Islamic University of Technology (IUT)

Report on Lab 1

Submitted By

Shanta Maria (ID:200042172)

CSE 4410 Database Management Systems II Lab

Submitted To

Dr. Abu Raihan Mostofa Kamal

Professor, Department of CSE

Zannatun Naim Srsity

Lecturer, Department of CSE

Md. Rafid Haque

Lecturer, Department of CSE

January 9, 2023

Introduction

This lab class was a recap of the previous semester's tasks.

1 Question and DDL

- 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.
2. Write SQL statements for the following queries:
   (a) Find the total number of customers for each franchise.
   (b) Find the avg rating for each menu item among all franchises.
   (c) Find the 5 top most popular items. It should be based on the number of times they were ordered.
   (d) Find the names of all customers who have preferred food that is offered from at least 2 different franchises.
   (e) Find the names of all customers who have not placed any orders.

1.1 DDL

The implemented DDLs are as below:

```sql
drop table franchise cascade constraint;
drop table customer cascade constraint;
drop table customer_membership constraint;
drop table menu cascade constraint;
drop table cuisine cascade constraint;
drop table preferred_cuisine cascade constraint;
drop table rating cascade constraint;
drop table branch cascade constraint;
drop table chef cascade constraint;
drop table cuisine_items cascade constraint;
drop table special_menu cascade constraint;
drop table order cascade constraint;
drop table menu_offered cascade constraint;


create table franchise
(
    franchise_id int not null,
    franchise_name varchar2(25),
    constraint pk_franchise_id primary key(franchise_id)
);

create table customer
(
    customer_id int not null,
    customer_name varchar2(25),
    constraint pk_customer_id primary key(customer_id),
);

create table customer_membership
(
    customer_id int not null,
    franchise_id int not null,
    constraint pk_id primary key(franchise_id, customer_id),
    constraint fk_franchise_id foreign key(franchise_id) references franchise(franchise_id)
    constraint fk_customer_id foreign key(franchise_id) references franchise(franchise_id)
);
```

```
create table menu
(
    menu_id int not null,
    menu_name varchar2(25),
    main_ingrediant varchar2(25),
    price number,
    calorie_count int,
    constraint pk_menu_id primary key(menu_id)
);

create table cuisine
(
    cuisine_id int not null,
    cuisine_name varchar2(25),
    constraint pk_cuisine_id primary key(cuisine_id),
);

create table preferred_cuisine
(
    customer_id int not null,
    cuisine_id int not null,
    constraint pk_preferred_cuisine_id primary key(cuisine_id, customer_id),
    constraint fk_customer_id foreign key(customer_id) references customer(customer_id),
    constraint fk_cuisine_id foreign key(cuisine_id) references cuisine(cuisine_id)
);

create table rating
(
    customer_id int not null,
    menu_id int not null,
    rating number,
    constraint pk_rating_id primary key(customer_id, menu_id),
    constraint fk_customer_id foreign key(customer_id) references customer(customer_id),
    constraint fk_menu_id foreign key(menu_id) references menu(menu_id)
);
```

```
create table branch
(
    branch_id int not null,
    branch_name varchar2(25),
    location varchar2(25),
    franchise_id int not null,
    constraint pk_branch_id primary key(branch_id),
    constraint fk_franchise_id foreign key(franchise_id) references franchise(franchise_id)
);

create table chef
(
    chef_id int not null,
    cuisine_expertise_id int,
    branch_id int not null,
    constraint pk_chef_id primary key(chef_id),
    constraint fk_cuisine_expertise_id foreign key(cuisine_expertise_id) references cuisine(cuisine_id),
    constraint fk_branch_id foreign key(branch_id) references branch(branch_id)
);

create table cuisine_items
(
    cuisine_id int,
    menu_id int,
    constraint pk_c_item primary key(cuisine_id, menu_id),
    constraint fk_cuisine_id_items foreign key (cuisine_id) references cuisine(cuisine_id),
    constraint fk_menu_id_items foreign key (menu_id) references menu(menu_id),
)

create table special_menu
(
    menu_id int not null,
    chef_id int not null,
    menu_number int,
    constraint pk_special_menu_id primary key(menu_id, chef_id, menu_number),
    constraint fk_s_chef_id foreign key(chef_id) references chef(chef_id),
    constraint fk_s_menu_id foreign key(menu_id) references menu(menu_id),
    constraint check_five check (menu_number > 0 and menu_number <= 5)
);
```

```
create table order
(
    order_id int not null,
    customer_id int,
    menu_id int,
    constraint pk_order_id primary key(order_id),
    constraint fk_o_customer_id foreign key(customer_id) references customer(customer_id),
    constraint fk_o_menu_id foreign key(menu_id) references menu(menu_id)
);

create table menu_offered
(
    menu_id int,
    franchise_id int,
    constraint pk_menu_offered primary key(menu_id, franchise_id),
    constraint fk_om_menu_id foreign key(menu_id) references menu(menu_id),
    constraint fk_om_franchise_id foreign key(franchise_id) references franchise(franchise_id)
);
```

