DATA MANAGEMENT AND DATABASE DESIGN
FINAL PROJECT

Topic Name: Restaurant Recommendation and Reservation System

GitHub Repository:
https://github.com/snehalpadekar/Restaurant-Recommendation-and-Reservation-System

Group Name: R3

Group Members:
Anjali Kshirsagar (002743547)
Gayatri Kenkare (002743776)
Snehal Padekar (002737903)
Mahek Gangadia (002797094)

CONTENTS:

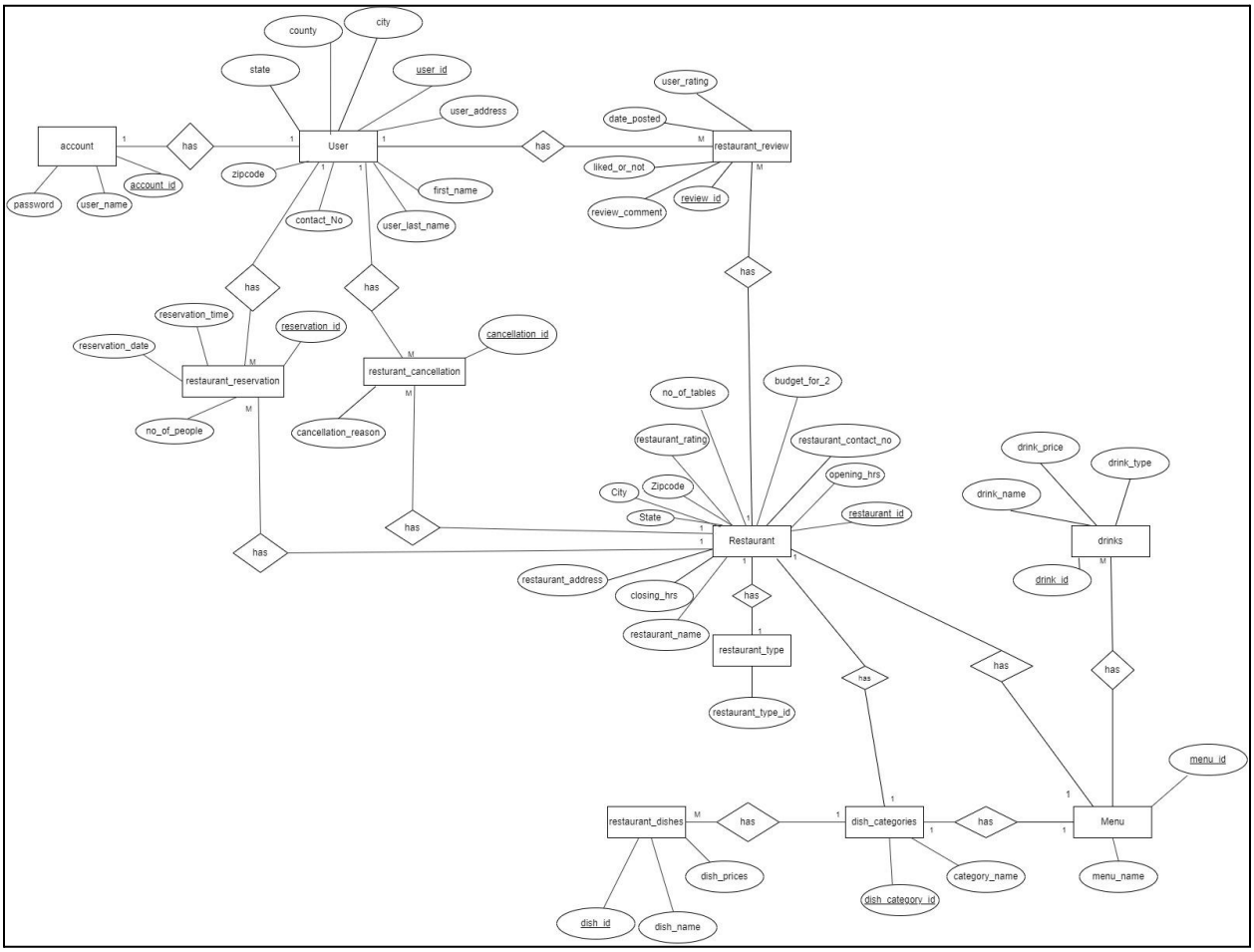
A Model on Restaurant Recommendation and Reservation System

The project idea is to offer a Restaurant Recommendation and Reservation System. It is a platform for people who love to eat. The system will take the user's food preferences into account, which will then suggest good restaurants in the nearby area to the user. This recommendation will solely be based on the cost, the quality of the meal, customer reviews or ratings, accessibility, etc. The user can reserve a table after choosing the restaurant. The Reservation System offers features like booking a table at a specific restaurant, canceling the reservation, etc. The purpose of this system is to let people get ideas about which restaurant will be great for them. This system can give people some suggestions; also you can get others' opinions from this site. Besides viewing others' opinions, you can give suggestions to other people by rating restaurants. In this system, there are many ways to search restaurants, including by zip code, dish type, budget, dish name, Restaurant type, price, and by recommendation search.

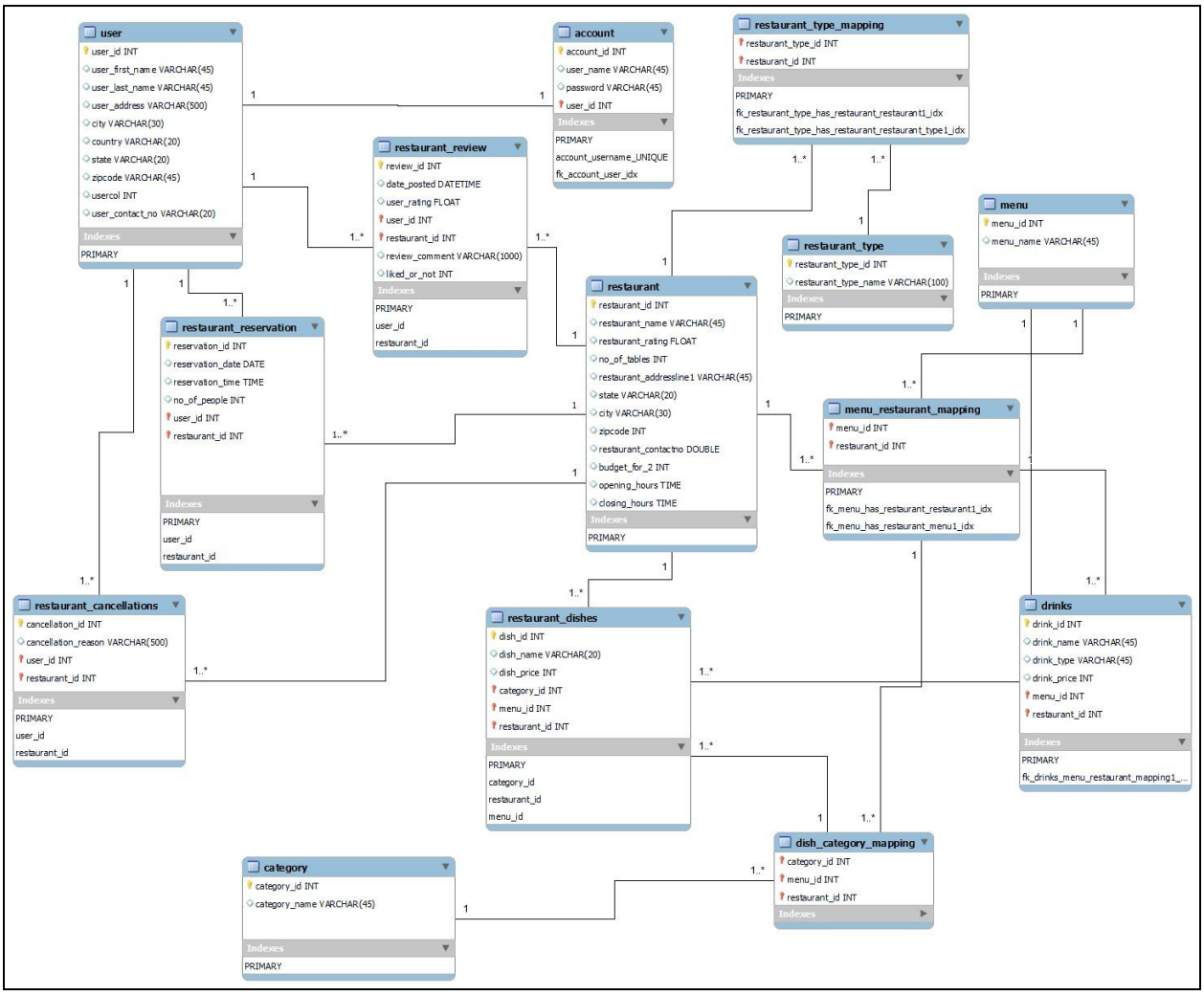
Motivation:
Being international students, we have a hard time finding a decent restaurant that is both within our budget and conveniently located. As a result, we had the idea to build a system where users could check out a restaurant's menu and other details and reserve a table. Making a recommendation system will help users find a good restaurant within their budget.

Scope:
The user can browse through the restaurants and view the reviews. They can post reviews for the restaurant after logging in or signing up for the system. The project offers two primary features. The recommendation module comes first, followed by the reservation module. The project's scope is limited to just the eateries in Boston City. The recommendation module will be able to respond to a variety of user inquiries, like "A decent restaurant serving Burgers", The reservation module offers a variety of features, including the ability to reserve a table at a restaurant, cancel a reservation, or check for review for any restaurant.

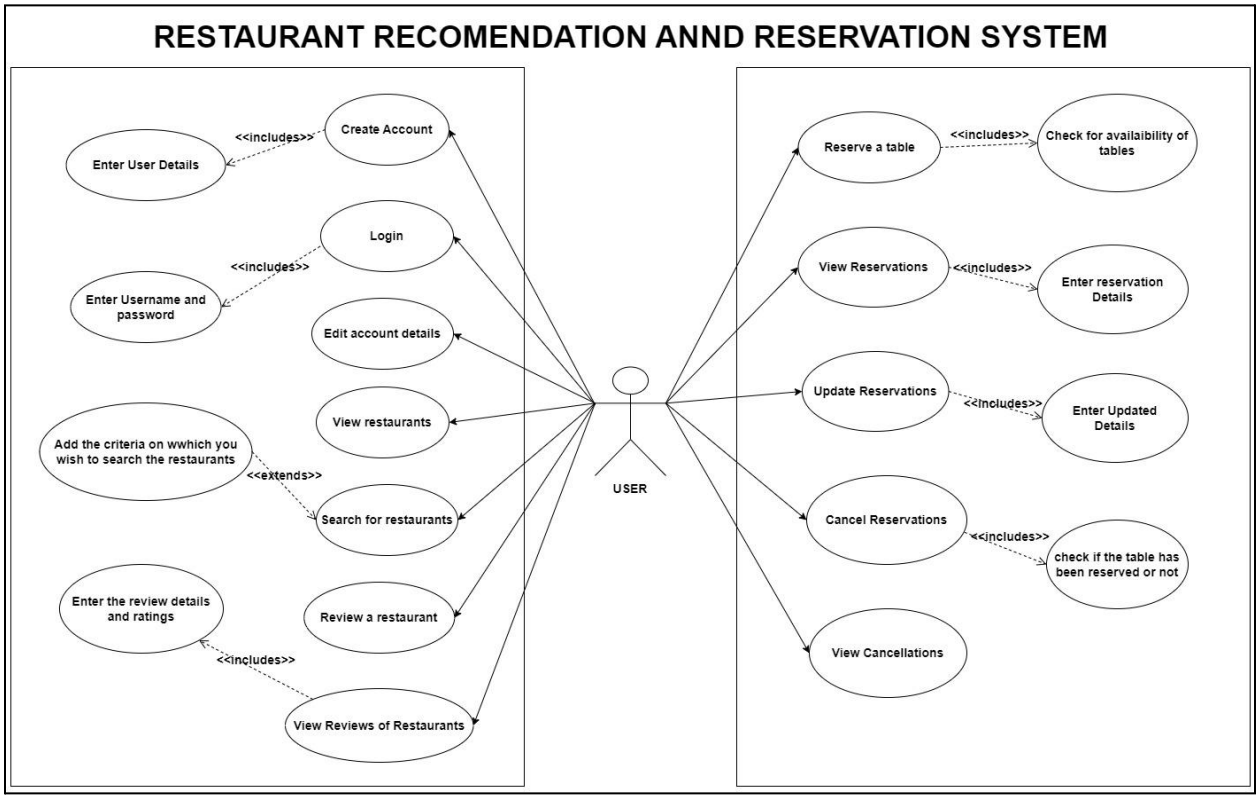
ER diagram of the online Restaurant Recommendation and Reservation System.



Physical Model diagram of the Restaurant Recommendation and Reservation System.



UML diagram (Use Case Diagram)



Explanation of some of the design decisions:

- The Restaurant Recommendation and Reservation System account has a username and password. The **account** table has a primary key named username. At the same time, every user has a unique id as user_id which is the primary key of the **user** table.
- The **restaurant_reservation** table has a primary key as ‘reservation_id’ which uniquely defines each reservation. It also has a foreign key ‘user_id’ and ’restaurant_id’ of the user and restaurant table respectively which distinguishes each reservation. Each restaurant can have many reservations.
- The **restaurant_cancellation** table has a primary key as ‘cancellation_id’ which uniquely defines each cancellation. It also has a foreign key ‘user_id’ and ’restaurant_id’ of the user and restaurant table respectively which distinguishes each cancellation. Each restaurant can have many cancellations.
- The **restaurant_review** table has a primary key as ‘review_id’ which uniquely defines each review. It also has a foreign key ‘user_id’ of the user and restaurant_id of the restaurant which shows exactly which person is giving a review abt which restaurant. Each restaurant can have many reviews.

- The list of all the restaurants is stored on a **restaurant** table. In this table, each restaurant is uniquely identified by the restaurant_id key which is a primary key of the table.
- A restaurant has many tables which can be reserved/canceled by the user. The number of tables available at a particular restaurant is also mentioned in the **restaurant** table
- The **restaurant** table is also linked with the '**menu**' table whose primary key is menu_id.
- The menu consists of two main types, that is drinks and food.
- All the data related to drinks and beverages like their name, type and the price is saved on the **drinks** table.
- The type of cuisine served and its id is stored in **category** tables. It is further linked with the **restaurant_dish** table which specifies the name, price, and type of the dish. Three foreign keys are passed to this table to compare with their respective tables. They are 'category_id', 'restaurant_id', and 'menu_id'. 'restaurant_dish_id' is set as the is the primary key for this table.

