculating the total saving of that purchase.
    - Monthly sale activities: We can get the monthly sale activities of any employee. Using the employee id we are displaying the total sale, total number of quantities being sold and number of sales in that particular month.
    - Total employee: Displaying total number of employees in a shop
    - Total customers: Displaying total number of customers in a shop

4) Showing all data of a Table:-
    - Show customers
    - Show employee
    - Show discounts
    - Show purchases
    - Show suppliers
    - Show supplies
    - Show products
    - Show logs(for trigger)

b. Entity-Relationship Diagram (Image)



c. Table Design (Data Dictionary)

Table- Employee

COLUMN NAME	DATA TYPE	CONSTRAINTS	FORMAT	DESCRIPTION
eid	char(3)	Primary key	E5	Unique employee id for each employee
Name	varchar2(15)	Check constraint	Janvi	Name of the employee
Telephone#	char(12)	Check constraint	7876787678, 789-987-9988	phone number of employee
Email	varchar2(20)	Check constraint	janvi@gmail.com	Email id of employee

Table- Customer

COLUMN NAME	DATA TYPE	CONSTRAINTS	FORMAT	DESCRIPTION
cid	char(4)	Primary key	C5	Unique Customer id for each customer
Name	varchar2(15)	Check constraint	Janvi	Name of the customer
Telephone#	char(12)	Check constraint	7876787678, 789-987-9988	phone number of customer
visits_made	number(4)	Check constraint	4	How Many time customer visited in stationery
last_visit_date			12-OCT-2019	Last time customer Visited in stationery

Table- Discounts

COLUMN NAME	DATA TYPE	CONSTRAINTS	FORMAT	DESCRIPTION
discnt_category	number(1)	Primary key	1	Discount category on product
discnt_rate	number(3,2)	Check constraint	0.1	Discount rate on product

Table- products

COLUMN NAME	DATA TYPE	CONSTRAINTS	FORMAT	DESCRIPTION
pid	char(4)	Primary key	P5	Unique Product id for each product
Name	varchar2(15)	Check constraint	pencil	Name of the product
qoh	number(5)	Check constraint	7	Quantity number of product
qoh_threshold	number(5)	Check constraint	4	j
original_price	number(6,2)	hhh	67.98	Price of product
discnt_category	number(1)	Foreign Key Referring discounts table	4	Discount category on product

Table- Purchases

COLUMN NAME	DATA TYPE	CONSTRAINTS	FORMAT	DESCRIPTION
pur#	number(6)	Primary key	1001	Purchase number of customer
eid	char(3)	Check constraint	E4	Employee id
pid	char(4)	Check constraint	P7	Product id
cid	char(4)	Check constraint	C6	Customer id
qty	number(5)		5	How many number of quantity Customer buy
ptime	date		23-OCT-2021	Purchase time
total_price	number(7,2)		89.67	Total price customer Have to pay

Table- Suppliers

COLUMN NAME	DATA TYPE	CONSTRAINTS	FORMAT	DESCRIPTION
sid	char(2)	Primary key	S5	Unique supplier id for each suppliers
Name	varchar2(15)	Check constraint	kevin	Name of the supplier
city	varchar2(15)		Ahmedabad	City of supplier

Telephone#	char(12)	Check constraint	7876787678, 789-987-9988	phone number of supplier
Email	varchar2(20)	Check constraint	kevin@gmail.com	Email id of supplier

Table- Supplies

COLUMN NAME	DATA TYPE	CONSTRAINTS	FORMAT	DESCRIPTION
sup#	number(4)	Primary key	1002	Unique supply number for each supplies
pid	char(4)	Check constraint	P3	Product id
sid	char(2)		S4	Supplier id
sdate	date	Check constraint	12-AUG-2021	Supply date
quantity	number(5)	Check constraint	67	How many quantity they supplies

Table- Logs

COLUMN NAME	DATA TYPE	CONSTRAINTS	FORMAT	DESCRIPTION
log#	number(5)	Primary key	1002	Unique lod number for each new inserted Or updated value (from trigger)
user_name	varchar2(12)	Not null	system	U

operation	varchar2(6)	Not null	Insert, update	Type of operation You performed
op_time	date	Not null	12-AUG-2021	Current time of operation Which you have performed
table_name	varchar2(20)	Not null	customer	In which table you have Inserted the new value
tuple_pkey	varchar2(6)		C3	

d. Stored Procedure, Functions, and Triggers (With code and statement to call

The procedure, a function is written on the front-end)

e. Screenshots of results generated after procedure and function are called on

The front-end

f. Screenshots of errors generated on the front-end when the trigger is fired

Procedures:

In this Section, we have first written the code of SQL procedure, then attached the Screenshot of calling it in java, and then attached the output Screenshot.

1. Add customer:

Description: The following procedure inserts the customer data into the database when he/she buys.

Procedure:

PROCEDURE add_customer(

c_id customers.cid%TYPE,

c_name customers.name%TYPE,

c_telephone# customers.telephone#%TYPE)

is

begin

insert into customers(cid,name,telephone#,visits_made,last_visit_date) values (c_id,c_name,c_telephone#,'1',to_char(SYSDATE,'DD-MON-YYYY'));

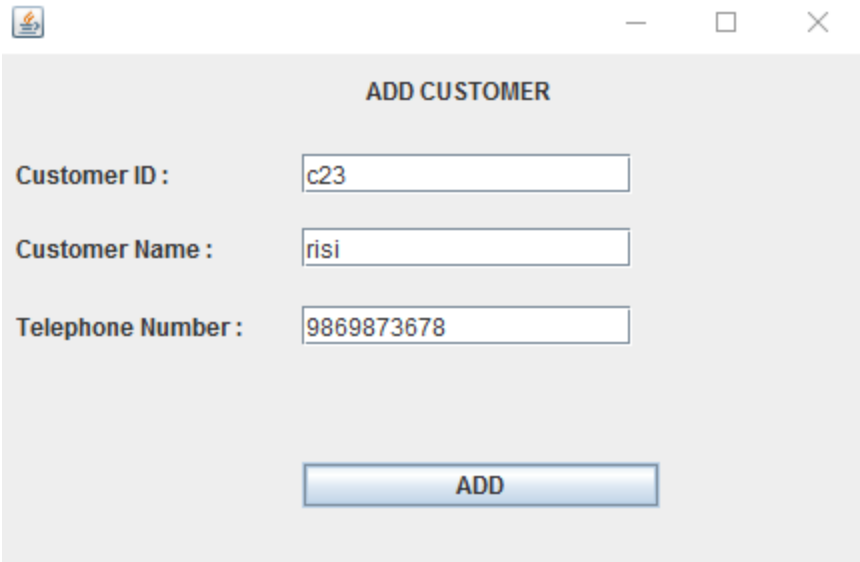
commit;

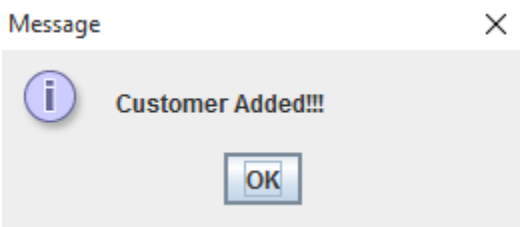
End;

Front-end calling code Screenshot:

```
public void customr_add(String id, String name, String telephone) {
    try {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        Connection con = DriverManager.getConnection(
                "jdbc:oracle:thin:@localhost:1521:xe", "system", "Nimisha123#");
        CallableStatement cs = con.prepareCall("{ call procesureandfunction.add_customer(?,?,?)}")
        cs.setString(1, id);
        cs.setString(2, name);
        cs.setString(3, telephone);
        cs.execute();
        JOptionPane.showMessageDialog(null, "Customer Added!!!");
        cs.close();
        con.close();
    } catch (SQLException ex) {
        JOptionPane.showMessageDialog(null, ex.getMessage());
        System.out.println(ex.getMessage());
    }
      catch (Exception e) {
        System.out.print("Exception ---" + e.getStackTrace());
    }
}
```

Output Screenshot:





2. Delete purchase:

Description: The following procedure deletes the purchase order when he/she types his/her purchase id.

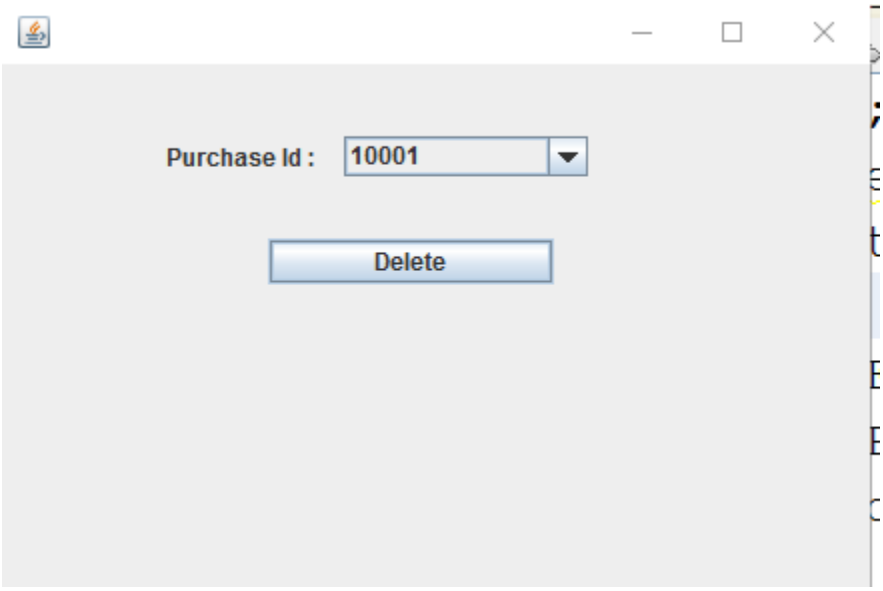
Procedure:

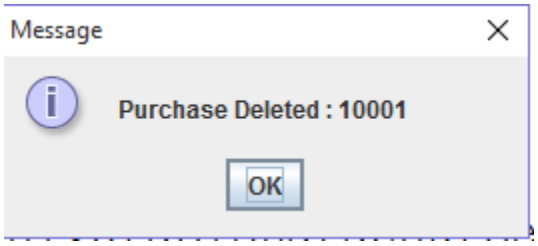
```
procedure delete_purchase(

  pur purchases.pur#%TYPE )

  is

  begin

  delete from purchases where pur# = pur;

  commit;

  end;
```

Front-end calling code Screenshot:

```java
public void delete_pur(String pur_id) {
    try {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        Connection con = DriverManager.getConnection(
                "jdbc:oracle:thin:@localhost:1521:xe", "system", "Nimisha123#");
        CallableStatement cs = con.prepareCall("{ call procesureandfunction.delete_purchase
        cs.setString(1, pur_id);
        cs.execute();
        JOptionPane.showMessageDialog(null, "Purchase Deleted : " + pur_id);
        cs.close();
        con.close();
    } catch (SQLException ex) {
        JOptionPane.showMessageDialog(null, "Something went Wrong!!");
        System.out.println("\n*** SQLException caught ***\n" + ex.getMessage());
    } catch (Exception e) {
        System.out.print("Exception ---" + e.getStackTrace());
    }

}
```

Output Screenshot:





3. Add employee:

Description: The following procedure inserts the employee data into the database when he/she put details.

