Introduction: The Restaurant Management System has been proposed to implement to replace the manual system. The main aim of this project is computerization of all processes which happens in restaurants. It is a distributed database system for creating a selective retrieve of information, for subsequent analysis, manipulation and application.

Relations and Sites:

Global Relations:

CUSTOMERS (Customer_id, Customer_name, Phone, Location)

MEALS (Meal_id, Date_of_meal, Cost_of_meal, Customer_id, Staff_id)

STAFF_RULE (staff_rule_code, rule_describtion)

STAFF (staff_id, staff_name, staff_salary, staff_rule_code)

MENU_ITEM (menu_item_id, menu_item_name, menu_item_price)

MEAL_DISH (meal_id, menu_item_id)

Fragmentation Schema:

Customer1 = $SL_{\text{Location = ‘Chittagong’}}$ (Customers)

Customer2 = $SL_{\text{Location = ‘Dhaka’}}$ (Customers)

Staff1 = $SL_{\text{Location = ‘Chittagong’}}$ (Staff)

Staff2 = $SL_{\text{Location = ‘Dhaka’}}$ (Staff)

Transaction1 = $SL_{\text{Location = ‘Chittagong’}}$ (Meals)

Transaction2 = $SL_{\text{Location = ‘Dhaka’}}$ (Meals)

Allocation Schema:

There are two sites.

Site1 (Chittagong): Customer1, Staff1, Transaction1

Site2 (Dhaka): Customer2, Staff2, Transaction2

Individual Contribution:

ID: 16.01.04.035 (Tonmoy Mohajan)

1. Procedures, Functions, Triggers, Exception, Views:

Procedure:

1. Bonus_Procedure (staff_rule_code, percentage): The procedure was created to provide the bonus to specific staffs category-wise.

2. ctgCustomerProc: This procedure will show all customers details at site 1. (Chittagong).

3. DhkCustomerProc: This procedure will show all customers details at site 2. (Dhaka).

4. Search_customer_procedure (customer_id): This will show details of customers when the customer_id is provided.

5. Search_Which_staffed_served_specific_customer(customer_id): This procedure will show which staff served the food to the given id's customer.

6. salaryCount_function (salary): Show all staffs whose salary is below 20,000:

7. Menu_item_price_update (item_id, percentage): This procedure will increase the menu item price.

8. Generate_bill (customer_id, date): This function will show the total cost of customer in a given date.

Functions:

1. countStaff (salary): This function will show details of staffs who gets more than the given salary.

2. FindCustomer_name(customer_id): This function will return the customer name.

Triggers:

1. Insert_trigger: In inserting the customers, the trigger will insert it according to its respective sites.

2. Update_delete_trigger: When the location of the customers will be updated then the trigger will change the location and put it to it's new site and delete from the previous site.

Exceptions:

1) No_customer_found: It is a predefined exception. In customer_search function when the customer_id don't present in the main table then it throws a expression that no customer found on that id!

2) Wrong_input: It is a user defined exception. In staff_bonus procedure when the wrong input is given then it throws an exception to choose to correct one.

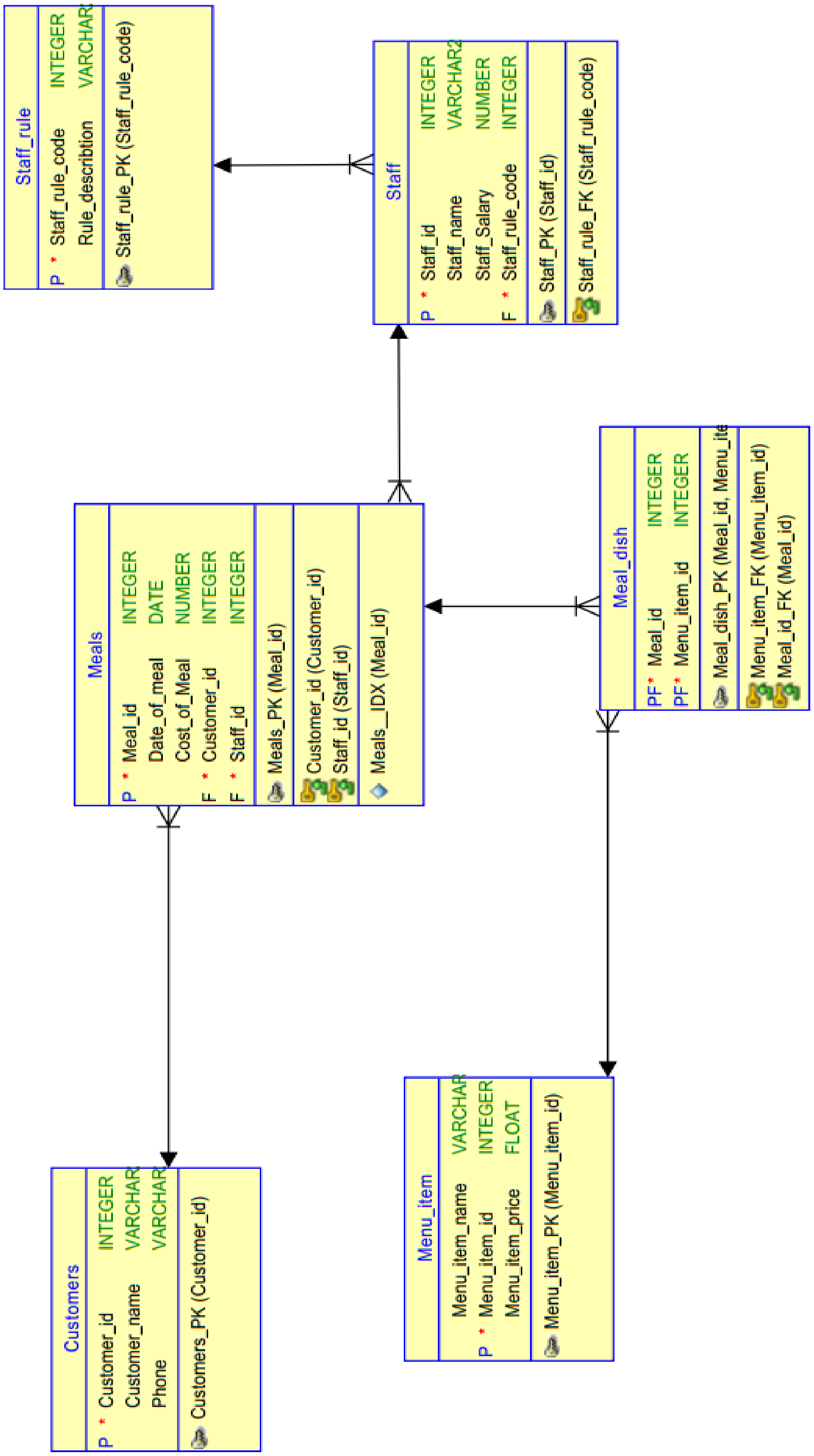
Views:

1) CtgCustomer: The view is created to store site1 customer's information.

2) DhkCustomer: The view is created to store site2 customer's information.

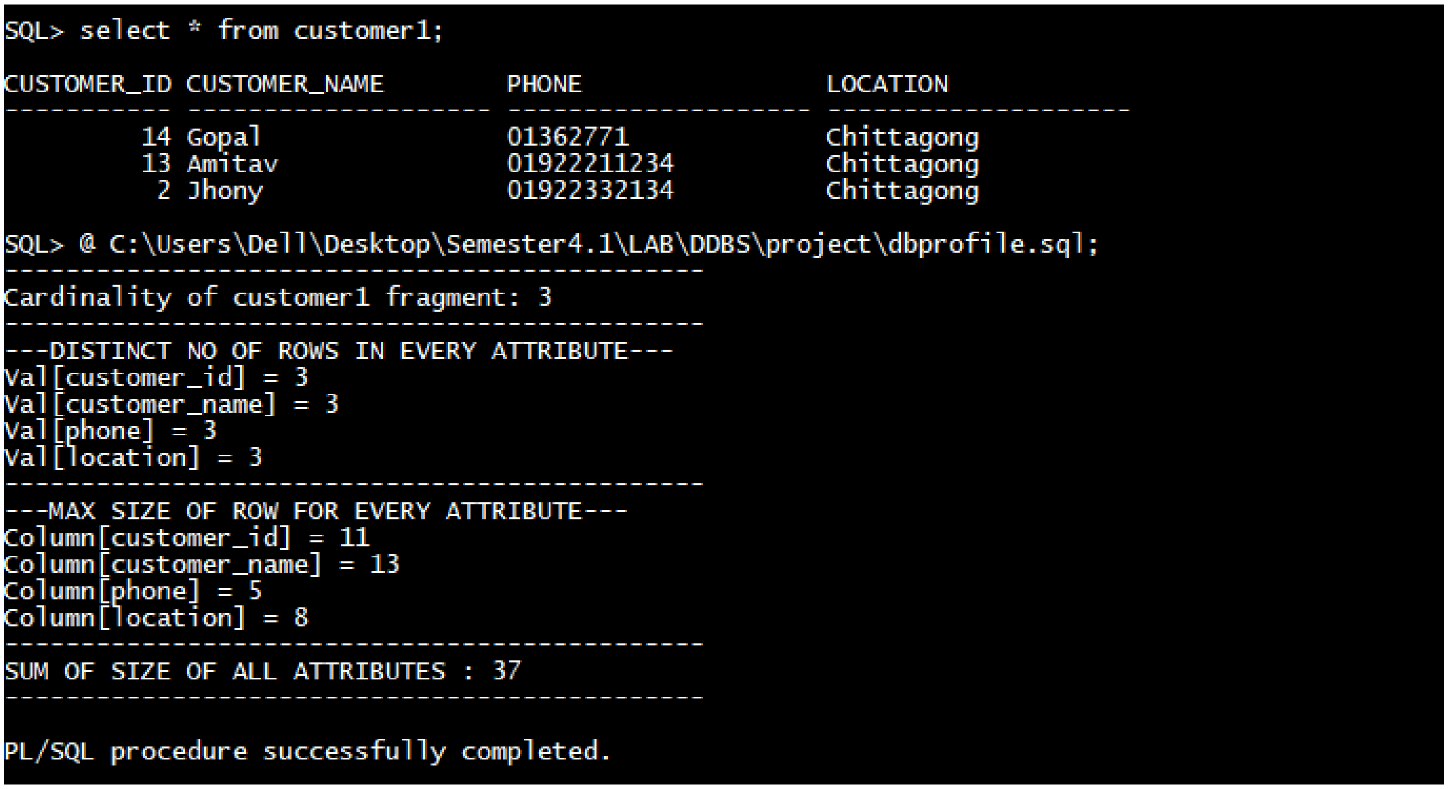
3) topTransactions: This view is created to store the top 3 transaction records.

