DMDD ASSIGNMENT 3
Topic Name: Restaurant Recommendation and Reservation System

Group Name: R3

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

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

1. 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 their 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.

2. Files and their description (Download and reformat the data to fit your database schema) :

- **Assignment 3 Web scraping:** Contains python script to scrape restaurant details from google and restaurant menu details from different websites and clean it. This file exports all the table data frames to csvs' which are imported into the "Assignment 3 inserting scraped data into database" file.
- **Assignment 3 Inserting Scraped Data into Database:** This file contains python scripts to insert the scraped data into MySQL workbench. The csv files exported from web scraping scripts are imported into this file and then inserted into the database after minor changes.

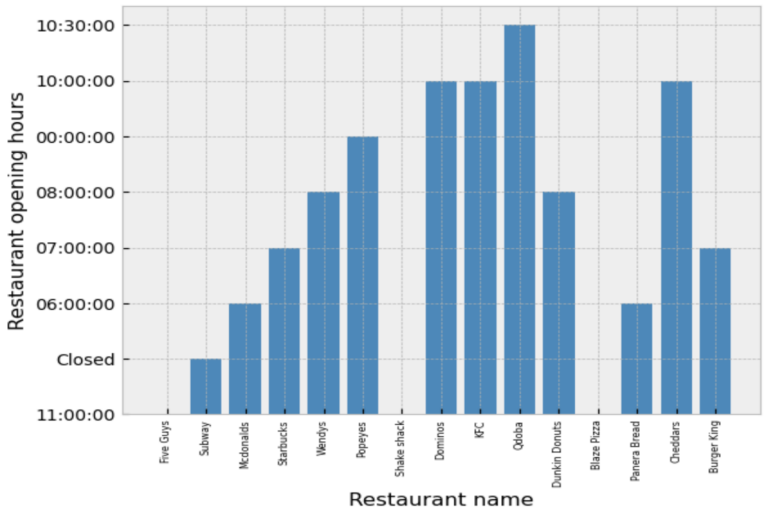
- **Scrapped data frames from each table:** This folder contains all the data frames exported after scrapping the information and cleaning it.
- **Assignment 3 Database Audit:** This file contains auditing validity, auditing completeness, and auditing consistency of the database.

3. Audit validity/ accuracy:

It's the process of checking the integrity, accuracy, and quality of data before it is used for a business purpose. The idea is to compare a data set against certain defined rules to ensure the correctness of the data both in structure and content. Auditing validity is determining what the constraints are on the individual fields and checking to ensure the field values adhere to these constraints. The constraints we have used are primary key, foreign key, mandatory meaning not null, unique, etc. The restaurant data that we have scraped is valid and adheres to all the constraints. For example, we have restaurant_id as the primary key in the restaurant table and we have restaurant_id as a foreign key in the review table. Therefore the review table will only have restaurants that are present on the restaurant table. Another example is the cross-field constraints for the opening and closing hours of a restaurant. For the opening and closing hours to be valid, the opening hours should be less than the closing hours. As we are scraping the opening and closing hours from google, they are valid and real-time.

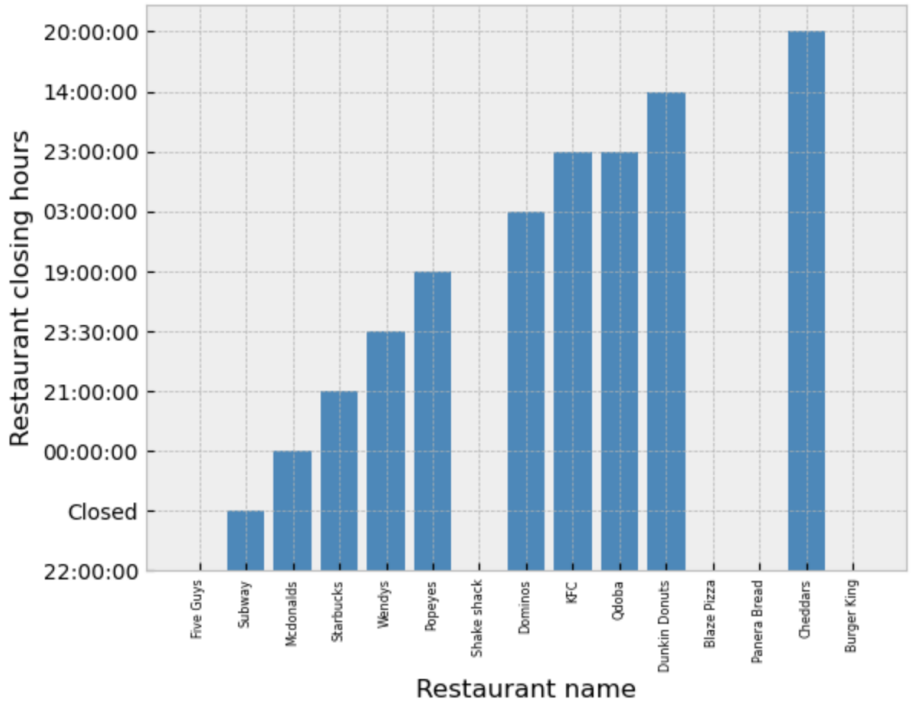
Example 1: Below is a graph between restaurants and their restaurant opening hours. From the below graph, we can say that the restaurant opening hours are valid as they are in the morning.

```
In [870]: df['restaurant_contact_no'].str.len()
          xvalue = df['restaurant_name']
          yvalue = df['opening_hrs']

          plt.xlabel('Restaurant name')
          plt.ylabel('Restaurant opening hours')
          plt.xticks(rotation=90, fontsize = 'xx-small')
          plt.bar(xvalue,yvalue)

Out[870]: <BarContainer object of 15 artists>
```



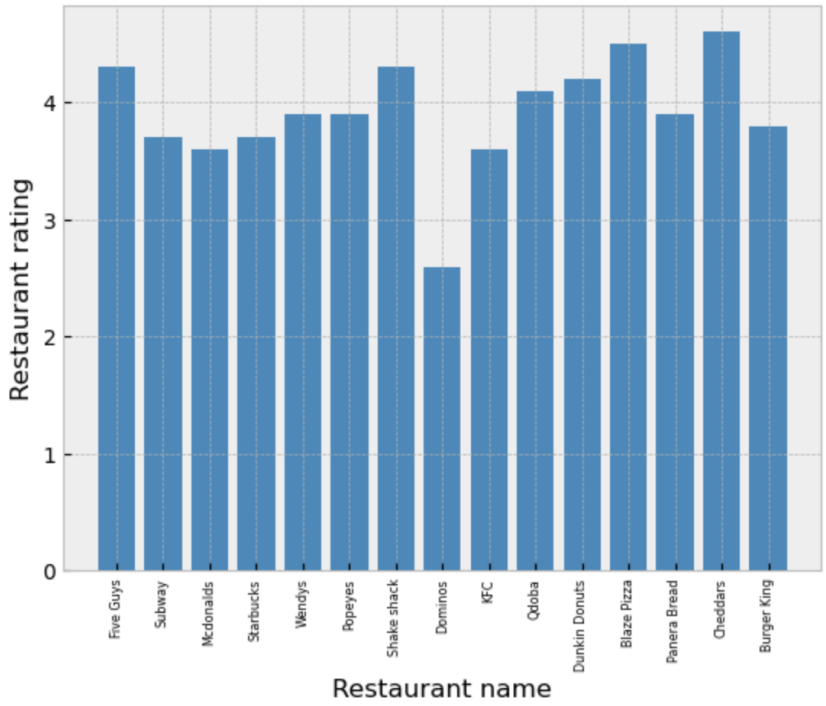
Example 2: Below is a graph between restaurants and their closing hours. From the below graph, we can say that the restaurant closing hours are valid as they are at night.

```
In [871]: xvalue = df['restaurant_name']
          yvalue = df['closing_hrs']

          plt.xlabel('Restaurant name')
          plt.ylabel('Restaurant closing hours')
          plt.xticks(rotation=90, fontsize = 'xx-small')
          plt.bar(xvalue,yvalue)

Out[871]: <BarContainer object of 15 artists>
```



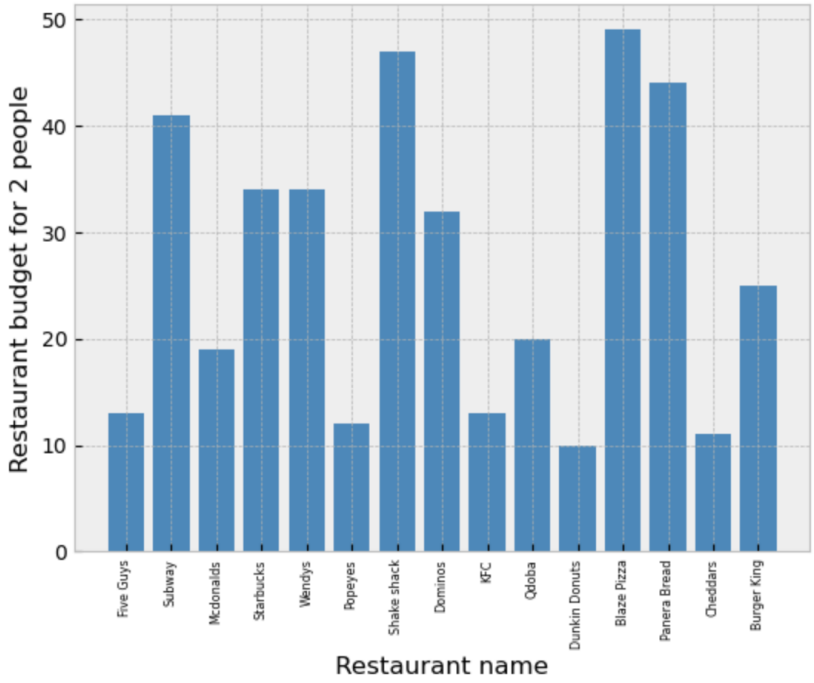
Example 3: Below is a graph between the restaurants and its rating. From the below graph, we can say that the rating is valid as its value lies between 1 and 5.

```
In [895]: xvalue = df['restaurant_name']
          yvalue = df['restaurant_rating']

          plt.xlabel('Restaurant name')
          plt.ylabel('Restaurant rating')
          plt.xticks(rotation=90, fontsize = 'xx-small')
          plt.bar(xvalue,yvalue)

Out[895]: <BarContainer object of 15 artists>
```