Procedure:

PROCEDURE add_employee(

e_id employees.eid%TYPE,

e_name employees.name%TYPE,

e_telephone# employees.telephone#%TYPE,

e_email employees.email%TYPE)

is

begin

insert into employees(eid,name,telephone#,email) values (e_id,e_name,e_telephone#,e_email);

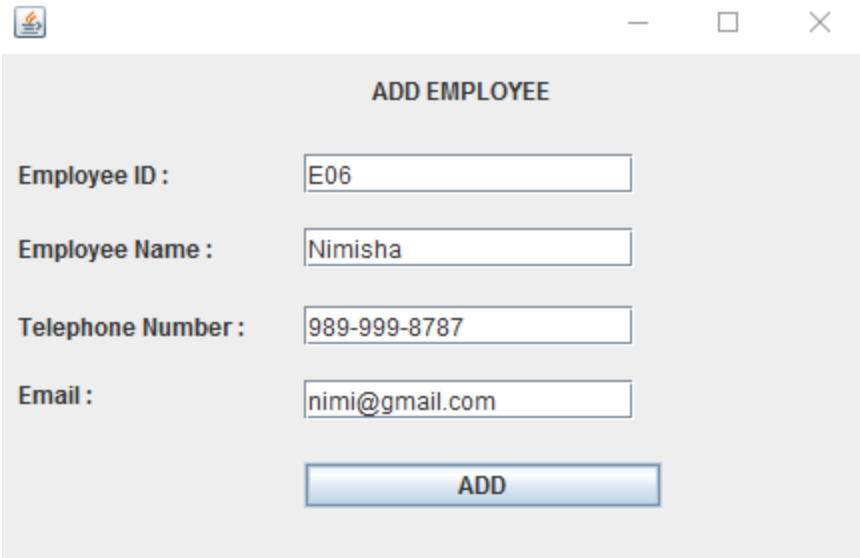
commit;

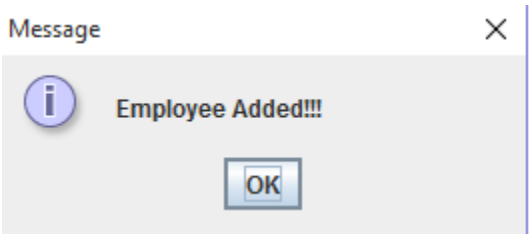
End;

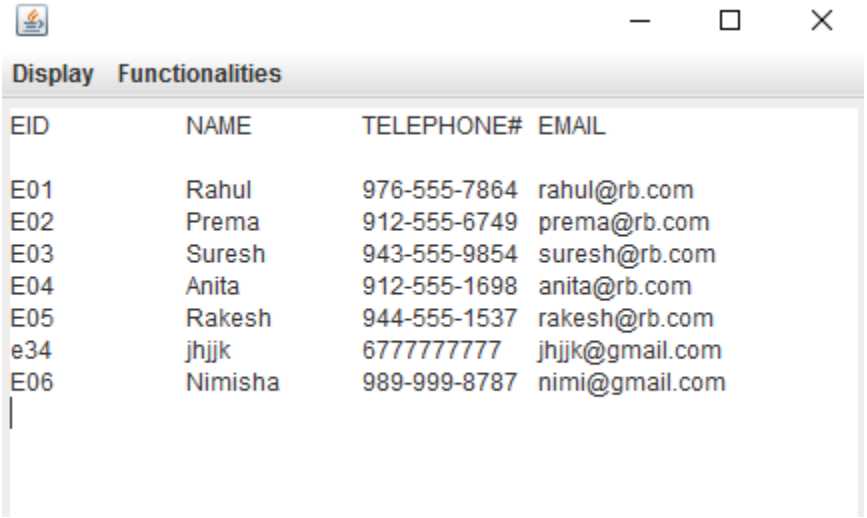
Front-end calling code Screenshot:

```
public void employee_add(String id, String name, String telephone,String email) {
    try {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        Connection con = DriverManager.getConnection(
                "jdbc:oracle:thin:@localhost:1521:xe", "system", "Nimisha123#");
        CallableStatement cs = con.prepareCall("{ call procesureandfunction.add_employee(?,?,?,?)}");
            cs.setString(1, id);
            cs.setString(2, name);
            cs.setString(3, telephone);
            cs.setString(4, email);
            cs.execute();
            JOptionPane.showMessageDialog(null, "Employee Added!!!");
        cs.close();
        con.close();
    } catch (SQLException ex) {
        JOptionPane.showMessageDialog(null, ex.getMessage());
        System.out.println(ex.getMessage());
    }
      catch (Exception e) {
        System.out.print("Exception ---" + e.getStackTrace());
    }
}
```

Output Screenshot:







4. Delete employee:

Description: The following procedure deletes the employee data when he/she leaves the shop and he/she types his/her employee id.

Procedure:

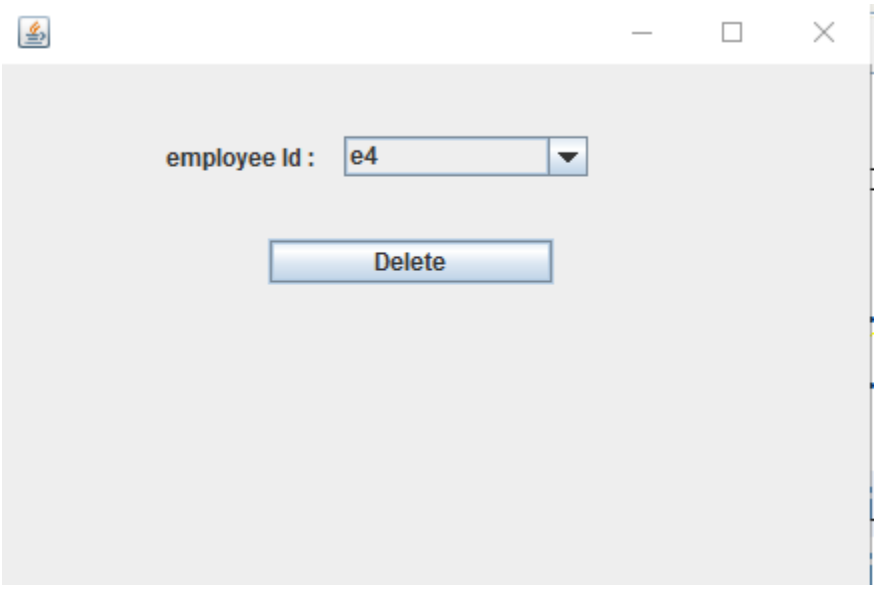
procedure delete_employee(

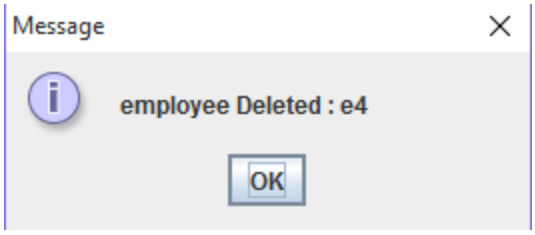
```
emp employees.eid%TYPE )
is
begin
delete from employees where eid = emp;
commit;
end;
```

Front-end calling code Screenshot:

```
void delete_emp(String eid) {
{
  Class.forName("oracle.jdbc.driver.OracleDriver");
  Connection con = DriverManager.getConnection(
          "jdbc:oracle:thin:@localhost:1521:xe", "system", "Nimisha123#");
  CallableStatement cs = con.prepareCall("{ call procesureandfunction.delete_employee(?)}");
  cs.setString(1, eid);
  cs.execute();
  JOptionPane.showMessageDialog(null, "employee Deleted : " + eid);
  cs.close();
  con.close();
catch (SQLException ex) {
  JOptionPane.showMessageDialog(null, "Something went Wrong!!");
  System.out.println("\n*** SQLException caught ***\n" + ex.getMessage());
catch (Exception e) {
  System.out.print("Exception ---" + e.getStackTrace());
```

Output Screenshot:





5. Delete customer:

Description: The following procedure deletes the customer data when he/she leaves the city and he/she types his/her customer id.

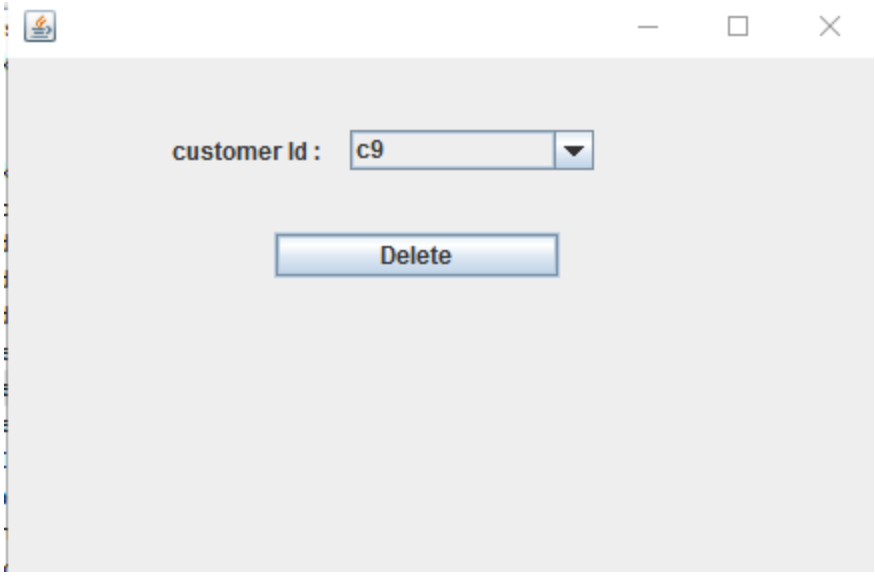
Procedure:

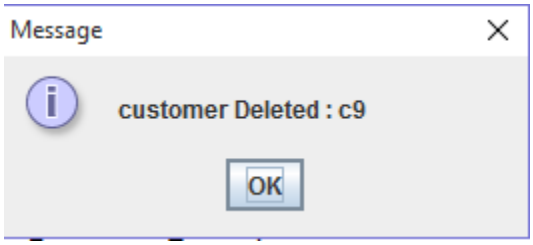
```
procedure delete_customer(

  cus customers.cid%TYPE )

  is

  begin

  delete from customers where cid = cus;

  commit;

  end;
```

Front-end calling code Screenshot:

```
ic void delete_cus(String cid) {
try {
    Class.forName("oracle.jdbc.driver.OracleDriver");
    Connection con = DriverManager.getConnection(
            "jdbc:oracle:thin:@localhost:1521:xe", "system", "Nimisha123#");
    CallableStatement cs = con.prepareCall("{ call procesureandfunction.delete_customer(?)}")
    cs.setString(1, cid);
    cs.execute();
    JOptionPane.showMessageDialog(null, "customer Deleted : " + cid);
    cs.close();
    con.close();
} catch (SQLException ex) {
    JOptionPane.showMessageDialog(null, "Something went Wrong!!");
    System.out.println("\n*** SQLException caught ***\n" + ex.getMessage());
} catch (Exception e) {
    System.out.print("Exception ---" + e.getStackTrace());
}
```

Output Screenshot:





6. Add supplier:

Description: The following procedure inserts the supplier data into the database when he/she put details.

Procedure:

```
procedure add_supplier(

 s_id suppliers.sid%TYPE,

  s_name suppliers.name%TYPE,

  s_city suppliers.city%TYPE,

  s_telephone# suppliers.telephone#%TYPE,

 s_email suppliers.email%TYPE )

 is
```

begin

insert into suppliers(sid,name,city,telephone#,email) values (s_id,s_name,s_city,s_telephone#,s_email);

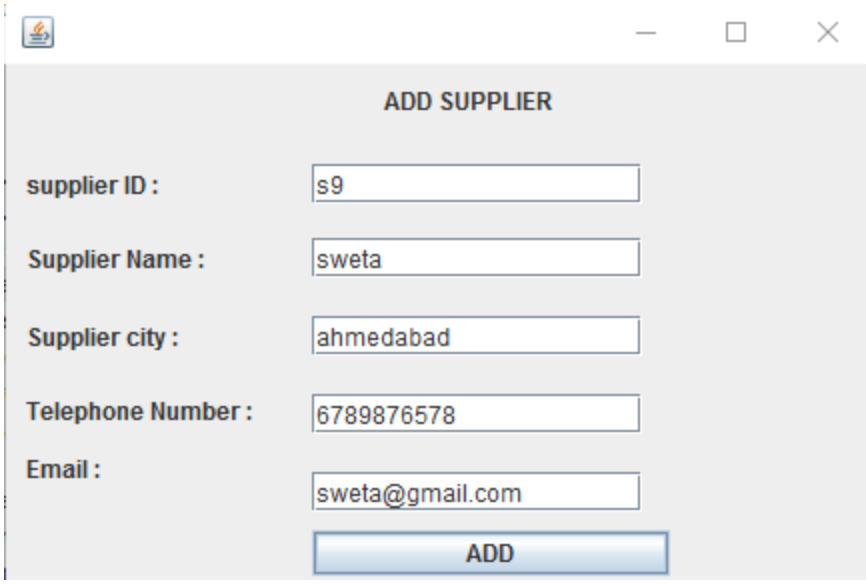
commit;

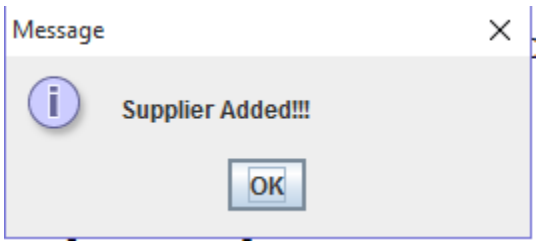
End;

Front-end calling code Screenshot:

```
blic void supplier_add(String id, String name, String city, String telephone, String email) {
  try {
      Class.forName("oracle.jdbc.driver.OracleDriver");
      Connection con = DriverManager.getConnection(
              "jdbc:oracle:thin:@localhost:1521:xe", "system", "Nimisha123#");
      CallableStatement cs = con.prepareCall("{ call procesureandfunction.add_supplier(?,?,?,?,?)}");
          cs.setString(1, id);
          cs.setString(2, name);
          cs.setString(3, city);
          cs.setString(4, telephone);
          cs.setString(5, email);
          cs.execute();
          JOptionPane.showMessageDialog(null, "Supplier Added!!!");
      cs.close();
      con.close();
  } catch (SQLException ex) {
      JOptionPane.showMessageDialog(null, ex.getMessage());
      System.out.println(ex.getMessage());
  }
    catch (Exception e) {
      System.out.print("Exception ---" + e.getStackTrace());
  }
```

Output Screenshot:





FUNCTIONS:

In this Section, we have first written the code of the SQL function, then attached the Screenshot of calling it in java, and then attached the output Screenshot.