1.2 Analysis and Explanation

There are a total of 13 tables:

1. `franchise`
    (a) Primary Key: `franchise_id`
    (b) Foreign Keys: `none`
    (c) Cardinality :
        i. One to One Relationships: `none`
        ii. One to Many Relationships: `customer_membership, menu_offered`
        iii. Many to Many Relationships: `customer` through junction table `customer_membership`, `menu` through the junction table `menu_offered`
2. `customer`
    (a) Primary Key: `customer_id`
    (b) Foreign Keys: `none`
    (c) Cardinality :
        i. One to One Relationships: `none`
        ii. One to Many Relationships: `customer_membership`, `preferred_cuisine`, `rating, order`
        iii. Many to Many Relationships: `franchise` through junction table `customer_membership`, `cuisine` through junction table `preferred_cuisine`, `menu` through junction table `rating`, `menu` through the junction table `order`
3. `customer_membership`
    (a) Primary Key: `franchise_id, customer_id`
    (b) Foreign Keys: `franchise_id` from `franchise`, `customer_id` from `customer`
    (c) Cardinality :

i. One to One Relationships: none

ii. One to Many Relationships: franchise, customer

iii. Many to Many Relationships: none

4. cuisine

(a) Primary Key: cuisine_id

(b) Foreign Keys: none

(c) Cardinality :

i. One to One Relationships: none

ii. One to Many Relationships: preferred_cuisine, cuisine_items

iii. Many to Many Relationships: customer through junction table preferred_cuisine, menu through junction table cuisine_items

5. preferred_cuisine

(a) Primary Key: cuisine_id, customer_id

(b) Foreign Keys: customer_id from customer, cuisine_id from cuisine

(c) Cardinality :

i. One to One Relationships: none

ii. One to Many Relationships: cuisine, customer

iii. Many to Many Relationships: none

6. menu

(a) Primary Key: menu_id

(b) Foreign Keys: none

(c) Cardinality :

i. One to One Relationships: none

ii. One to Many Relationships: rating, cuisine_items, special_menu, order, menu_offered

iii. Many to Many Relationships: `customer` through the junction table `rating`, `cuisine` through the junction table `cuisine_items`, `chef` through the junction table `special_menu`, `customer` through the junction table `order`, `franchise` through the junction table `menu_offered`

7. `rating`

   (a) Primary Key: `customer_id, menu_id`

   (b) Foreign Keys: `customer_id` from `customer`, `menu_id` from `menu`

   (c) Cardinality :

      i. One to One Relationships: `none`

      ii. One to Many Relationships: `customer, menu`

      iii. Many to Many Relationships: `none`

8. `branch`

   (a) Primary Key: `branch_id`

   (b) Foreign Keys: `franchise_id` from `franchise`

   (c) Cardinality :

      i. One to One Relationships: `none`

      ii. One to Many Relationships: `franchise`

      iii. Many to Many Relationships: `none`

9. `chef`

   (a) Primary Key: `chef_id`

   (b) Foreign Keys: `cuisine_expertise_id` from `cuisine`, `branch_id` from `branch`

