Lab Report

Database Management Systems-II Lab

Name: Zaara Zabeen Arpa

ID: 200042101

Date of Submission: 05.01.2023

Lab-1 Overview

In this lab, we were given a scenario based on which we were asked to create DDL using standard SQL. Then we were given some queries which we had to solve using SQL statements.

Task:1

Analysis of the problem:

In this task we converted the scenario into DDL.

Working Code:

```
1.
create table franchise(
   franchise_id number,
   franchise_name varchar(20),
   number_of_countries number,
   number_of_customers number,
   check (number_of_countries>=20),
   check (number_of_customers>=100000),
   constraint pk_franchise primary key (franchise_id));
```

```
create table customer(
   customer_id number,
   customer_name varchar(20),
   country varchar(20),
   constraint pk_customer primary key (customer_id)
);
create table registered_customer(
   customer_id number,
   franchise_id number,
   constraint pk_registered_customer primary key (franchise_id,customer_id),
            constraint fk_registered_customer1 foreign key (franchise_id) references
franchise(franchise_id),
            constraint fk_registered_customer2 foreign key (customer_id) references
customer(customer_id)
);
create table branch(
   branch_id number,
   branch_name varchar(20),
   franchise_id number,
   country varchar(20),
   constraint pk_branch primary key (branch_id),
   constraint fk_branch foreign key (franchise_id) references franchise(franchise_id)
);
Create  table chef(
   chef_id number,
   chef_name varchar(20),
   branch_id number,
   specialty varchar(20),
   constraint pk_chef primary key (chef_id),
   constraint fk_chef1 foreign key (branch_id) references branch(branch_id),
   constraint fk_chef2 foreign key (developed_menu) references menu(menu_name)
);
```

```
create table menu(
   menu_name varchar(20),
   cuisine varchar(20),
   main_ingredients varchar(20),
   price number,
   calorie_count number,
   constraint pk_menu primary key (menu_name)
);
create table chef_menu(
   menu_name varchar(20),
   chef_id number,
   constraint pk_chef_menu primary key (menu_name, chef_id),
   constraint fk_chef_menu1 foreign key (menu_name) references menu,
   constraint fk_chef_menu2 foreign key (chef_id) references chef
);
create table franchise_menu(
   franchise_id number,
   menu_name varchar(20),
   location varchar(20),
   constraint pk_franchise_menu primary key (franchise_id, menu_name),
   constraint fk_franchise_menu1 foreign key (franchise_id) references franchise,
   constraint fk_franchise_menu2 foreign key (menu_name) references menu
);
create table customer_review(
   review_id number,
   customer_id number,
   branch_id number,
   preferred_cuisine varchar(20),
   ordered_food varchar(20),
   rating number,
   constraint pk_customer_review primary key (review_id),
      constraint fk_customer_review1 foreign key (customer_id) references
customer(customer_id),
```

```
constraint fk_customer_review2 foreign key (ordered_food) references menu(menu_name)
);
create table orders(
   order_id number,
   customer_id number,
   menu_name varchar2(20),
   constraint pk_order primary key (order_id),
   constraint fk_order1 foreign key (customer_id) references customer(customer_id),
   constraint fk_order2 foreign key (menu_name) references menu(menu_name)
);
```

Explanation:

As the scenario includes multiple franchises with multiple customers, I have created two tables named Franchise and Customer. Franchise and customer has many to many cardinalities between them. Franchises of the food chain spread across over 20 countries and each of the franchises gets at least 100000 customers per year, I used “check” to check this constraint. Franchise and Customer has a junction table between them which is registered_customer. This table contains the customers registered under multiple franchises. Then I have a table named Branch which will represent branches under franchises. Franchise and Branch have one to many cardinality between them. Each branch has its own team of chefs so I have a Chef table which has a one to many cardinality with the Branch table. I have created a Menu table which has a many to many cardinality with Chef table and Franchise table. The junction table between Chef and Menu is Chef_Menu, Franchise and Menu is Franchise_Menu. As ratings are to be stored I have created a Customer_Review table which will store the review and ratings of a customer.

This table has a one to many cardinality with Customer and Menu table. Also I have created a table which is Orders representing which customer ordered which food. This table has a one to many cardinality with Customer and Menu.
Explanation of attributes-

- Franchise table has franchise_id (primary key), franchise_name, number_of_countries the franchise is available, number_of_customers registered under a franchise.
- Customer table has customer_id(primary key), customer_name, country the customer belongs to.
- Registered_Customer consists of the primary keys of Franchise and Customer table. These two foreign keys are the primary keys of this table.
- Branch table has branch_id(primary key), branch_name, franchise_id(foreign key), country the branch is located in.
- Chef table has chef_id(primary key), chef_name, branch_id(foreign key), specialty, developed_menu(foreign key)
- Menu has menu_name(primary key), cuisine, main_ingredients, price, calorie_count.
- Chef_menu consists of the primary keys of chef and menu table. These two foreign keys are also the primary keys of this table.
- Franchise_Menu consists of the primary keys of franchise and menu table. These two foreign keys are also the primary keys of this table.
- Customer_Review has review_id (primary key), customer_id(foreign key), ordered_food(foreign key), branch_id(foreign key), preferred_cuisine and rating.

- Orders table has order_id(primary key), customer_id(foreign key), menu_name(primary key).

Task:2

Analysis of the problem:

In this problem, we had to solve some queries using SQL statements.

Working Code:

```
(a)
select franchise_id, count(customer_id)
from registered_customer
group by franchise_id;

(b)
select ordered_food, avg(rating)
from customer_review
group by ordered_food;

(c)select *
from (select *
from (select menu_name, count(order_id) as number_of_orders
from orders
group by menu_name)
order by number_of_orders desc)
where rownum <=5;
```

```
(d) select customer.customer_name
from customer , menu , franchise_menu , franchise , customer_review
where customer.customer_id = franchise.customer_id and franchise.franchise_id =
franchise_menu.franchise_id and menu.menu_name = franchise.menu_name and
menu.cuisine = customer_review.preferred_cuisine and customer_review.customer_id =
customer.customer_id
group by customer.customer_name
having count(distinct franchise.franchise_id) >= 2;


(e)select customer_id
from customer
where customer_id NOT IN
 (select customer_id
from customer_review);
```

Explanation:

a. To find the total number of customers for each franchise I have first counted all the customer_id from registered_customer table grouping them by franchise_id using ‘group by’. Thus I can get the total number of customers under every franchise.

b. To find the average rating of each menu item I have selected the name of ordered food and using aggregate function ‘avg’ found the average rating and grouped them by menu_name which is basically the ordered_food attribute.

c. I have to find out the top 5 most popular items based on the number of times they were ordered. So first I find out which menu was ordered how many times using menu_name and order_id attributes and aggregate function

‘count’ in a subquery. Then I sorted them in a descending order then I found out the top 5 items using ‘rownum’.

d. Here I have to find out the names of the customers who have preferred food offered from at least 2 different franchises. At first I joined the tables customer, menu, franchise_menu, franchise, customer_review and showed the customer name. In the same query, using the ‘having’ clause I found out which customers ordered food that has the number of franchises >=2.
e. To find out the customers who have not placed any order I used the ‘NOT IN’ clause. First in a subquery I found out all the customers from customer table then I separated those from the customers of customer_review table using the ‘not in’ clause.

Problems faced while solving the tasks:

- While writing the DDL I was a bit confused about the scenario. Also I faced some problems in chef and menu relations. First I thought of using Varray. Later I used a junction table between them.
- In task 2-(d) it was a bit complicated for me at first. I couldn't understand the statement at first. It took a lot of time for me to solve the problem.