1) purchase_saving:

Description: The following function takes a value of purchase id from data and gives the value of the total saving of that purchase id.

Function:

```
FUNCTION purchase_saving(pur IN NUMBER) RETURN NUMBER

        IS saving NUMBER;

        BEGIN

        select original_price * qty - total_price into saving

        from purchases pu, products pr
```

where pu.pid = pr.pid and pu.pur# = pur;

RETURN saving;

EXCEPTION

WHEN NO_DATA_FOUND THEN

saving:=-1;

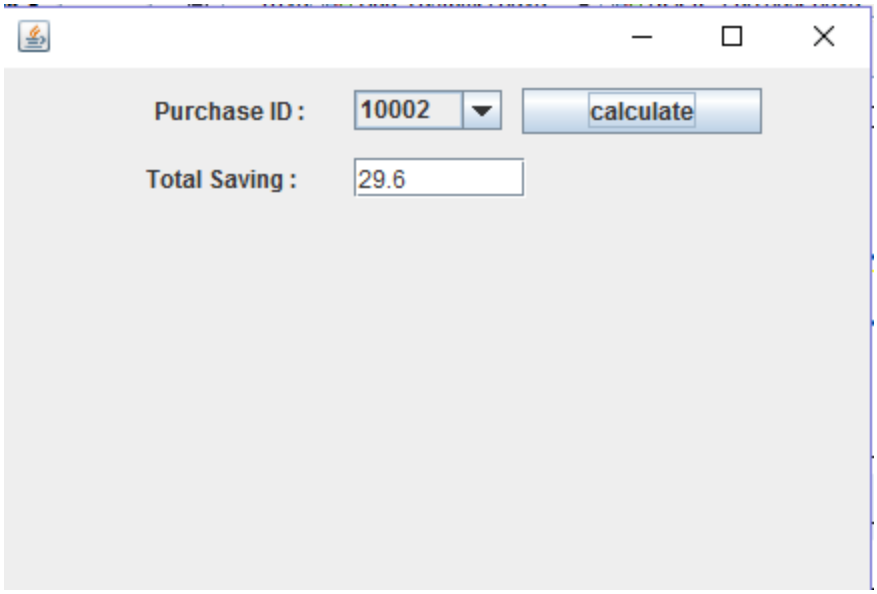
Return saving;

END;

Front-end calling code Screenshot:

```
public void saving(String id) {
    try {
        Class.forName("com.mysql.jdbc.Driver");
        Connection con = DriverManager.getConnection(
                "jdbc:oracle:thin:@localhost:1521:xe", "system", "Nimisha123#");
        CallableStatement cs = con.prepareCall("begin ? := procesureandfunction.purchase_saving(?); er
        cs.registerOutParameter(1, OracleTypes.NUMBER);
        cs.setString(2, id);
        cs.execute();
        float result = cs.getFloat(1);
        if (result == -1) {
            JOptionPane.showMessageDialog(null, "OOPS!! NO DATA FOUND!!");
        } else {
            textField_1.setText(result + "");
        }
        cs.close();
        con.close();
    } catch (SQLException e1) {
        JOptionPane.showMessageDialog(null, "OOps Can not find The Purchase ID..");
    } catch (Exception e1) {
        System.out.print("Exception ---" + e1.getStackTrace());
    }
}
```

Output Screenshot:



2) monthly_sale_activities:

Description:

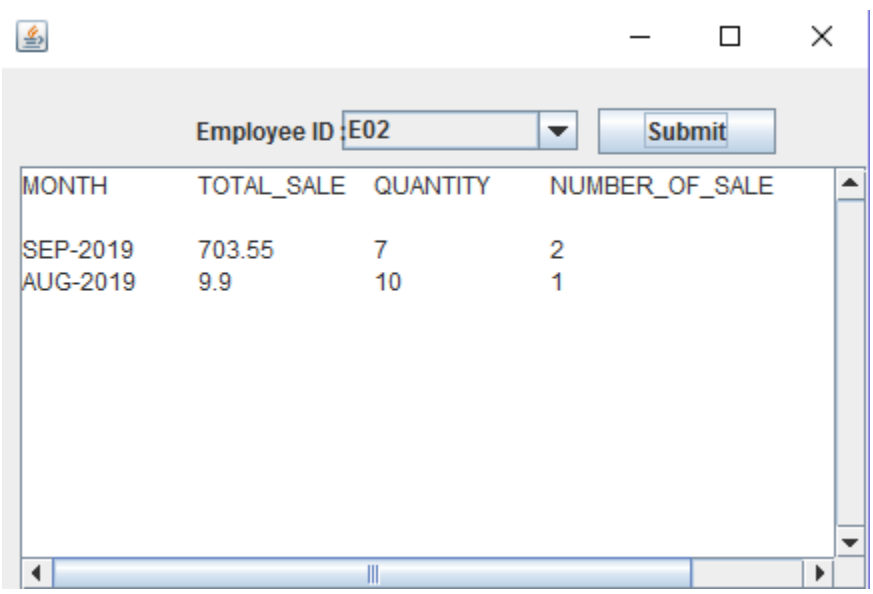
Function:

```
function monthly_sale_activities(emp_data in char)

 return ref_cursor is rc ref_cursor;

begin

open rc for

select to_char(ptime,'MON-YYYY') as month,  sum(pu.total_price) as total_sale, sum(pu.qty) as
quantity, count(pu.eid) as number_of_sale, emp.name from purchases pu, (select name from
employees where eid = emp_data) emp where

pu.eid = emp_data group by to_char(pu.ptime,'MON-YYYY'), emp.name;

return rc;

end;
```

Front-end calling code Screenshot:

```
] _sale(String eid) {

 me("oracle.jdbc.driver.OracleDriver");
 con = DriverManager.getConnection(
 bc:oracle:thin:@localhost:1521:xe", "system", "Nimisha123#");
 tement cs = con.prepareCall("begin ? := procesureandfunction.monthly_sale_activities(?); end;
 OutParameter(1, OracleTypes.CURSOR);
 g(2, eid);
 );
 s = (ResultSet) cs.getObject(1);
 mn_data = "MONTH" + "\t" + "TOTAL_SALE" + "\t" + "QUANTITY" + "\t" + "NUMBER_OF_SALE" + "\t"
 pend(column_data);
 ext()) {
 a.append(rs.getString(1) + "\t" + rs.getString(2) + "\t" + rs.getString(3) + "\t" + rs.getStr


 ;
 eption ex) {
 .showMessageDialog(null, "Something went Wrong!!");
 println("\n*** SQLException caught ***\n" + ex.getMessage());
 ion e) {
 print("Exception ---" + e.getStackTrace());
```

Output Screenshot:



3) total employees:

Description: This function counts how many employees are working in this shop.

Function:

FUNCTION totalEmployees

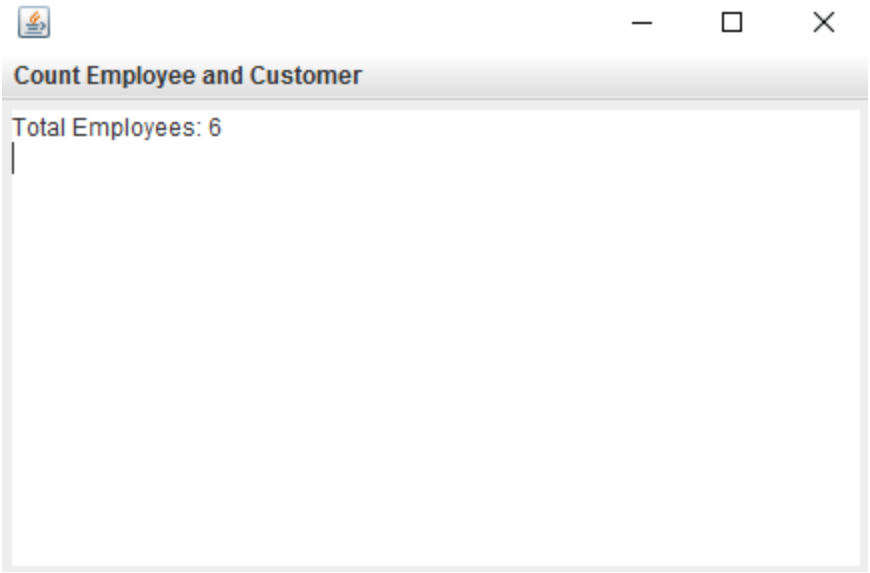
RETURN number IS

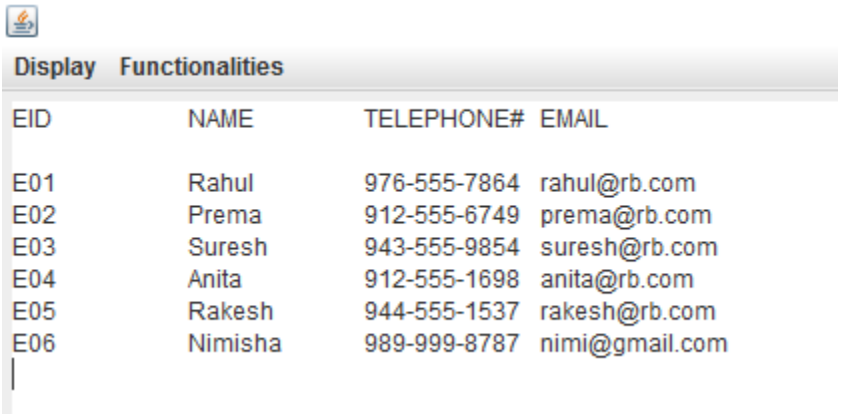
```
  total number(2) := 0;
BEGIN
  SELECT count(*) into total
  FROM employees;
  RETURN total;
END;
```

Front-end calling code Screenshot:

```
public void total_Employees() {
    try {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        Connection con = DriverManager.getConnection(
                "jdbc:oracle:thin:@localhost:1521:xe", "system", "Nimisha123#");
        System.out.print("connected");
        CallableStatement cs = con.prepareCall("begin ? := procesureandfunction.totalEmp
         cs.registerOutParameter(1, OracleTypes.NUMBER);
       cs.execute();
       int result = (int) cs.getFloat(1);
       textArea.append("Total Employees: "+result+ "\n");
        cs.close();
        con.close();
    } catch (SQLException ex) {
        JOptionPane.showMessageDialog(null, "Something went Wrong!!");
        System.out.println("\n*** SQLException caught ***\n" + ex.getMessage());
    } catch (Exception e) {
        System.out.print("Exception ---" + e.getStackTrace());
    }
}
```

Output Screenshot:





4) total customers:

Description: This function counts how many customers came to visit this shop.