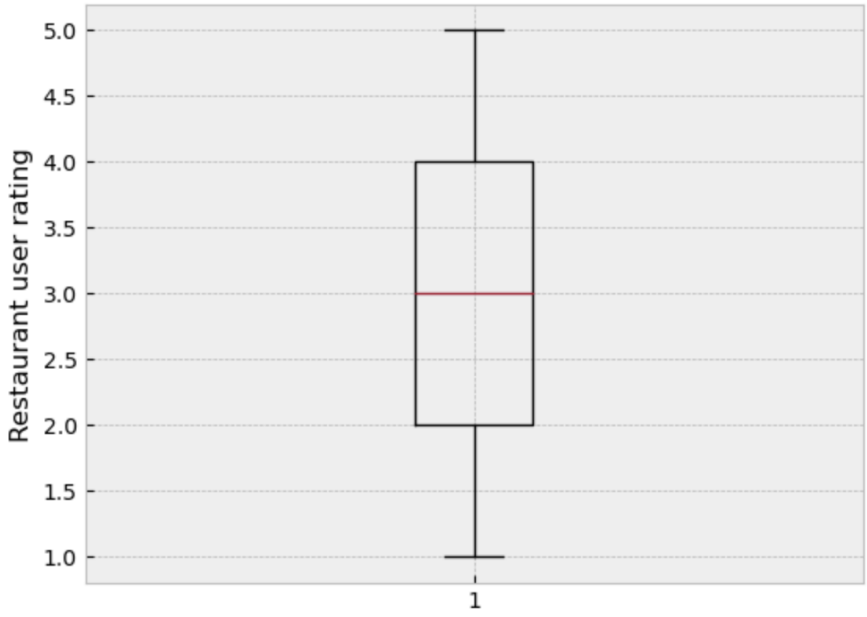
Example 4: Below is a graph between the restaurant and its budget for 2. From the below graph, we can say that there are a few restaurants that have a higher budget for 2 while some restaurants which have a lower budget for 2 people.

```
In [896]: xvalue = df['restaurant_name']
          yvalue = df['budget_for_2']

          plt.xlabel('Restaurant name')
          plt.ylabel('Restaurant budget for 2 people')
          plt.xticks(rotation=90, fontsize = 'xx-small')
          plt.bar(xvalue,yvalue)

Out[896]: <BarContainer object of 15 artists>
```



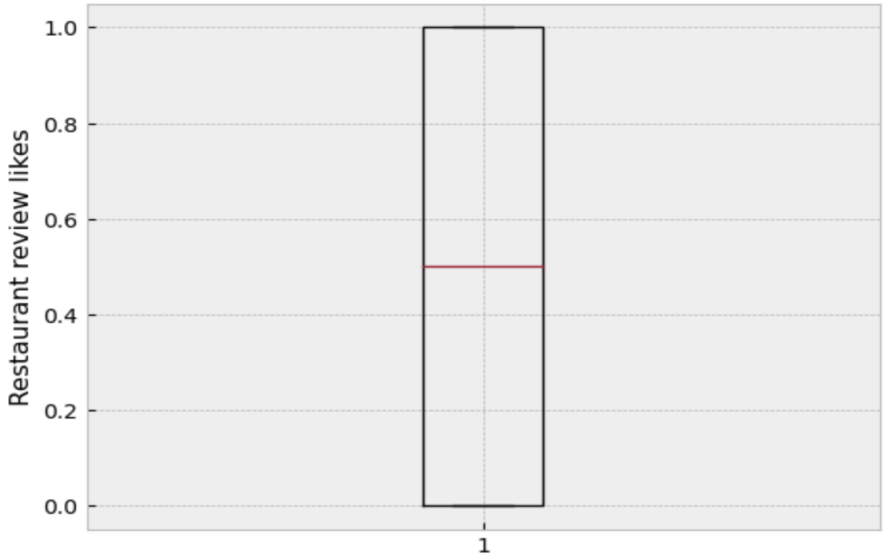
Example 5: Below is a box plot for restaurant user ratings. The user rating exists from 1 to 5. The below graph shows that there are no outliers and 3 is the median user rating.

```
In [898]: plt.boxplot(df.user_rating)
          plt.ylabel('Restaurant user rating')

Out[898]: Text(0, 0.5, 'Restaurant user rating')
```



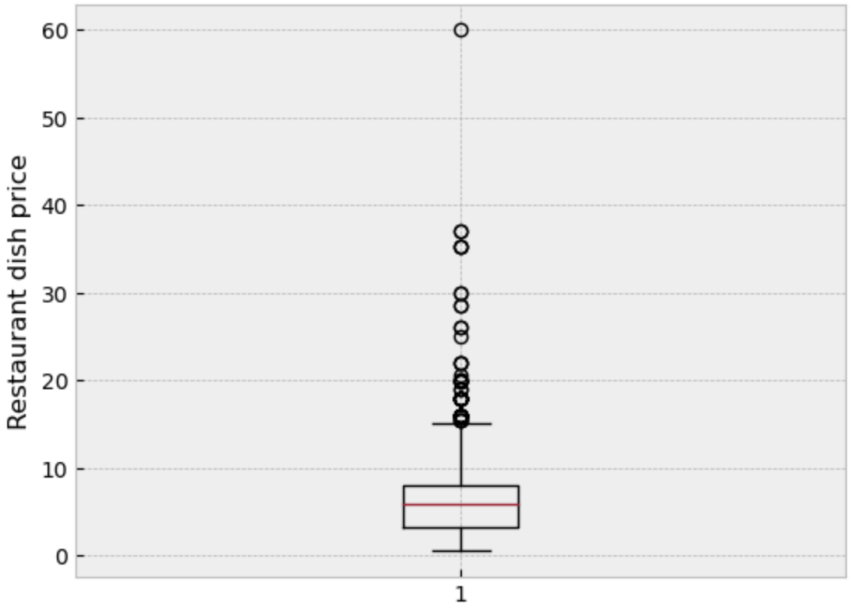
Example 6: Below is a plot for restaurant review likes. The restaurant review can be either liked or not liked. Therefore the value is either 0 or 1 as shown below.

```
In [899]: plt.boxplot(df.review_likes)
          plt.ylabel('Restaurant review likes')

Out[899]: Text(0, 0.5, 'Restaurant review likes')
```



Example 7: Below is a plot for restaurant dish prices. The restaurant dish's prices vary from a range of 5$ up to 60$.

```
In [901]: plt.boxplot(df.dish_price)
          plt.ylabel('Restaurant dish price')

Out[901]: Text(0, 0.5, 'Restaurant dish price')
```



4. Auditing Completeness:

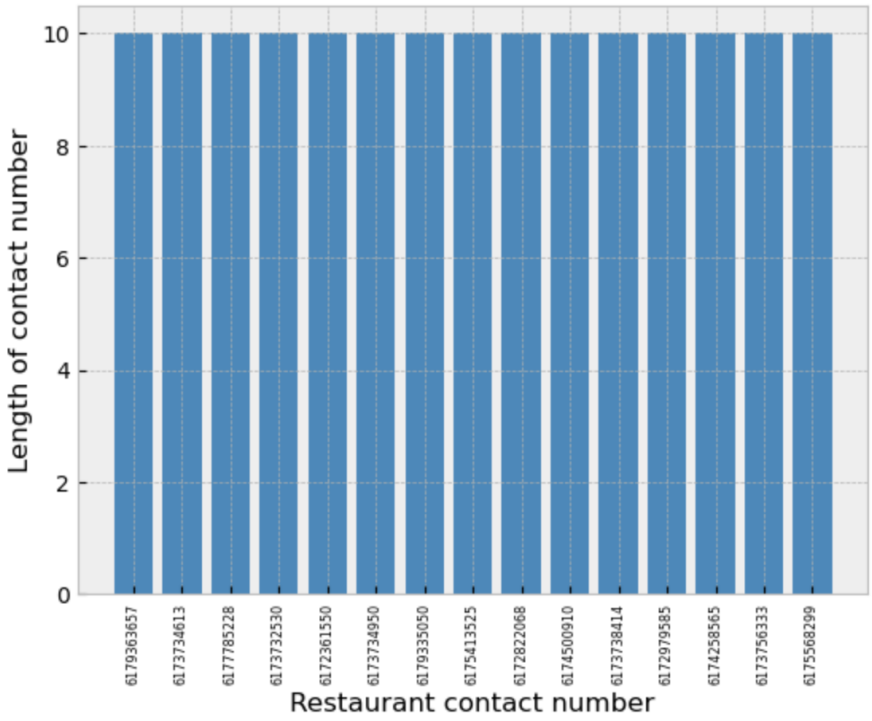
In the data quality framework, data completeness refers to the degree to which all data in a data set is available. A measure of data completeness is the percentage of missing data entries. For instance, a column of 500 with 100 missing fields has a completeness degree of 80%. Depending on your business, 20% of missing entries can translate into losing hundreds of thousands of dollars in prospects and leads. Data completeness is not about ensuring 100% of your fields are complete. It's about determining what items of information are critical and optional. For instance, you would need the phone numbers of the restaurant, but you may not need fax numbers.

Completeness indicates the level of information and knowledge you have about your customer and how accurate this information is. For example, in contact data, children and the elderly may not have email addresses or some contacts may not have landlines or work numbers. Data completeness, therefore, does not imply that all data attributes must be present or populated, rather you will have to classify and choose which data sets are important to keep and which can be ignored.

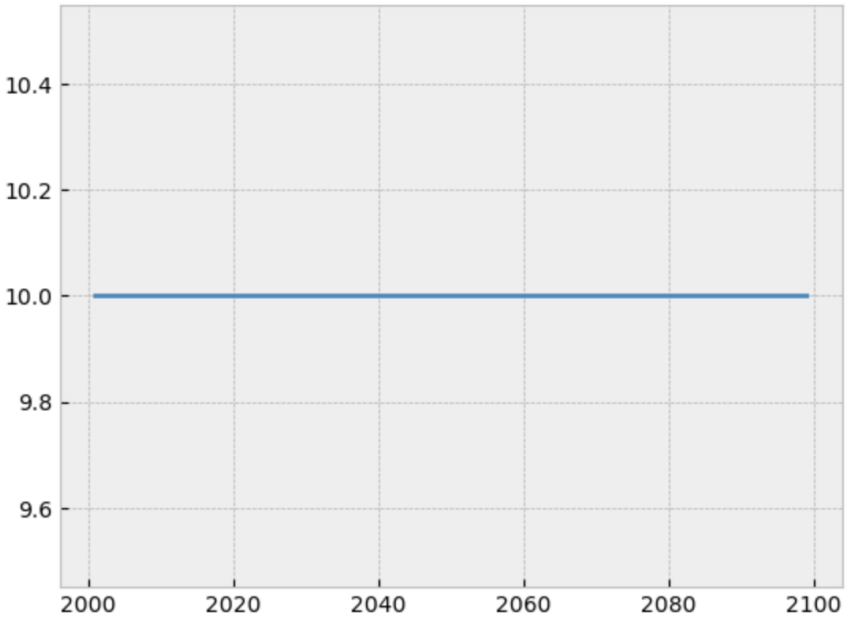
In the data, we have scraped only the essential columns which are required and ignored the unnecessary columns while scraping the data from the google and food menu pages website. The data which cannot be scraped from websites is populated randomly in order to be complete.