2. <u>ER-Diagram:</u> Entity Relationship Diagram plays a very important role in designing a database.



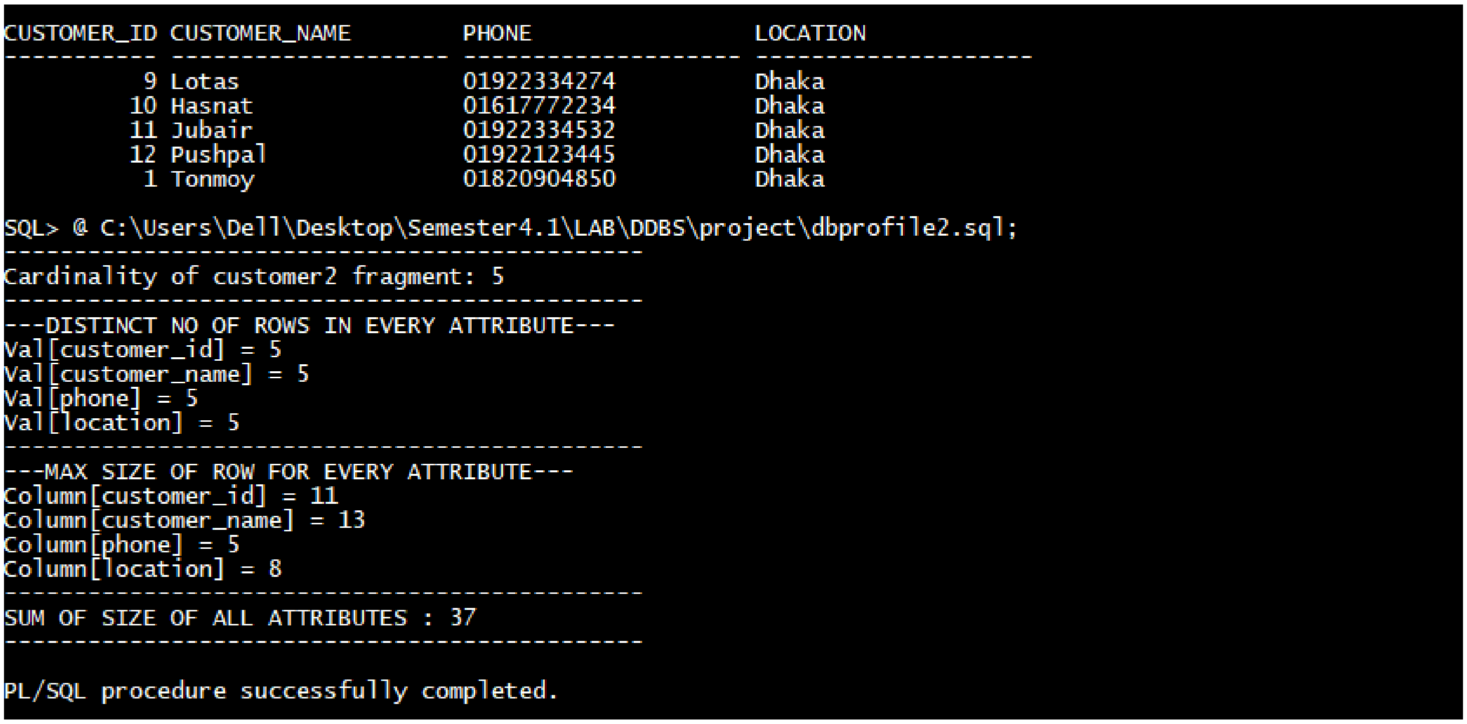
3. DBProfile:

At site-1:



```
SQL> select * from customer1;

CUSTOMER_ID CUSTOMER_NAME        PHONE                LOCATION
----------- -------------------- -------------------- --------------------
         14 Gopal                01362771             Chittagong
         13 Amitav               01922211234          Chittagong
          2 Jhony                01922332134          Chittagong

SQL> @ C:\Users\Dell\Desktop\Semester4.1\LAB\DDBS\project\dbprofile.sql;
----------------------------------------------
Cardinality of customer1 fragment: 3
----------------------------------------------
---DISTINCT NO OF ROWS IN EVERY ATTRIBUTE---
Val[customer_id] = 3
Val[customer_name] = 3
Val[phone] = 3
Val[location] = 3
----------------------------------------------
---MAX SIZE OF ROW FOR EVERY ATTRIBUTE---
Column[customer_id] = 11
Column[customer_name] = 13
Column[phone] = 5
Column[location] = 8
----------------------------------------------
SUM OF SIZE OF ALL ATTRIBUTES : 37
----------------------------------------------

PL/SQL procedure successfully completed.
```

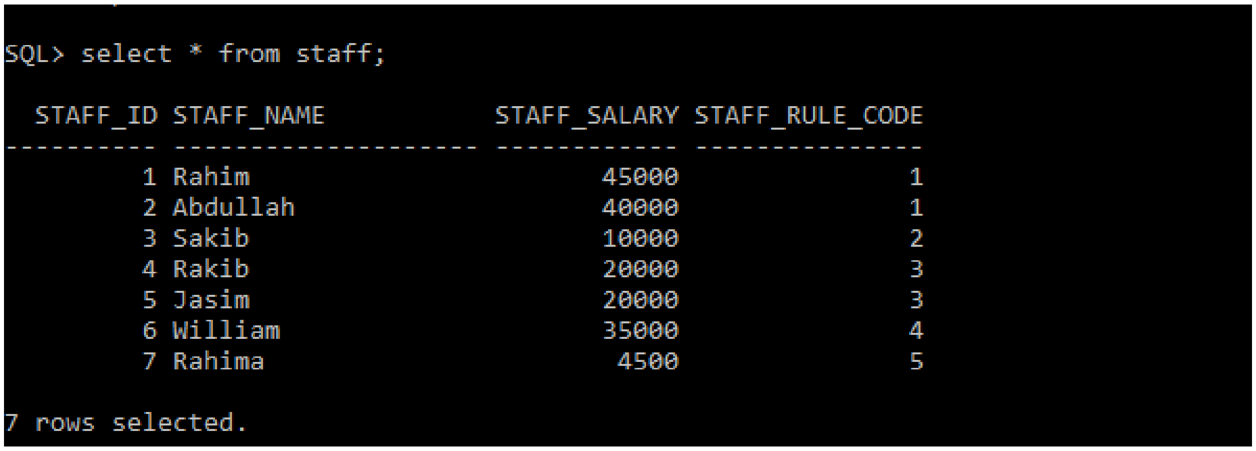
At site-2:



```
CUSTOMER_ID CUSTOMER_NAME        PHONE                LOCATION
----------- -------------------- -------------------- --------------------
          9 Lotas                01922334274          Dhaka
         10 Hasnat               01617772234          Dhaka
         11 Jubair               01922334532          Dhaka
         12 Pushpal              01922123445          Dhaka
          1 Tonmoy               01820904850          Dhaka

SQL> @ C:\Users\Dell\Desktop\Semester4.1\LAB\DDBS\project\dbprofile2.sql;
----------------------------------------------
Cardinality of customer2 fragment: 5
----------------------------------------------
---DISTINCT NO OF ROWS IN EVERY ATTRIBUTE---
Val[customer_id] = 5
Val[customer_name] = 5
Val[phone] = 5
Val[location] = 5
----------------------------------------------
---MAX SIZE OF ROW FOR EVERY ATTRIBUTE---
Column[customer_id] = 11
Column[customer_name] = 13
Column[phone] = 5
Column[location] = 8
----------------------------------------------
SUM OF SIZE OF ALL ATTRIBUTES : 37
----------------------------------------------

PL/SQL procedure successfully completed.
```