SOURCES OF DATA

We scraped information about restaurants from Google. To scrape data, Selenium and Chrome web drivers were utilized. Data has been scraped from three sources. Google is the first source, and the website food menu prices are the second. The restaurant data was directly taken from Google. Google will look up the restaurant, and information on the closest restaurants will be collected. The information about the restaurant, such as the review, the type of restaurant, the address, the phone number, the hours of operation, etc. The information, including meal categories, items, and costs, is scraped from a website's food menu prices. Both websites' data is real-time and extremely accurate. The information is dynamic and valid. The third source is Kaggle. We have taken a few datasets from Kaggle related to user comments and restaurant reservation details

NORMALIZATION

1st NF check:

- All the tables have a primary Key
    - Restaurant_id in restaurant table
    - User_id in user table
    - Account_id in account table
    -
- The values in each column of a tables are atomic
    - In menu sections some of the restaurants had both food as well as drinks menu, which resulted in non atomic values for the table. Thus a separate

table for the menu has been created and been mapped with restaurants in the restaurant_menu_mapping table.

- There are no repeating groups in the any of the table

2nd NF check:

- All requirements for 1st NF are met
- No partial dependencies in the above tables. All non key attributes depend on the primary key.
    - In menu sections some of the restaurants had both food as well as drinks menu, which resulted in non atomic values for the table. Thus a separate table for the menu has been created and been mapped with restaurants in the restaurant_menu_mapping table
    - The restaurant_type table has been created as a separate table which was earlier included into the restaurant table.
    - The categories were included into the restaurant_dishes table which showed partial dependency and thus it has been separated to a different table named dish_categories.
- There is No calculated data in the above tables

3rd NF check:

- All the requirements for 2nd NF are met.
- There is no transitive dependency for non-prime attributes
    - Earlier the user table contained account_id which was dependent on the primary key ‘user_id’ and user_name attribute which was dependent upon the account_id. Thus two separate tables have been formed, user table (with user details) and account table(with account details and the foreign key user_id from user table.

EXPLAINING OUR DATABASE

We have created 14 tables that contain in total 2631 data entries. Along with this, we have created 5 views plus 20 views for different use cases , 13 stored procedures, 5 Indexes, and 2 Triggers.

Tables:-

1. Restaurant
2. Restaurant_type
3. Resturant_type_mapping
4. User
5. Account
6. Menu
7. Menu_resturant_mapping
8. Drinks
9. Restaurant_dishes
10. Dish_category
11. Dish_category_mapping
12. Resturant_reservation
13. Resturant_cancellation
14. Resturant_review

Indexes:-

1. Index_restaurant_id
2. Index_name
3. Index_restaurant_review
4. Index_restaurant_review
5. index_restaurant_cancellation

Views:-

1. Dishprice
2. Drinktable
3. rescategory
4. restype
5. Userrating
6. Other list views of different use cases (mentioned below)

Stored Procedures:-

1. Searchby_budget
2. Searchby_category
3. Searchby_dishname

4. Searchby_dishprice
5. Searchby_drinktype
6. Searchby_rating
7. searchby_resturant_name
8. Searchby_restype
9. searchby_userrating
10. Searchby_zip
11. add_review_restaurant
12. restaurant_reservation
13. Restaurant_cancellation
14. Updatenooftables (Trigger)
15. Updatenooftables1 (Trigger)

SQL STATEMENTS FOR THE CONCEPTUAL MODEL

Restaurant table :

```
CREATE TABLE restaurant (
restaurant_id INT,
restaurant_name VARCHAR(100),
restaurant_rating FLOAT,
restaurant_contact_no VARCHAR(100),
opening_hrs time,
closing_hrs time,
budget_for_2 INT,
restaurant_address VARCHAR(100),
city VARCHAR(30),
state VARCHAR(20),
zipcode INT,
no_of_tables Int,
PRIMARY KEY (restaurant_id)
);
```

Restaurant_type table :

```
CREATE TABLE restaurant_type(
restaurant_type_id INT,
restaurant_type VARCHAR(100),
PRIMARY KEY (restaurant_type_id)
);
```

Restaurant_type_mapping table:

```
CREATE TABLE restaurant_type_mapping(
restaurant_type_id INT,
restaurant_id INT,
FOREIGN KEY (restaurant_type_id) REFERENCES restaurant_type(restaurant_type_id),
FOREIGN KEY (restaurant_id) REFERENCES restaurant(restaurant_id)
);
```

User table :

```
CREATE TABLE user(
user_id INT,
user_first_name VARCHAR(100),
user_last_name VARCHAR(100),
user_address VARCHAR(500),
city VARCHAR(30),
county VARCHAR(20),
state VARCHAR(20),
zipcode INT,
user_contact_no VARCHAR(20),
PRIMARY KEY (user_id)
);
```

Account table :

```
CREATE TABLE account (
account_id INT,
user_name VARCHAR(100),
password VARCHAR(100),
user_id INT,
PRIMARY KEY (account_id),
FOREIGN KEY (user_id) REFERENCES user(user_id)
);
```

Menu :

```
CREATE TABLE menu(
menu_id INT,
menu_name VARCHAR(100),
PRIMARY KEY (menu_id)
);
```

Menu_resturant_mapping table :

```
CREATE TABLE menu_restaurant_mapping(
menu_id INT,
restaurant_id INT,
FOREIGN KEY (menu_id) REFERENCES menu(menu_id),
FOREIGN KEY (restaurant_id) REFERENCES restaurant(restaurant_id)
);
```

Drinks table :

```
CREATE TABLE drinks(
drink_id INT,
drink_name VARCHAR(100),
drink_type VARCHAR(100),
drink_price float,
menu_id INT,
restaurant_id INT,
PRIMARY KEY (drink_id),
FOREIGN KEY (menu_id) REFERENCES  menu(menu_id),
FOREIGN KEY (restaurant_id) REFERENCES  restaurant(restaurant_id)
);
```

Cuisine table :

```
CREATE TABLE cuisine(
cuisine_id INT,
cuisine_name VARCHAR(50),
PRIMARY KEY (cuisine_id)
);
```

Dish_categories table :

```
CREATE TABLE dish_categories(
category_id INT,
category_name VARCHAR(100),
PRIMARY KEY (category_id)
);
```

Dish_category_mapping table :

```
CREATE TABLE dish_category_mapping(
category_id INT,
menu_id INT,
restaurant_id INT,
```

```
FOREIGN KEY (category_id) REFERENCES dish_categories(category_id),
FOREIGN KEY (menu_id) REFERENCES  menu(menu_id),
FOREIGN KEY (restaurant_id) REFERENCES restaurant(restaurant_id)
);
```

Restaurant_dishes table :

```
CREATE TABLE restaurant_dishes(
dish_id INT,
dish_name VARCHAR(500),
dish_price FLOAT,
-- cuisines_id INT,
category_id INT,
menu_id INT,
restaurant_id INT,
PRIMARY KEY (dish_id),
FOREIGN KEY (category_id) REFERENCES dish_categories(category_id),
FOREIGN KEY (menu_id) REFERENCES  menu(menu_id),
FOREIGN KEY (restaurant_id) REFERENCES restaurant(restaurant_id)
);
```

Restaurant_reservations :

```
CREATE TABLE restaurant_reservation(
reservation_id INT,
no_of_people INT,
reservation_time time,
reservation_date date,
restaurant_id int,
user_id INT,
PRIMARY KEY (reservation_id),
FOREIGN KEY (restaurant_id) REFERENCES restaurant (restaurant_id),
FOREIGN KEY (user_id) REFERENCES user(user_id)
);
```

Restaurant_review :

```
CREATE TABLE restaurant_review(
review_id INT,
review_comment VARCHAR(1000),
liked_or_not INT,
user_rating FLOAT,
date_posted date,
```

```
restaurant_id INT,
user_id INT,
PRIMARY KEY (review_id),
FOREIGN KEY (restaurant_id) REFERENCES restaurant (restaurant_id),
FOREIGN KEY (user_id) REFERENCES user(user_id)
);
```

Restaurant_cancellation :

```
CREATE TABLE restaurant_cancellation(
cancellation_id INT,
cancellation_reason Varchar(500),
restaurant_id INT,
user_id INT,
PRIMARY KEY (cancellation_id),
FOREIGN KEY (restaurant_id) REFERENCES restaurant (restaurant_id),
FOREIGN KEY (user_id) REFERENCES user(user_id)
);
```

VIEWS CREATION STATEMENTS:

Dishprice View :

```
CREATE
    ALGORITHM = UNDEFINED
    DEFINER = `root`@`localhost`
    SQL SECURITY DEFINER
VIEW `dishprice` AS
    SELECT
        `r`.`restaurant_name` AS `restaurant_name`,
        `d`.`dish_name` AS `dish_name`,
        `d`.`dish_price` AS `dish_price`,
        `r`.`restaurant_rating` AS `restaurant_rating`,
        `r`.`opening_hrs` AS `opening_hrs`,
        `r`.`closing_hrs` AS `closing_hrs`,
        `r`.`budget_for_2` AS `budget_for_2`,
        `r`.`restaurant_address` AS `restaurant_address`,
        `r`.`zipcode` AS `zipcode`,
        `r`.`city` AS `city`,
        `r`.`state` AS `state`
    FROM
```

```
    (`restaurant` `r`
    JOIN `restaurant_dishes` `d` ON ((`r`.`restaurant_id` = `d`.`restaurant_id`)))
```

Drinktable View :

```
CREATE
  ALGORITHM = UNDEFINED
  DEFINER = `root`@`localhost`
  SQL SECURITY DEFINER
VIEW `drinktable` AS
  SELECT
    `r`.`restaurant_name` AS `restaurant_name`,
    `d`.`drink_name` AS `drink_name`,
    `d`.`drink_type` AS `drink_type`,
    `d`.`drink_price` AS `drink_price`,
    `r`.`restaurant_rating` AS `restaurant_rating`,
    `r`.`opening_hrs` AS `opening_hrs`,
    `r`.`closing_hrs` AS `closing_hrs`,
    `r`.`budget_for_2` AS `budget_for_2`,
    `r`.`restaurant_address` AS `restaurant_address`,
    `r`.`zipcode` AS `zipcode`,
    `r`.`city` AS `city`,
    `r`.`state` AS `state`
  FROM
    (`restaurant` `r`
    JOIN `drinks` `d` ON ((`r`.`restaurant_id` = `d`.`restaurant_id`)))
```

Rescategory View :

```
CREATE
  ALGORITHM = UNDEFINED
  DEFINER = `root`@`localhost`
  SQL SECURITY DEFINER
VIEW `restype` AS
  SELECT DISTINCT
    `r`.`restaurant_name` AS `restaurant_name`,
    `r`.`restaurant_rating` AS `restaurant_rating`,
    `r`.`opening_hrs` AS `opening_hrs`,
    `r`.`closing_hrs` AS `closing_hrs`,
    `r`.`budget_for_2` AS `budget_for_2`,
    `r`.`restaurant_address` AS `restaurant_address`,
    `r`.`zipcode` AS `zipcode`,
```

```
    `r`.`city` AS `city`,
    `r`.`state` AS `state`,
    `t`.`restaurant_type` AS `restaurant_type`
  FROM
    ((`restaurant` `r`
    JOIN `restaurant_type_mapping` `tm` ON ((`r`.`restaurant_id` = `tm`.`restaurant_id`)))
    JOIN `restaurant_type` `t` ON ((`t`.`restaurant_type_id` = `tm`.`restaurant_type_id`)))
```

User Rating View :

```
CREATE
  ALGORITHM = UNDEFINED
  DEFINER = `root`@`localhost`
  SQL SECURITY DEFINER
VIEW `userrating` AS
  SELECT
    `r`.`restaurant_name` AS `restaurant_name`,
    `r`.`restaurant_rating` AS `restaurant_rating`,
    max(`re`.`user_rating`) AS `user_rating`,
    `r`.`opening_hrs` AS `opening_hrs`,
    `r`.`closing_hrs` AS `closing_hrs`,
    `r`.`budget_for_2` AS `budget_for_2`,
    `r`.`restaurant_address` AS `restaurant_address`,
    `r`.`zipcode` AS `zipcode`,
    `r`.`city` AS `city`,
    `r`.`state` AS `state`
  FROM
    (`restaurant` `r`
    JOIN `restaurant_review` `re` ON ((`r`.`restaurant_id` = `re`.`restaurant_id`)))
```

Restype

```
CREATE
  ALGORITHM = UNDEFINED
  DEFINER = `root`@`localhost`
  SQL SECURITY DEFINER
VIEW `r3_db_system`.`restype` AS
  SELECT DISTINCT
    `r`.`restaurant_name` AS `restaurant_name`,
    `r`.`restaurant_rating` AS `restaurant_rating`,
    `r`.`opening_hrs` AS `opening_hrs`,
    `r`.`closing_hrs` AS `closing_hrs`,
```

```
        `r`.`budget_for_2` AS `budget_for_2`,
        `r`.`restaurant_address` AS `restaurant_address`,
        `r`.`zipcode` AS `zipcode`,
        `r`.`city` AS `city`,
        `r`.`state` AS `state`,
        `t`.`restaurant_type` AS `restaurant_type`
    FROM
        ((`r3_db_system`.`restaurant` `r`
        JOIN `r3_db_system`.`restaurant_type_mapping` `tm` ON ((`r`.`restaurant_id` =
`tm`.`restaurant_id`)))
        JOIN `r3_db_system`.`restaurant_type` `t` ON ((`t`.`restaurant_type_id` =
`tm`.`restaurant_type_id`)))
```

STORED PROCEDURES:

Review Restaurant

```
DELIMITER //
CREATE DEFINER=`root`@`localhost` PROCEDURE `add_restaurant_review`(rev_comment
VARCHAR(1000),
likedornot INT,
userrating FLOAT,
res_name VARCHAR(100),
user_fname VARCHAR(100))
BEGIN
DECLARE resid INT;
DECLARE userid INT;
DECLARE reviewid INT;
set @resname = res_name;
select restaurant_id INTO resid from restaurant where restaurant_name like
CONCAT('%',@resname,'%');
set @uname = user_fname;
select user_id INTO userid from user where user_first_name = @uname;
select max(review_id)+1 into  reviewid from restaurant_review;
INSERT INTO restaurant_review values (reviewid, rev_comment, likedornot, userrating,
CURDATE(), resid, userid);
END
//
```

Restaurant Reservation

```
DELIMITER //
CREATE PROCEDURE `restaurant_reservation` (no_of_people int, reservation_time time,
reservation_date date,IN  res_name varchar(100), IN user_firstname varchar(100))
BEGIN
DECLARE resid INT;
DECLARE userid INT;
DECLARE reservid INT;
set @resname = res_name;
select restaurant_id INTO resid from restaurant where restaurant_name like
CONCAT('%',@resname,'%');
set @uname = user_firstname;
select user_id INTO userid from user where user_first_name = @uname;
select max(reservation_id)+1 into  reservid from restaurant_reservation;
INSERT INTO restaurant_reservation values (reservid,no_of_people, reservation_time,
reservation_date, resid, userid);
END;
//
```