   (c) Cardinality :

      i. One to One Relationships: `branch, cuisine`

      ii. One to Many Relationships: `special_menu`

      iii. Many to Many Relationships: `menu` through the junction table `special_menu`

10. cuisine_items

    (a) Primary Key: cuisine_id, menu_id

    (b) Foreign Keys: cuisine_id from cuisine, menu_id from menu

    (c) Cardinality :

        i. One to One Relationships: none

        ii. One to Many Relationships: menu, cuisine

        iii. Many to Many Relationships: none

11. special_menu

    (a) Primary Key: menu_id, chef_id

    (b) Foreign Keys: chef_id from chef, menu_id from menu

    (c) Cardinality :

        i. One to One Relationships: none

        ii. One to Many Relationships: chef, menu

        iii. Many to Many Relationships: none

12. order

    (a) Primary Key: order_id

    (b) Foreign Keys: menu_id from menu, customer_id from customer

    (c) Cardinality :

        i. One to One Relationships: none

        ii. One to Many Relationships: menu, customer

        iii. Many to Many Relationships: none

13. menu_offered

    (a) Primary Key: menu_id, franchise_id

    (b) Foreign Keys: franchise_id from franchise, menu_id from menu

(c) Cardinality :

i. One to One Relationships: `none`

ii. One to Many Relationships: `menu, franchise`

iii. Many to Many Relationships: `none`

1.3 Difficulties

I faced difficulties in figuring out the junction tables and how to connect special menu with chef and menu tables. It was also difficult to find the connection between 2 tables using smaller interconnected tables.

2 SQL Queries

```
-----task(a)-----
select franchise_id, count(customer_id) as customer
from customer_membership
group by franchise_id;

-----task(b)-----
select menu_id, avg(rating) as rating
from rating
group by menu_id;

-----task(c)-----
select menu_id
from (select menu_id, count(order_id)
        from order
        group by menu_id
        order by count(order_id))
where rownum <= 5;

-----task(d)-----
select cid, customer_name
from(select cid, count(fid)
    from (select preferred_cuisine.customer_id as cid, menu_offered.franchise_id as fid
        from preferred_cuisine, menu_offered, cuisine_items
        where preferred_cuisine.cuisine_id = cuisine_items.cuisine_id and menu_offered.menu_id = cuisine_items.menu_id)
    group by cid
    where count(fid) >= 2), customer
where customer_id = cid;

-----task(e)-----
select customer_name
from (select customer_id
        from customer
        minus
        select distinct customer_id
        from order) natural join customer;
```

2.1 Analysis and Explanation

For task(a), I used the `count()` function to count the total number of customers under each franchise by grouping the result according to `franchise_id`.

For task(b), I used the `avg()` function to calculate the average rating of each menu item by grouping the result according to `menu_id`.

For task(c), I used a nested query to first calculate all the menu items with the number of times they were ordered in descending order according to the number of orders and then showed the top 5 results.

For task(d), I used 2 nested queries. One nested query is used to find all the customers with their associated franchises according to their preferred cuisines. Then the next nested query is used to count the number of franchises each customer is associated with and showed the results for franchises numbering equal to or more than 2. Lastly, I queried the customer names according to the customer IDs from the nested queries.

For task(e), I used a nested query to find all the customer IDs who never placed an order using the `minus` keyword and then used natural join with `customer` table to find the respective customer names.

2.2 Difficulties

I did not face any mentionable difficulties when writing the queries.

Conclusion

As shown in the report, I have drawn a possible database system for the given scenario and solved the SQL queries. All the commands used were written in VS Code which was then saved with .sql extension. The .sql file was then run through the SQL command line to execute all the commands.