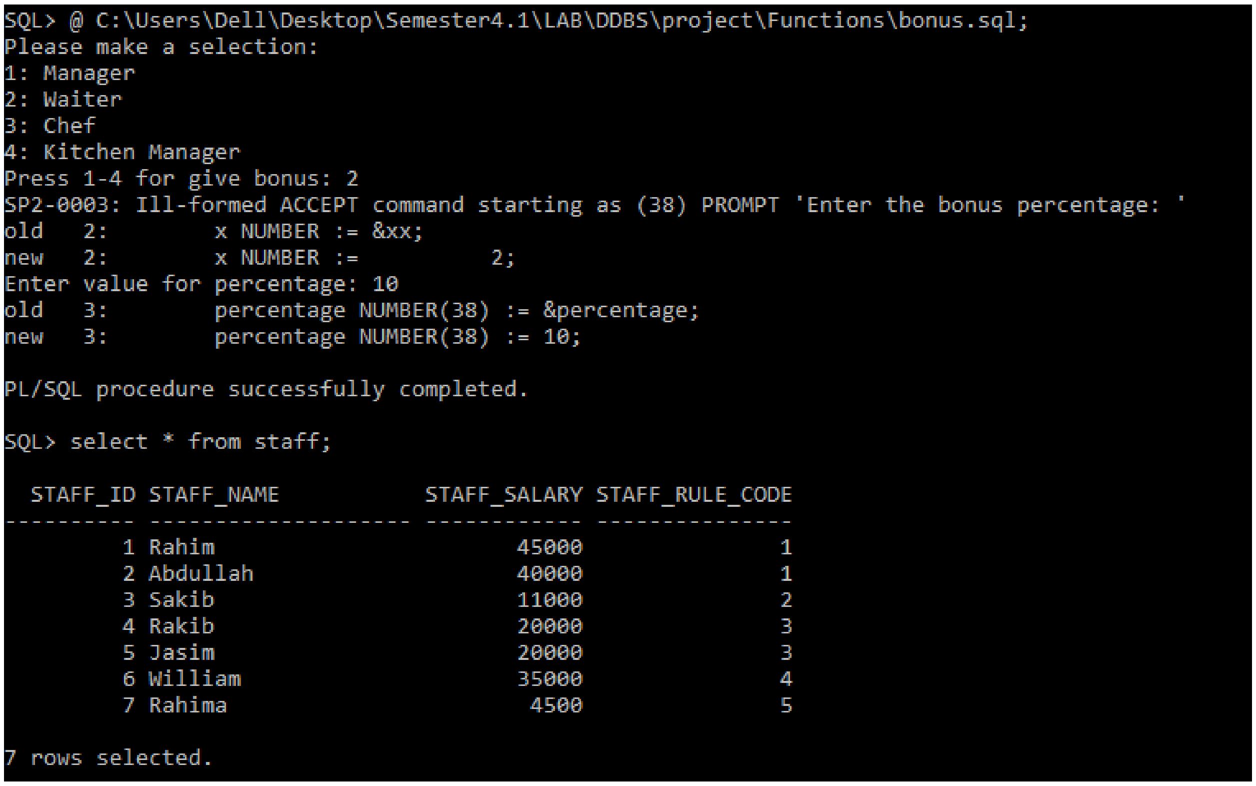
3. Function & Procedures sample output:

A. Bonus_Procedure (staff_rule_code, percentage): The procedure was created to provide the bonus to specific staffs category-wise.



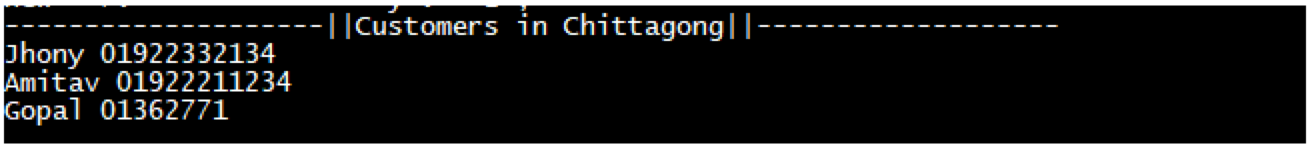
```
SQL> select * from staff;

  STAFF_ID STAFF_NAME           STAFF_SALARY STAFF_RULE_CODE
---------- -------------------- ------------ ---------------
         1 Rahim                       45000               1
         2 Abdullah                    40000               1
         3 Sakib                       10000               2
         4 Rakib                       20000               3
         5 Jasim                       20000               3
         6 William                     35000               4
         7 Rahima                       4500               5

7 rows selected.
```

After adding 10% bonus to Waiter new updated table -



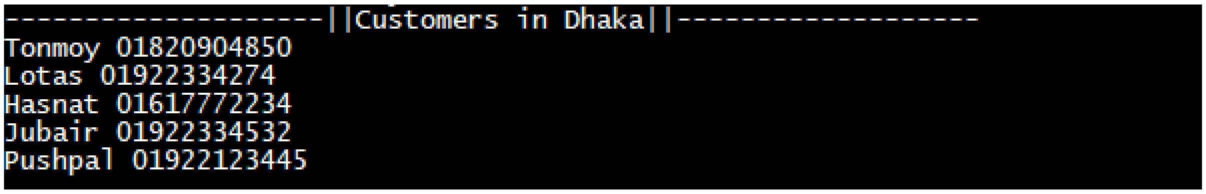
```
SQL> @ C:\Users\Dell\Desktop\Semester4.1\LAB\DDBS\project\Functions\bonus.sql;
Please make a selection:
1: Manager
2: Waiter
3: Chef
4: Kitchen Manager
Press 1-4 for give bonus: 2
SP2-0003: Ill-formed ACCEPT command starting as (38) PROMPT 'Enter the bonus percentage: '
old   2:        x NUMBER := &xx;
new   2:        x NUMBER :=         2;
Enter value for percentage: 10
old   3:        percentage NUMBER(38) := &percentage;
new   3:        percentage NUMBER(38) := 10;

PL/SQL procedure successfully completed.

SQL> select * from staff;

  STAFF_ID STAFF_NAME           STAFF_SALARY STAFF_RULE_CODE
---------- -------------------- ------------ ---------------
         1 Rahim                       45000               1
         2 Abdullah                    40000               1
         3 Sakib                       11000               2
         4 Rakib                       20000               3
         5 Jasim                       20000               3
         6 William                     35000               4
         7 Rahima                       4500               5

7 rows selected.
```

B. ctgCustomerProc: This procedure will show all customers details at site 1. (Chittagong).



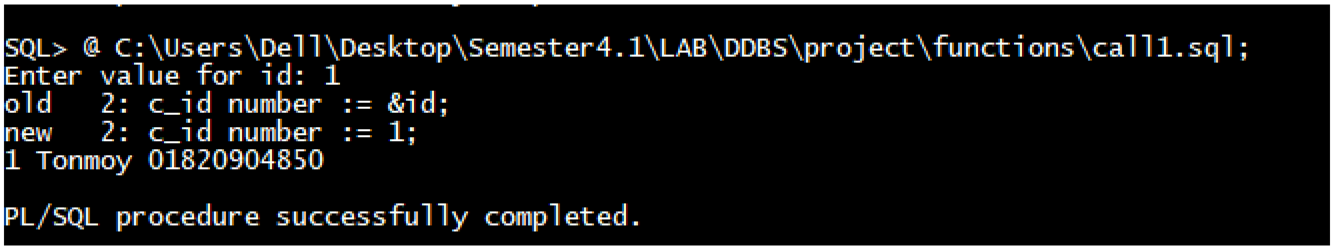
```
--------------------||Customers in Chittagong||--------------------
Jhony 01922332134
Amitav 01922211234
Gopal 01362771
```

C. DhkCustomerProc: This procedure will show all customers details at site 2. (Dhaka).



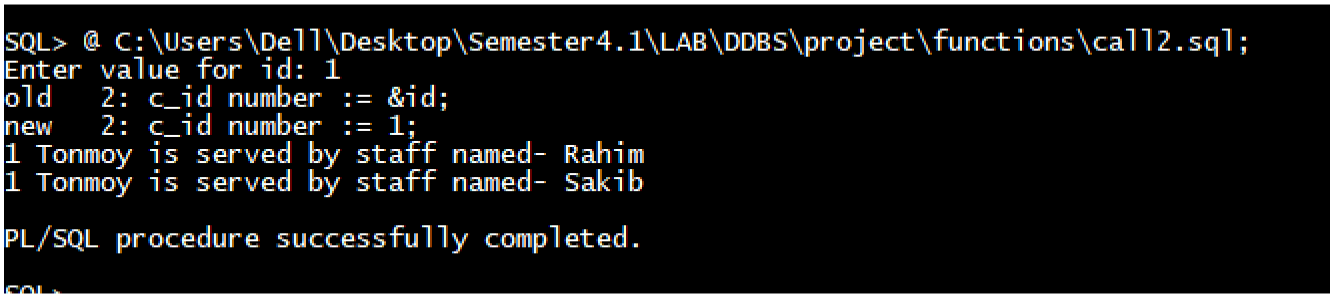
```
--------------------||Customers in Dhaka||-------------------
Tonmoy 01820904850
Lotas 01922334274
Hasnat 01617772234
Jubair 01922334532
Pushpal 01922123445
```