Restaurant Cancellation

```
DELIMITER //
CREATE PROCEDURE `restaurant_cancellation` (IN cancelreason varchar(100), IN
reserv_id INT)
BEGIN
DECLARE userid INT;
DECLARE cancelid INT;
DECLARE restaurantid INT;
    IF  (select reservation_id from restaurant_reservation where reservation_id = reserv_id)
THEN
            BEGIN
            select user_id INTO userid from restaurant_reservation where reservation_id =
reserv_id;
            select restaurant_id INTO restaurantid from restaurant_reservation where
reservation_id = reserv_id;
            select max(cancellation_id)+1 into  cancelid from restaurant_cancellation;
            INSERT INTO restaurant_cancellation values (cancelid, cancelreason, restaurantid,
userid);
            UPDATE restaurant set no_of_tables = no_of_tables + 1 where restaurant_id =
restaurantid;
            DELETE from restaurant_reservation where reservation_id = reserv_id;
```

```
            END;
    else
            BEGIN
       select 'Reservation ID does not exist';
    END;
   END IF;
END;
//
```

Search restaurant by budget

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `searchby_budget`(IN budget int)
BEGIN
SELECT restaurant_name,restaurant_rating, opening_hrs, closing_hrs,budget_for_2,
restaurant_address, zipcode, city, state
FROM restaurant where budget_for_2 <= budget ;
END
```

Search restaurant by zipcode

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `searchby_zip`(in zip int)
BEGIN
SELECT restaurant_name,restaurant_rating, opening_hrs, closing_hrs,budget_for_2,
restaurant_address, zipcode, city, state
FROM restaurant where zipcode = zip ;
END
```

Search restaurant by category

```
CREATE  PROCEDURE `searchby_category`(IN category varchar(100))
BEGIN
set @categoryname = category;
SELECT distinct restaurant_name,restaurant_rating, opening_hrs, closing_hrs,budget_for_2,
restaurant_address, zipcode, city, state
FROM restaurant r INNER JOIN restaurant_dishes d
ON r.restaurant_id=d.restaurant_id
WHERE d.category_id in (
    select category_id from dish_categories
   where category_name like CONCAT('%',@categoryname,'%'));
END;
```

Search restaurant by Dish Name

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `searchby_dishname`(IN dishname
varchar(100))
BEGIN
SET @dish_name = dishname;
 SELECT restaurant_name,dish_name,dish_price,restaurant_rating,budget_for_2,
restaurant_address, zipcode, city, state
 FROM dishprice
  WHERE dish_name like  CONCAT('%',@dish_name,'%');
END
```

Search restaurant by Dish Price

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `searchby_dishprice`(IN price int)
BEGIN
 SELECT restaurant_name,dish_name,dish_price,restaurant_rating, opening_hrs,
closing_hrs,budget_for_2, restaurant_address, zipcode, city, state
 FROM dishprice
  WHERE dish_price <= price;
END
```

Search restaurant by Drink Price

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `searchby_drinkprice`(IN price int)
BEGIN
 SELECT restaurant_name,drink_name,drink_type,drink_price,budget_for_2,
restaurant_address, zipcode
 FROM drinktable
  WHERE drink_price <= price;
END
```

Search restaurant by Drink Type

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `searchby_drinktype1`(IN drinktype
varchar(100))
BEGIN
SET @drink_type = drinktype;
SELECT restaurant_name,drink_type,restaurant_rating, opening_hrs,
closing_hrs,budget_for_2, restaurant_address, zipcode, city, state
 FROM drinktable
WHERE drink_type like CONCAT('%',@drink_type,'%');
END
```

Search restaurant by Rating

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `searchby_rating`(IN rating float)
BEGIN
SELECT restaurant_name,restaurant_rating, opening_hrs, closing_hrs,budget_for_2,
restaurant_address, zipcode, city, state
FROM restaurant where restaurant_rating >= rating ;
END
```

Search restaurant by Restaurant Name

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `searchby_restaurant_name`(IN
resname varchar(100))
BEGIN
SELECT restaurant_name,restaurant_rating, opening_hrs, closing_hrs,budget_for_2,
restaurant_address, zipcode, city, state
FROM restaurant where restaurant_name = resname ;
END
```

Search restaurant by Restaurant Type

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `searchby_restaurant_name`(IN
resname varchar(100))
BEGIN
SELECT restaurant_name,restaurant_rating, opening_hrs, closing_hrs,budget_for_2,
restaurant_address, zipcode, city, state
FROM restaurant where restaurant_name = resname ;
END
```

Search restaurant by User rating

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `searchby_userrating`(IN rate float)
BEGIN
SELECT restaurant_name,restaurant_rating,user_rating, opening_hrs,
closing_hrs,budget_for_2, restaurant_address, zipcode, city, state
FROM userrating
WHERE user_rating >= rate;
END
```

TRIGGERS:

Updatenooftables: **Trigger to update number of tables after restaurant reservation**

```
CREATE TRIGGER updatenooftables
AFTER INSERT
ON restaurant_reservation
```

```
FOR EACH ROW
UPDATE restaurant set no_of_tables = no_of_tables - 1 where restaurant_id =
new.restaurant_id;
```

Updatenooftables1: Trigger to update no of tables after cancelling the reservation

```
CREATE TRIGGER updatenooftables1
AFTER INSERT
ON restaurant_cancellation
FOR EACH ROW
UPDATE restauran set no_of_tables = no_of_tables + 1 where restaurant_id =
new.restaurant_id;
```

INDEXES

index_restaurant_id

```
CREATE INDEX index_restaurant_id ON restaurant(restaurant_id);
```

index_name

```
CREATE INDEX index_name ON user(user_first_name, user_last_name);
```

index_restaurant_review

```
CREATE UNIQUE INDEX index_restaurant_review ON restaurant_review(review_id);
```

index_restaurant_reservation

```
CREATE UNIQUE INDEX index_restaurant_reservation ON
restaurant_reservation(reservation_id);
```

index_restaurant_cancellation

```
CREATE UNIQUE INDEX index_restaurant_cancellation ON
restaurant_cancellation(cancellation_id);
```

USE CASE

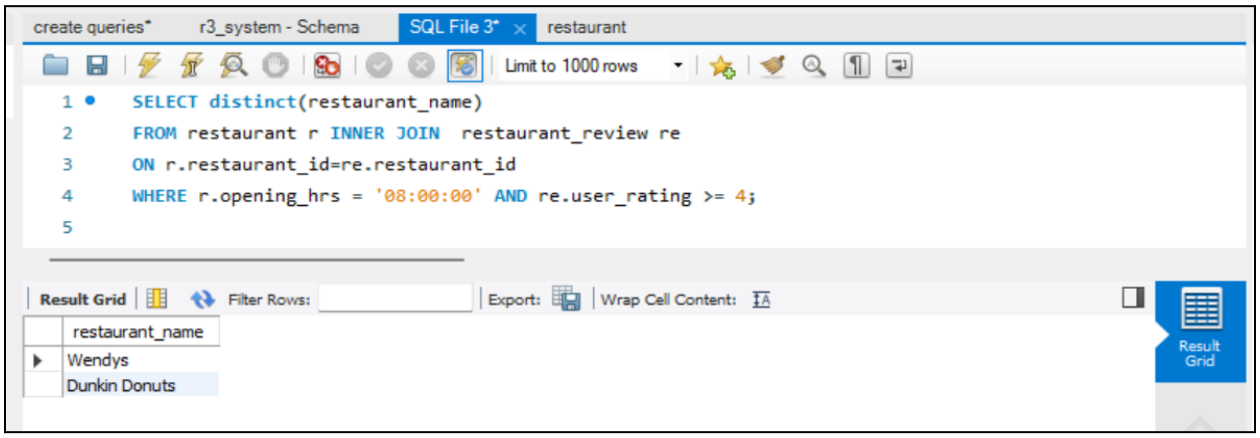
Use Case 1: Search for a Restaurant that opens at 8 am with a rating above 4.

Description: The user searches for restaurants above a rating of 4 that opens at 8 am.
Actors: User
Precondition: The user must be logged in.
Steps:
Actor Action: The user looks for restaurants above the rating of 4 that opens at 8 am.

System Response: Display all the details of the best restaurants that are open at 8 am with a rating above 4.
Postcondition: The user will decide which restaurant to visit.

SQL Query:

```
SELECT distinct(restaurant_name)
FROM restaurant r INNER JOIN  restaurant_review re
ON r.restaurant_id=re.restaurant_id
WHERE r.opening_hrs = '08:00:00' AND re.user_rating >= 4;
```

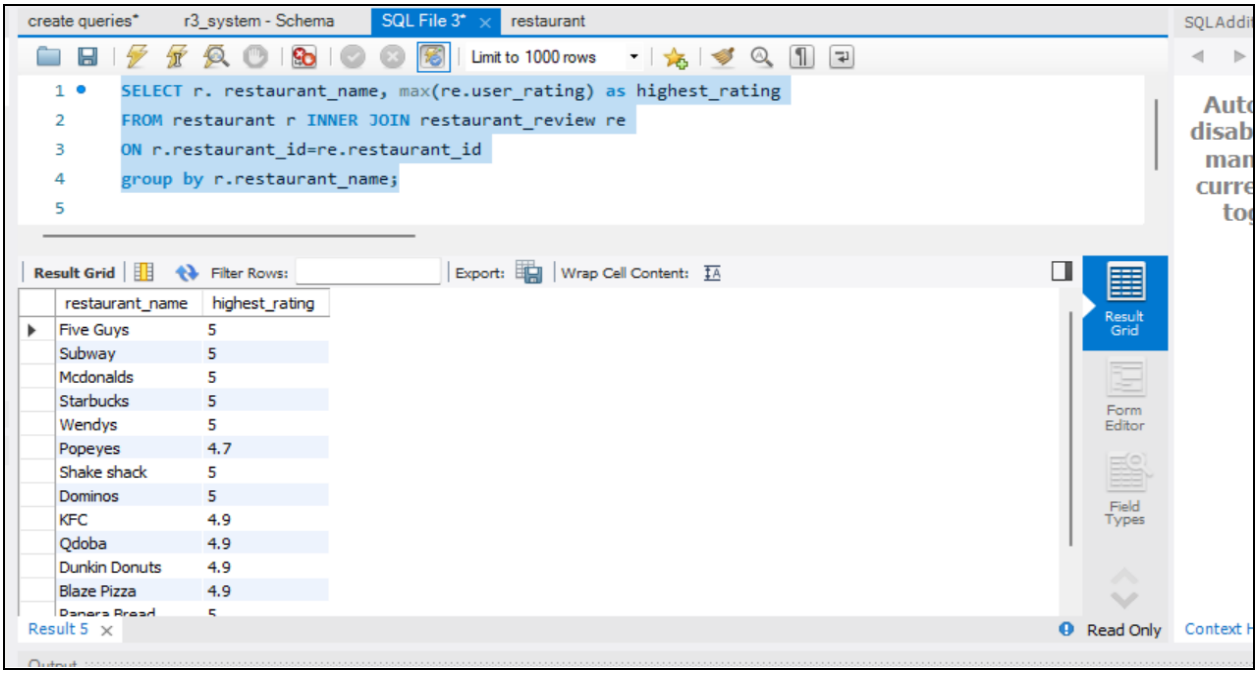


Use Case 2: Check the restaurant with the best User Rating.

Description: The user checks which restaurant has the best rating.
Actors: User
Precondition: The user must be logged in to their account.
Steps:
Actor Action: The user searches for user ratings for restaurants in reviews.
System Response: Show all the restaurants according to rating.
Post Condition: System will display the list of restaurants according to rating.

SQL Query:

```
SELECT r. restaurant_name, max(re.user_rating) as highest_rating
FROM restaurant r INNER JOIN restaurant_review re
ON r.restaurant_id=re.restaurant_id
group by r.restaurant_name;
```

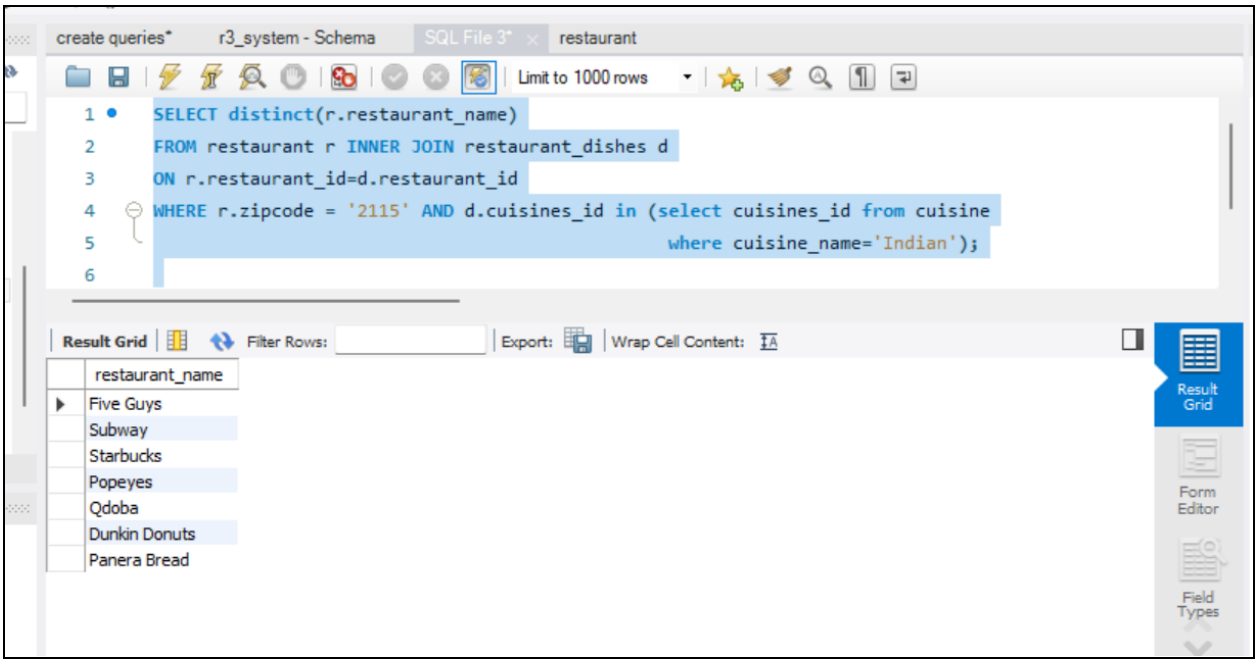


Use Case 3: Search for restaurants offering Indian cuisine in the zip code 2115

Description: The user searches for a restaurant that offers Indian food in a particular zip code
Actors: User
Precondition: The user must be logged in from his/her account
Steps:
Actor action – The user searches for restaurants offering Indian food
System Responses – Displays all the restaurants offering Indian food in that zip code
Post Condition: The user can view all the restaurants and reserve a table at a restaurant of his choice.
Alternate Path: The user request is not correct and the system throws an error
Error: No restaurants found that offer this combination of features

SQL Query:

```
SELECT distinct(r.restaurant_name)
FROM restaurant r INNER JOIN restaurant_dishes d
ON r.restaurant_id=d.restaurant_id
WHERE r.zipcode = '2115' AND d.cuisines_id in (select cuisines_id from cuisine
                                                where cuisine_name='Indian');
```