Function:

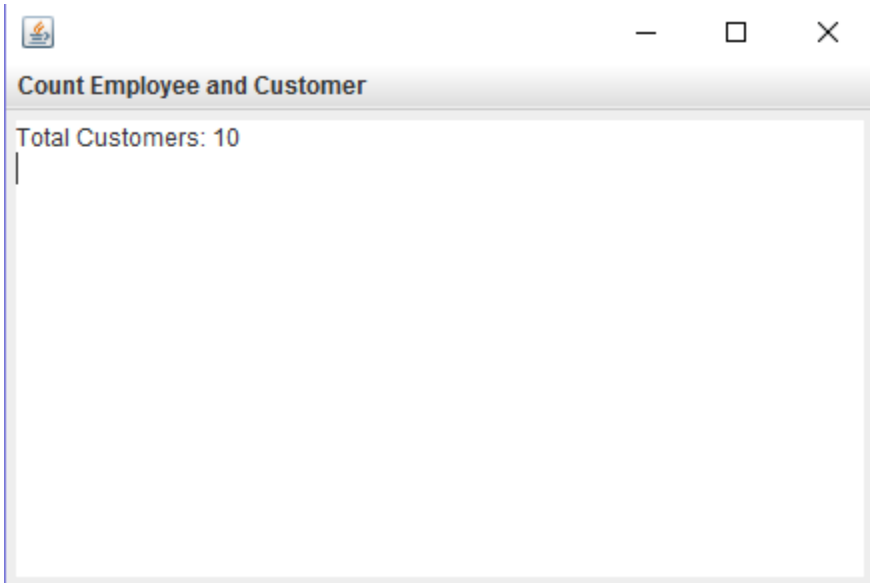
```
FUNCTION totalCustomers

RETURN number IS

  total number(2) := 0;

BEGIN

  SELECT count(*) into total

  FROM customers;

  RETURN total;
```

END;

Front-end calling code Screenshot:

```
public void total_Customers() {
    try {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        Connection con = DriverManager.getConnection(
                "jdbc:oracle:thin:@localhost:1521:xe", "system", "Nimisha123#");
        System.out.print("connected");
        CallableStatement cs = con.prepareCall("begin ? := procesureandfunction.totalCus
         cs.registerOutParameter(1, OracleTypes.NUMBER);
       cs.execute();
       int result = (int) cs.getFloat(1);
       textArea.append("Total Customers: "+result+ "\n");
        cs.close();
        con.close();
    } catch (SQLException ex) {
        JOptionPane.showMessageDialog(null, "Something went Wrong!!");
        System.out.println("\n*** SQLException caught ***\n" + ex.getMessage());
    } catch (Exception e) {
        System.out.print("Exception ---" + e.getStackTrace());
    }
}
```

Output Screenshot:



Display Functionalities

CID	NAME	TELEPHONE#	VISITS_MADE	LAST_VISIT_MADE
C1	Jayesh	966-555-9825	9	2021-04-08
C2	Kiran	988-555-6473	1	2019-09-18
C3	Geeta	966-555-3427	3	2019-09-19
C4	Rakesh	999-555-3142	1	2019-10-12
C5	Rakesh	954-555-9805	2	2019-08-12
C6	Kusum	976-555-6474	2	2019-08-15
C7	Kavya	989-555-7853	1	2019-10-16
C8	Joy	923-555-9840	1	2019-10-18
c23	risi	9869873678	1	2021-04-08
c56	jasmi	9876786789	1	2021-04-09

5) add_purchase:

Description: This function takes a value of purchase id, customer id, employee id, and quantity, and in the output will be added in purchases, in the time column will take the current time automatically. Also, the purchase number will take automatically.

Function:

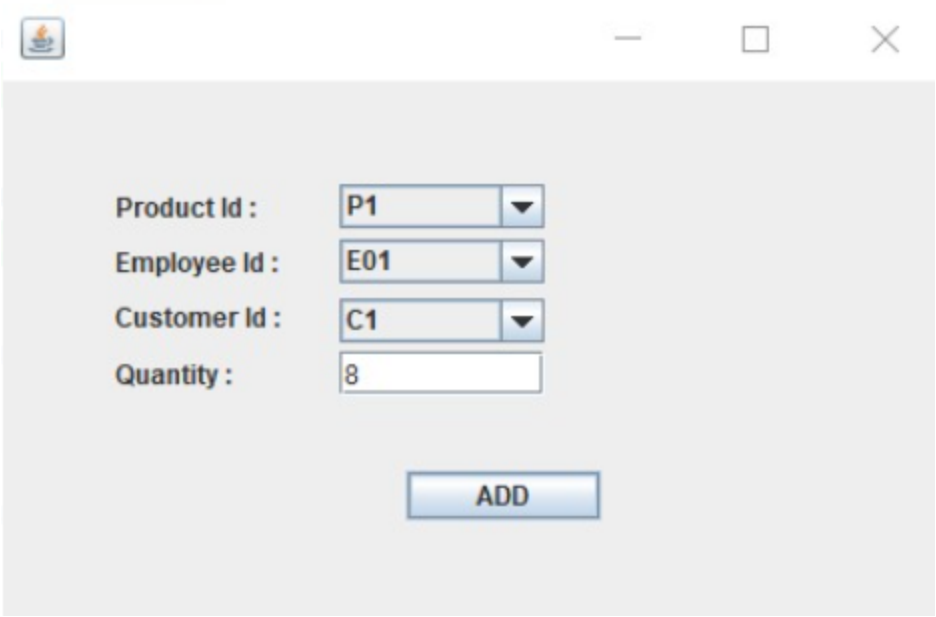
```
function add_purchase(e_id in CHAR, p_id in CHAR, c_id in CHAR, pur_qty in NUMBER) return
NUMBER is q_oh NUMBER; discount_cat NUMBER; discount NUMBER(3,2); price
number(6,2); total_pr number(6,2);

begin

Select qoh into q_oh from products where pid=p_id;

If q_oh >= pur_qty THEN

dbms_output.put_line('GOOD');

Select discnt_category into discount_cat from products where pid=p_id;

                Select discnt_rate into discount from discounts where
discnt_category=discount_cat;

                Select original_price into price from products where pid=p_id;



                total_pr:=price*(1-discount)*pur_qty;

                q_oh:=q_oh-pur_qty;
```

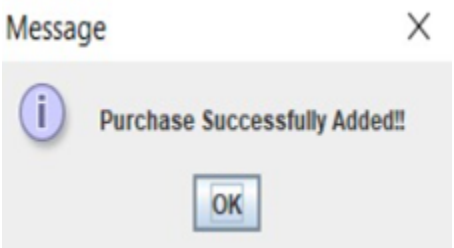
```
                UPDATE products SET qoh = q_oh WHERE pid=p_id;
insert into purchases values(pur_sequence.NEXTVAL,e_id,p_id,c_id,pur_qty,sysdate, total_pr);
select qoh into q_oh from products where pid = p_id;

ELSE
                q_oh := -123;
                dbms_output.put_line('Insufficient quantity in stock. ');
END IF;
Return q_oh;
end;
```

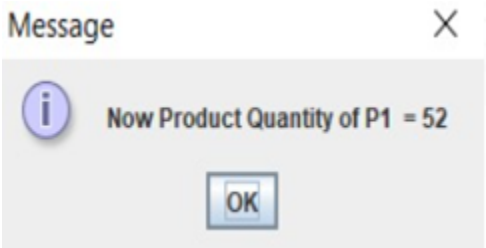
Front-end calling code Screenshot:

```
public void purchase_add() {

    try {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        Connection con = DriverManager.getConnection(
                "jdbc:oracle:thin:@localhost:1521:xe", "system", "Nimisha123#");
        CallableStatement cs = con.prepareCall("begin ? := project2.add_purchase(?,?,?,?); e
        cs.registerOutParameter(1, OracleTypes.NUMBER);
        if (textField.getText().matches("^\\d+(\\.\\d+)?")) {
            cs.setString(2, comboBox_1.getSelectedItem().toString());
            System.out.println(comboBox_1.getSelectedItem().toString());
            cs.setString(3, comboBox.getSelectedItem().toString());
            cs.setString(4, comboBox_2.getSelectedItem().toString());
            cs.setInt(5, Integer.parseInt(textField.getText()));
            cs.execute();
            int result = cs.getInt(1);
            if (result == -123) {
                JOptionPane.showMessageDialog(null, "Insufficient quantity in stock!!");
            } else {

                JOptionPane.showMessageDialog(null, "Purchase Successfully Added!!");
```

Output Screenshot:







6) show_customers :

Description: The following function helps the employee to show customer details.

Function:

```
function show_customers
   return ref_cursor is
   rcc ref_cursor;
   begin
   open rcc for
   select * from customers;
   return rcc;
   end;
```

Front-end calling code Screenshot:

```
ıblic void show_customer() {
    try {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        Connection con = DriverManager.getConnection(
                "jdbc:oracle:thin:@localhost:1521:xe", "system", "Nimisha123#");
        System.out.print("connected");
        CallableStatement cs = con.prepareCall("begin ? := procesureandfunction.show_customers(); end;")
        cs.registerOutParameter(1, OracleTypes.CURSOR);
        cs.execute();
        ResultSet rs = (ResultSet) cs.getObject(1);
        String column_name = "CID" + "\t" + "NAME" + "\t" + "TELEPHONE#" + "\t" + "VISITS_MADE" + "\t" +
        textArea.append(column_name);
        while (rs.next()) {
            textArea.append(rs.getString(1) + "\t" + rs.getString(2) + "\t" + rs.getString(3) + "\t" + r
        }
        cs.close();
        con.close();
    } catch (SQLException ex) {
        JOptionPane.showMessageDialog(null, "Something went Wrong!!");
        System.out.println("\n*** SQLException caught ***\n" + ex.getMessage());
    } catch (Exception e) {
        System.out.print("Exception ---" + e.getStackTrace());
```

Output Screenshot:

Display Functionalities

CID	NAME	TELEPHONE#	VISITS_MADE	LAST_VISIT_MADE
C1	Jayesh	966-555-9825	6	2021-04-08
C2	Kiran	988-555-6473	1	2019-09-18
C3	Geeta	966-555-3427	3	2019-09-19
C4	Rakesh	999-555-3142	1	2019-10-12
C5	Rakesh	954-555-9805	2	2019-08-12
C6	Kusum	976-555-6474	2	2019-08-15
C7	Kavya	989-555-7853	1	2019-10-16
C8	Joy	923-555-9840	1	2019-10-18
c9	nim	879-988-9999	1	2021-04-07
c23	risi	9869873678	1	2021-04-08

7) show_employees:

Description: The following function helps to show employee details.

Function:

function show_employees

return ref_cursor is

rce ref_cursor;

begin

open rce for

select * from employees;

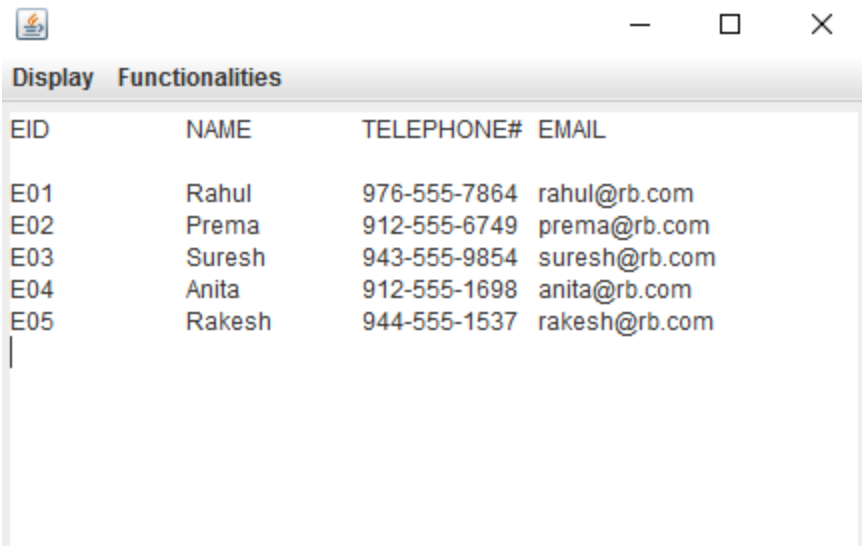
return rce;

end;

Front-end calling code Screenshot:

```
ıblic void show_employees() {
    try {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        Connection con = DriverManager.getConnection(
                "jdbc:oracle:thin:@localhost:1521:xe", "system", "Nimisha123#");
        CallableStatement cs = con.prepareCall("begin ? := procesureandfunction.show_employees(); end;")
        cs.registerOutParameter(1, OracleTypes.CURSOR);
        String Column_name = "EID" + "\t" + "NAME" + "\t" + "TELEPHONE#" + "\t" + "EMAIL\n\n";
        textArea.append(Column_name);
        cs.execute();
        ResultSet rs = (ResultSet) cs.getObject(1);
        while (rs.next()) {
            textArea.append(rs.getString(1) + "\t" + rs.getString(2) + "\t" + rs.getString(3) + "\t" + r
        }
        cs.close();
        con.close();
    } catch (SQLException ex) {
        JOptionPane.showMessageDialog(null, "Something went Wrong!!");
        System.out.println("\n*** SQLException caught ***\n" + ex.getMessage());
    } catch (Exception e) {
        System.out.print("Exception ---" + e.getStackTrace());
    }
```

Output Screenshot:



8) show_products:

Description: The following function helps to show Product details.