5. Auditing Consistency :

Example 1: 2 tables have contact numbers and both have the same format and same length this shows that the data is consistent. Below are the

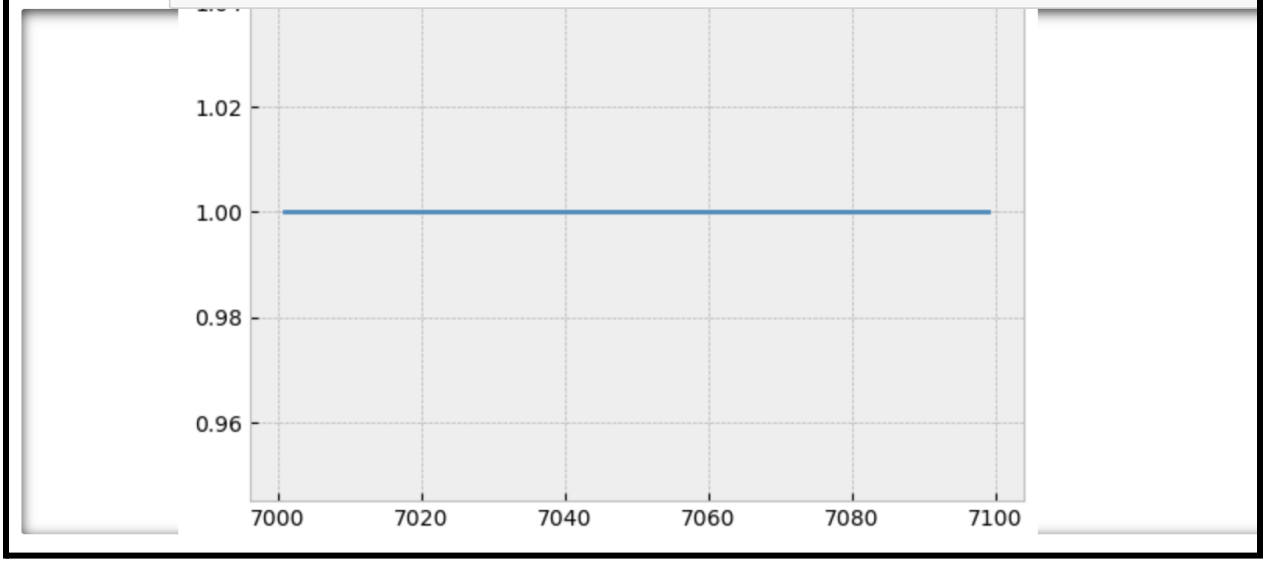


In [903]:
```
length = []
user_id=list(df['user_id'])
for i,row in df.iterrows():
    length.append(len(row['user_contact_no']))

plt.plot(user_id,length)
plt.show()
```



Example 2 : Below is a graph for user name and user id. All the users have a certain email id format and the format remains consistent. Below is the graph to show consistency in the user email id.

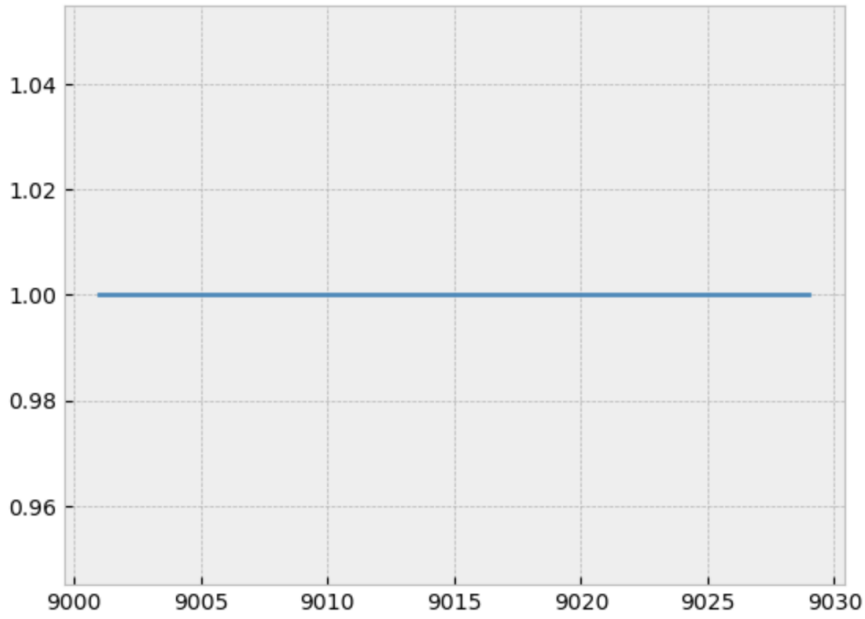
```
In [905]: import re
          correct = []
          user_id = list(df['account_id'])
          for i,row in df.iterrows():
              if '@' in row['user_name']:
                  correct.append(1)
                  print(row['user_name'])


          plt.plot(df['account_id'],correct)
          plt.show()
```



Example 3 : Below is a graph for user reservation time format and user reservation id. All the users have a certain time format and the format remains consistent. Below is the graph to show consistency in the user reservation time.

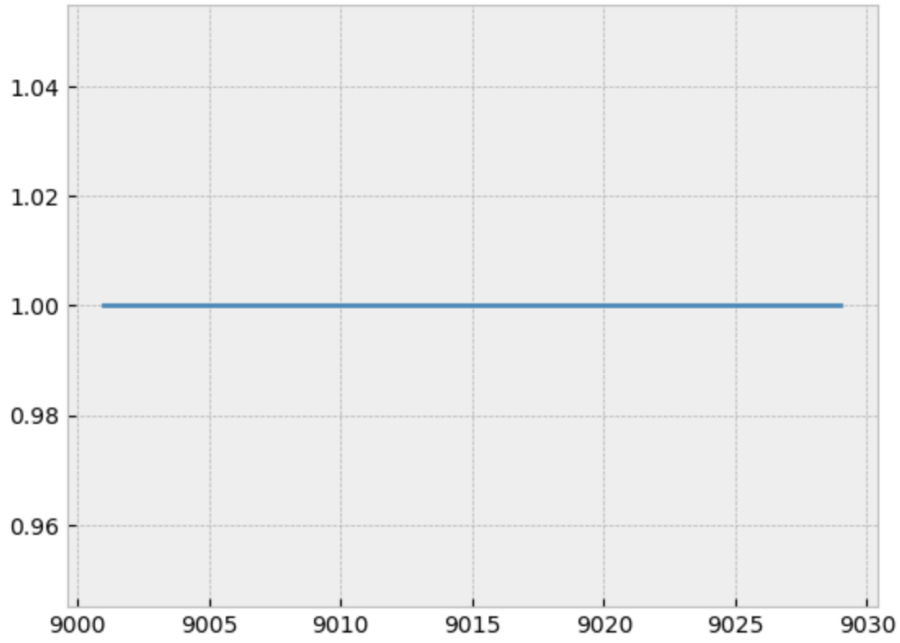
In [996]:

```
timeformat = []
reservation_id = list(df['reservation_id'])
for i,row in df.iterrows():
    if ':' in row['time']:
        timeformat.append(1)


plt.plot(df['reservation_id'],timeformat)
plt.show()
```



Example 4 : Below is a graph for user reservation date format and user reservation id. All the users have a certain date format and the format remains consistent. Below is the graph to show consistency in the user reservation date.

```
In [997]: dateformat = []
reservation_id = list(df['reservation_id'])
for i,row in df.iterrows():
    if '-' in row['date']:
        dateformat.append(1)


plt.plot(df['reservation_id'],dateformat)
plt.show()
```



6. SQL to insert the data into your database

Schema:
use r3_system;

Restaurant Table:
CREATE TABLE restaurant (
restaurant_id INT,
restaurant_name VARCHAR(100),
restaurant_rating FLOAT,
restaurant_type VARCHAR(100),
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 VARCHAR(20),
no_of_tables Int,
PRIMARY KEY (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 VARCHAR(20),
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 Table:

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

Menu Restaurant 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)
);
```

Restaurant Dishes Table:

```
CREATE TABLE restaurant_dishes(
dish_id INT,
restaurant_id INT,
dish_name VARCHAR(500),
Dish_type VARCHAR(50),
dish_price FLOAT,
cuisines_id INT,
menu_id INT,
PRIMARY KEY (dish_id),
FOREIGN KEY (restaurant_id) REFERENCES restaurant (restaurant_id),
FOREIGN KEY (menu_id) REFERENCES menu(menu_id)
);
```

Restaurant Reservation Table:

```
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 Table:

```
CREATE TABLE restaurant_review(
review_id INT,
review_comment VARCHAR(1000),
review_like 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 Table:

```
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)
);
```

7. Scripts to clean your data

- Assignment 3 Web Scraping
- Assignment 3 Inserting scraped Data into database

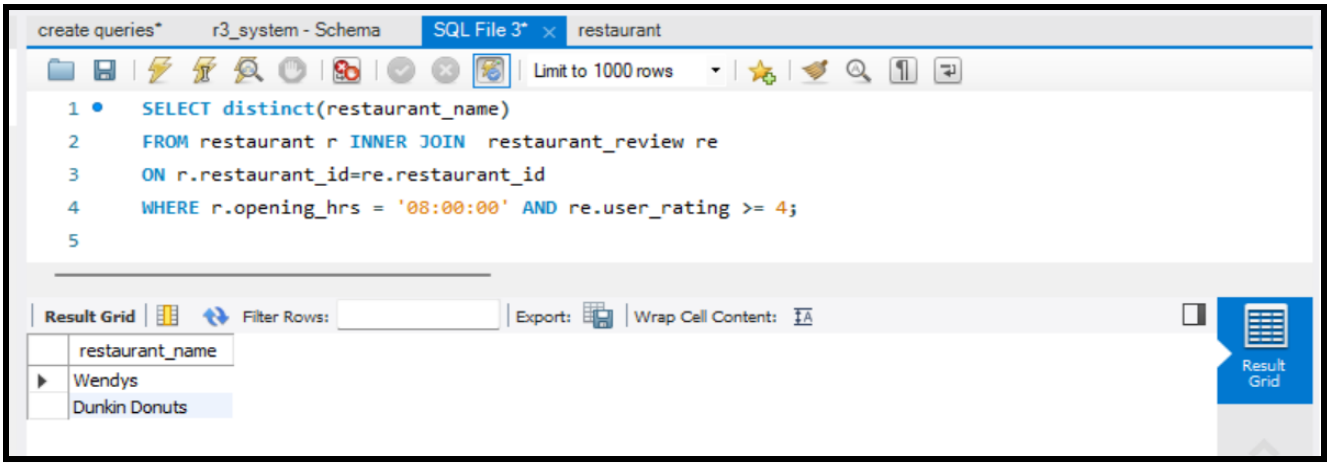
8. Use cases and Tests with their output

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;
```