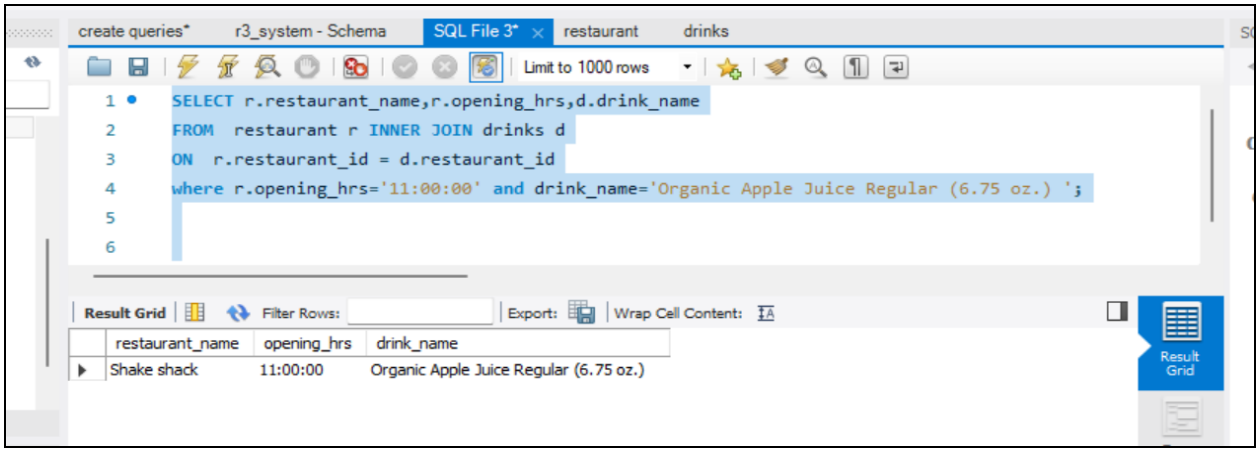


Use Case 4: View a restaurant that offers Organic Apple Juice Regular (6.75 oz.) and opens at 11 am

Description: The user views a restaurant that offers Organic Apple Juice Regular (6.75 oz.) and opens at 11 am
Actors: User
Precondition: The user must be logged in from his/her account
Steps:
Actor action: The user views a restaurant that offers Organic Apple Juice Regular (6.75 oz.) and opens at 11 am
System Responses: Displays details of those restaurants
Post Condition: System will display those restaurants
Error: User not logged into his account or No restaurants found that offer Frappuccino at 9 am

SQL Query:

```
SELECT r.restaurant_name,r.opening_hrs,d.drink_name
FROM  restaurant r INNER JOIN drinks d
ON  r.restaurant_id = d.restaurant_id
where r.opening_hrs='11:00:00' and drink_name='Organic Apple Juice Regular (6.75 oz.) ';
```

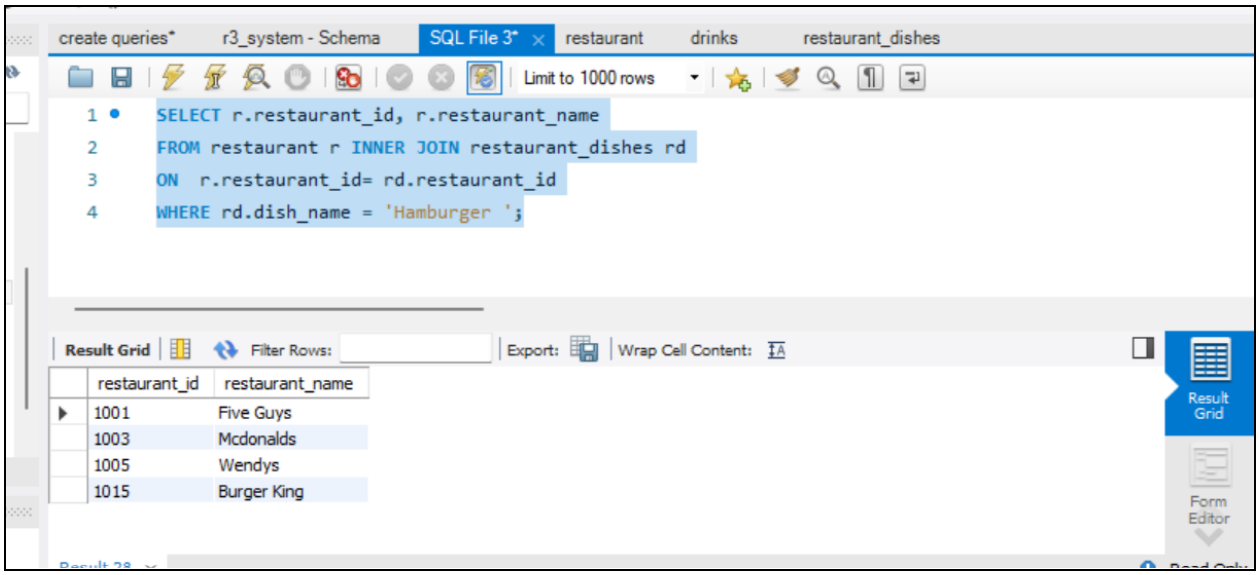


Use Case 5: Search for a restaurant that serves Hamburgers.

Description: The user searches for different restaurants that serve Hamburgers
Actor: User
Precondition: The user needs to log in to his account
Steps:
Actor action: The user requests the Restaurant details which serve Hamburgers
System Responses: Details of all the Restaurants serving Hamburgers will be displayed to the user
Post Condition: The user will be able to filter out many more features and select a particular restaurant that he/she likes
Error: The user cannot find the restaurant that serves Hamburgers

SQL Query:-

```
SELECT r.restaurant_id, r.restaurant_name
FROM restaurant r INNER JOIN restaurant_dishes rd
ON  r.restaurant_id= rd.restaurant_id
WHERE rd.dish_name = 'Hamburger ';
```

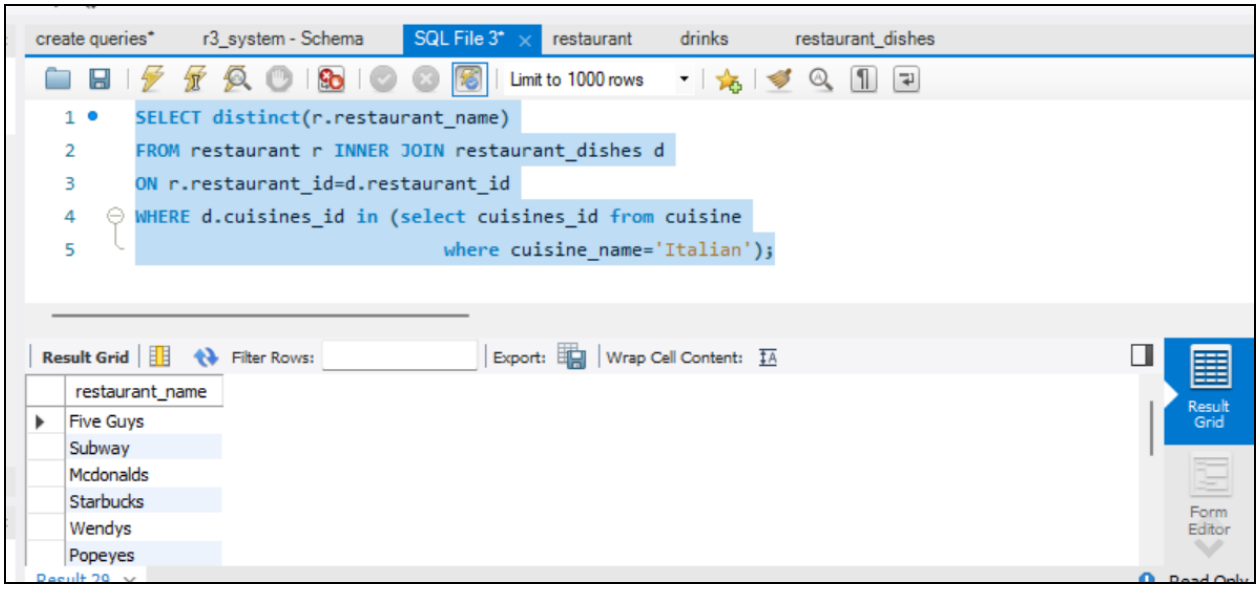


Use case 6: Search for a restaurant offering Italian cuisine.

Description: The user searches for different restaurants offering Italian cuisine
Actor: User
Precondition: The user needs to log in to his account
Steps:
Actor action: The user requests the Details of Restaurants having Italian Cuisine
System Responses: Details of all the Restaurants offering Italian Cuisines will be displayed to the user
Post Condition: The user will be able to filter out many more features and select a particular restaurant that he/she likes
Alternate Path: If no such cuisine is present in the database the system will show a message that no such cuisine is provided by the restaurant's
Error: Non-alpha-numeric characters allowed

SQL Query:-

```
SELECT distinct(r.restaurant_name)
FROM restaurant r INNER JOIN restaurant_dishes d
ON r.restaurant_id=d.restaurant_id
WHERE d.cuisines_id in (select cuisines_id from cuisine
                                                    where cuisine_name='Italian');
```

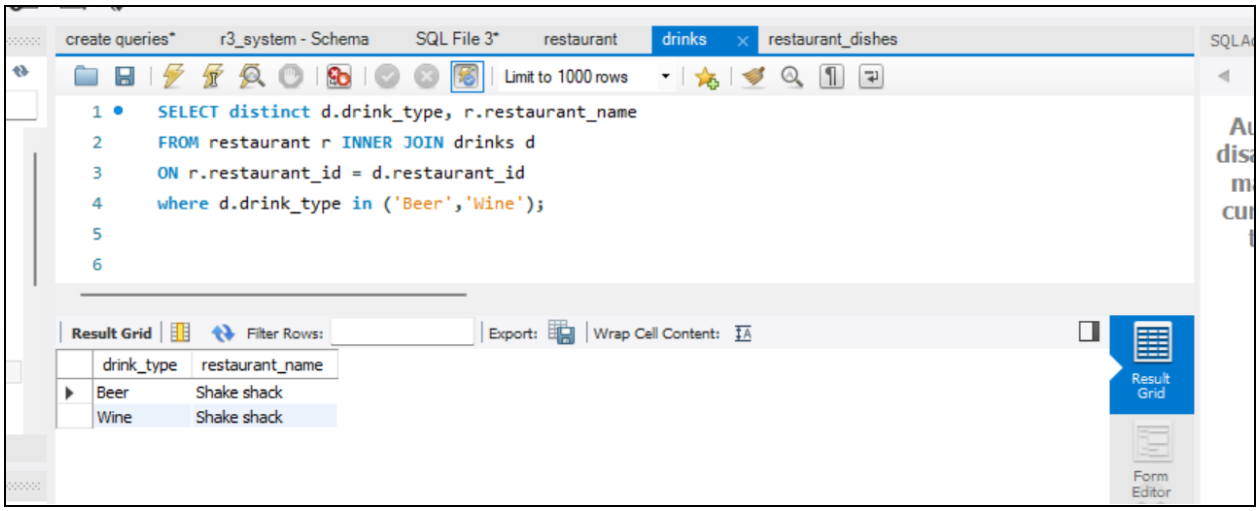


Use Case 7: View the restaurant which serves alcoholic (Beer, wine) drinks.

Description: The user searches for different restaurants offering alcoholic (Beer, wine) drinks
Actor: User
Precondition: The user must be logged into his account
Steps:
Actor action: The user requests a list of restaurants that serve alcoholic drinks
System Responses: Details of the restaurants meeting the criteria are displayed
Post Condition: The user will be able to filter out many more features and select a particular restaurant that he/she likes and further reserve a table if he/she wants
Alternate Path: The user has not logged into his account
Error: User not logged in

SQL Query:

```
SELECT distinct d.drink_type, r.restaurant_name
FROM restaurant r INNER JOIN drinks d
ON r.restaurant_id = d.restaurant_id
where d.drink_type in ('Beer', 'Wine');
```

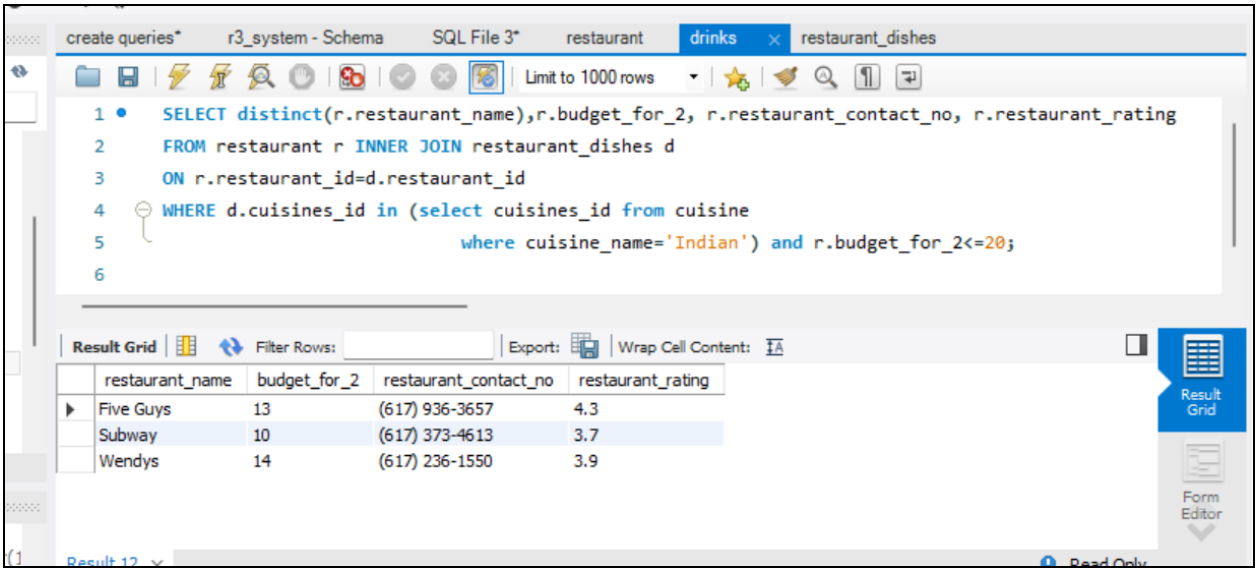


Use Case 8: View a restaurant with Indian cuisine with a specific budget (say less than $20)

Description: The user views a restaurant within a specific price
Actors: User
Precondition: The user must be logged in
Steps:
Actor action – The user views a restaurant with a budget of 2 below 20$
System Responses – restaurant details would be displayed
Post Condition: system displays restaurant reviews
Error: No restaurants found within the user's budget

SQL Query:-

```
SELECT distinct(r.restaurant_name),r.budget_for_2, r.restaurant_contact_no,
r.restaurant_rating
FROM restaurant r INNER JOIN restaurant_dishes d
ON r.restaurant_id=d.restaurant_id
WHERE d.cuisines_id in (select cuisines_id from cuisine where cuisine_name='Indian')
and r.budget_for_2<=20;
```

