Islamic University of Technology
Lab Task 01
CSE 4410 : DBMS-II Lab

Submitted To :

Dr. Abu Raihan Mostofa Kamal
Professor, CSE Department
Islamic University of Technology

Zannatun Naim Sristy
Lecturer, CSE Department
Islamic University of Technology

Submitted By :

Sian Ashsad
ID : 200042151
Prog. : BSc in SWE
Dept. : CSE

Report :

Analysis : In this task, we need to convert the given scenario into DDL with appropriate constraints. On the coded DDL, we have to perform required SQL queries.

DDL code :

```
create table customer(
    c_id VARCHAR2(20),
    c_name VARCHAR2(100),

    CONSTRAINT PK_c_id PRIMARY KEY (c_id)

);

create table customer_franchise(
    c_id VARCHAR2(20),
    f_name VARCHAR2(50),

    CONSTRAINT fk_c_id_cf FOREIGN KEY f_name REFERENCES customer(c_id),
    CONSTRAINT fk_f_name_cf FOREIGN KEY f_name REFERENCES
franchise(f_name)

)

create table franchise(
    f_name VARCHAR2(50),
    CONSTRAINT pk_f_name PRIMARY KEY (f_name)
);

create table branch(
    branch_id VARCHAR2(20),
    f_name VARCHAR2(50),

    CONSTRAINT pk_branch_id PRIMARY KEY (branch_id),
    CONSTRAINT fk_f_name FOREIGN KEY f_name REFERENCES franchise(f_name)
);
```

```
create table chef(
    chef_id VARCHAR2(20),
    branch_id VARCHAR2(20),
    cuisine_id varchar2(20),

    CONSTRAINT pk_chef_id PRIMARY KEY (chef_id),
    CONSTRAINT fk_branch_id FOREIGN KEY (branch_id) REFERENCES
branch(branch_id),
    CONSTRAINT fk_cuisine_id FOREIGN KEY (cuisine_id) REFERENCES
cuisine(cuisine_id)
);

CREATE OR REPLACE TYPE ingredients AS VARRAY (20) OF VARCHAR2 (20) ;

create table cuisine(
    cuisine_id VARCHAR2(20),
    menu_id VARCHAR2(20),
    recipe ingredients,
    price int,
    calorie_count varchar2(10),
    CONSTRAINT pk_cuisine_id PRIMARY KEY(cuisine_id),
    CONSTRAINT fk_menu_id_cuisine FOREIGN KEY(menu_id) REFERENCES
menu(menu_id)
);


create table menu(
    menu_id VARCHAR2(20),
    menu_name varchar2(100),

    CONSTRAINT pk_menu_id PRIMARY KEY(menu_id)
);

create table franchise_menu(
    f_name VARCHAR2(50),
    menu_id VARCHAR2(20),

    CONSTRAINT fk_f_name_fm FOREIGN KEY f_name REFERENCES
franchise(f_name),
    CONSTRAINT fk_menu_id_fm FOREIGN KEY f_name REFERENCES menu(menu_id)
```

```
);

CREATE SEQUENCE menu_number_seq
MINVALUE 1
MAXVALUE 5
START WITH 1
INCREMENT BY 1
CACHE 5;

CREATE OR REPLACE
TRIGGER MENU_NUMBER_INCREMENT
BEFORE INSERT ON special_menu
FOR EACH ROW
BEGIN
    :NEW.menu_number := menu_number_seq . NEXTVAL ;
END ;
/

create table special_menu(
    chef_id VARCHAR2(20),
    menu_id VARCHAR2(20),
    menu_number int not null,

    CONSTRAINT fk_chef_id_owner FOREIGN KEY(chef_id) REFERENCES
chef(chef_id),
    CONSTRAINT fk_menu_id_owner FOREIGN KEY menu_id REFERENCES
menu(menu_id),
    check (menu_number <= 5)

)

create table prefered_cuisine(
    c_id VARCHAR2(20),
    cuisine_id VARCHAR2(20),

    CONSTRAINT fk_c_id_prefer FOREIGN KEY(c_id) REFERENCES customer(c_id),
    CONSTRAINT fk_cuisine_id_prefer FOREIGN KEY(cuisine_id) REFERENCES
cuisine(cuisine_id)
);
```

```
create table rating(
    c_id VARCHAR2(20),
    f_name VARCHAR2(50),
    menu_id VARCHAR2(20),
    cuisine_id VARCHAR2(20),
    rating int,

    CONSTRAINT fk_c_id_rating FOREIGN KEY(c_id) REFERENCES customer(c_id),
    CONSTRAINT fk_cuisine_id_rating FOREIGN KEY(cuisine_id) REFERENCES
cuisine(cuisine_id),
    CONSTRAINT fk_f_name_rating FOREIGN KEY f_name REFERENCES
franchise(f_name),
    CONSTRAINT fk_menu_id_rating FOREIGN KEY f_name REFERENCES
menu(menu_id)
);

create table order(
    order_id VARCHAR2(20),
    c_id VARCHAR2(20),
    cuisine_id VARCHAR2(20),
    f_name VARCHAR2(50),


    CONSTRAINT pk_order_id PRIMARY KEY (order_id),
    CONSTRAINT fk_c_id_order FOREIGN KEY(c_id) REFERENCES customer(c_id),
    CONSTRAINT fk_cuisine_id_order FOREIGN KEY(cuisine_id) REFERENCES
cuisine(cuisine_id),
    CONSTRAINT fk_f_name_order FOREIGN KEY f_name REFERENCES
franchise(f_name)
)
```

Explanation of solution (DDL) : The ER diagram has 12 tables.