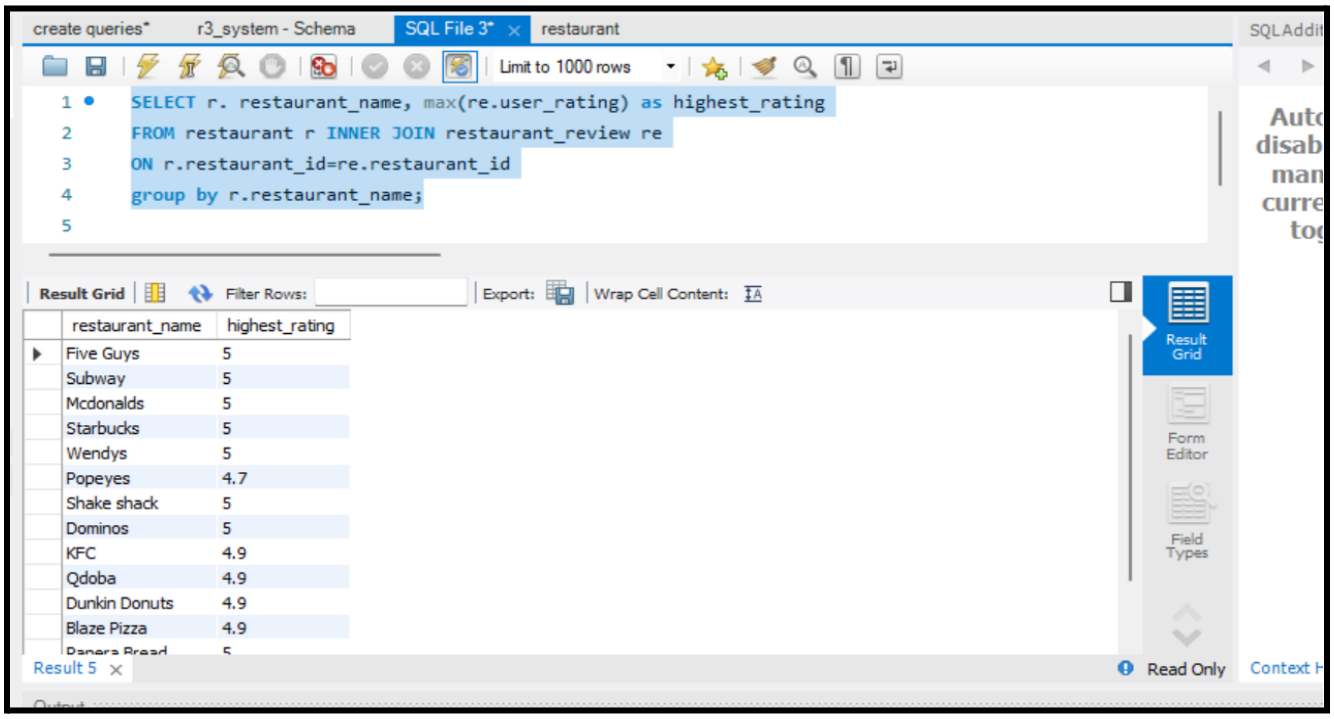


Use Case 2: Check the restaurant with the best User Rating.

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;



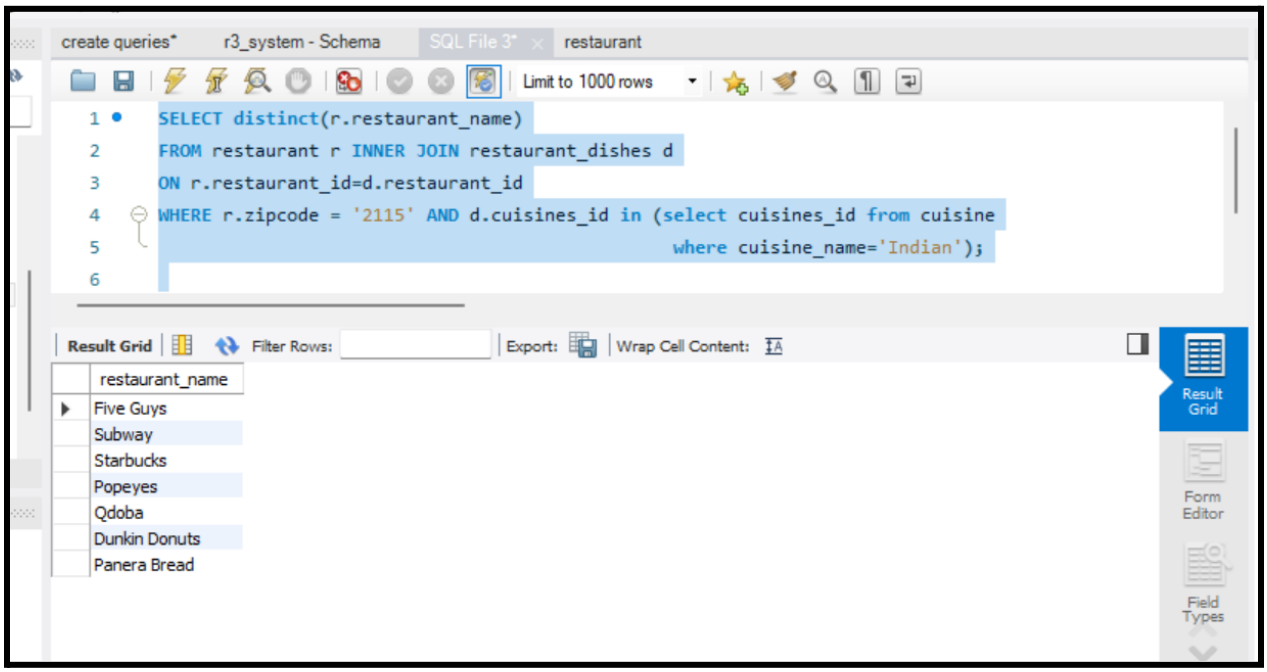
Use Case 3: Search for restaurants offering Indian cuisine in the zip code 2115

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');
```



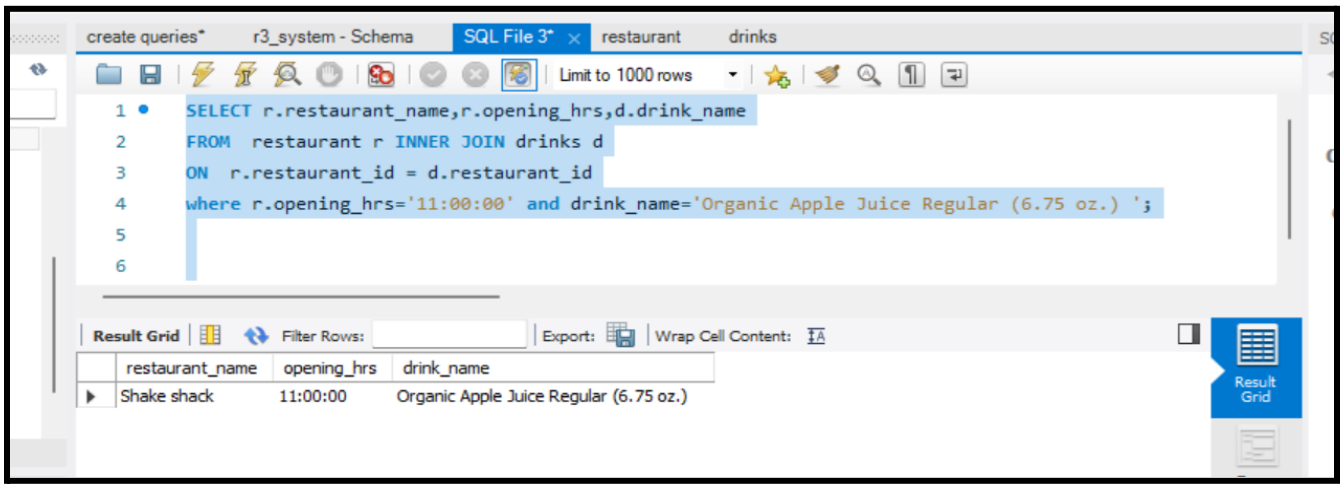
Use Case 4: View a restaurant that offers Organic Apple Juice Regular (6.75 oz.) and opens at 11 am

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.) ';
```

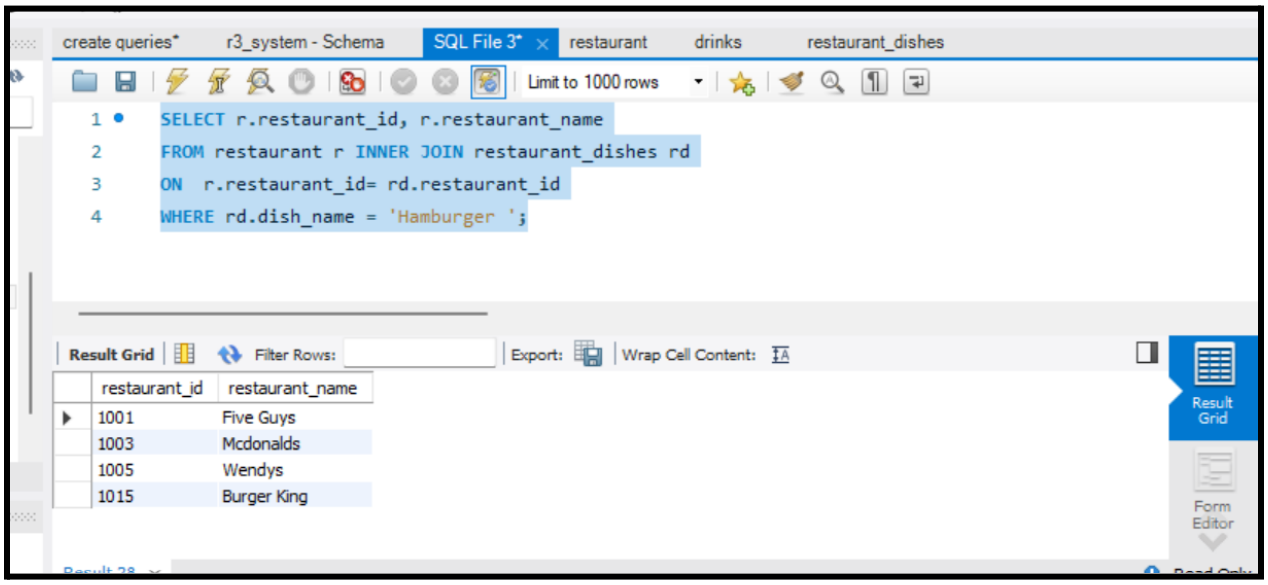


Use Case 5: Search for a restaurant that serves Hamburgers.