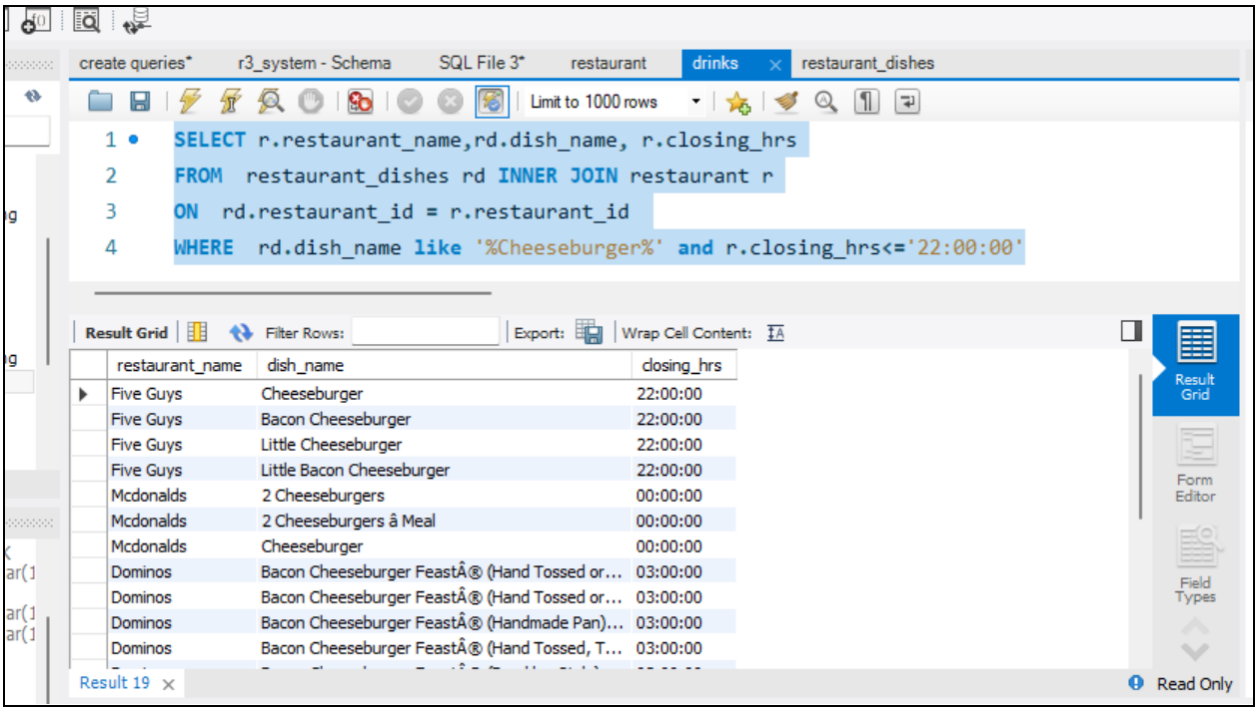


Use Case 9: View a restaurant that serves Cheeseburger and is open till 10 pm

Description: The user searches for different restaurants offering CheeseBurger and is open till 10 pm
Actor: User
Precondition: The user must be logged into his account
Steps:
Actor action: The user requests the details of a specific restaurant that serves Cheeseburger and closes at 10 pm
System Responses: Displays the list of restaurants
Post Condition: The user can decide which restaurant he wants to visit
Alternate Path: The user enters the wrong dish name
Error: No such restaurant is available

SQL Query:

```
SELECT r.restaurant_name,rd.dish_name, r.closing_hrs
FROM  restaurant_dishes rd INNER JOIN restaurant r
ON  rd.restaurant_id = r.restaurant_id
WHERE  rd.dish_name like '%Cheeseburger%' and r.closing_hrs<='22:00:00'
```

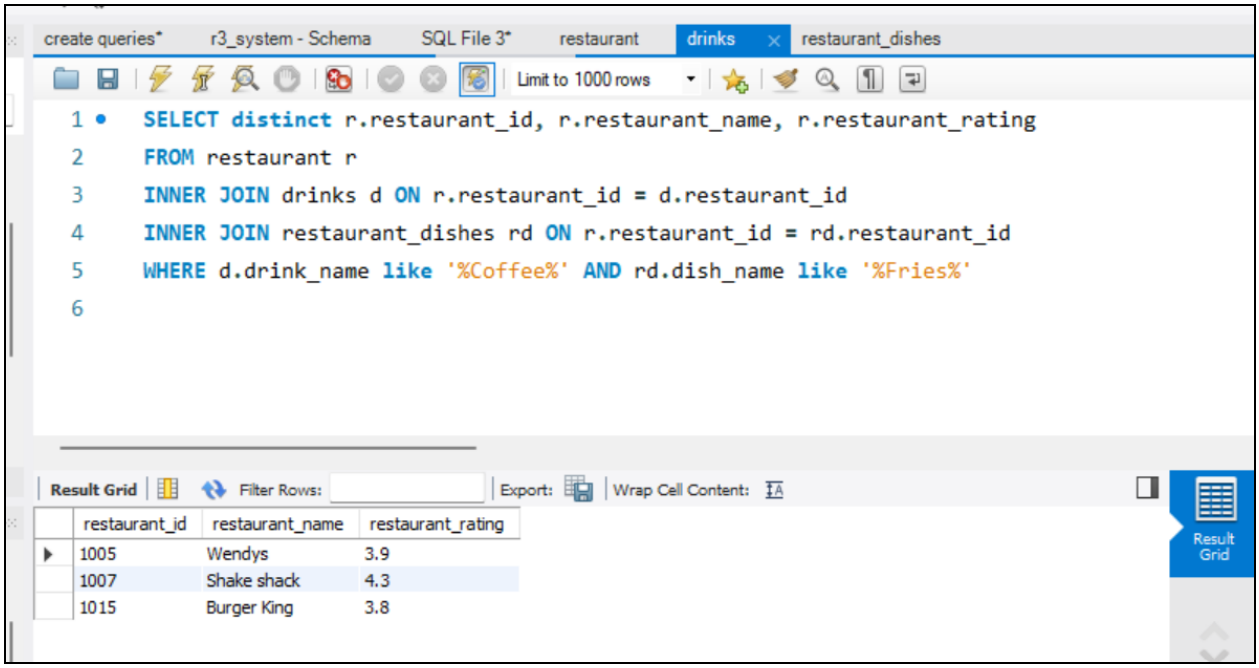


Use case 10: View the restaurant's offerings of Coffee and Fries

Description: The user searches for a restaurant offering coffee and fries
Actor: User
Precondition: The user needs to log in to his account.
Steps:
Actor action: The user requests the Details of Restaurants having coffee and fries
System Responses: Details of all the Restaurants offering Coffee and Fries
Post Condition: The user will be able to filter out many more features and select a particular restaurant that he/she likes
Alternate Path: If no such restaurant is present in the database the system will show a message that no such restaurant is present serving coffee and fries
Error: Non-alpha-numeric characters allowed

SQL Query:-

```
SELECT distinct r.restaurant_id, r.restaurant_name, r.restaurant_rating
FROM restaurant r
INNER JOIN drinks d ON r.restaurant_id = d.restaurant_id
INNER JOIN restaurant_dishes rd ON r.restaurant_id = rd.restaurant_id
WHERE d.drink_name like '%Coffee%' AND rd.dish_name like '%Fries%'
```

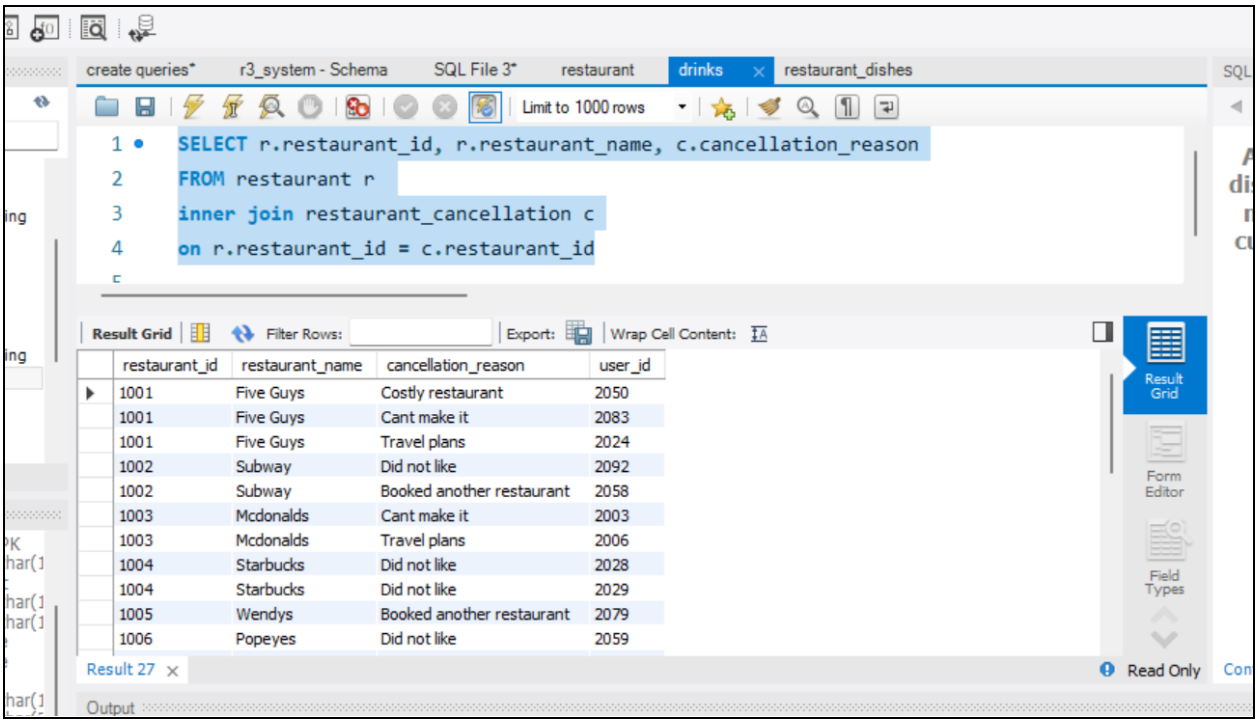


Use case 11: View cancellations done for a restaurant and their reasons

Description: The user searches for cancellations done for a restaurant and its reasons
Actor: User
Precondition: The user needs to log in to his account
Steps:
Actor action: The user requests the details of cancellations done for a restaurant and their reasons
System Responses: Details of cancellations done for a restaurant and their reasons
Post Condition: The user will be able to filter out many more features and select a particular restaurant that he/she likes
Alternate Path: The user enters the wrong restaurant name
Error: No such restaurant is available

SQL Query:-

SELECT r.restaurant_id, r.restaurant_name, c.cancellation_reason
FROM restaurant r
inner join restaurant_cancellation c
on r.restaurant_id = c.restaurant_id



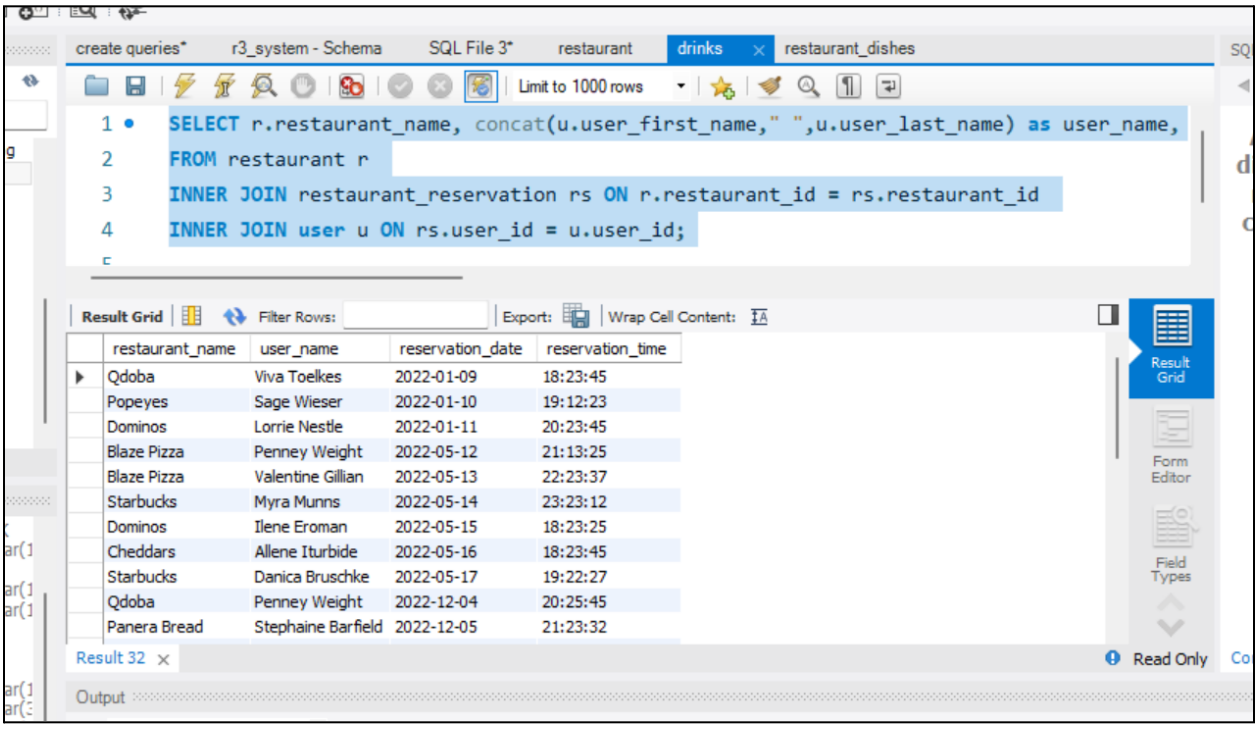
Use Case 12: What are the restaurant details, user details, and reservation details where the users reserved a table?

Description: The user searches for restaurant details, user details, and reservation details where the user reserved a table
Actor: User
Precondition: The User needs to log in to his account.
Steps:
Actor action: The user requests the Details of the Restaurant
System Responses: Details of the Restaurant appear.
Post Condition: The user will be able to check all the details of the restaurant he booked a table.
Alternate Path: If no such reservation is present in the database the system will show a message that no such reservation is present.