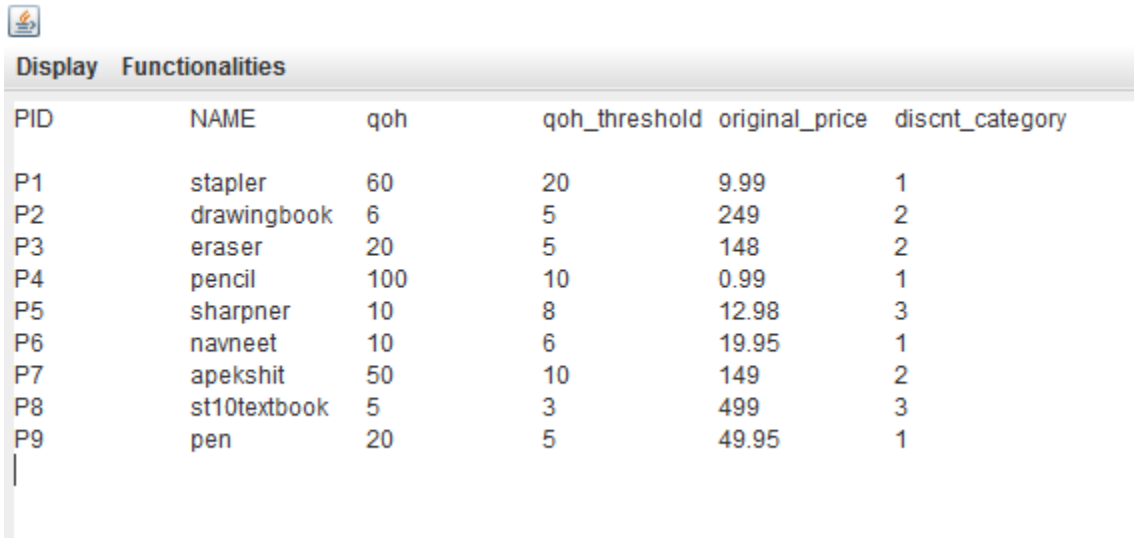
Function:

```
function show_products
   return ref_cursor is
   rcp ref_cursor;
   begin
   open rcp for
   select * from products;
   return rcp;
   end;
```

Front-end calling code Screenshot:

```java
public void show_products() {
  try {
      Class.forName("oracle.jdbc.driver.OracleDriver");
      Connection con = DriverManager.getConnection(
              "jdbc:oracle:thin:@localhost:1521:xe", "system", "Nimisha123#");
      CallableStatement cs = con.prepareCall("begin ? := procesureandfunction.show_products(); end;");
      cs.registerOutParameter(1, OracleTypes.CURSOR);
      String Column_name = "PID" + "\t" + "NAME" + "\t" + "qoh" + "\t"+"qoh_threshold" + "\t"+"origina
      textArea.append(Column_name);
      cs.execute();
      ResultSet rs = (ResultSet) cs.getObject(1);
      while (rs.next()) {
          textArea.append(rs.getString(1) + "\t" + rs.getString(2) + "\t" + rs.getString(3) + "\t" + r
      }
      cs.close();
      con.close();
  } catch (SQLException ex) {
      JOptionPane.showMessageDialog(null, "Something went Wrong!!");
      System.out.println("\n*** SQLException caught ***\n" + ex.getMessage());
  } catch (Exception e) {
      System.out.print("Exception ---" + e.getStackTrace());
  }
```

Output Screenshot:



Display Functionalities

PID	NAME	qoh	qoh_threshold	original_price	discnt_category
P1	stapler	60	20	9.99	1
P2	drawingbook	6	5	249	2
P3	eraser	20	5	148	2
P4	pencil	100	10	0.99	1
P5	sharpner	10	8	12.98	3
P6	navneet	10	6	19.95	1
P7	apekshit	50	10	149	2
P8	st10textbook	5	3	499	3
P9	pen	20	5	49.95	1

9) show_purchases:

Description: The following function helps to show Purchases detail of products.

Function:

function show_purchases

return ref_cursor is

rcpu ref_cursor;

begin

open rcpu for

select * from purchases;

return rcpu;

end;

Front-end calling code Screenshot:

```
ıblic void show_purchases() {
    try {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        Connection con = DriverManager.getConnection(
                "jdbc:oracle:thin:@localhost:1521:xe", "system", "Nimisha123#");
        CallableStatement cs = con.prepareCall("begin ? := procesureandfunction.show_purchases(); end;")
        cs.registerOutParameter(1, OracleTypes.CURSOR);
        cs.execute();
        ResultSet rs = (ResultSet) cs.getObject(1);
        String column_name = "PUR#" + "\t" + "EID" + "\t" + "PID" + "\t" + "CID" + "\t" + "QTY" + "\t" +
        textArea.append(column_name);
        while (rs.next()) {
            textArea.append(rs.getString(1) + "\t" + rs.getString(2) + "\t" + rs.getString(3) + "\t" + r
            );
        }
        cs.close();
        con.close();
    } catch (SQLException ex) {
        JOptionPane.showMessageDialog(null, "Something went Wrong!!");
        System.out.println("\n*** SQLException caught ***\n" + ex.getMessage());
    } catch (Exception ex) {
        System.out.print("Exception ---" + ex.getStackTrace());
    }
```

Output Screenshot:

Display Functionalities

PUR#	EID	PID	CID	QTY	PTIME	TOTAL_PRICE
10002	E01	P3	C1	1	2019-09-18	118.4
10003	E02	P4	C2	5	2019-09-18	4.95
10004	E01	P5	C3	2	2019-09-19	18.17
10005	E04	P7	C4	1	2019-10-12	119.2
10006	E03	P8	C1	1	2019-10-12	349.3
10007	E03	P6	C3	2	2019-09-19	35.91
10008	E03	P6	C5	1	2019-08-12	17.96
10009	E03	P1	C7	1	2019-10-16	8.99
10010	E04	P2	C6	1	2019-09-19	211.65
10011	E02	P4	C6	10	2019-08-15	9.9
10012	E02	P8	C3	2	2019-09-12	698.6
10013	E04	P6	C5	2	2019-08-30	35.91
10014	E03	P9	C8	3	2019-10-18	134.84

10) show_logs:

Description: This function is used for update or insert triggers.

Function:

```
function show_logs
   return ref_cursor is
   rcl ref_cursor;
   begin
   open rcl for
   select * from logs;
   return rcl;
   end;
```

Front-end calling code Screenshot:

```
ıblic void show_logs() {
    try {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        Connection con = DriverManager.getConnection(
                "jdbc:oracle:thin:@localhost:1521:xe", "system", "Nimisha123#");
        CallableStatement cs = con.prepareCall("begin ? := procesureandfunction.show_logs(); end;");
        cs.registerOutParameter(1, OracleTypes.CURSOR);
        cs.execute();
        ResultSet rs = (ResultSet) cs.getObject(1);
        String column_names = "LOG#" + "\t" + "USER_NAME" + "\t" + "OPERATION" + "\t" + "OP_TIME" + "\t"
        textArea.append(column_names);
        while (rs.next()) {
            textArea.append(rs.getString(1) + "\t" + rs.getString(2) + "\t" + rs.getString(3) + "\t" + r
            );
        }
        cs.close();
        con.close();
    } catch (SQLException ex) {
        JOptionPane.showMessageDialog(null, "Something went Wrong!!");
        System.out.println("\n*** SQLException caught ***\n" + ex.getMessage());
    } catch (Exception ex) {
        System.out.print("Exception ---" + ex.getStackTrace());
    }
```

Output Screenshot:

Display Functionalities

LOG#	USER_NAME	OPERATION	OP_TIME	TABLE_NAME	TUPLE_KEY
10000	SYSTEM	insert	2021-04-07	customers	c9
10001	SYSTEM	update	2021-04-07	products	P2
10002	SYSTEM	update	2021-04-07	customers	C1
10003	SYSTEM	update	2021-04-08	customers	C1
10004	SYSTEM	insert	2021-04-08	purchases	10001
10020	SYSTEM	insert	2021-04-08	customers	c23
10021	SYSTEM	update	2021-04-08	products	P2
10022	SYSTEM	update	2021-04-08	customers	C1

11) show_discounts:

Description: The following function helps to show discount in the product.

Function:

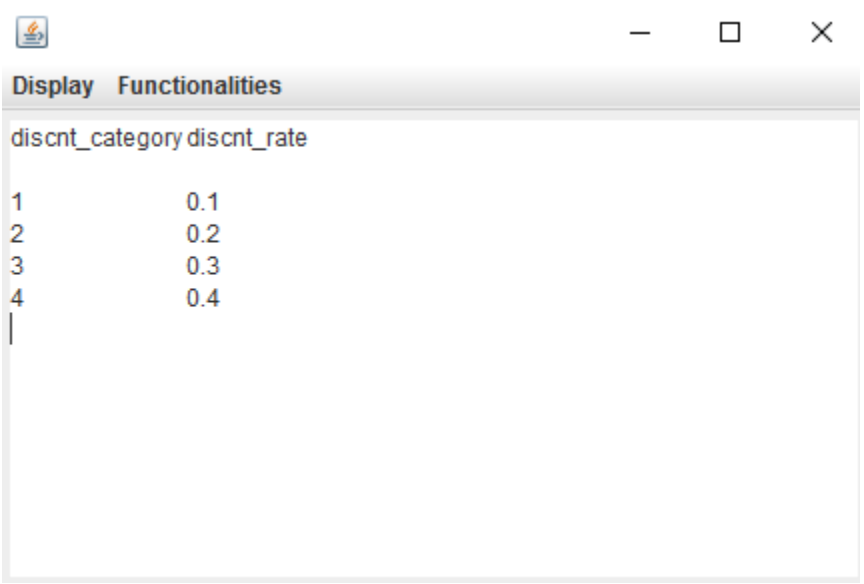
```
function show_discounts
    return ref_cursor is
    rds ref_cursor;
```

```
begin
open rds for
select * from discounts;
return rds;
end;
```

Front-end calling code Screenshot:

```java
ıblic void show_discounts() {
    try {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        Connection con = DriverManager.getConnection(
                "jdbc:oracle:thin:@localhost:1521:xe", "system", "Nimisha123#");
        System.out.print("connected");
        CallableStatement cs = con.prepareCall("begin ? := procesureandfunction.show_discounts(); end;")
        cs.registerOutParameter(1, OracleTypes.CURSOR);
        cs.execute();
        ResultSet rs = (ResultSet) cs.getObject(1);
        String column_name = "discnt_category" + "\t" + "discnt_rate" +"\n\n";
        textArea.append(column_name);
        while (rs.next()) {
            textArea.append(rs.getString(1) + "\t" + rs.getString(2) + "\n");
        }
        cs.close();
        con.close();
    } catch (SQLException ex) {
        JOptionPane.showMessageDialog(null, "Something went Wrong!!");
        System.out.println("\n*** SQLException caught ***\n" + ex.getMessage());
    } catch (Exception e) {
        System.out.print("Exception ---" + e.getStackTrace());
    }
```

Output Screenshot:



12) show_suppliers:

Description: The following function helps to show suppliers details.

Function:

```
function show_suppliers
    return ref_cursor is
    rcsupplier ref_cursor;
    begin
    open rcsupplier for
    select * from suppliers;
    return rcsupplier;
    end;
```

Front-end calling code Screenshot:

```
ublic void show_suppliers() {
    try {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        Connection con = DriverManager.getConnection(
                "jdbc:oracle:thin:@localhost:1521:xe", "system", "Nimisha123#");
        CallableStatement cs = con.prepareCall("begin ? := procesureandfunction.show_suppliers(); end;")
        cs.registerOutParameter(1, OracleTypes.CURSOR);
        cs.execute();
        ResultSet rs = (ResultSet) cs.getObject(1);
        ResultSetMetaData rsmd = rs.getMetaData();
        String col_names = rsmd.getColumnName(1) + "\t" + rsmd.getColumnName(2) + "\t" + rsmd.getColumnN
        textArea.append(col_names);
        while (rs.next()) {
            textArea.append(rs.getString(1) + "\t" + rs.getString(2) + "\t" + rs.getString(3) + "\t" + r
            );
        }
        cs.close();
        con.close();
    } catch (SQLException ex) {
        JOptionPane.showMessageDialog(null, "Something went Wrong!!");
        System.out.println("\n*** SQLException caught ***\n" + ex.getMessage());
    } catch (Exception e) {
        System.out.print("Exception ---" + e.getStackTrace());
    }
```

Output Screenshot:

Display Functionalities

SID	NAME	CITY	TELEPHONE#	EMAIL
S1	SuperMarket	Vastral	966-555-3215	supermarket@rb.com
S2	BigMark	ramdevnagar	956-555-5643	bigmark@rb.com
S3	BigStore	madhuli	936-555-8976	bigstore@rb.com

13) show_supplies:

Description: The following function helps to show supplies details of the product.