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 ';
```



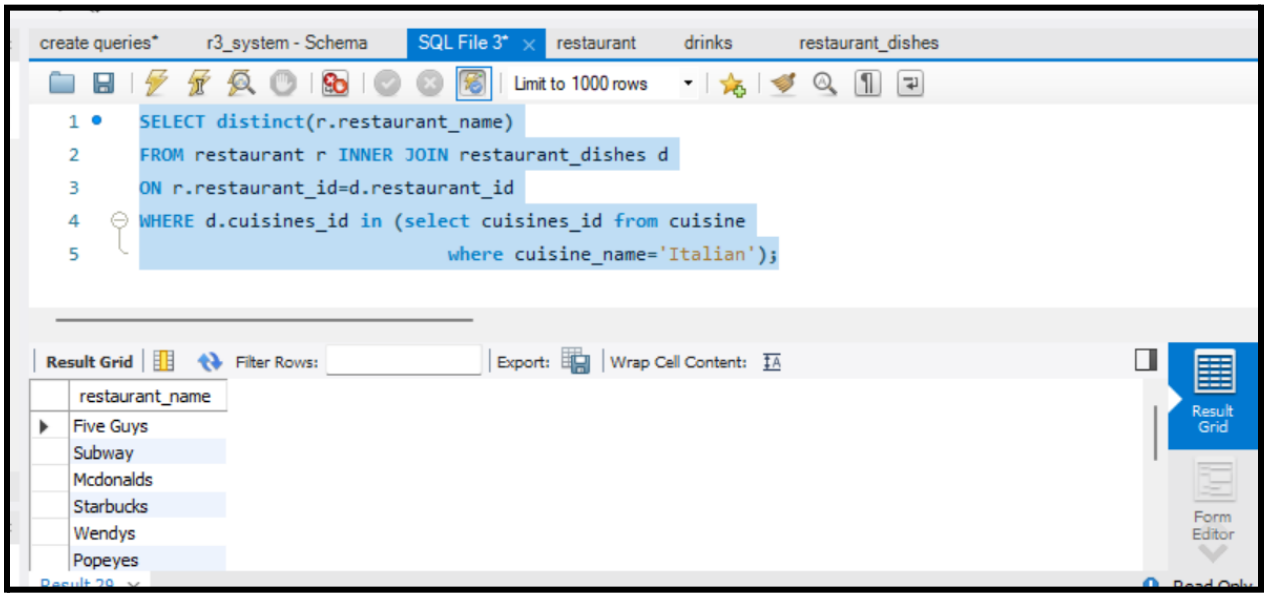
Use case 6: Search for a restaurant offering Italian cuisine.

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');

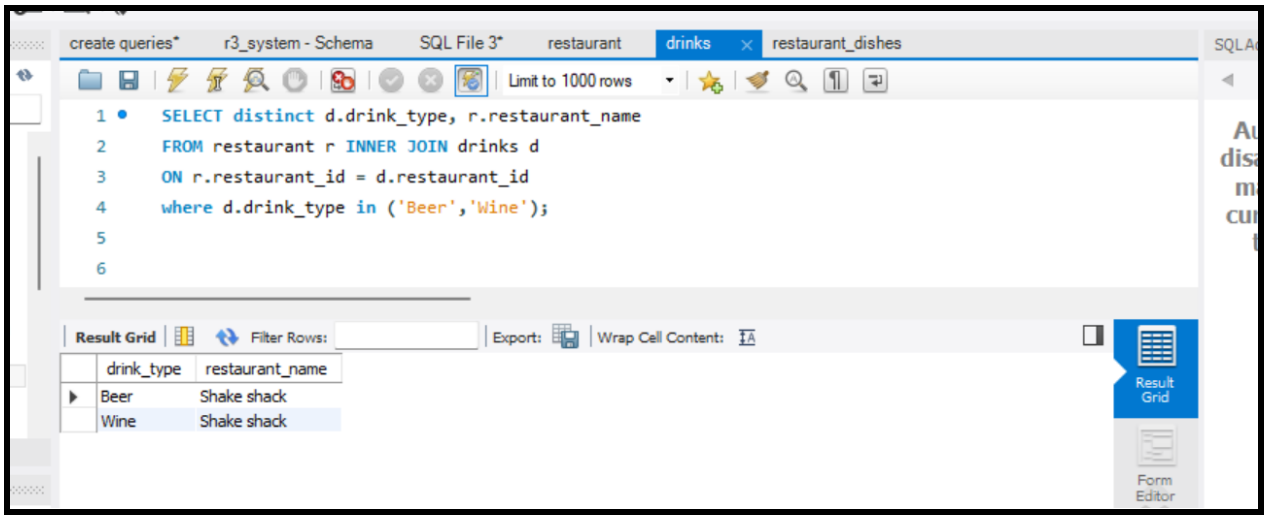


Use Case 7: View the restaurant which serves alcoholic (Beer, wine) drinks.

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');

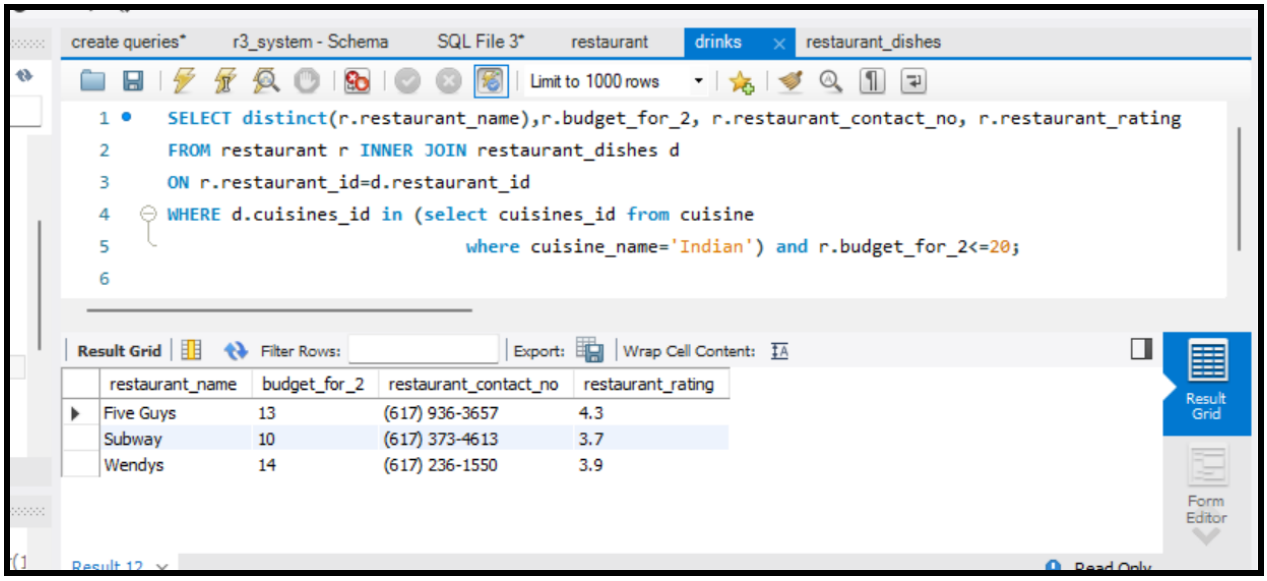


Use Case 8: View a restaurant with Indian cuisine with a specific budget (say less than $20)

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;
```