SQL Query:-

```
SELECT r.restaurant_name, concat(u.user_first_name," ",u.user_last_name) as
user_name, rs.reservation_date,rs.reservation_time
FROM restaurant r
INNER JOIN restaurant_reservation rs ON r.restaurant_id = rs.restaurant_id
```

INNER JOIN user u ON rs.user_id = u.user_id;

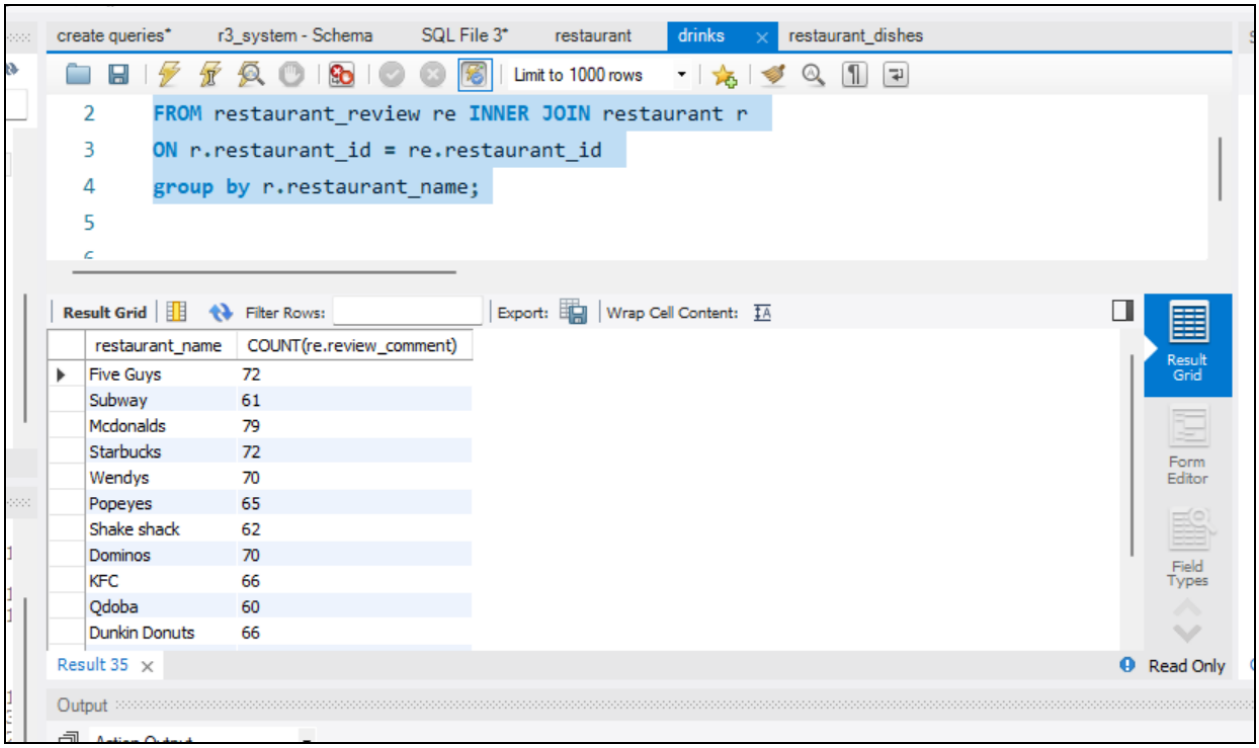


Use Case 13: How many reviews have a restaurant received to date?

Description: The user search for reviews a restaurant has received to date
Actor: User
Precondition: The user needs to log in to his account.
Steps:
Actor action: The user requests for reviews a restaurant has received to date
System Responses: Shows details of that restaurant received to date
Post Condition: The user will be able to see how many people have reviewed that restaurant.

SQL Query:-

SELECT r.restaurant_name,COUNT(re.review_comment)
FROM restaurant_review re INNER JOIN restaurant r
ON r.restaurant_id = re.restaurant_id
group by r.restaurant_name;

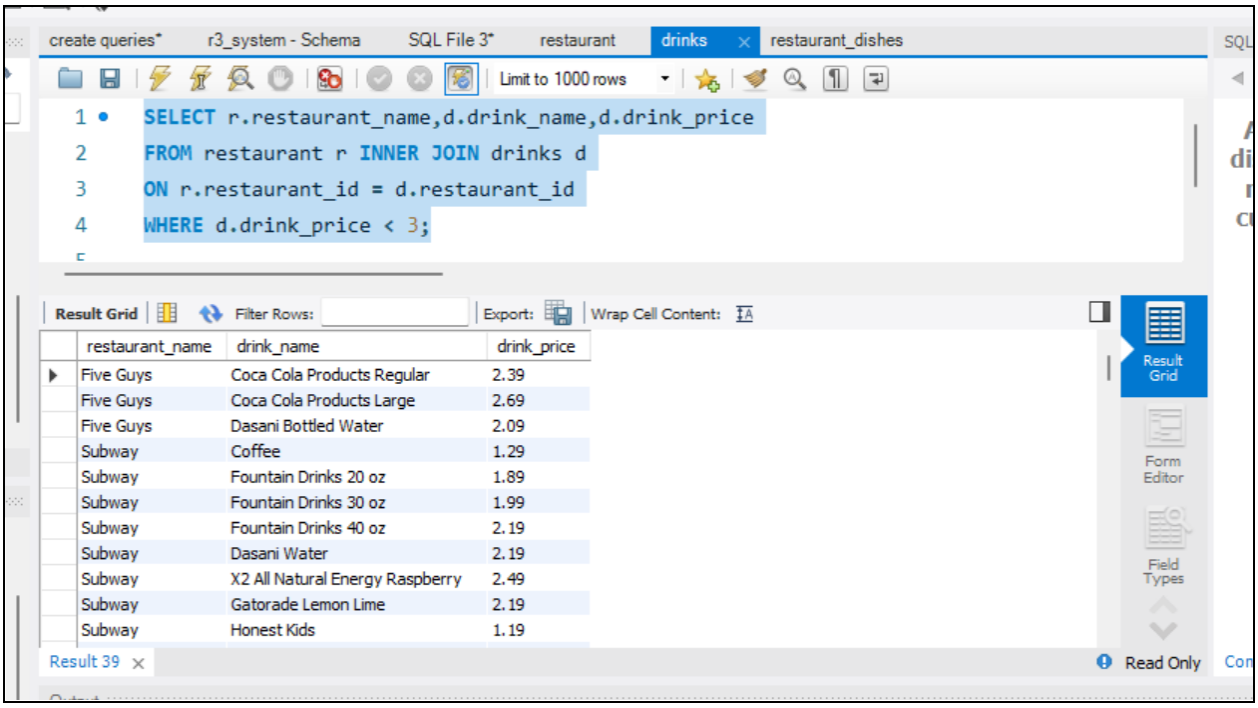


Use Case 14: Restaurants with drinks below 3$?

Description: The user searches for a restaurant that offers drinks below 3 dollars
Actor: User
Precondition: The user needs to log in to his account
Steps:
Actor action: The user requests the list of restaurants where drinks are available below 3$
System Responses: Restaurant details would be displayed
Post Condition: system displays restaurant reviews
Error: No restaurants found within the user's budget

SQL Query:-

```
SELECT r.restaurant_name,d.drink_name,d.drink_price
FROM restaurant r INNER JOIN drinks d
ON r.restaurant_id = d.restaurant_id
WHERE d.drink_price < 3;
```

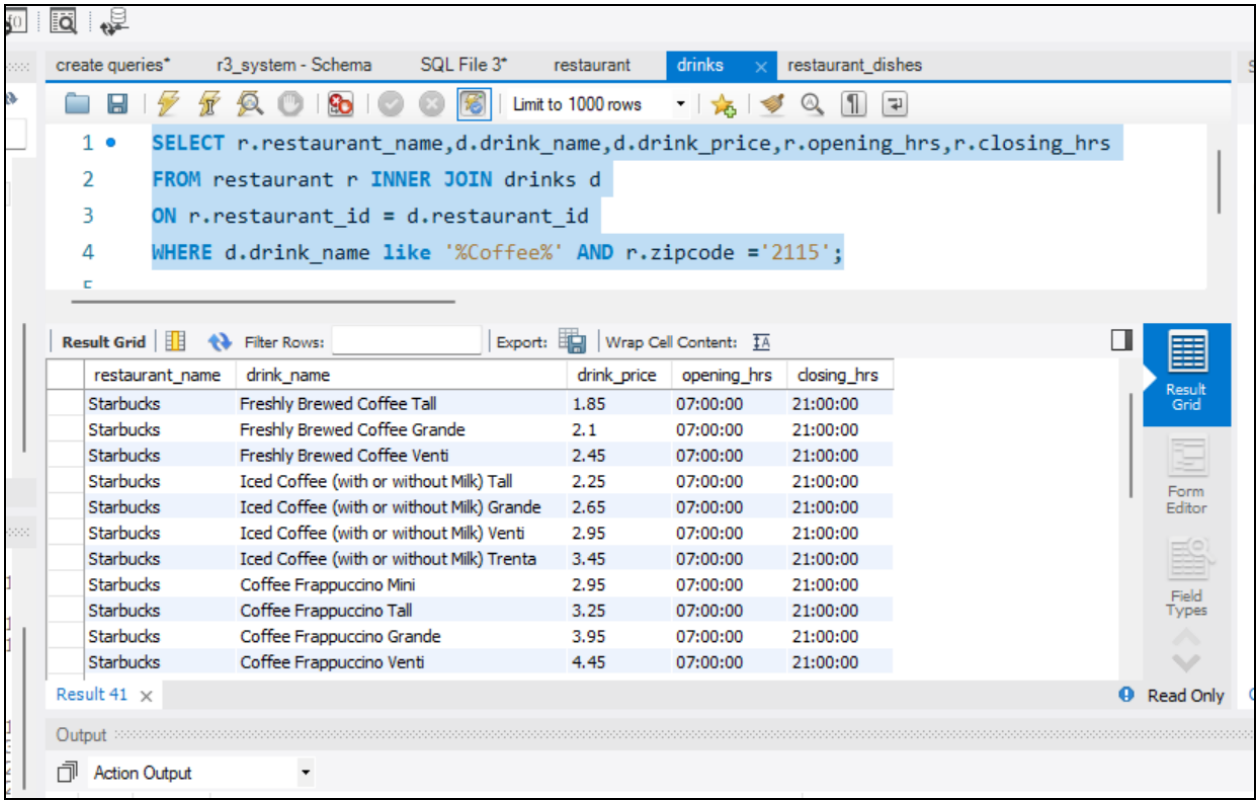


Use Case 15: Restaurants that serve coffee in zip code 02115?

Description: The User searches for a restaurant that serves Coffee in a restaurant located at zipcode 02115
Actor: User
Precondition: The user needs to log in to his account
Steps:
Actor action: The user requests the list of restaurants where coffee is sold in zipcode 02115
System Responses: Restaurant details would be displayed by the system
Post Condition: System displays restaurant reviews
Error: No restaurants found within the given zip code

SQL Query:-

```
SELECT r.restaurant_name,d.drink_name,d.drink_price,r.opening_hrs,r.closing_hrs
FROM restaurant r INNER JOIN drinks d
ON r.restaurant_id = d.restaurant_id
WHERE d.drink_name like '%Coffee%' AND r.zipcode ='2115';
```

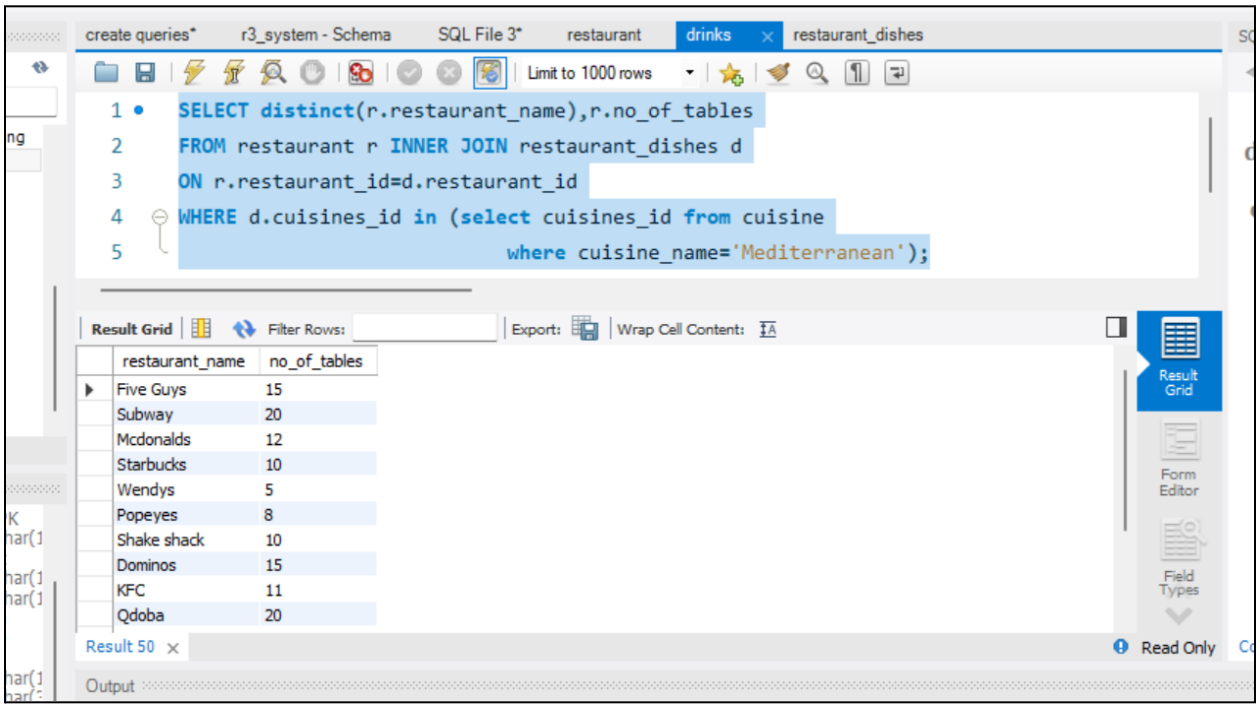


Use Case 16: Check if the table is available for Reservation for a Mediterranean restaurant.

Description:- The user checks if a table is available for reservation in a Mediterranean restaurant.
Actors: User
Precondition: The user must be logged in to his/her account
Steps:
Actor Action: The user searches for the restaurant and tries to make reservations
System Response: Displays all the tables available for making a reservation
Post Condition: The user will decide if he/her wants to book that particular table or not.