D. Search_customer_procedure(customer_id): This will show details of customers when the id is provided.



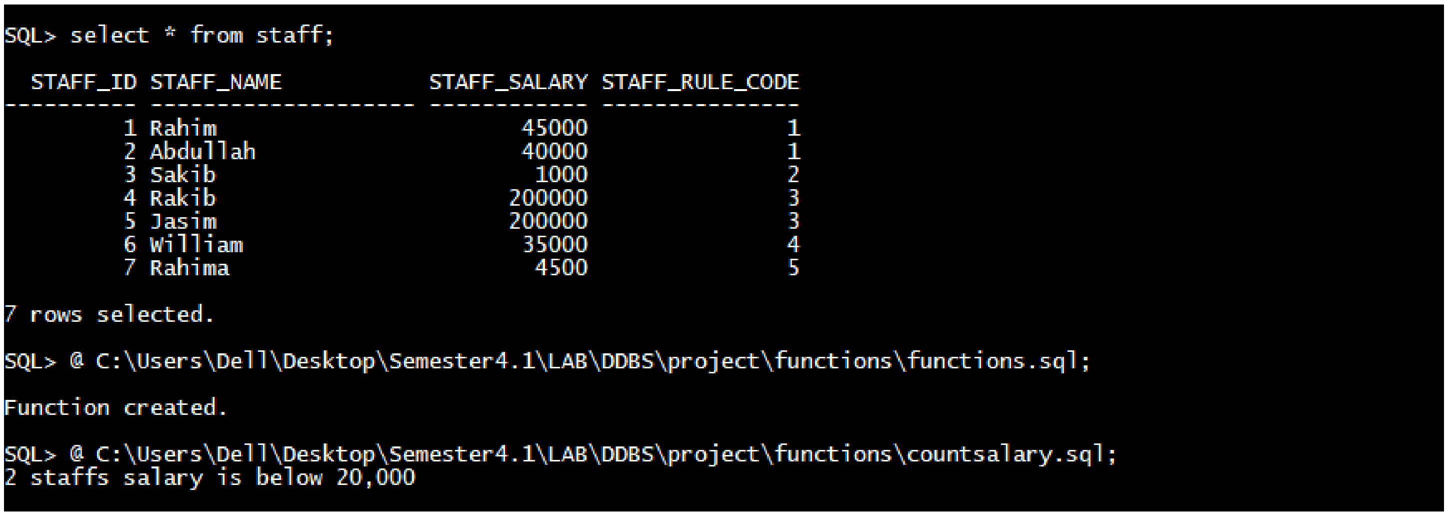
```
SQL> @ C:\Users\Dell\Desktop\Semester4.1\LAB\DDBS\project\functions\call1.sql;
Enter value for id: 1
old   2: c_id number := &id;
new   2: c_id number := 1;
1 Tonmoy 01820904850

PL/SQL procedure successfully completed.
```

E. Search_Which_staffed_served_specific_customer(customer_id): This procedure will show which staff served the food to the given id's customer.



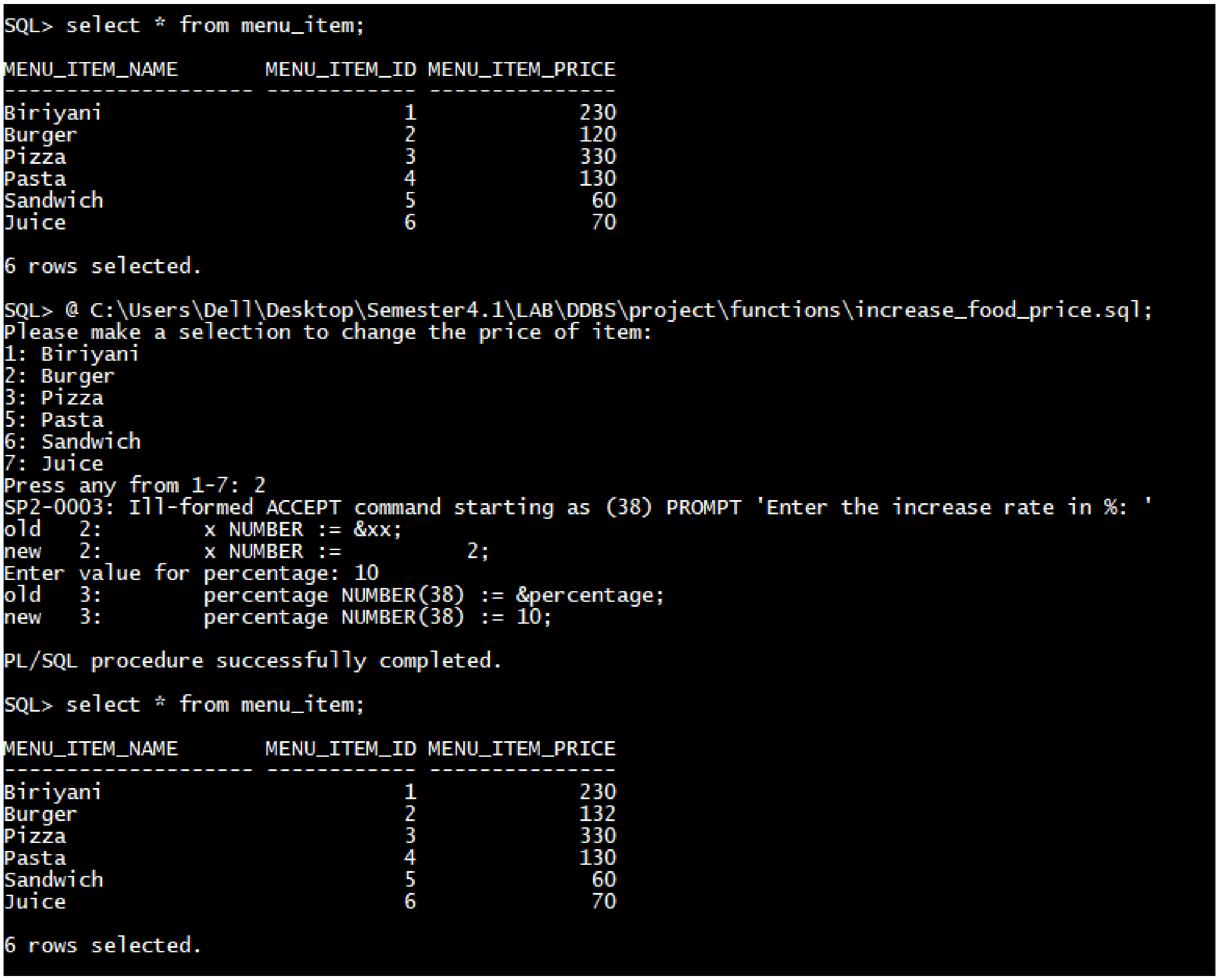
```
SQL> @ C:\Users\Dell\Desktop\Semester4.1\LAB\DDBS\project\functions\call2.sql;
Enter value for id: 1
old   2: c_id number := &id;
new   2: c_id number := 1;
1 Tonmoy is served by staff named- Rahim
1 Tonmoy is served by staff named- Sakib

PL/SQL procedure successfully completed.
```