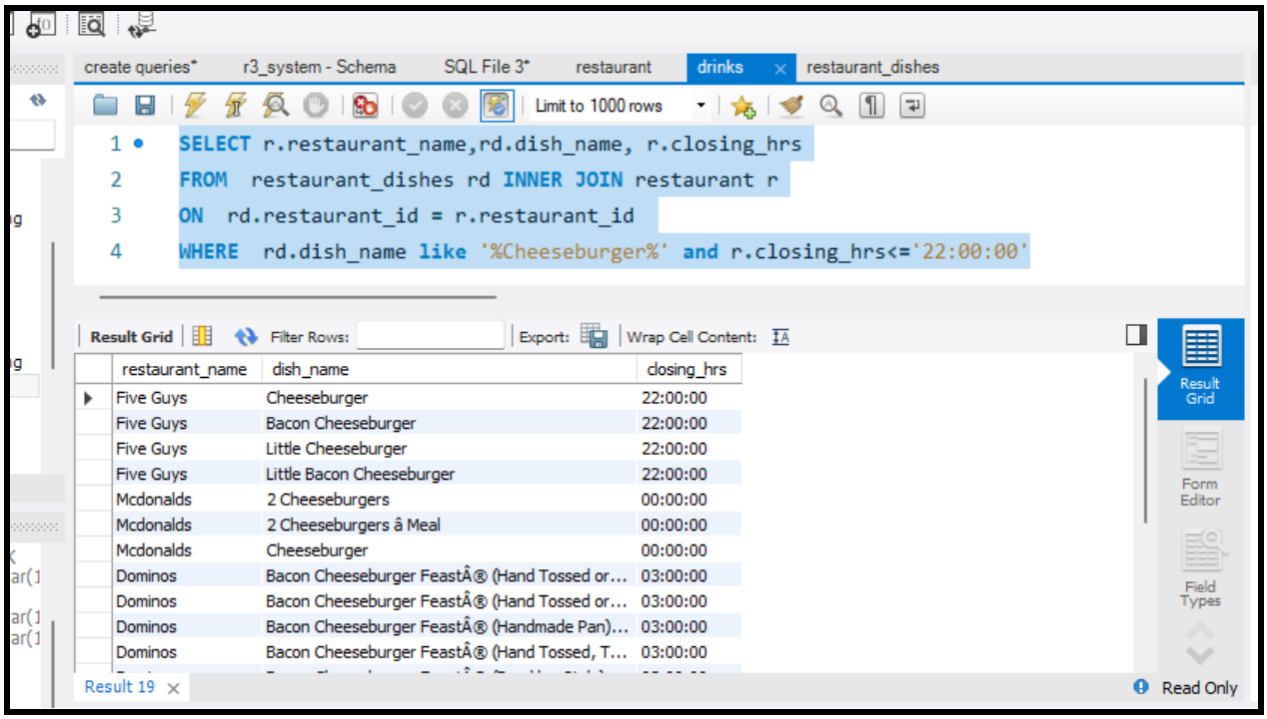


Use Case 9: View a restaurant that serves Cheeseburger and is open till 10 pm

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'
```

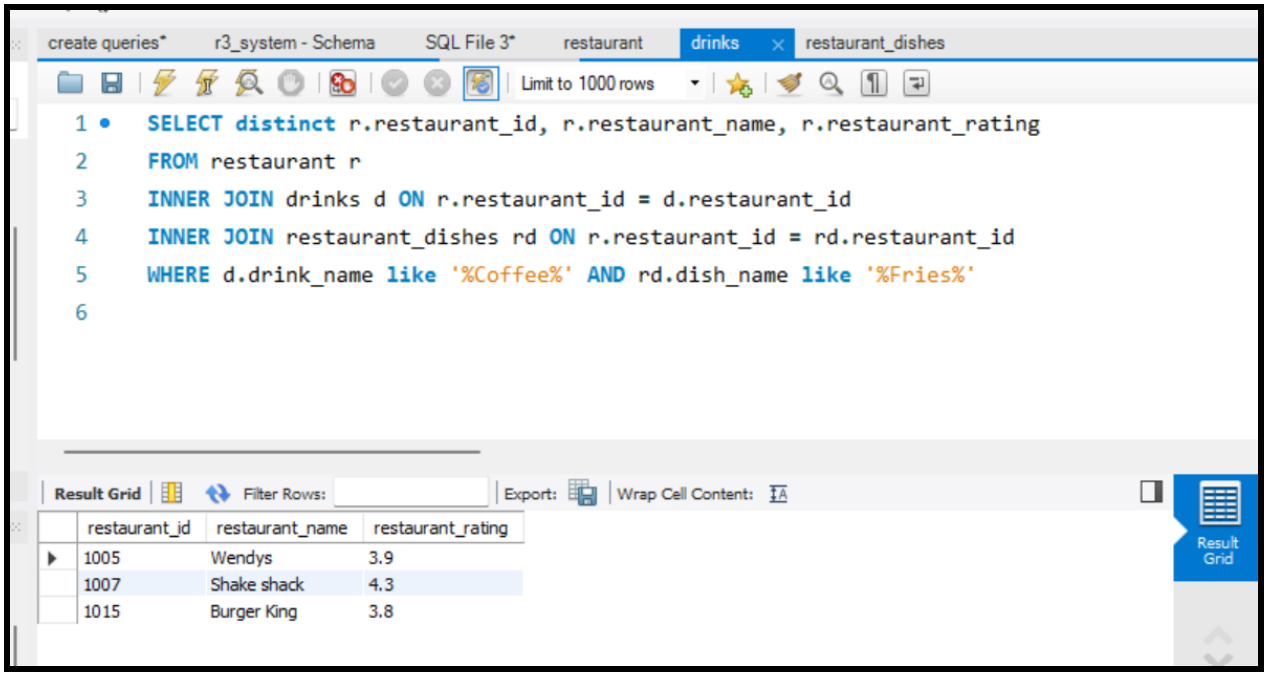


Use case 10: View the restaurant's offerings of Coffee and Fries

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%'
```