SQL Query:

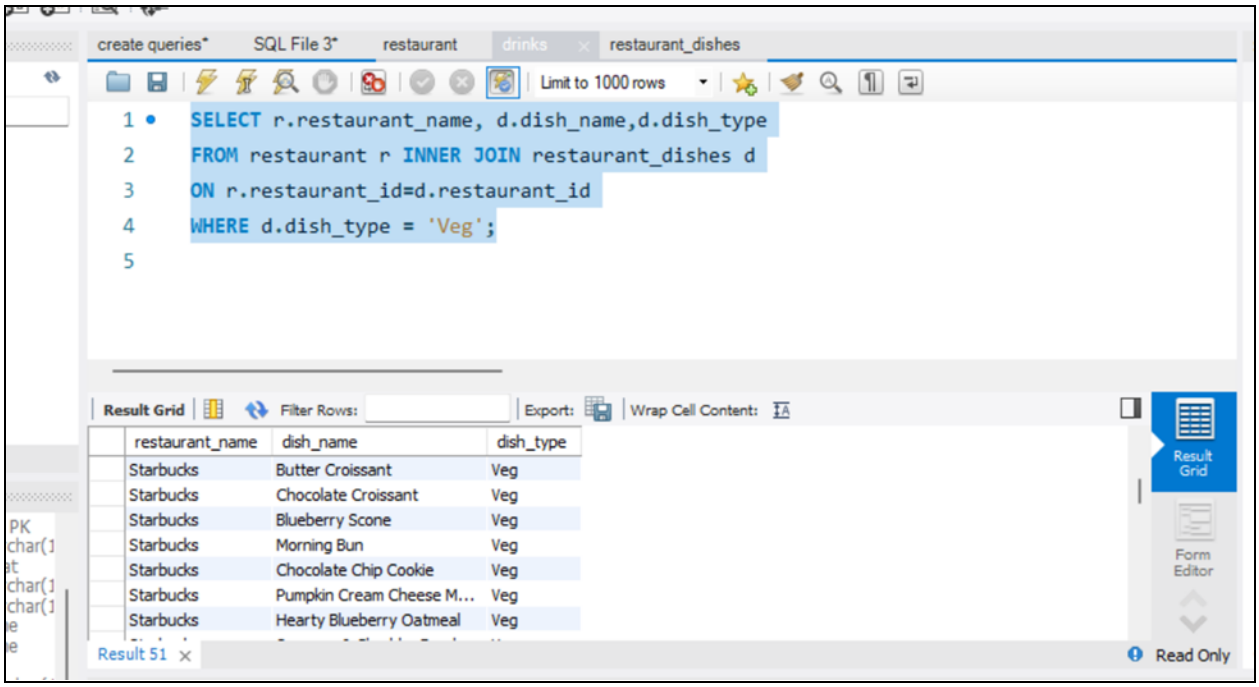
```
SELECT distinct(r.restaurant_name),r.no_of_tables
FROM restaurant r INNER JOIN restaurant_dishes d
ON r.restaurant_id=d.restaurant_id
WHERE d.cuisines_id in (select cuisines_id from cuisine
                        Where cuisine_name='Mediterranean');
```



Use Case 17: Search for restaurants that offer vegetarian food
Description: The user searches for restaurants that offer vegetarian food
Actors: User
Precondition: The user must be logged in from his account.
Steps:
Actor action: The user searches for details of restaurants that offer entirely vegetarian dishes.
System Responses: Displays details of the restaurants offering vegetarian food.
Post Condition: Users will be able to select restaurants by viewing other features and previous reviews of those restaurants.
Alternate Path: If no such restaurant is available which offers vegetarian food, the system will show an error.
Error: No restaurants found that offer vegetarian food.

SQL Query:

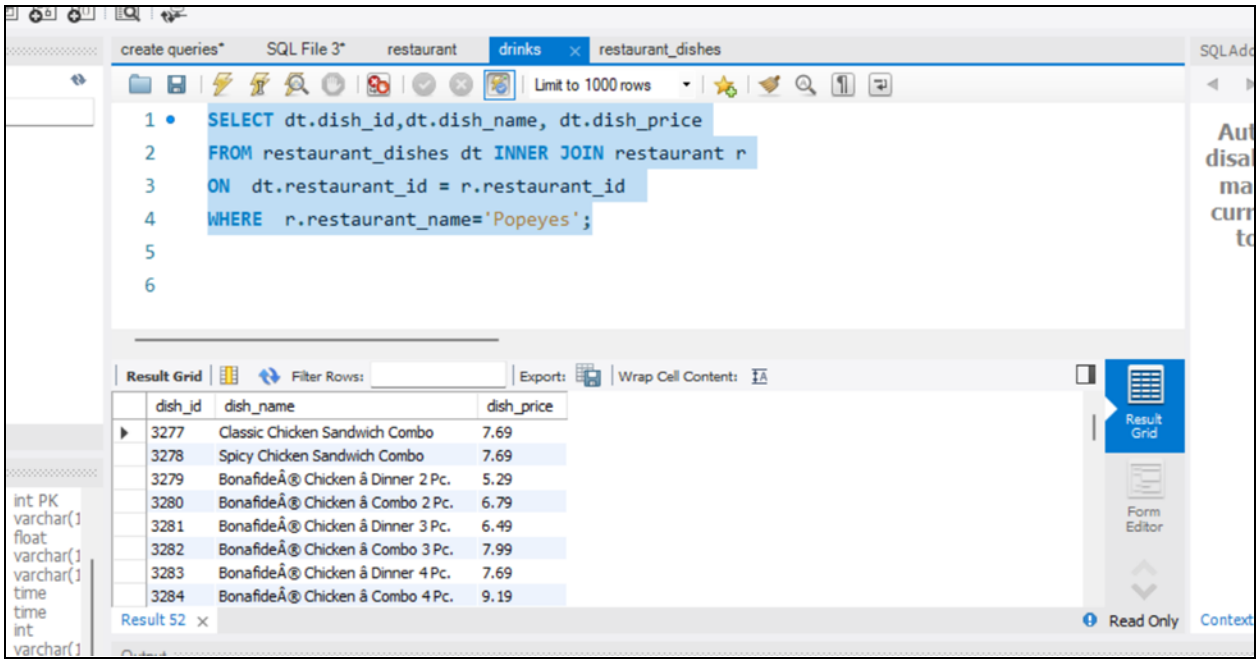
```
SELECT r.restaurant_name, d.dish_name,d.dish_type
FROM restaurant r INNER JOIN restaurant_dishes d
ON r.restaurant_id=d.restaurant_id
WHERE d.dish_type = 'Veg';
```



Use Case 18: View the food menu for a specific restaurant.
Description: The user searches for the food menu for a specific restaurant
Actor: User
Precondition: The user must be logged into his account
Steps:
Actor action: The user requests the details of the food menu for a specific restaurant
System Responses: Displays the food menu of that restaurant.
Post Condition: The user can decide which dish he wants to order when he checks the food menu
Alternate Path: The user enters the wrong restaurant name.
Error: No such restaurant is available

SQL Query:

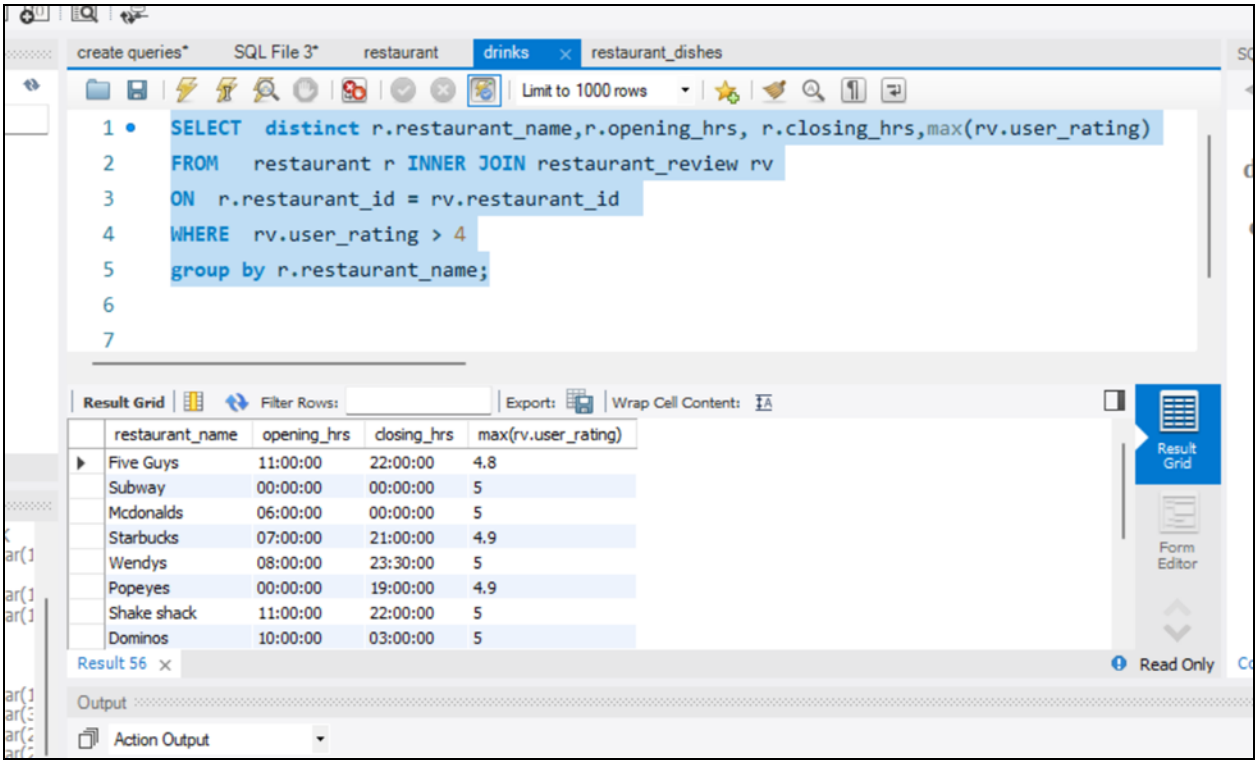
```
SELECT dt.dish_id,dt.dish_name, dt.dish_price
FROM restaurant_dishes dt INNER JOIN restaurant r
ON  dt.restaurant_id = r.restaurant_id
WHERE  r.restaurant_name='Popeyes';
```



Use Case 19: View Opening and Closing Hours for a specific restaurant that has a good user rating (considering good as a rating above 4)
Description: The user searches for an opening and closing hours for a specific restaurant that has a good user rating
Actor: User
Precondition: The user must be logged into his account
Steps:
Actor action: The user requests restaurant hours and review rating
System Responses: Displays the working hour
Post Condition: The user can decide if you want the restaurant to visit or not
Alternate Path: The user enters the wrong restaurant name
Error: User not logged in.

SQL Query:

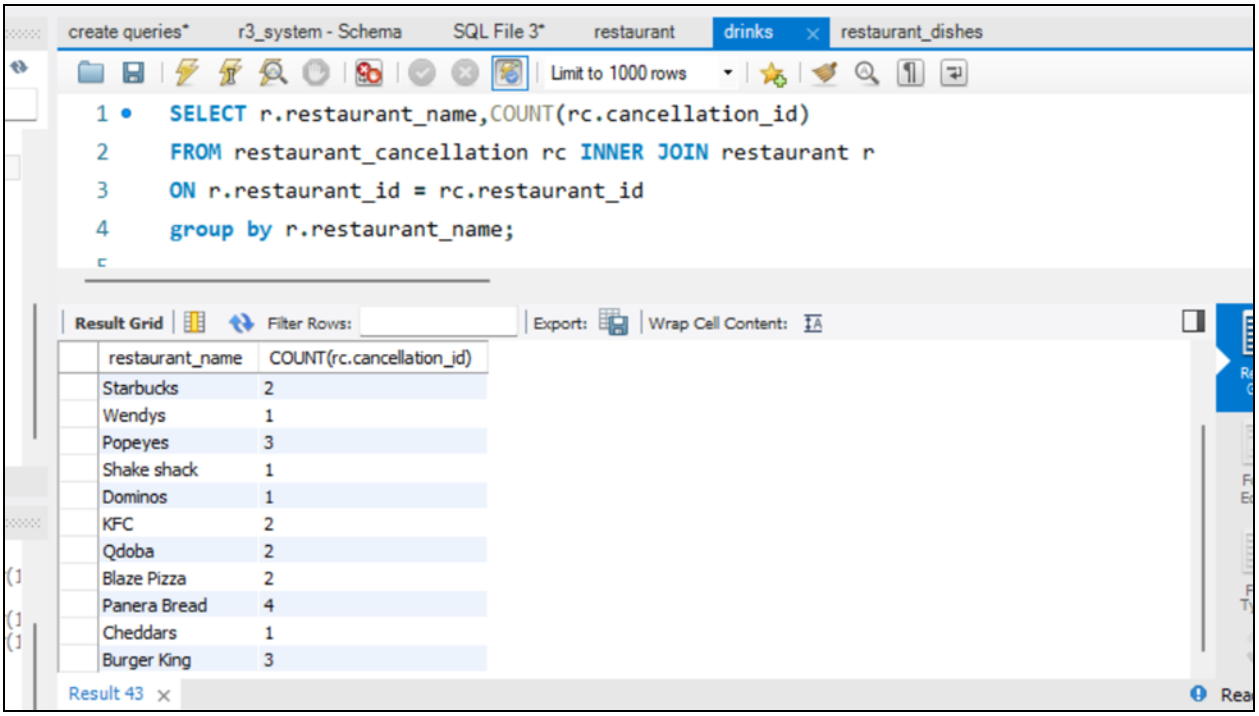
```
SELECT distinct r.restaurant_name, r.opening_hrs, r.closing_hrs,
max(rv.user_rating)
FROM   restaurant r INNER JOIN restaurant_review rv
ON  r.restaurant_id = rv.restaurant_id
WHERE  rv.user_rating > 4
group by r.restaurant_name;
```



Use Case 20: How many cancellations have a restaurant received to date?
Description: The user searches for cancellations for a restaurant received to date
Actor: User
Precondition: The user needs to log in to his account
Steps:
Actor action: The user requests the cancellations for a restaurant received to date
System Responses: Displays cancellations a restaurant received to date
Post Condition: The user will be able to see all cancellations to date

SQL Query:-

```
SELECT r.restaurant_name,COUNT(rc.cancellation_id)
FROM restaurant_cancellation rc INNER JOIN restaurant r
ON r.restaurant_id = rc.restaurant_id
group by r.restaurant_name;
```



PROGRAM OUTPUTS:

Review Restaurant

```
Select Among the Following Tasks
Task 1 : Review a Restaurant
Task 2 : Reserve a table at Restaurant
Task 3 : Cancel Reservation

Enter your choice : 1
Review a Restaurant
Enter restaurant name : Qdoba
Enter user first name : Lenna
Enter review comment : Liked it!
Did you like the restaurant? (yes:1,no:0) : 1
Enter user rating : 4

 --- OUTPUT---

Review Details:
(7011, 'Liked it!', 1, 4.0, datetime.date(2022, 12, 14), 1010, 2004)


End Task
```

Reserve Restaurant

```
Would you like to continue ? (yes:y/Y, no: n/N) : Y

Select Among the Following Tasks
Task 1 : Review a Restaurant
Task 2 : Reserve a table at Restaurant
Task 3 : Cancel Reservation

Enter your choice : 2
Reserve a table at a restaurant
Enter restaurant name : Qdoba
Enter user first name : Lenna
Enter no of people : 5
Enter Date (YYYY-MM-dd) : 2022-12-20
Enter Time (HH:MM:SS) : 20:00:00

 --- OUTPUT---

Restaurant Details Before reservation
(1010, 'Qdoba', 4.1, '(617) 450-0910', datetime.timedelta(seconds=37800), datetime.timedelta(seconds=82800), 33, '393 Huntin
gton Ave Ste 101, Boston, ', 'Boston', 'MA', 2115, 8)


Reservation Deatils
(9034, 3, datetime.timedelta(seconds=73425), datetime.date(2022, 1, 11), 1006, 2041)


Restaurant Details After reservation
(1010, 'Qdoba', 4.1, '(617) 450-0910', datetime.timedelta(seconds=37800), datetime.timedelta(seconds=82800), 33, '393 Huntin
gton Ave Ste 101, Boston, ', 'Boston', 'MA', 2115, 7)


End Task
```