Function:

```
function show_supplies
   return ref_cursor is
   rcsupplies ref_cursor;
   begin
   open rcsupplies for
   select * from supplies;
   return rcsupplies;
   end;
```

Front-end calling code Screenshot:

```java
ıblic void show_supplies() {
    try {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        Connection con = DriverManager.getConnection(
                "jdbc:oracle:thin:@localhost:1521:xe", "system", "Nimisha123#");
        CallableStatement cs = con.prepareCall("begin ? := procesureandfunction.show_supplies(); end;");
        cs.registerOutParameter(1, OracleTypes.CURSOR);
        cs.execute();
        ResultSet rs = (ResultSet) cs.getObject(1);
        ResultSetMetaData rsmd = rs.getMetaData();
        String col_names = rsmd.getColumnName(1) + "\t" + rsmd.getColumnName(2) + "\t" + rsmd.getColumnN
        textArea.append(col_names);
        while (rs.next()) {
            textArea.append(rs.getString(1) + "\t" + rs.getString(2) + "\t" + rs.getString(3) + "\t" + r
            );
        }
        cs.close();
        con.close();
    } catch (SQLException ex) {
        JOptionPane.showMessageDialog(null, "Something went Wrong!!");
        System.out.println("\n*** SQLException caught ***\n" + ex.getMessage());
    } catch (Exception e) {
        System.out.print("Exception ---" + e.getStackTrace());
    }
```

Output Screenshot:

Display Functionalities

SUP#	PID	SID	SDATE	QUANTITY
100	P1	S1	2019-08-20	61
101	P2	S1	2019-10-01	8
102	P3	S1	2019-09-18	21
103	P4	S2	2019-09-18	115
104	P5	S2	2019-09-19	8
105	P6	S2	2019-08-12	15
106	P7	S3	2019-10-12	51
107	P8	S3	2019-09-12	8
108	P9	S3	2019-10-18	23
109	P5	S3	2019-08-23	4

TRIGGERS:

In this Section, we have first written the code of SQL trigger and then attached the output Screenshot after the trigger has been fired.

1.customers

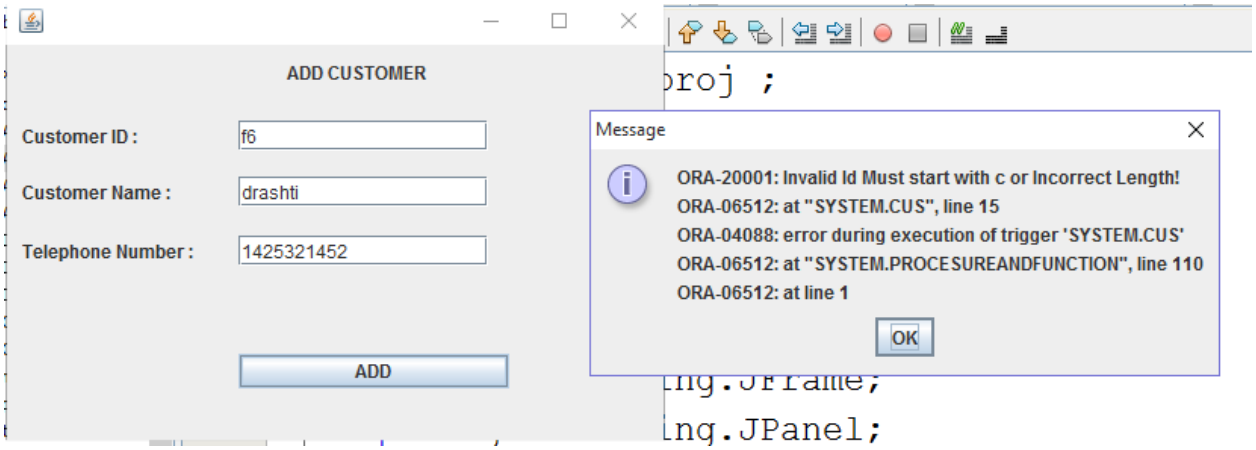
Description: This trigger is fired when you write less than 10 numbers in telephone number and if the customer id's first character is not started with c or C and it's length less than 2.

Trigger:

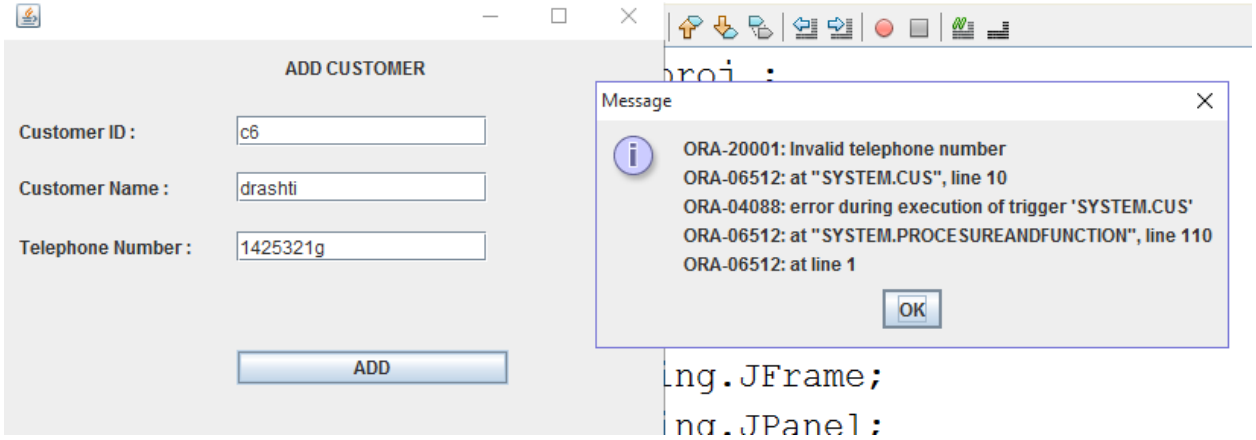
```
CREATE OR REPLACE TRIGGER cus BEFORE

   INSERT OR UPDATE ON customers

   FOR EACH ROW

declare act varchar2(50);

BEGIN

   IF( length(:new.telephone#) < 10 ) THEN

      raise_application_error(-20001, 'Invalid telephone number');

   END IF;

   IF (length(:new.cid) <= 1) THEN

      raise_application_error(-20001, 'Incorrect Length!');

   END IF;

   SELECT SUBSTR( :new.cid, 1, 1 ) SUBSTRING into act FROM dual;

   IF (act != 'c' or act != 'C') THEN

      raise_application_error(-20001, 'Invalid Id Must start with c or Incorrect Length!');

   END IF;

END;

/
```

Trigger Firing and output Screenshot:

(1) when customer id is incorrect:



(2) IF PHONE NUMBER IS NOT OF 10 DIGITS:



2. Purchases

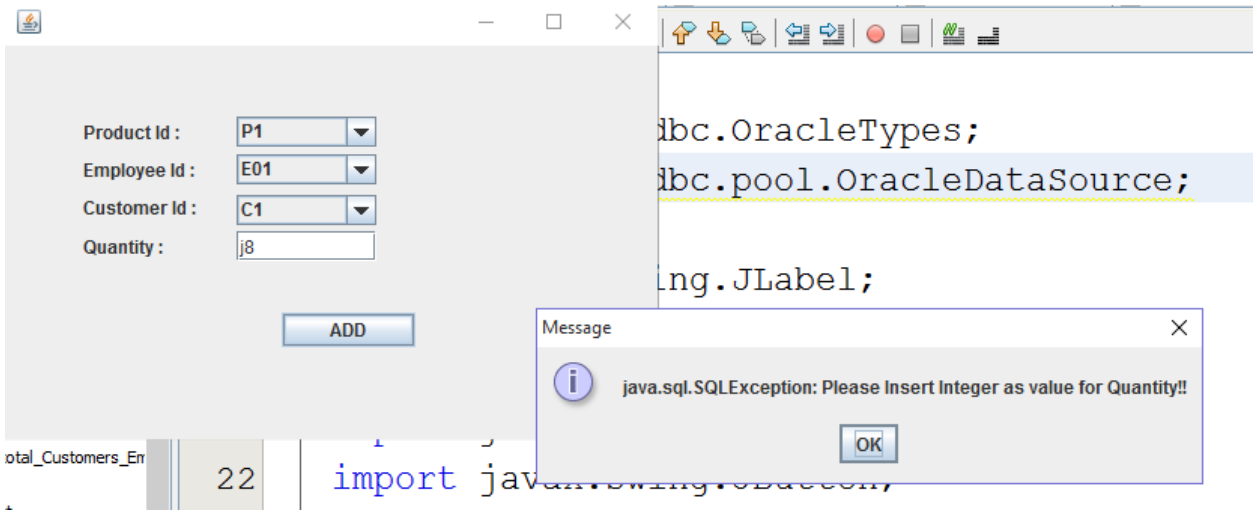
Description: when you insert or update on purchase row, if quantity is negative then this trigger will fired.

Trigger:

```
CREATE OR REPLACE TRIGGER pur BEFORE

   INSERT OR UPDATE ON purchases

   FOR EACH ROW

BEGIN

   IF ( :new.qty < 0 ) THEN

      raise_application_error(-20001, 'Please Insert Integer as value for Quantity!!');
```

```
    END IF;
END;
/
```

Trigger Firing and output Screenshot:



3. Employees

Description: This trigger is fired when you write less than 10 numbers in telephone number, invalid email and if the Employee id's first character is not started with e or E and it's length less than 2.

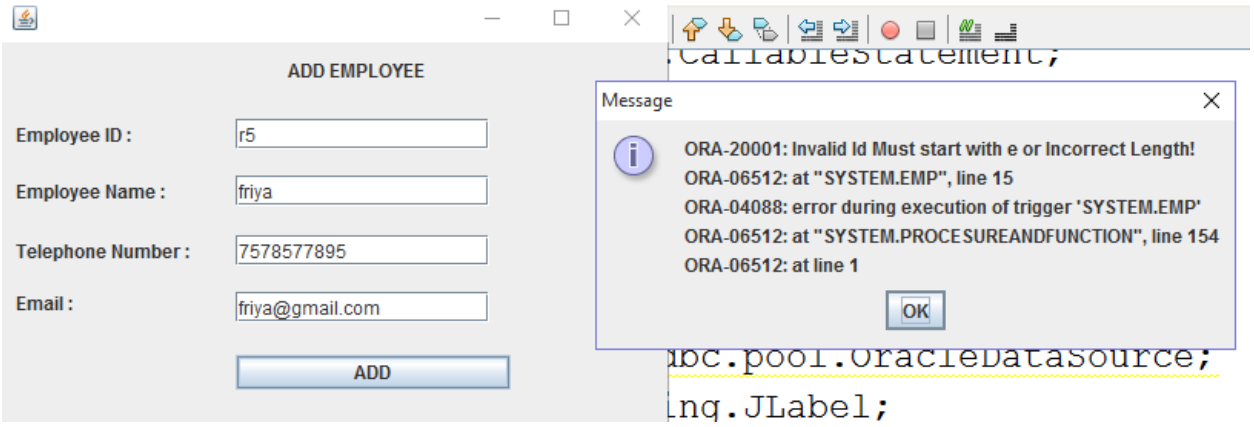
Trigger:

```
CREATE OR REPLACE TRIGGER emp BEFORE
   INSERT OR UPDATE ON employees
   FOR EACH ROW
declare act varchar2(50);
BEGIN

   IF( length(:new.telephone#) < 10 ) THEN
      raise_application_error(-20001, 'Invalid telephone number');
```