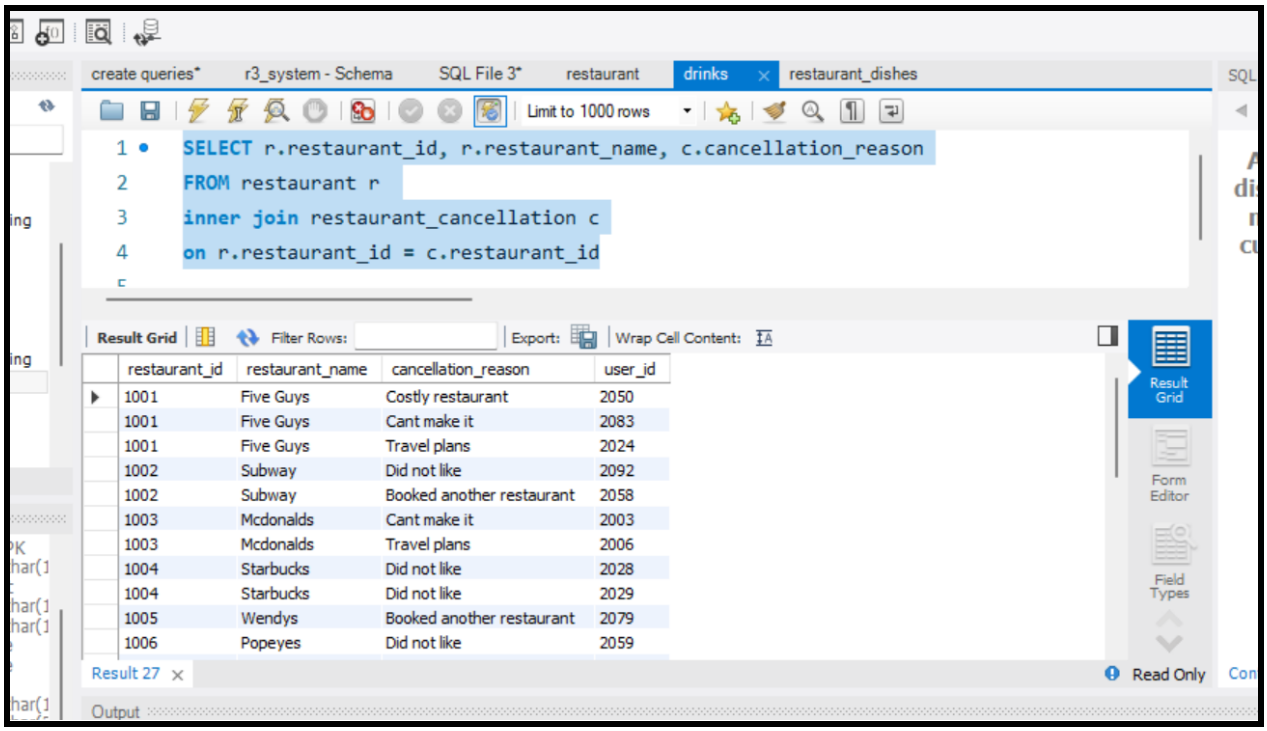


Use case 11: View cancellations done for a restaurant and their reasons

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
```



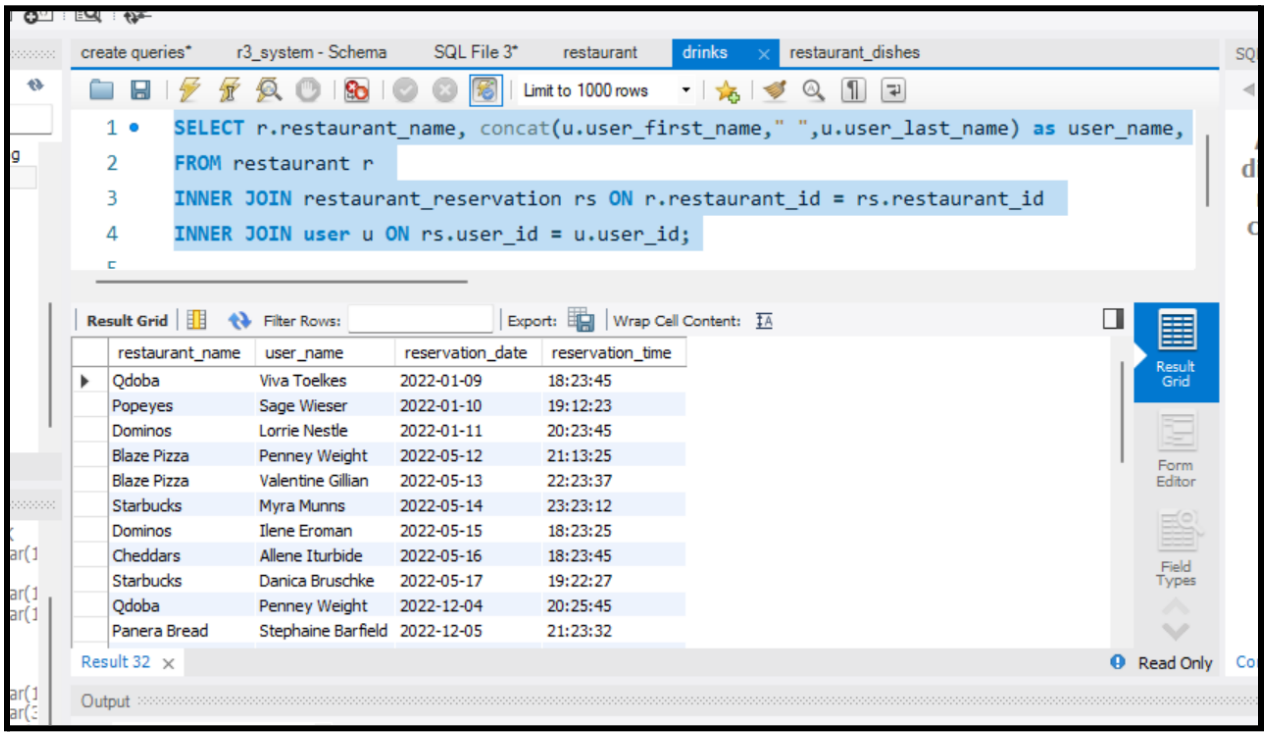
Use Case 12: What are the restaurant details, user details, and reservation details where the users reserved a table?

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;

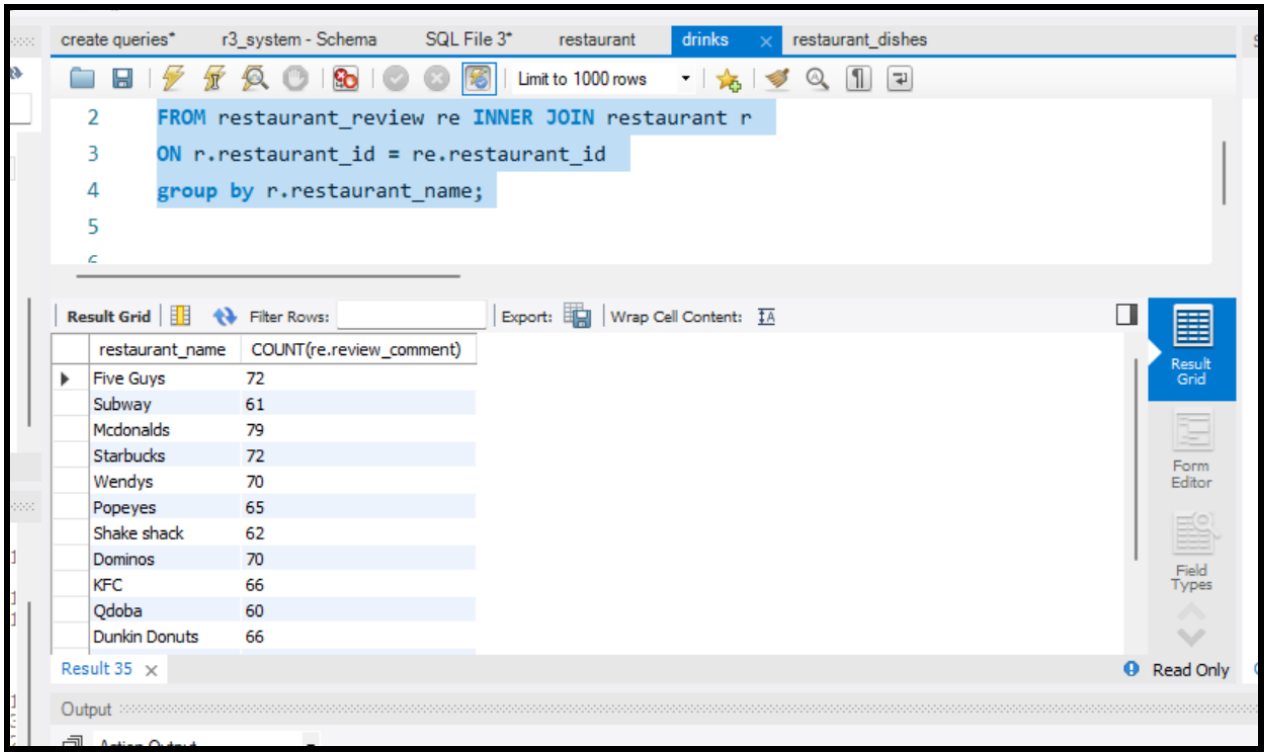


Use Case 13: How many reviews have a restaurant received to date?

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;

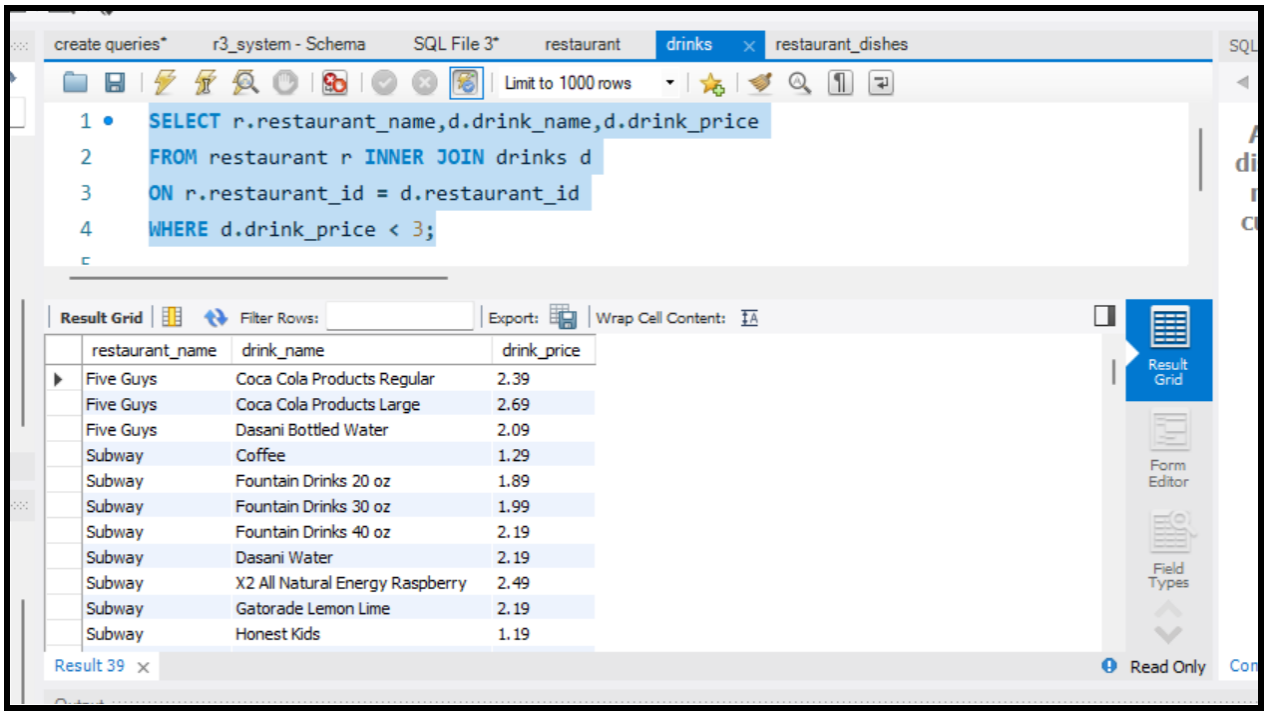


Use Case 14: Restaurants with drinks below 3$?

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;
```

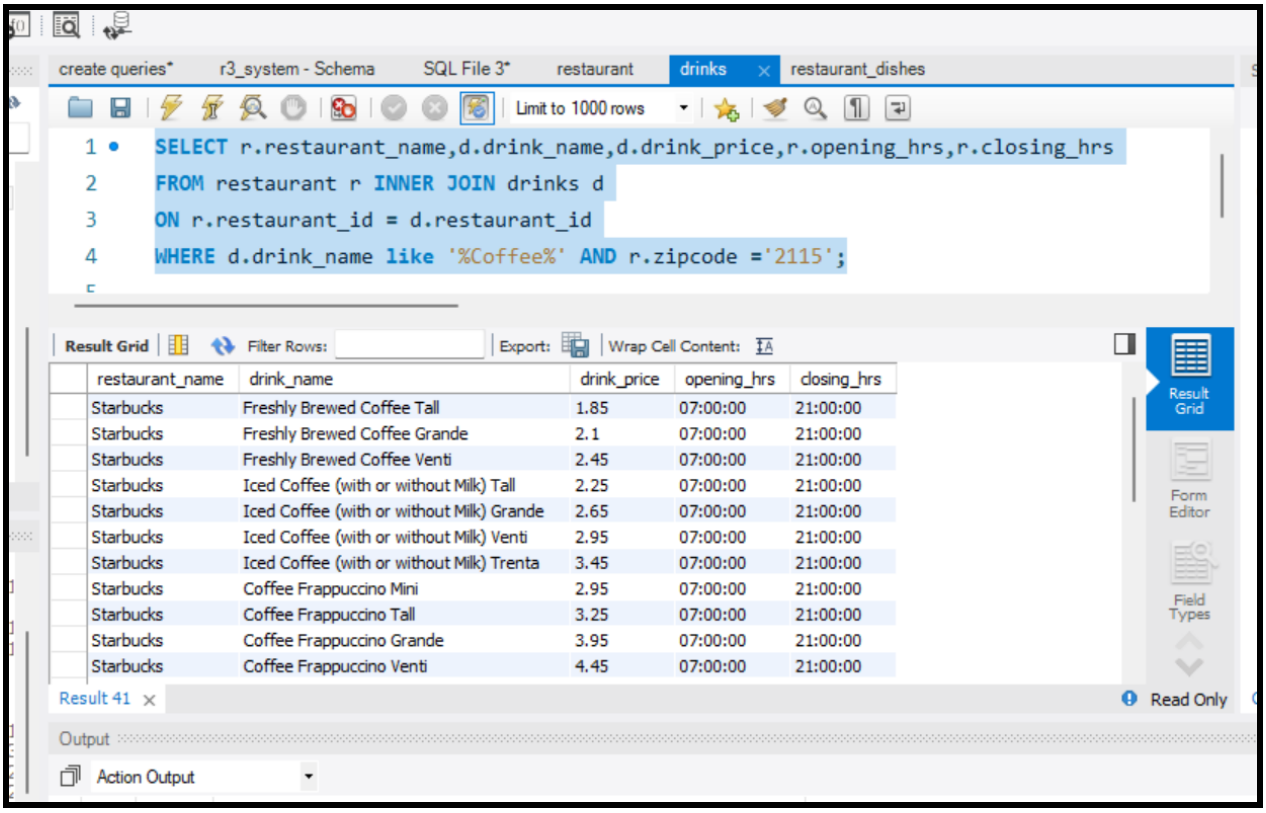


Use Case 15: Restaurants that serve coffee in zip code 02115?