Cancel Reservation

```
Would you like to continue ? (yes:y/Y, no: n/N) : Y

Select Among the Following Tasks
Task 1 : Review a Restaurant
Task 2 : Reserve a table at Restaurant
Task 3 : Cancel Reservation

Enter your choice : 3

Cancel a Reservation
Enter cancellation reason : Plan cancelled
Enter restaurant name : Qdoba
Enter reservationId : 9034

 --- OUTPUT---

Restaurant Details Before cancellation
(1010, 'Qdoba', 4.1, '(617) 450-0910', datetime.timedelta(seconds=37800), datetime.timedelta(seconds=82800), 33, '393 Huntin
gton Ave Ste 101, Boston, ', 'Boston', 'MA', 2115, 7)


Cancellation Details
(8037, 'Plan cancelled', 1010, 2004)


Restaurant Details After cancellation
(1010, 'Qdoba', 4.1, '(617) 450-0910', datetime.timedelta(seconds=37800), datetime.timedelta(seconds=82800), 33, '393 Huntin
gton Ave Ste 101, Boston, ', 'Boston', 'MA', 2115, 8)


End Task
```

Search by budget:

```
search_by_budget
Enter Budget : 30

 --- OUTPUT---
('Starbucks', 3.7, datetime.timedelta(seconds=25200), datetime.timedelta(seconds=75600), 26, '360 Huntington Ave, Boston, ',
2115, 'Boston', 'MA')

('Wendys', 3.9, datetime.timedelta(seconds=28800), datetime.timedelta(seconds=84600), 24, '157-A Massachusetts Ave, Boston,
', 2127, 'Boston', 'MA')

('Popeyes', 3.9, datetime.timedelta(0), datetime.timedelta(seconds=68400), 24, 'Northeastern University, 360 Huntington Ave,
Boston, ', 2115, 'Boston', 'MA')

('KFC', 3.6, datetime.timedelta(seconds=36000), datetime.timedelta(seconds=82800), 22, '695 Columbia Rd, Dorchester, ', 212
5, 'Boston', 'MA')

('Dunkin Donuts', 4.2, datetime.timedelta(seconds=28800), datetime.timedelta(seconds=50400), 17, 'Northeastern - Shillman Ha
ll, 115 Forsyth St, Boston, ', 2115, 'Boston', 'MA')

('Blaze Pizza', 4.5, datetime.timedelta(seconds=39600), datetime.timedelta(seconds=79200), 11, '1282 Boylston St, Boston, ',
2215, 'Boston', 'MA')
```

Search by Category:

```
search_by_category
Enter Category : pizza

 --- OUTPUT---
('Dominos', 2.6, datetime.timedelta(seconds=36000), datetime.timedelta(seconds=10800), 42, '1400 Tremont St Roxbury Crossing
Station - Space D, Crossing, ', 2120, 'Boston', 'MA')

('Blaze Pizza', 4.5, datetime.timedelta(seconds=39600), datetime.timedelta(seconds=79200), 11, '1282 Boylston St, Boston, ',
2215, 'Boston', 'MA')
```

Search by dish price::

```
search_by_dishprice
Enter Dish price : 20

 --- OUTPUT---
('Five Guys', 'Hamburger ', 6.99, 4.3, datetime.timedelta(seconds=39600), datetime.timedelta(seconds=79200), 41, '263 Hun
tington Ave, Boston, ', 2115, 'Boston', 'MA')

('Five Guys', 'Cheeseburger ', 7.69, 4.3, datetime.timedelta(seconds=39600), datetime.timedelta(seconds=79200), 41, '263
Huntington Ave, Boston, ', 2115, 'Boston', 'MA')

('Five Guys', 'Bacon Burger ', 7.99, 4.3, datetime.timedelta(seconds=39600), datetime.timedelta(seconds=79200), 41, '263
Huntington Ave, Boston, ', 2115, 'Boston', 'MA')

('Five Guys', 'Bacon Cheeseburger ', 8.69, 4.3, datetime.timedelta(seconds=39600), datetime.timedelta(seconds=79200), 41,
'263 Huntington Ave, Boston, ', 2115, 'Boston', 'MA')

('Five Guys', 'Little Hamburger ', 4.99, 4.3, datetime.timedelta(seconds=39600), datetime.timedelta(seconds=79200), 41,
'263 Huntington Ave, Boston, ', 2115, 'Boston', 'MA')
```

Search by drink price:

```
search_by_drinkprice
Enter Drink price : 10

 --- OUTPUT---
('Five Guys', 'Coca Cola Products Regular ', 'Drinks', 2.39, 41, '263 Huntington Ave, Boston, ', 2115)

('Five Guys', 'Coca Cola Products Large ', 'Drinks', 2.69, 41, '263 Huntington Ave, Boston, ', 2115)

('Five Guys', 'Dasani Bottled Water ', 'Drinks', 2.09, 41, '263 Huntington Ave, Boston, ', 2115)

('Subway', 'Coffee ', 'Drinks', 1.29, 45, 'Ryder hall, Leon St, Boston, ', 2115)

('Subway', 'Fountain Drinks 20 oz ', 'Drinks', 1.89, 45, 'Ryder hall, Leon St, Boston, ', 2115)

('Subway', 'Fountain Drinks 30 oz ', 'Drinks', 1.99, 45, 'Ryder hall, Leon St, Boston, ', 2115)

('Subway', 'Fountain Drinks 40 oz ', 'Drinks', 2.19, 45, 'Ryder hall, Leon St, Boston, ', 2115)

('Subway', 'Dasani Water ', 'Drinks', 2.19, 45, 'Ryder hall, Leon St, Boston, ', 2115)
```

Search by drink type:

```
Enter Drink type : Coffee

 --- OUTPUT---
('Starbucks', 'Caffe Latte Tall ', 'Espresso, Coffee & Tea', 2.95, 26, '360 Huntington Ave, Boston, ', 2115)

('Starbucks', 'Caffe Latte Grande ', 'Espresso, Coffee & Tea', 3.65, 26, '360 Huntington Ave, Boston, ', 2115)

('Starbucks', 'Caffe Latte Venti ', 'Espresso, Coffee & Tea', 4.15, 26, '360 Huntington Ave, Boston, ', 2115)

('Starbucks', 'Caffe Mocha Tall ', 'Espresso, Coffee & Tea', 3.45, 26, '360 Huntington Ave, Boston, ', 2115)

('Starbucks', 'Caffe Mocha Grande ', 'Espresso, Coffee & Tea', 4.15, 26, '360 Huntington Ave, Boston, ', 2115)

('Starbucks', 'Caffe Mocha Venti ', 'Espresso, Coffee & Tea', 4.65, 26, '360 Huntington Ave, Boston, ', 2115)

('Starbucks', 'White Chocolate Mocha Tall ', 'Espresso, Coffee & Tea', 3.75, 26, '360 Huntington Ave, Boston, ', 2115)

('Starbucks', 'White Chocolate Mocha Grande ', 'Espresso, Coffee & Tea', 4.45, 26, '360 Huntington Ave, Boston, ', 2115)
```

Search by rating:

```
search_by_rating
Enter Rating : 4

 --- OUTPUT---
('Five Guys', 4.3, datetime.timedelta(seconds=39600), datetime.timedelta(seconds=79200), 41, '263 Huntington Ave, Boston, ',
2115, 'Boston', 'MA')

('Shake shack', 4.3, datetime.timedelta(seconds=39600), datetime.timedelta(seconds=79200), 43, '234-236 Newbury St, Boston,
', 2116, 'Boston', 'MA')

('Qdoba', 4.1, datetime.timedelta(seconds=37800), datetime.timedelta(seconds=82800), 33, '393 Huntington Ave Ste 101, Bosto
n, ', 2115, 'Boston', 'MA')

('Dunkin Donuts', 4.2, datetime.timedelta(seconds=28800), datetime.timedelta(seconds=50400), 17, 'Northeastern - Shillman Ha
ll, 115 Forsyth St, Boston, ', 2115, 'Boston', 'MA')

('Blaze Pizza', 4.5, datetime.timedelta(seconds=39600), datetime.timedelta(seconds=79200), 11, '1282 Boylston St, Boston, ',
2215, 'Boston', 'MA')

('Cheddars', 4.6, datetime.timedelta(seconds=36000), datetime.timedelta(seconds=72000), 41, '1638 Washington St, Boston, ',
2118, 'Boston', 'MA')
```

Search by restaurant name:

```
search_by_restaurantname
Enter Restaurant Name : Shake shack

 --- OUTPUT---
('Shake shack', 4.3, datetime.timedelta(seconds=39600), datetime.timedelta(seconds=79200), 43, '234-236 Newbury St, Boston,
', 2116, 'Boston', 'MA')
```

Search by dish name:

```
search_by_dishname
Enter Dish Name : Hamburger

 --- OUTPUT---
('Five Guys', 'Hamburger ', 6.99, 4.3, 41, '263 Huntington Ave, Boston, ', 2115, 'Boston', 'MA')

('Five Guys', 'Little Hamburger ', 4.99, 4.3, 41, '263 Huntington Ave, Boston, ', 2115, 'Boston', 'MA')

('Mcdonalds', 'Hamburger ', 2.49, 3.6, 38, '540 Commonwealth Ave, Boston, ', 2215, 'Boston', 'MA')

('Wendys', 'Hamburger ', 3.39, 3.9, 24, '157-A Massachusetts Ave, Boston, ', 2127, 'Boston', 'MA')

('Shake shack', 'Hamburger Single ', 4.29, 4.3, 43, '234-236 Newbury St, Boston, ', 2116, 'Boston', 'MA')

('Shake shack', 'Hamburger Double ', 6.59, 4.3, 43, '234-236 Newbury St, Boston, ', 2116, 'Boston', 'MA')

('Burger King', 'Hamburger ', 1.0, 3.8, 37, '128 Tremont St, Boston, ', 2108, 'Boston', 'MA')

('Burger King', 'Hamburger â\x80\x93 Meal ', 3.99, 3.8, 37, '128 Tremont St, Boston, ', 2108, 'Boston', 'MA')
```

Search by restaurant type:

```
search_by_restype
Enter Restaurant type : fast food

 --- OUTPUT---
('Five Guys', 4.3, datetime.timedelta(seconds=39600), datetime.timedelta(seconds=79200), 41, '263 Huntington Ave, Boston, ',
2115, 'Boston', 'MA', 'Fast food restaurant')

('Mcdonalds', 3.6, datetime.timedelta(seconds=21600), datetime.timedelta(0), 38, '540 Commonwealth Ave, Boston, ', 2215, 'Bo
ston', 'MA', 'Fast food restaurant')

('Wendys', 3.9, datetime.timedelta(seconds=28800), datetime.timedelta(seconds=84600), 24, '157-A Massachusetts Ave, Boston,
', 2127, 'Boston', 'MA', 'Fast food restaurant')
```

Search by user rating

```
search_by_userrating
Enter user rating : 4.5

 --- OUTPUT---
('Five Guys', 4.3, 5.0, datetime.timedelta(seconds=39600), datetime.timedelta(seconds=79200), 41, '263 Huntington Ave, Bosto
n, ', 2115, 'Boston', 'MA')

('Subway', 3.7, 4.9, datetime.timedelta(0), datetime.timedelta(0), 45, 'Ryder hall, Leon St, Boston, ', 2115, 'Boston', 'M
A')

('Mcdonalds', 3.6, 4.9, datetime.timedelta(seconds=21600), datetime.timedelta(0), 38, '540 Commonwealth Ave, Boston, ', 221
5, 'Boston', 'MA')

('Starbucks', 3.7, 5.0, datetime.timedelta(seconds=25200), datetime.timedelta(seconds=75600), 26, '360 Huntington Ave, Bosto
n, ', 2115, 'Boston', 'MA')

('Wendys', 3.9, 5.0, datetime.timedelta(seconds=28800), datetime.timedelta(seconds=84600), 24, '157-A Massachusetts Ave, Bos
ton, ', 2127, 'Boston', 'MA')

('Popeyes', 3.9, 4.9, datetime.timedelta(0), datetime.timedelta(seconds=68400), 24, 'Northeastern University, 360 Huntington
Ave, Boston, ', 2115, 'Boston', 'MA')

('Shake shack', 4.3, 5.0, datetime.timedelta(seconds=39600), datetime.timedelta(seconds=79200), 43, '234-236 Newbury St, Bos
ton, ', 2116, 'Boston', 'MA')

('Dominos', 2.6, 4.9, datetime.timedelta(seconds=36000), datetime.timedelta(seconds=10800), 42, '1400 Tremont St Roxbury Cro
ssing Station - Space D, Crossing, ', 2120, 'Boston', 'MA')

('KFC', 3.6, 5.0, datetime.timedelta(seconds=36000), datetime.timedelta(seconds=82800), 22, '695 Columbia Rd, Dorchester, ',
2125, 'Boston', 'MA')

('Qdoba', 4.1, 5.0, datetime.timedelta(seconds=37800), datetime.timedelta(seconds=82800), 33, '393 Huntington Ave Ste 101, B
oston, ', 2115, 'Boston', 'MA')
```

Search by zip code:

```
search_by_zip
Enter Zip code : 2115

 --- OUTPUT---
('Five Guys', 4.3, datetime.timedelta(seconds=39600), datetime.timedelta(seconds=79200), 41, '263 Huntington Ave, Boston, ',
2115, 'Boston', 'MA')

('Subway', 3.7, datetime.timedelta(0), datetime.timedelta(0), 45, 'Ryder hall, Leon St, Boston, ', 2115, 'Boston', 'MA')

('Starbucks', 3.7, datetime.timedelta(seconds=25200), datetime.timedelta(seconds=75600), 26, '360 Huntington Ave, Boston, ',
2115, 'Boston', 'MA')

('Popeyes', 3.9, datetime.timedelta(0), datetime.timedelta(seconds=68400), 24, 'Northeastern University, 360 Huntington Ave,
Boston, ', 2115, 'Boston', 'MA')

('Qdoba', 4.1, datetime.timedelta(seconds=37800), datetime.timedelta(seconds=82800), 33, '393 Huntington Ave Ste 101, Bosto
n, ', 2115, 'Boston', 'MA')

('Dunkin Donuts', 4.2, datetime.timedelta(seconds=28800), datetime.timedelta(seconds=50400), 17, 'Northeastern - Shillman Ha
ll, 115 Forsyth St, Boston, ', 2115, 'Boston', 'MA')

('Panera Bread', 3.9, datetime.timedelta(seconds=21600), datetime.timedelta(seconds=79200), 49, '289 Huntington Ave, Boston,
', 2115, 'Boston', 'MA')
```

CONCLUSION:

The proposed Restaurant Reservation and Recommendation system fulfills all the requirements of the user like searching for a restaurant, restaurant reservation and cancellation of any reservation made for the restaurants in Boston city. Thus the system allows users to easily browse through the system and check for any specific requirements he has and get restaurant details accordingly.