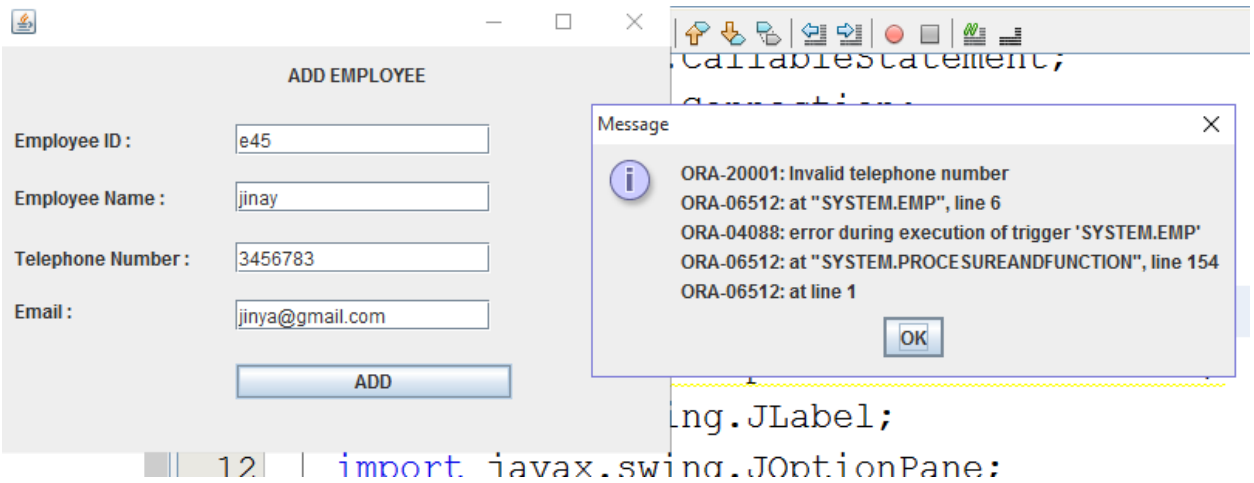
```
    END IF;

    IF (length(:new.eid) <= 1) THEN
        raise_application_error(-20001, 'Incorrect Length!');
    END IF;

    SELECT SUBSTR( :new.eid, 1, 1 ) SUBSTRING into act FROM dual;

    IF (act != 'e' or act != 'E') THEN
        raise_application_error(-20001, 'Invalid Id Must start with e or Incorrect Length!');
    END IF;

    IF (length(:new.email) != '^[a-zA-Z0-9_+&*-]+(?:\\."+
                        "[a-zA-Z0-9_+&*-]+)*@" +
                        "(?:[a-zA-Z0-9-]+\\.)+[a-z" +
                        "A-Z]{2,7}$' ) THEN
        raise_application_error(-20001, 'Invalid email address');
    END IF;
END;
/
```

Trigger Firing and output Screenshot:

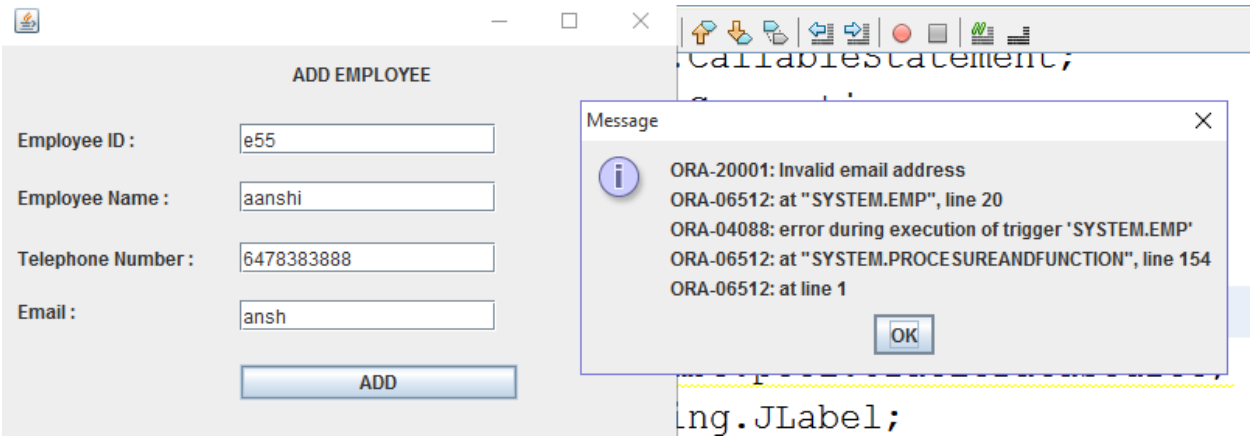
(1) when employee id is incorrect:



(2) IF PHONE NUMBER IS NOT OF 10 DIGITS:



(3) IF USER ENTERS INVALID EMAIL(CHECK EMAIL SYNTAX):-



4.suppliers

Description: This trigger is fired when you write less than 10 numbers in telephone number, invalid email and if supplier id's first character is not started with s or S and it's length less than 2.

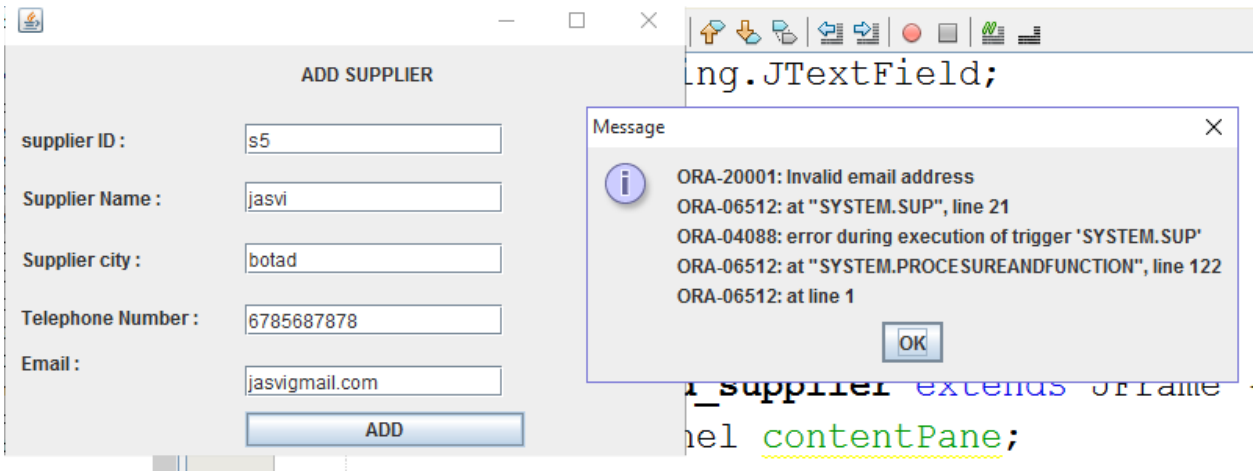
Trigger:

```
CREATE OR REPLACE TRIGGER sup BEFORE

   INSERT OR UPDATE ON suppliers

   FOR EACH ROW

declare act varchar2(50);

BEGIN



   IF( length(:new.telephone#) < 10 ) THEN

      raise_application_error(-20001, 'Invalid telephone number');

   END IF;



   IF (length(:new.sid) <= 1) THEN

      raise_application_error(-20001, 'Incorrect Length!');

   END IF;



   SELECT SUBSTR( :new.sid, 1, 1 ) SUBSTRING into act FROM dual;



   IF (act != 's' or act != 'S') THEN

      raise_application_error(-20001, 'Invalid Id Must start with s or Incorrect Length!');

   END IF;



   IF (:new.email != '^[a-zA-Z0-9_+&*-]+(?:\\."+

                        "[a-zA-Z0-9_+&*-]+)*@" +

                        "(?:[a-zA-Z0-9-]+\\.)+[a-z" +
```

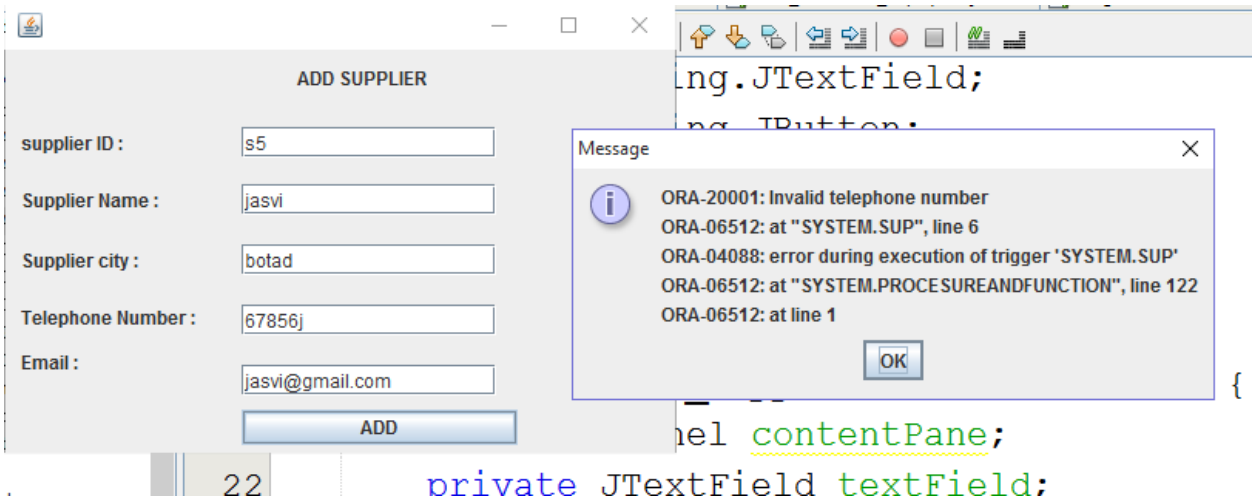
```
                    "A-Z]{2,7}$' ) THEN
            raise_application_error(-20001, 'Invalid email address');
        END IF;
    END;
    /
```

Trigger Firing and output Screenshot:

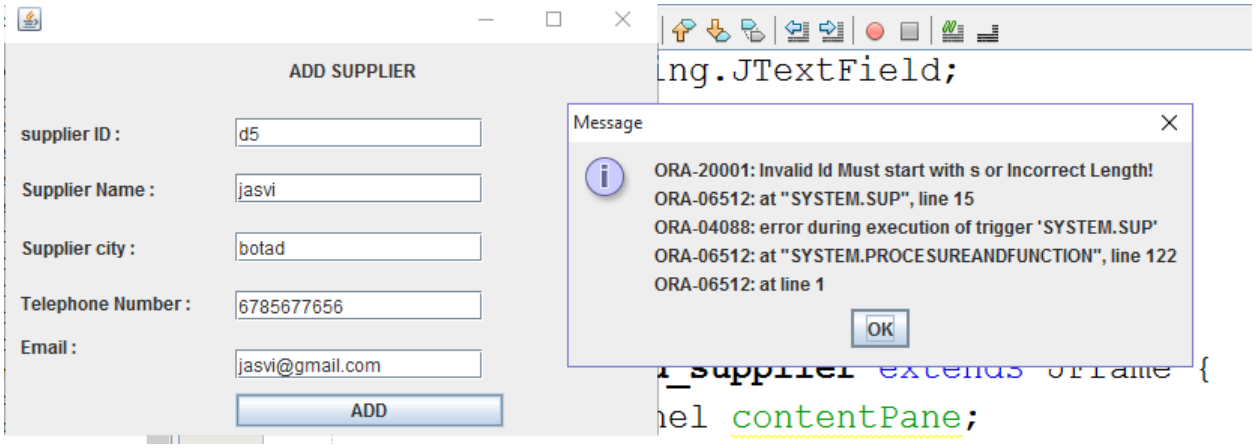
(1) IF USER ENTERS INVALID EMAIL:



(2) IF PHONE NUMBER IS LESS THAN 10 DIGITS:



(3) when supplier id is incorrect:



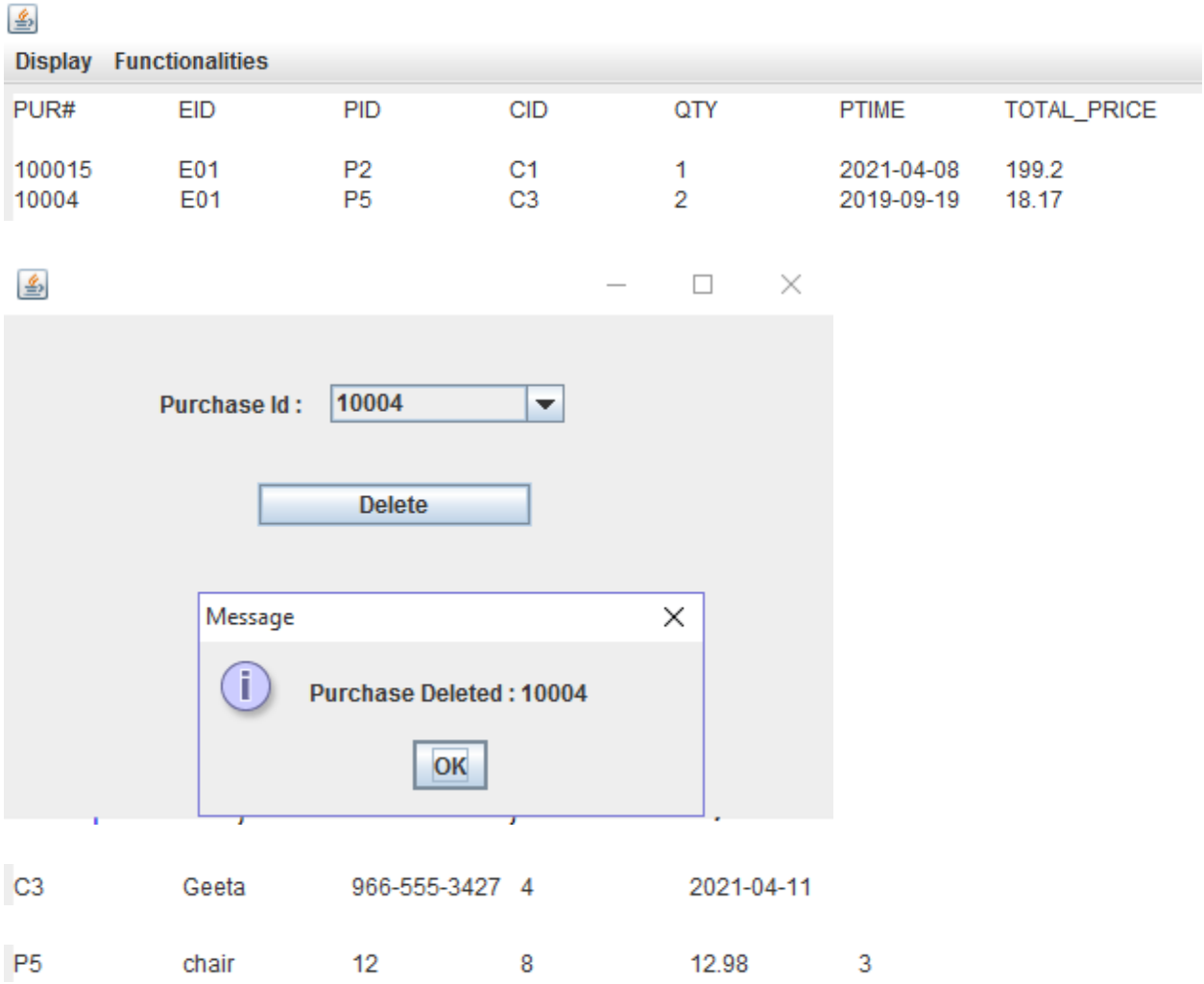
5. Delete trigger

Description: if we delete purchases row then this trigger will update the value of quantity number in products table. Because one customer returns the product then this is required to change the number of quantity details. And also update the value of visit_made to one and last visit date to current date in customer table because customer visits one more time into stationery.

Trigger:

```
create or replace trigger delete_trigger

after delete on purchases

for each row

declare

begin

update products set qoh = qoh + :old.qty where pid = :old.pid;

update customers set visits_made = visits_made + 1, last_visit_date =
to_char(sysdate,'DD-MON-YY')  where cid = :old.cid;

end;

/
```

Trigger Firing and output Screenshot:



6.purchase_made

Trigger:

```
create or replace trigger purchase_made after insert on purchases
for each row
Declare
new_date date;
visits_no number(4);
qoh_temp number(5);
final_qoh number(5);
Qoh_th number(4);
```

```
M number(5);
M_temp number(5);
Sid_no char(2);
Begin
Select last_visit_date into new_date from customers where cid= :new.cid;
Select visits_made into visits_no from customers where cid= :new.cid;
IF new_date != SYSDATE THEN
visits_no := visits_no+1;
Update customers set last_visit_date= to_char(sysdate,'DD-MON-YY') where cid= :new.cid;
Update customers set visits_made = visits_no where cid= :new.cid;
END IF;
Select qoh into qoh_temp from products where pid = :new.pid;
Select qoh_threshold  into qoh_th from products where pid = :new.pid;
IF qoh_temp < qoh_th THEN
M := qoh_th - qoh_temp + 1;
M_temp := M +qoh_temp +10;
final_qoh := qoh_temp + M_temp;
Select sid into Sid_no from supplies where pid = :new.pid and ROWNUM=1
Order by sid;
Insert into supplies values(sup_sequence.NEXTVAL,:new.pid, sid_no, sysdate, M_temp);
dbms_output.put_line('new qoh=' || final_qoh);
Update products set qoh = final_qoh where pid=:new.pid;
END IF;
end;
/
```

7. Insert and update trigger

Description: After we add into the customers table the new inserted detail shows in the log table like log number generated by sequence, user is system, here insert operation performed, date column shows current date when you add customer detail and tuple key will become customers id.

Trigger:

```
create or replace trigger t

after insert on customers

for each row

declare

 begin

 insert into logs
values(log_sequence.nextval,user,'insert',to_char(sysdate,'DD-MON-YY'),'customers',:new.cid);

end;

/
```

Trigger Firing and output Screenshot:

Display Functionalities

LOG#	USER_NAME	OPERATION	OP_TIME	TABLE_NAME	TUPLE_KEY
10033	SYSTEM	insert	2021-04-10	customers	c33

Description: After we update the last visit date in the customer table the new updated detail shows in the log table like log number generated by sequence, user is system, here update operation performed, date column shows current date when you update quantity number of product and tuple key will become customer id.

Trigger:

```
create or replace trigger t1

after update of last_visit_date on customers

for each row
```

```
begin

insert into logs
values(log_sequence.nextval,user,'update',to_char(sysdate,'DD-MON-YY'),'customers',:
new.cid);

end;

/
```

Trigger Firing and output Screenshot:

Display Functionalities

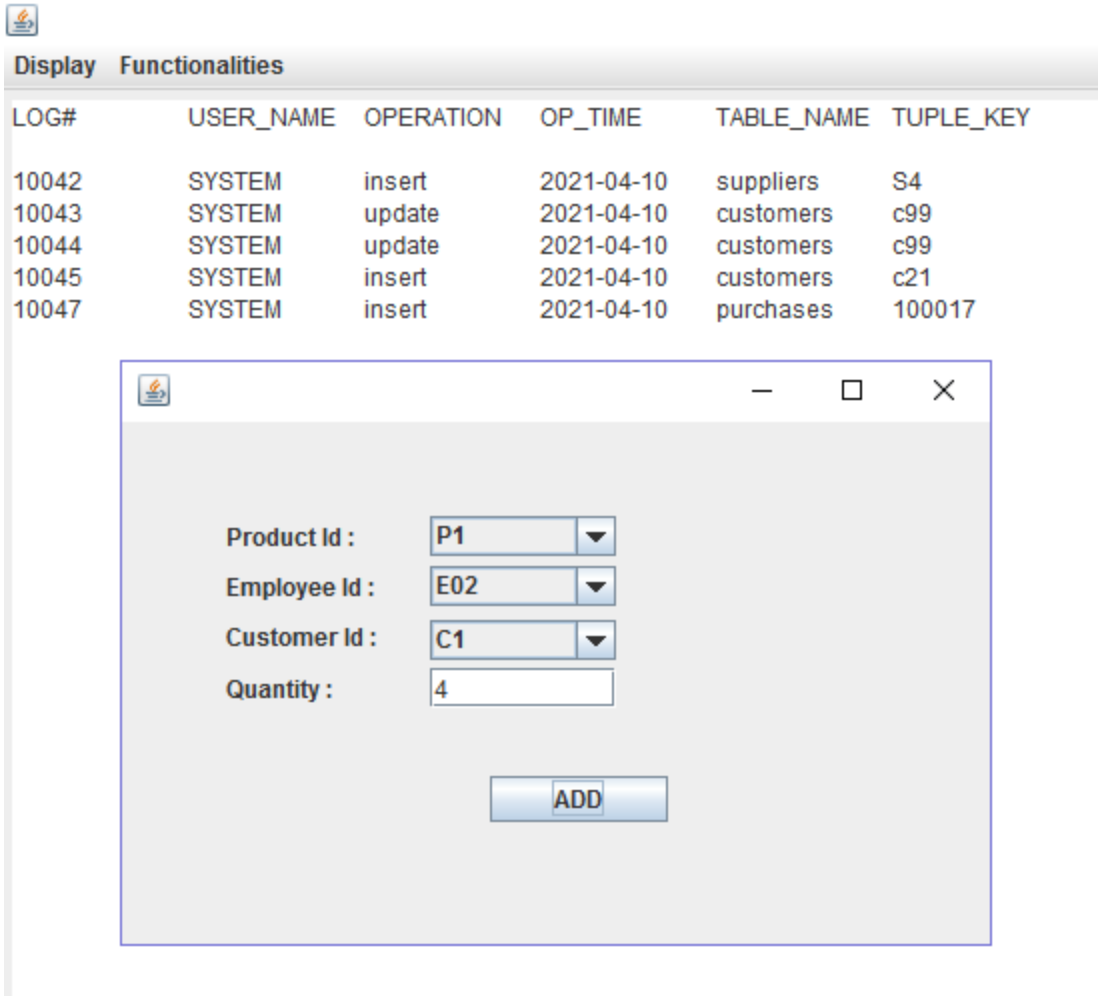
LOG#	USER_NAME	OPERATION	OP_TIME	TABLE_NAME	TUPLE_KEY
10042	SYSTEM	insert	2021-04-10	suppliers	S4
10043	SYSTEM	update	2021-04-10	customers	c99

Description: After we add into the purchases table the new inserted detail shows in the log table like log number generated by sequence, user is system, here insert operation performed, date column shows current date when you add purchase detail and tuple key will become purchase number it will generate by sequence.

Trigger:

```
create or replace trigger t2

after insert on purchases

for each row

begin

insert into logs values(log_sequence.nextval, user,
'insert',to_char(sysdate,'DD-MON-YY'), 'purchases', :new.pur#);

end;

/
```

Trigger Firing and output Screenshot:



Description: After we update quantity number in product table the new updated detail shows in the log table like log number generated by sequence, user is system, here update operation performed, date column shows current date when you update quantity number of product and tuple key will become product id.

Trigger:

```
create or replace trigger t3

after update of qoh on products

for each row

begin

insert into logs
values(log_sequence.nextval,user,'update',to_char(sysdate,'DD-MON-YY'),'products',:new.pid);
```

end;

/

Trigger Firing and output Screenshot:

Display Functionalities

LOG#	USER_NAME	OPERATION	OP_TIME	TABLE_NAME	TUPLE_KEY
10042	SYSTEM	insert	2021-04-10	suppliers	S4
10043	SYSTEM	update	2021-04-10	customers	c99
10044	SYSTEM	update	2021-04-10	customers	c99
10045	SYSTEM	insert	2021-04-10	customers	c21
10047	SYSTEM	insert	2021-04-10	purchases	100017
10048	SYSTEM	update	2021-04-10	products	P1

Description: After we add into the supplier table the new inserted detail shows in the log table like log number generated by sequence, user is system, here insert operation performed, date column shows current date when you insert supplier detail and tuple key will become supplier id.

Trigger:

create or replace trigger t4

after insert on suppliers

for each row

begin

insert into logs
values(log_sequence.nextval,user,'insert',to_char(sysdate,'DD-MON-YY'),'suppliers',:new.sid);

end;

/

Trigger Firing and output Screenshot:

Display Functionalities

LOG#	USER_NAME	OPERATION	OP_TIME	TABLE_NAME	TUPLE_KEY
10042	SYSTEM	insert	2021-04-10	suppliers	S4

ADD SUPPLIER

supplier ID : S4

Supplier Name : Dhruvi

Supplier city : Vapi

Telephone Number : 952-258-8988

Email : dhruvi.d@gamil.com

ADD