F. salaryCount_function(salary): Show all staffs whose salary is below 20,000



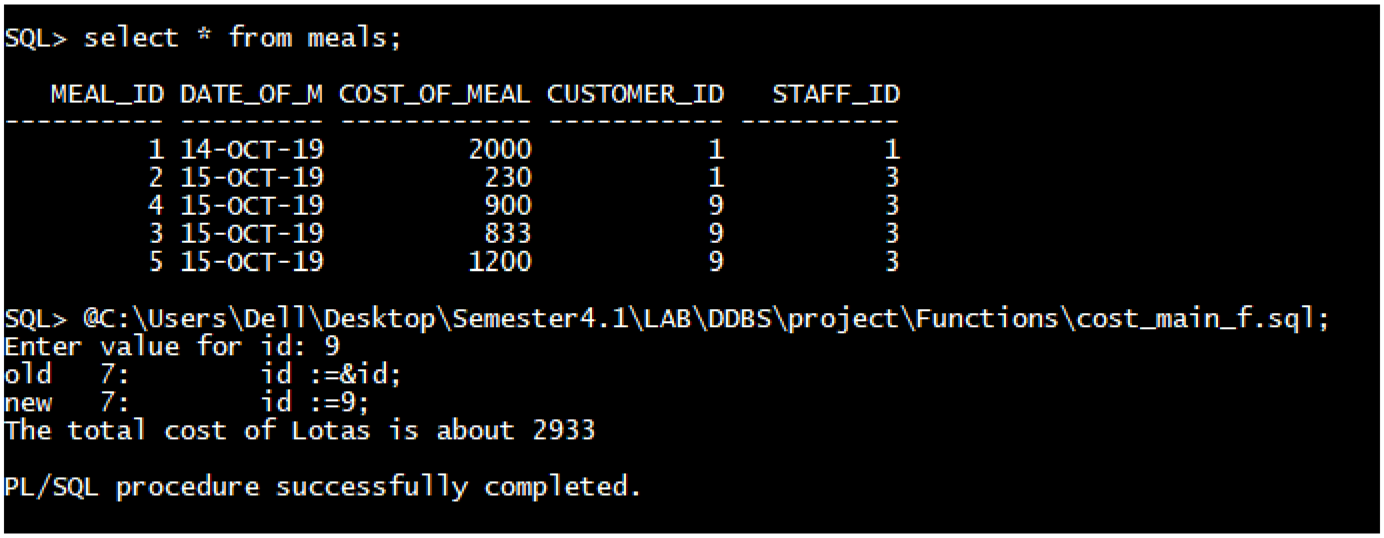
```
SQL> select * from staff;

  STAFF_ID STAFF_NAME           STAFF_SALARY STAFF_RULE_CODE
---------- -------------------- ------------ ---------------
         1 Rahim                       45000               1
         2 Abdullah                    40000               1
         3 Sakib                        1000               2
         4 Rakib                      200000               3
         5 Jasim                      200000               3
         6 William                     35000               4
         7 Rahima                       4500               5

7 rows selected.

SQL> @ C:\Users\Dell\Desktop\Semester4.1\LAB\DDBS\project\functions\functions.sql;

Function created.

SQL> @ C:\Users\Dell\Desktop\Semester4.1\LAB\DDBS\project\functions\countsalary.sql;
2 staffs salary is below 20,000
```

G. **Menu_item_price_update(item_id, percentage):** This procedure will increase the menu item price.



```
SQL> select * from menu_item;

MENU_ITEM_NAME       MENU_ITEM_ID MENU_ITEM_PRICE
-------------------- ------------ ---------------
Biriyani                        1             230
Burger                          2             120
Pizza                           3             330
Pasta                           4             130
Sandwich                        5              60
Juice                           6              70

6 rows selected.

SQL> @ C:\Users\Dell\Desktop\Semester4.1\LAB\DDBS\project\functions\increase_food_price.sql;
Please make a selection to change the price of item:
1: Biriyani
2: Burger
3: Pizza
5: Pasta
6: Sandwich
7: Juice
Press any from 1-7: 2
SP2-0003: Ill-formed ACCEPT command starting as (38) PROMPT 'Enter the increase rate in %: '
old   2:        x NUMBER := &xx;
new   2:        x NUMBER :=          2;
Enter value for percentage: 10
old   3:        percentage NUMBER(38) := &percentage;
new   3:        percentage NUMBER(38) := 10;

PL/SQL procedure successfully completed.

SQL> select * from menu_item;

MENU_ITEM_NAME       MENU_ITEM_ID MENU_ITEM_PRICE
-------------------- ------------ ---------------
Biriyani                        1             230
Burger                          2             132
Pizza                           3             330
Pasta                           4             130
Sandwich                        5              60
Juice                           6              70

6 rows selected.
```

H. Generate_bill(customer_id, date): This function will show the total cost of customer in a given date.

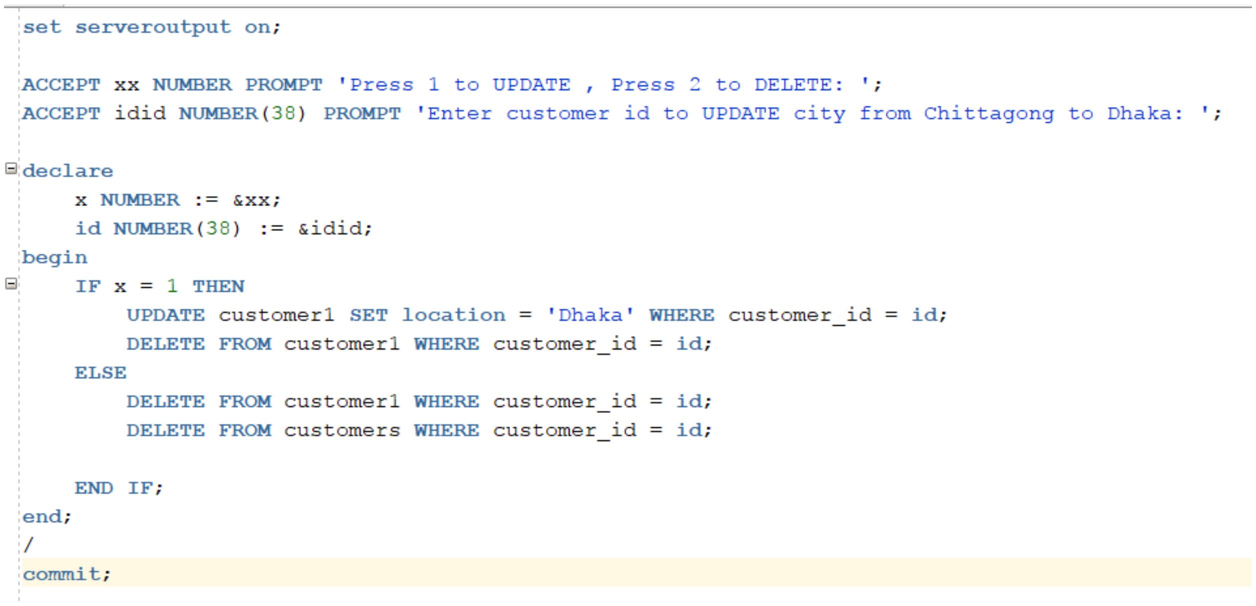


```
SQL> select * from meals;

   MEAL_ID DATE_OF_M COST_OF_MEAL CUSTOMER_ID   STAFF_ID
---------- --------- ------------ ----------- ----------
         1 14-OCT-19         2000           1          1
         2 15-OCT-19          230           1          3
         4 15-OCT-19          900           9          3
         3 15-OCT-19          833           9          3
         5 15-OCT-19         1200           9          3

SQL> @C:\Users\Dell\Desktop\Semester4.1\LAB\DDBS\project\Functions\cost_main_f.sql;
Enter value for id: 9
old   7:        id :=&id;
new   7:        id :=9;
The total cost of Lotas is about 2933

PL/SQL procedure successfully completed.
```

4. Level-3 Transparency:

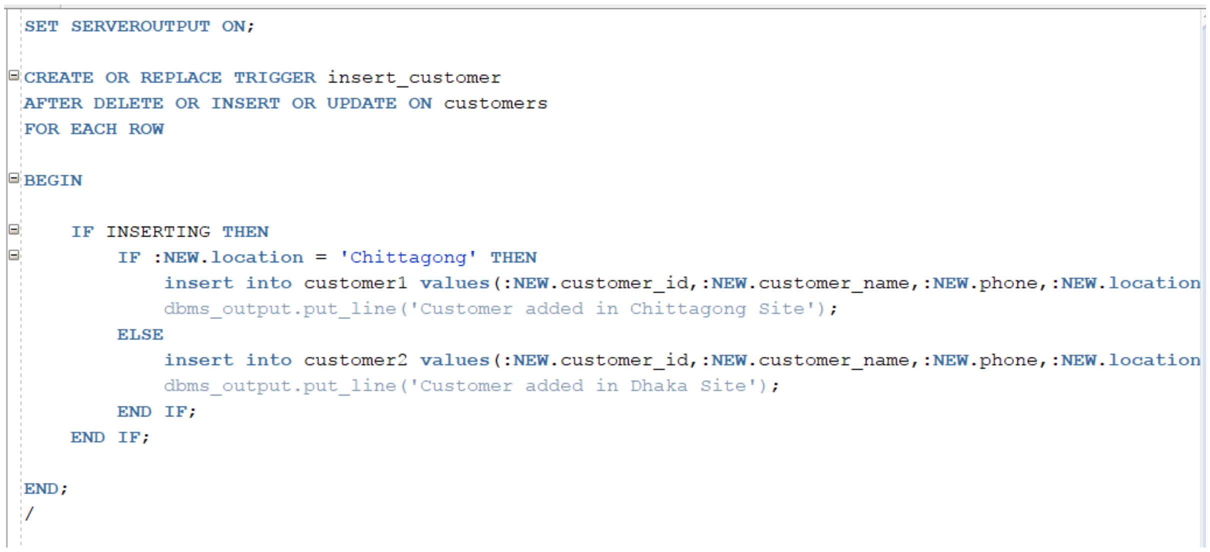
In this part the main job is to perform necessary changes when changes in a particular site is performed. Updating the location of a customer will effect on both sites. If a customer changes his location from Chittagong to Dhaka then we have to perform 3 operations

i) Insert into fragment-2 in site-2(Customer2)
ii) Update the Global Table selected ID from Chittagong to Dhaka (Customers)
iii) Delete from fragment-1 in site-1(Customer1)



