CSE 4410: DATABASE MANAGEMENT SYSTEMS II LAB

Lab 1 Report

Mizbaul Haque Maruf
ID: 200042125
mizbaulhaque@iut-dhaka.edu

TASK:

■ Suppose you are given the task of automating the operations of an international food chain via a single platform. There are multiple franchises (KFC, Chester's, Pizza hut, Domino's Pizza) of the food chain spread across over 20 countries. Each of the franchises gets at least 10,0000 customers per year.Customers can register themselves under different franchises to order food from different branches of that specific franchise. Each franchise has multiple branches spread around the country. Each branch has its own team of chefs. Each of them is an expert in a particular cuisine. And any customer can see the basic information of the chefs such as special menus developed by them (up to 5 menus). Each franchise has multiple menus (some are unique and some are common) that they offer to customers. The menus are identified by their own name, cuisine, main ingredients(optional), price, calorie_count etc. Further to build a proper food recommendation system for the food chain, information about customer details, their personal preferred cuisines (a person can have multiple preferred ones) and customers' ratings of any food need to be stored.
Now, your task is to:
1. Convert the scenario into DDL using standard SQL denoting the appropriate constraints.

```
CREATE TABLE CUSTOMERS(
    CUS_ID VARCHAR2(11) NOT NULL PRIMARY KEY,
    CUS_NAME VARCHAR2(50),
    PREF VARCHAR2(50)
);

CREATE TABLE FRANCHISE(
    RES_ID VARCHAR2(11) NOT NULL PRIMARY KEY,
    NAME VARCHAR2(50),
    BRANCH VARCHAR2(50),
    COUNTRY VARCHAR2(50),
    CUS_ID VARCHAR2(11),
    FOREIGN KEY (CUS_ID) REFERENCES CUSTOMERS(CUS_ID)
);

CREATE TABLE MENU(
    MENU_ID VARCHAR2(11) NOT NULL PRIMARY KEY,
    NAME VARCHAR2(50),
    CHEF_ID VARCHAR2(11),
    RES_ID VARCHAR2(11),
    FOREIGN KEY (CHEF_ID) REFERENCES CHEF(CHEF_ID),
    FOREIGN KEY (RES_ID) REFERENCES FRANCHISE(RES_ID)
);
```

```
CREATE TABLE ORDERS(
    ORDER_ID VARCHAR2(11) NOT NULL PRIMARY KEY,
    MENU_ID VARCHAR2(11),
    CUS_ID VARCHAR2(11),
    RATING NUMBER(1),
    FOREIGN KEY (MENU_ID) REFERENCES MENU(MENU_ID),
    FOREIGN KEY (CUS_ID) REFERENCES CUSTOMERS(CUS_ID)
);

CREATE TABLE CHEF(
    CHEF_ID VARCHAR2(11) NOT NULL PRIMARY KEY,
    NAME VARCHAR2(11),
    RES_ID VARCHAR2(11),
    MENU_ID1 VARCHAR2(11),
    MENU_ID2 VARCHAR2(11),
    MENU_ID3 VARCHAR2(11),
    MENU_ID4 VARCHAR2(11),
    MENU_ID5 VARCHAR2(11),
    FOREIGN KEY (RES_ID) REFERENCES FRANCHISE(RES_ID)
);
```

2. Write SQL statements for the following queries:
(a) Find the total number of customers for each franchise.

```
--1
SELECT NAME, COUNT(CUS_ID)
FROM FRANCHISE
GROUP BY NAME;
```

(b) Find the avg rating for each menu item among all franchises.

```
--2
SELECT M.MENU_ID, AVG(O.RATING)
FROM MENU M, ORDERS O
WHERE M.MENU_ID = O.MENU_ID
GROUP BY M.MENU_ID;
```

(c) Find the 5 top most popular items. It should be based on the number of times they were ordered.

```
--3
SELECT MENU_ID
FROM ORDERS
WHERE ROWNUM <= 5
ORDER BY RATING DESC;
```

(d) Find the names of all customers who have preferred food that is offered from at least 2 different franchises.

```
--4
SELECT C.CUS_ID, COUNT(F.NAME)
FROM CUSTOMERS C, FRANCHISE F
WHERE C.CUS_ID = F.CUS_ID
GROUP BY C.CUS_ID
HAVING COUNT(DISTINCT F.NAME) >= 2;
```

(e) Find the names of all customers who have not placed any orders.

```
--5
SELECT CUSTOMERS.CUS_NAME
FROM CUSTOMERS
WHERE CUSTOMERS.CUS_ID NOT IN (SELECT CUS_ID
                               FROM ORDERS
                               );
```

EXPLANATION:

(a) This query returns the total number of customers for each franchise. It does this by grouping the rows in the customers table by the franchise column and using the COUNT function to count the number of customer IDs in each group.

(b) This query returns the average rating for each menu item among all franchises. It does this by joining the menu and orders tables on the menu_id column and grouping the resulting rows by the name column of the menu table. It then uses the AVG function to compute the average rating for each group.

(c) This query returns the top 5 most popular items based on the number of times they were ordered. It does this by selecting the menu_id from orders and then take the top five by ROWNUM and then sorting using DESC.

(d) This query returns the customer IDs and the count of unique franchises visited by each customer. It does this by joining the customers and franchise tables on the cus_id column and grouping the resulting rows by the cus_id column of the customers table. It then uses the COUNT function to count the number of unique name values in the franchise table for each group. The HAVING clause is used to filter the results to include only the groups where the count of unique name values is greater than or equal to 2.

(e) This query returns the names of all customers who have not placed any orders. It does this by using the NOT IN operator to select only the customers who do not have a customer ID that appears in the customer_id column of the orders table. This effectively selects only the customers who have not placed any orders, as they have no corresponding rows in the orders table.