1. Customer table :
    a. Primary key : c_id
    b. Foreign key : None
    c. Relationships :
        i. One to one : None
        ii. One to many : rating and order table
        iii. Many to many : Franchise table, preferred_cuisine table
2. Franchise table :
    a. Primary key : f_name
    b. Foreign key : None
    c. Relationships :
        i. One to one : None table
        ii. One to many : branch, rating and order table
        iii. Many to many : menu table
3. Branch table :
    a. Primary key : branch_id
    b. Foreign key : f_name from the franchise table
    c. Relationships :
        i. One to one : None
        ii. One to many : chef table
        iii. Many to many : None
4. Chef table :
    a. Primary key : chef_id
    b. Foreign key : branch_id from branch table and cuisine_id from cuisine table
    c. Relationships :
        i. One to one : None
        ii. One to many : None
        iii. Many to many : special_menu table
5. Cuisine table :
    a. Primary key : cuisine_id
    b. Foreign key : menu_id from menu table
    c. Relationships :
        i. One to one : chef table
        ii. One to many : rating and order table
        iii. Many to many : preferred_cuisine table

6. Menu table :
    a. Primary key : menu_id
    b. Foreign key : None
    c. Relationships :
        i. One to one : None
        ii. One to many : cuisine and rating table
        iii. Many to many : franchise and special menu table
7. Rating table :
    a. Primary key : None
    b. Foreign key : c_id from customer, f_name from franchise, menu_id from menu and cuisine from cuisine table.
    c. Relationships :
        i. One to one : None
        ii. One to many : None
        iii. Many to many : None
8. Order table :
    a. Primary key : order_id
    b. Foreign key : c_id from customer, cuisine_id from cuisine and f_name from franchise table.
    c. Relationships :
        i. One to one : None
        ii. One to many : customer, cuisine and franchise table
        iii. Many to many : None
9. Customer_franchise table: Junction table between customer and franchise table
10. Franchise_menu table : Junction table between franchise and menu table
11. Special_menu table : Junction table between menu and chef table
12. Prefered_cuisine table : Junction table between customer and cuisine table

Here in the DDL, a trigger menu_number_increment has been used to automatically increment the number of special menus a chef has. A sequence menu_number_seq has been used to aid this logic.
Again a type ingredients has been created as a varray for the convenience of having multiple elements in an ingredient list.

Queries :

```
-->a<--

select f_name, total(c_id) as TotalCustomers
from customer_franchise
group by f_name;

-->b<--

select cuisine_id ,avg(rating) as AverageRating
from rating
group by cuisine_id;

-->c<--

select cuisine_id as Item_ID
from(select cuisine_id, total(order_id)
    from order
    group by cuisine_id
    order by DESC)
where ROWNUM <=5;

-->d<--

select c.c_name, a.count
from(select c.c_name, COUNT(m.f_name) as count
    from prefered_cuisine p, menu m, customer c
    where p.c_id = c.c_id AND p.cuisine_id = m.cuisine_id
    group by c.c_name
    ) a
where a.count>=2;

-->e<--

select c_id, c_name
from customer
MINUS
select C.c_id, C.c_name
from customer C, order O
where C.c_id = O.c_id;
```

Explanation of the queries :

a) Here the total number of customers per franchise is calculated from the customer_franchise junction table. The total aggregate function is used here for which group by clause must be called at the end for f_name.
b) Same as (a). Avg aggregate function has been used on the rating table and so group by clause has been used on cuisine_id.
c) Here the nested query gives cuisine_id along with the number of times it has been ordered in descending order. The total aggregate function has been used on order_id from the order table. Using the ROWNUM clause, we only take the first five entries.
d) Here in the nested query, 3 tables have been joined to figure out the names of customers and the number of franchises who offer their preferred cuisine. The outer query simply shows the entries who have the count more than 1.
e) To find out the customers who have no orders, we first find out the total list of customers and MINUS that with the list of customers who have orders.

Problems : Visualizing the entity relationship diagram proved to be quite a challenge. Furthermore some relationships were difficult to think up due to the complicacy of the given stem. While coding the DDL, some syntaxes had to be looked up from last year's lab course CSE - 4308 : DBMS.