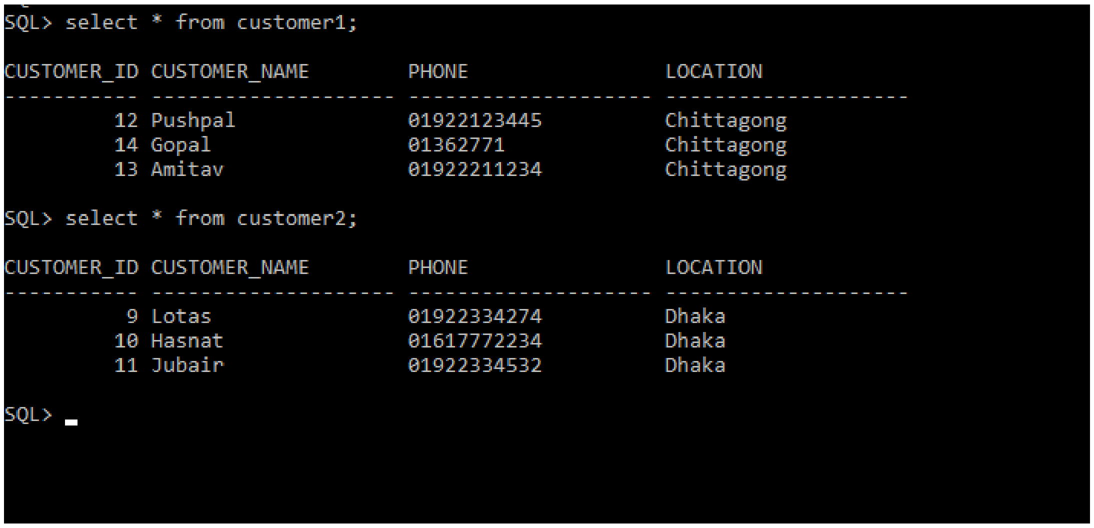
```
set serveroutput on;

ACCEPT xx NUMBER PROMPT 'Press 1 to UPDATE , Press 2 to DELETE: ';
ACCEPT idid NUMBER(38) PROMPT 'Enter customer id to UPDATE city from Chittagong to Dhaka: ';

declare
    x NUMBER := &xx;
    id NUMBER(38) := &idid;
begin
    IF x = 1 THEN
        UPDATE customer1 SET location = 'Dhaka' WHERE customer_id = id;
        DELETE FROM customer1 WHERE customer_id = id;
    ELSE
        DELETE FROM customer1 WHERE customer_id = id;
        DELETE FROM customers WHERE customer_id = id;

    END IF;
end;
/
commit;
```

➔ It is the Trigger where we inserted the newly updated value to ‘Customer1 ‘and also will update the global table.

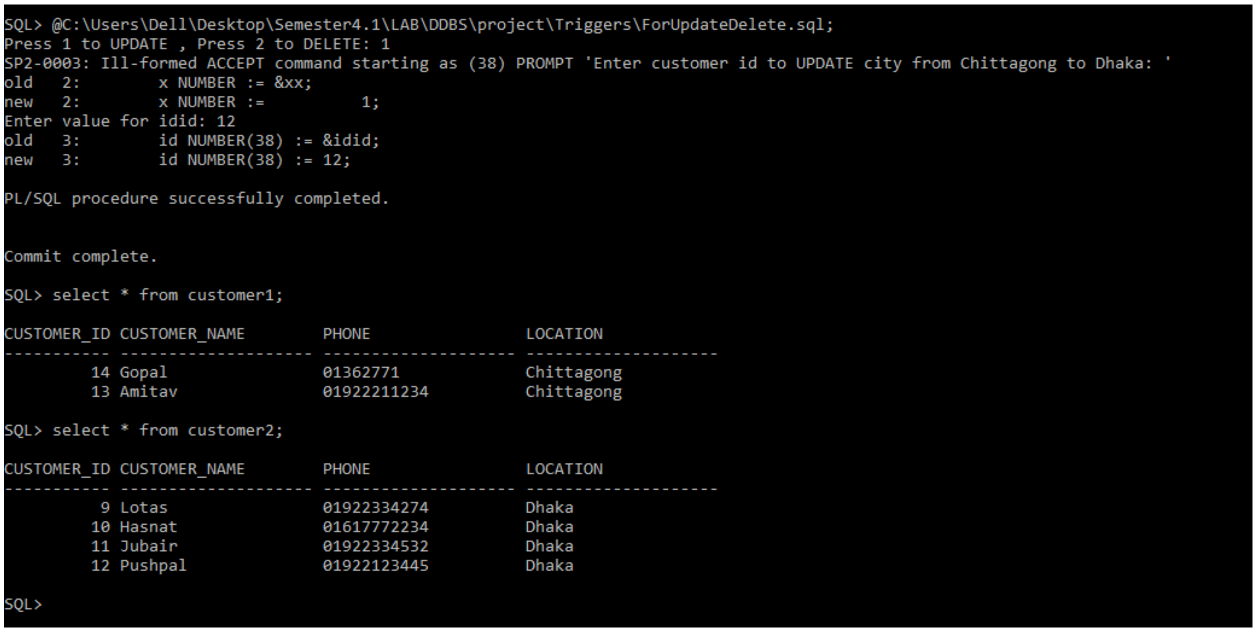


```
SET SERVEROUTPUT ON;

CREATE OR REPLACE TRIGGER insert_customer
AFTER DELETE OR INSERT OR UPDATE ON customers
FOR EACH ROW

BEGIN

    IF INSERTING THEN
        IF :NEW.location = 'Chittagong' THEN
            insert into customer1 values(:NEW.customer_id,:NEW.customer_name,:NEW.phone,:NEW.location
            dbms_output.put_line('Customer added in Chittagong Site');
        ELSE
            insert into customer2 values(:NEW.customer_id,:NEW.customer_name,:NEW.phone,:NEW.location
            dbms_output.put_line('Customer added in Dhaka Site');
        END IF;
    END IF;

END;
/
```

Result (Before updating customer1 and after updating customer1):

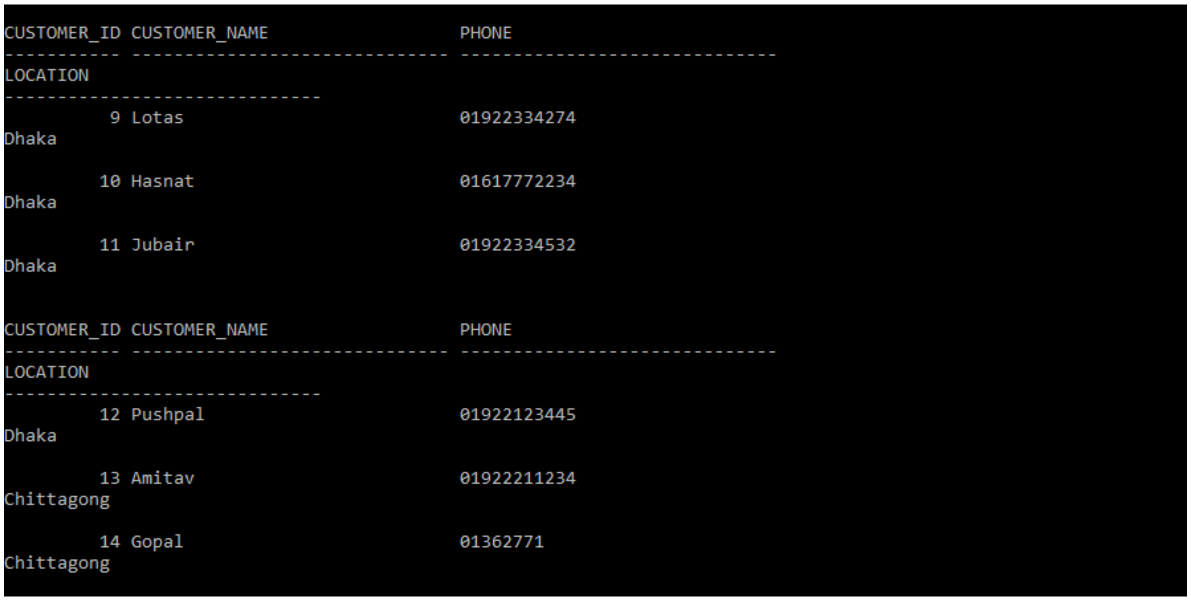


```
SQL> select * from customer1;

CUSTOMER_ID CUSTOMER_NAME        PHONE                LOCATION
----------- -------------------- -------------------- --------------------
         12 Pushpal              01922123445          Chittagong
         14 Gopal                01362771             Chittagong
         13 Amitav               01922211234          Chittagong

SQL> select * from customer2;

CUSTOMER_ID CUSTOMER_NAME        PHONE                LOCATION
----------- -------------------- -------------------- --------------------
          9 Lotas                01922334274          Dhaka
         10 Hasnat               01617772234          Dhaka
         11 Jubair               01922334532          Dhaka

SQL>
```



```
SQL> @C:\Users\Dell\Desktop\Semester4.1\LAB\DDBS\project\Triggers\ForUpdateDelete.sql;
Press 1 to UPDATE , Press 2 to DELETE: 1
SP2-0003: Ill-formed ACCEPT command starting as (38) PROMPT 'Enter customer id to UPDATE city from Chittagong to Dhaka: '
old   2:        x NUMBER := &xx;
new   2:        x NUMBER :=         1;
Enter value for idid: 12
old   3:        id NUMBER(38) := &idid;
new   3:        id NUMBER(38) := 12;

PL/SQL procedure successfully completed.


Commit complete.

SQL> select * from customer1;

CUSTOMER_ID CUSTOMER_NAME        PHONE                LOCATION
----------- -------------------- -------------------- --------------------
         14 Gopal                01362771             Chittagong
         13 Amitav               01922211234          Chittagong

SQL> select * from customer2;

CUSTOMER_ID CUSTOMER_NAME        PHONE                LOCATION
----------- -------------------- -------------------- --------------------
          9 Lotas                01922334274          Dhaka
         10 Hasnat               01617772234          Dhaka
         11 Jubair               01922334532          Dhaka
         12 Pushpal              01922123445          Dhaka

SQL>
```

And also, the Customers table updated →



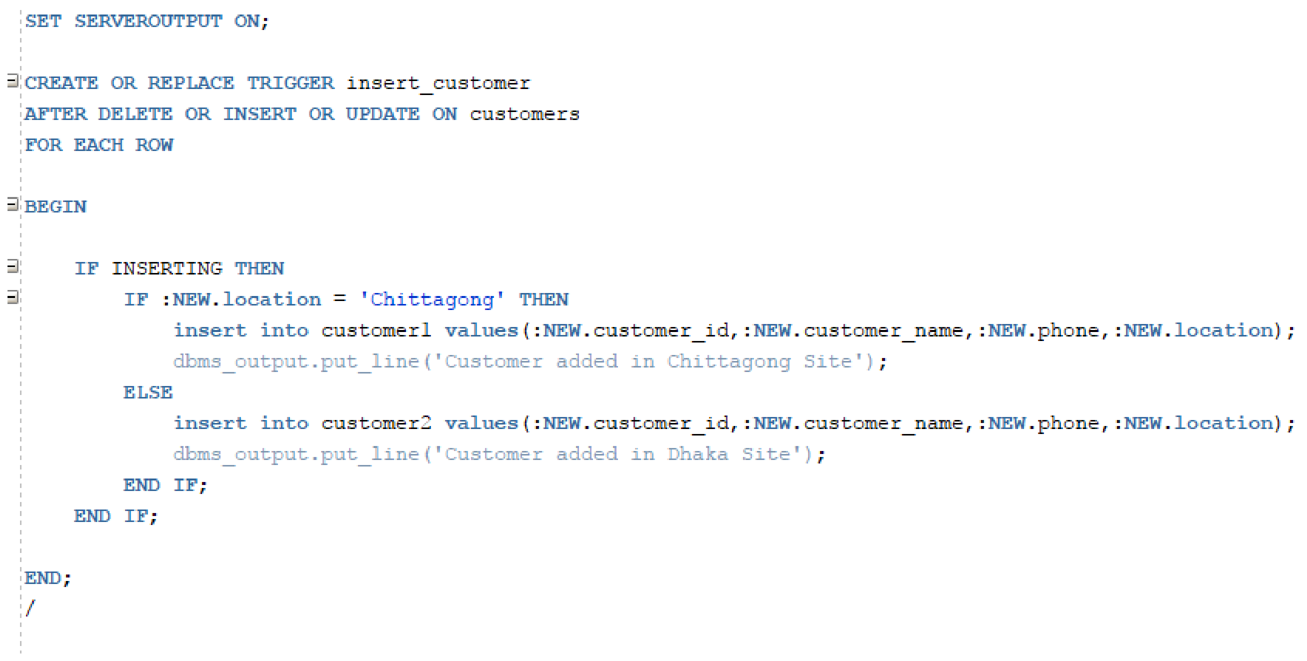
```
CUSTOMER_ID CUSTOMER_NAME                  PHONE
----------- ------------------------------ ------------------------------
LOCATION
------------------------------
          9 Lotas                          01922334274
Dhaka

         10 Hasnat                         01617772234
Dhaka

         11 Jubair                         01922334532
Dhaka


CUSTOMER_ID CUSTOMER_NAME                  PHONE
----------- ------------------------------ ------------------------------
LOCATION
------------------------------
         12 Pushpal                        01922123445
Dhaka

         13 Amitav                         01922211234
Chittagong

         14 Gopal                          01362771
Chittagong
```

5. Trigger:

The Update is done using trigger and also another activity is to insert new customer, if he's from Chittagong, his entry will be inserted into site-1 (Chittagong Branch) and after inserted into global relation Customers.

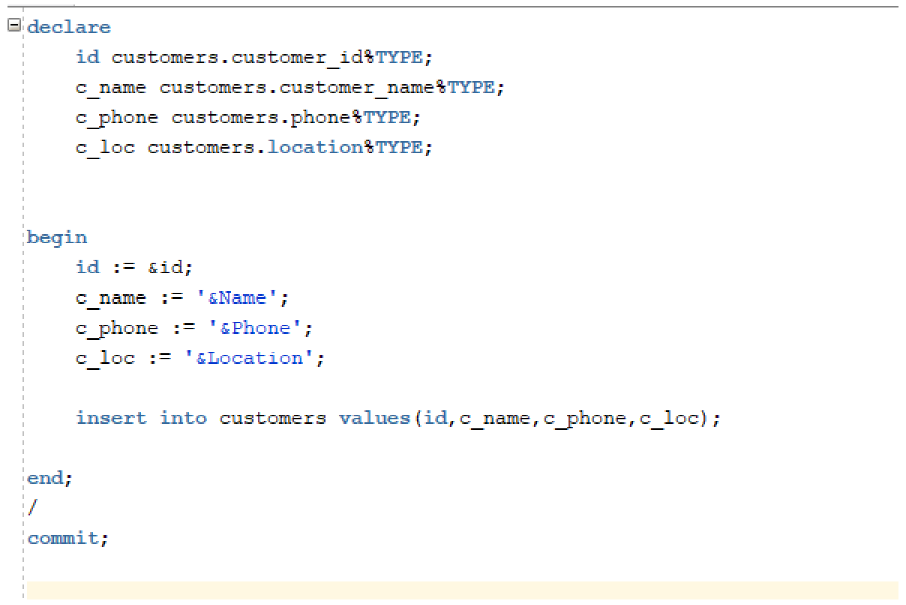
So, two actions are performed:
i) Insert into global relation (Customers)
ii) Insert Into corresponding sites according to Customer's location

➔ Here, we took user info to insert into Global Relation and then we called the trigger to insert into either 'Customer1' or 'Customer2' according to corresponding location.



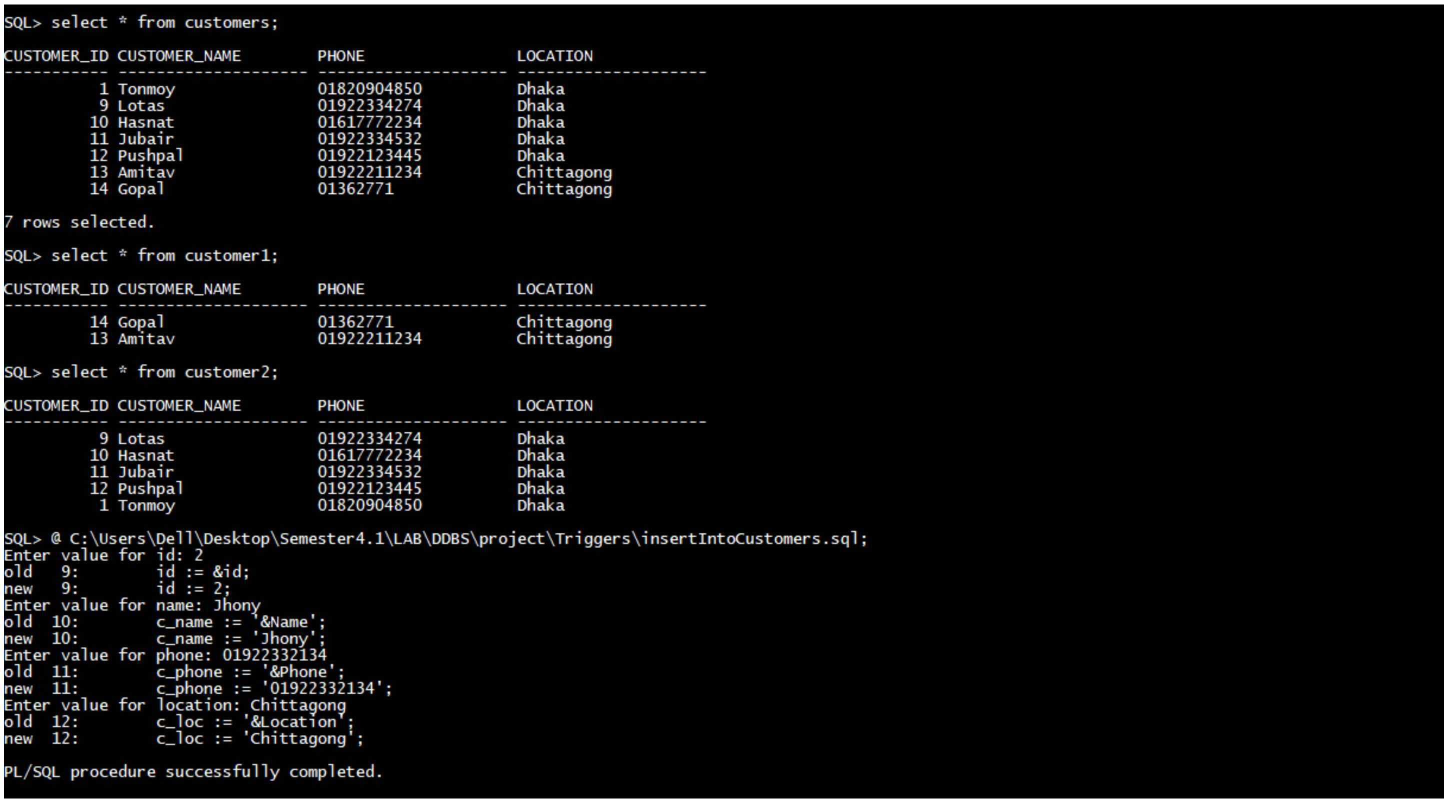
```
SET SERVEROUTPUT ON;

CREATE OR REPLACE TRIGGER insert_customer
AFTER DELETE OR INSERT OR UPDATE ON customers
FOR EACH ROW

BEGIN

    IF INSERTING THEN
        IF :NEW.location = 'Chittagong' THEN
            insert into customer1 values(:NEW.customer_id,:NEW.customer_name,:NEW.phone,:NEW.location);
            dbms_output.put_line('Customer added in Chittagong Site');
        ELSE
            insert into customer2 values(:NEW.customer_id,:NEW.customer_name,:NEW.phone,:NEW.location);
            dbms_output.put_line('Customer added in Dhaka Site');
        END IF;
    END IF;

END;
/
```

➔ Trigger is called to insert into either 'customer1 or 'customer2'

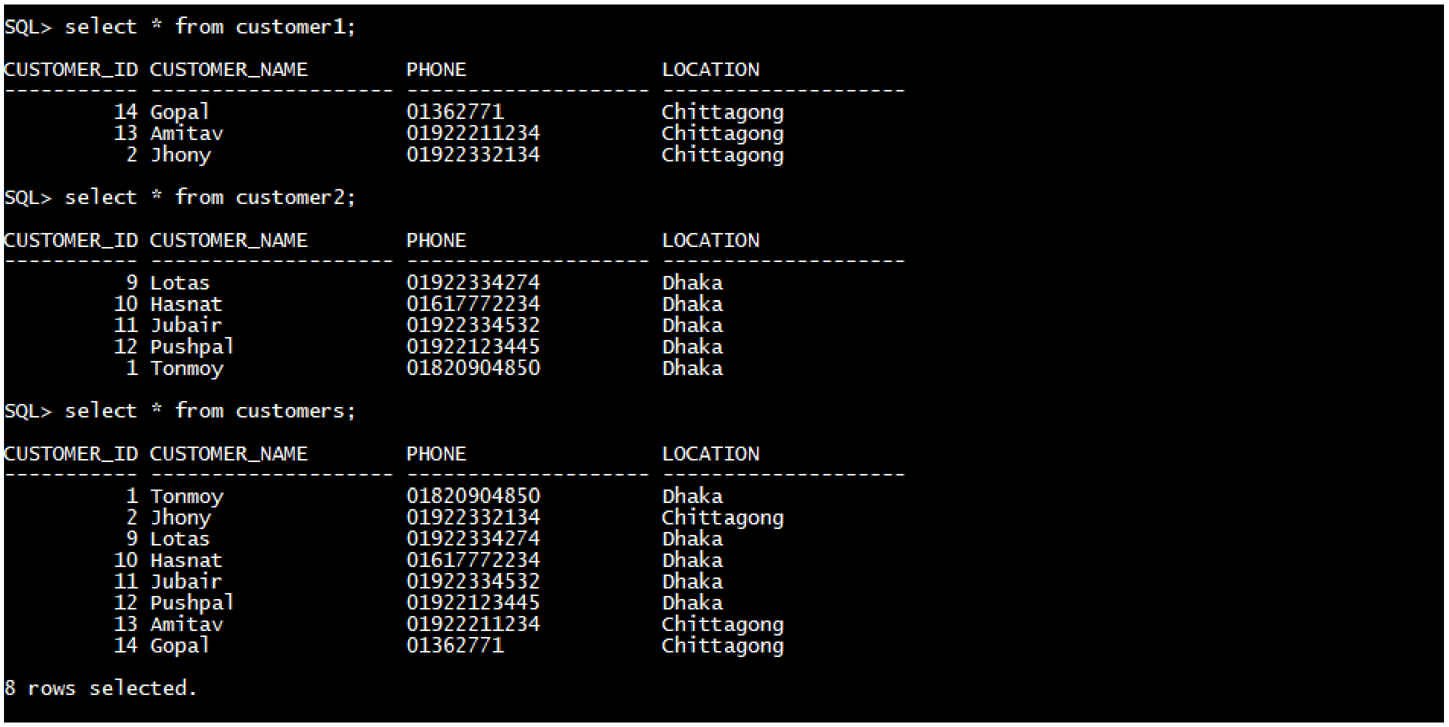


```
declare
    id customers.customer_id%TYPE;
    c_name customers.customer_name%TYPE;
    c_phone customers.phone%TYPE;
    c_loc customers.location%TYPE;


begin
    id := &id;
    c_name := '&Name';
    c_phone := '&Phone';
    c_loc := '&Location';

    insert into customers values(id,c_name,c_phone,c_loc);

end;
/
commit;
```

Result:



```
SQL> select * from customers;

CUSTOMER_ID CUSTOMER_NAME        PHONE                LOCATION
----------- -------------------- -------------------- --------------------
          1 Tonmoy               01820904850          Dhaka
          9 Lotas                01922334274          Dhaka
         10 Hasnat               01617772234          Dhaka
         11 Jubair               01922334532          Dhaka
         12 Pushpal              01922123445          Dhaka
         13 Amitav               01922211234          Chittagong
         14 Gopal                01362771             Chittagong

7 rows selected.

SQL> select * from customer1;

CUSTOMER_ID CUSTOMER_NAME        PHONE                LOCATION
----------- -------------------- -------------------- --------------------
         14 Gopal                01362771             Chittagong
         13 Amitav               01922211234          Chittagong

SQL> select * from customer2;

CUSTOMER_ID CUSTOMER_NAME        PHONE                LOCATION
----------- -------------------- -------------------- --------------------
          9 Lotas                01922334274          Dhaka
         10 Hasnat               01617772234          Dhaka
         11 Jubair               01922334532          Dhaka
         12 Pushpal              01922123445          Dhaka
          1 Tonmoy               01820904850          Dhaka

SQL> @ C:\Users\Dell\Desktop\Semester4.1\LAB\DDBS\project\Triggers\insertIntoCustomers.sql;
Enter value for id: 2
old   9:        id := &id;
new   9:        id := 2;
Enter value for name: Jhony
old  10:        c_name := '&Name';
new  10:        c_name := 'Jhony';
Enter value for phone: 01922332134
old  11:        c_phone := '&Phone';
new  11:        c_phone := '01922332134';
Enter value for location: Chittagong
old  12:        c_loc := '&Location';
new  12:        c_loc := 'Chittagong';

PL/SQL procedure successfully completed.
```



```
SQL> select * from customer1;

CUSTOMER_ID CUSTOMER_NAME        PHONE                LOCATION
----------- -------------------- -------------------- --------------------
         14 Gopal                01362771             Chittagong
         13 Amitav               01922211234          Chittagong
          2 Jhony                01922332134          Chittagong

SQL> select * from customer2;

CUSTOMER_ID CUSTOMER_NAME        PHONE                LOCATION
----------- -------------------- -------------------- --------------------
          9 Lotas                01922334274          Dhaka
         10 Hasnat               01617772234          Dhaka
         11 Jubair               01922334532          Dhaka
         12 Pushpal              01922123445          Dhaka
          1 Tonmoy               01820904850          Dhaka

SQL> select * from customers;

CUSTOMER_ID CUSTOMER_NAME        PHONE                LOCATION
----------- -------------------- -------------------- --------------------
          1 Tonmoy               01820904850          Dhaka
          2 Jhony                01922332134          Chittagong
          9 Lotas                01922334274          Dhaka
         10 Hasnat               01617772234          Dhaka
         11 Jubair               01922334532          Dhaka
         12 Pushpal              01922123445          Dhaka
         13 Amitav               01922211234          Chittagong
         14 Gopal                01362771             Chittagong

8 rows selected.
```

Conclusion: We tried our best to complete the project and make this project well-structured and well planned within our capability. We only considered the most important requirements only. The project has many opportunities to be developed into something better if enough time and dedication is provided. But as a small project this implements many of the features, we see in our real-life restaurant management systems.