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';
```

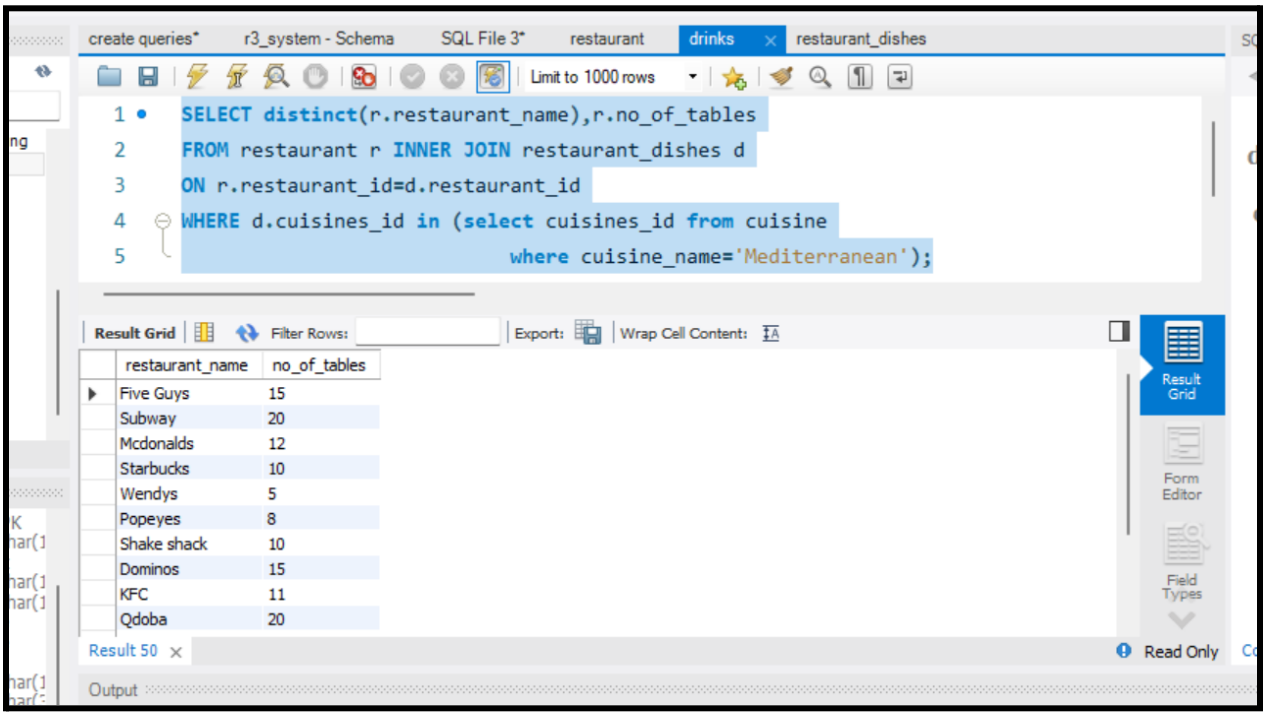


Use Case 16: Check if the table is available for Reservation for a Mediterranean restaurant.

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');
```

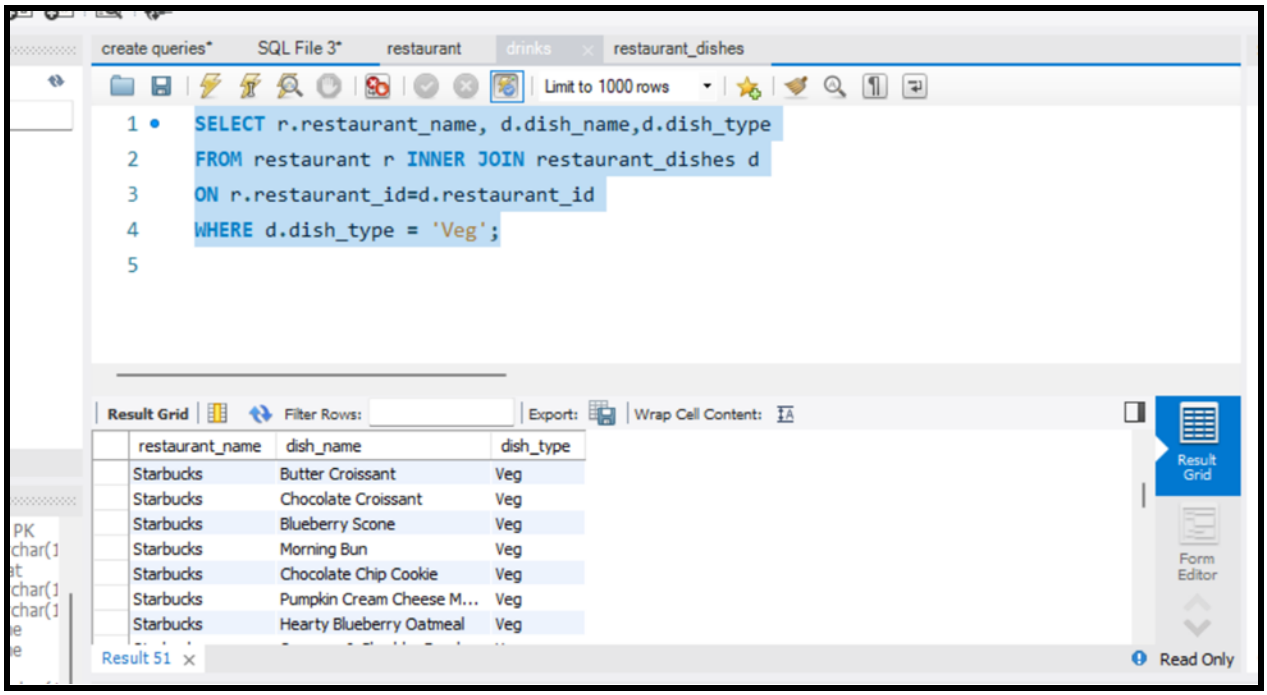


Use Case 17: Search for restaurants that offer vegetarian food

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';
```

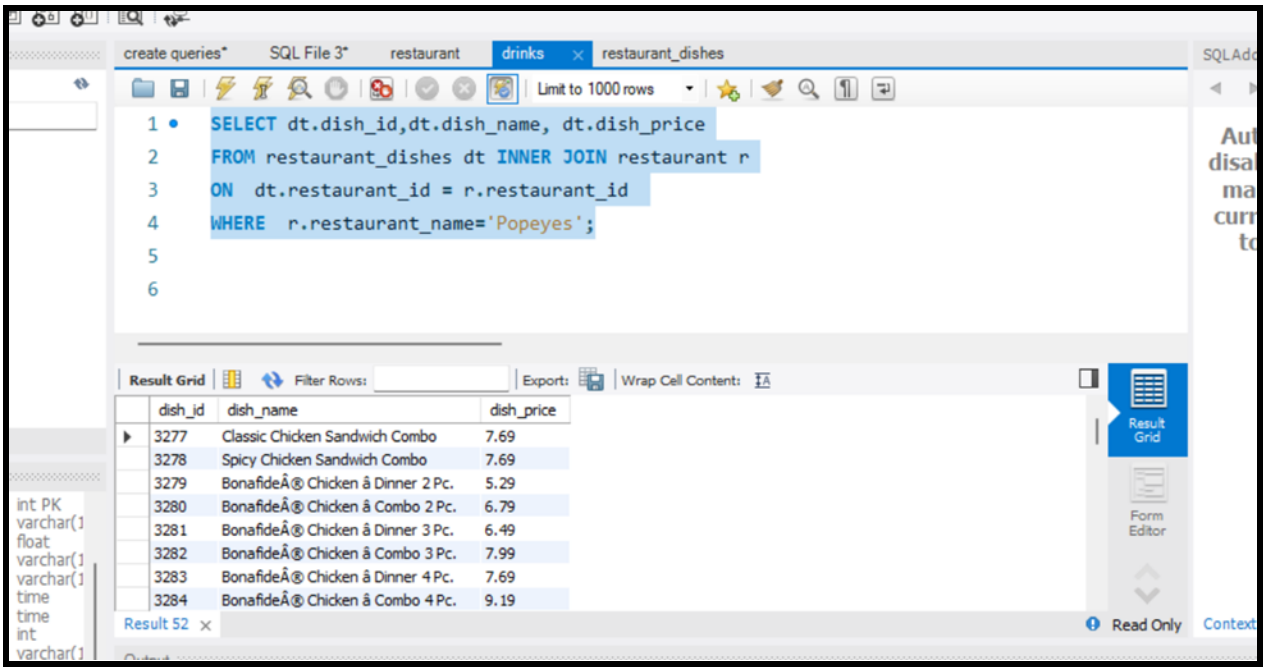


Use Case 18: View the food menu for a specific restaurant.

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';
```

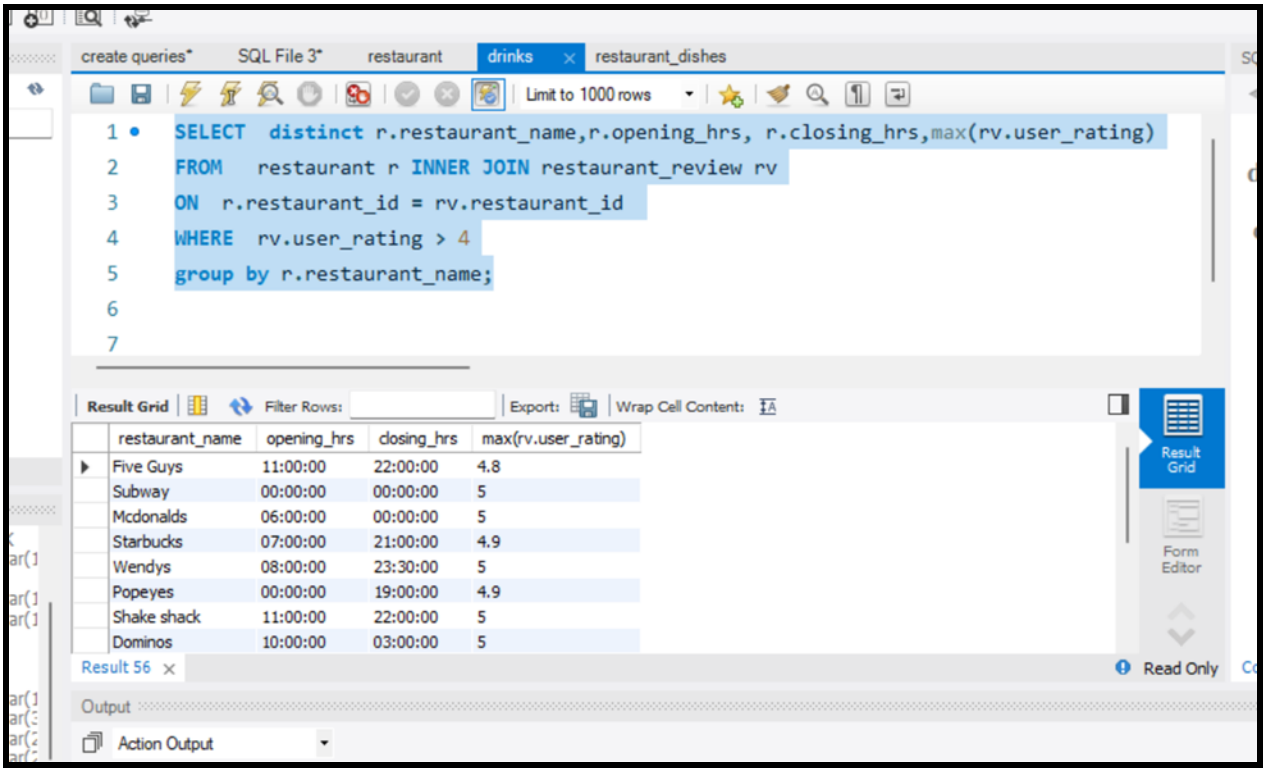


Use Case 19: View Opening and Closing Hours for a specific restaurant that has a good user rating (considering good as a rating above 4)

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;
```

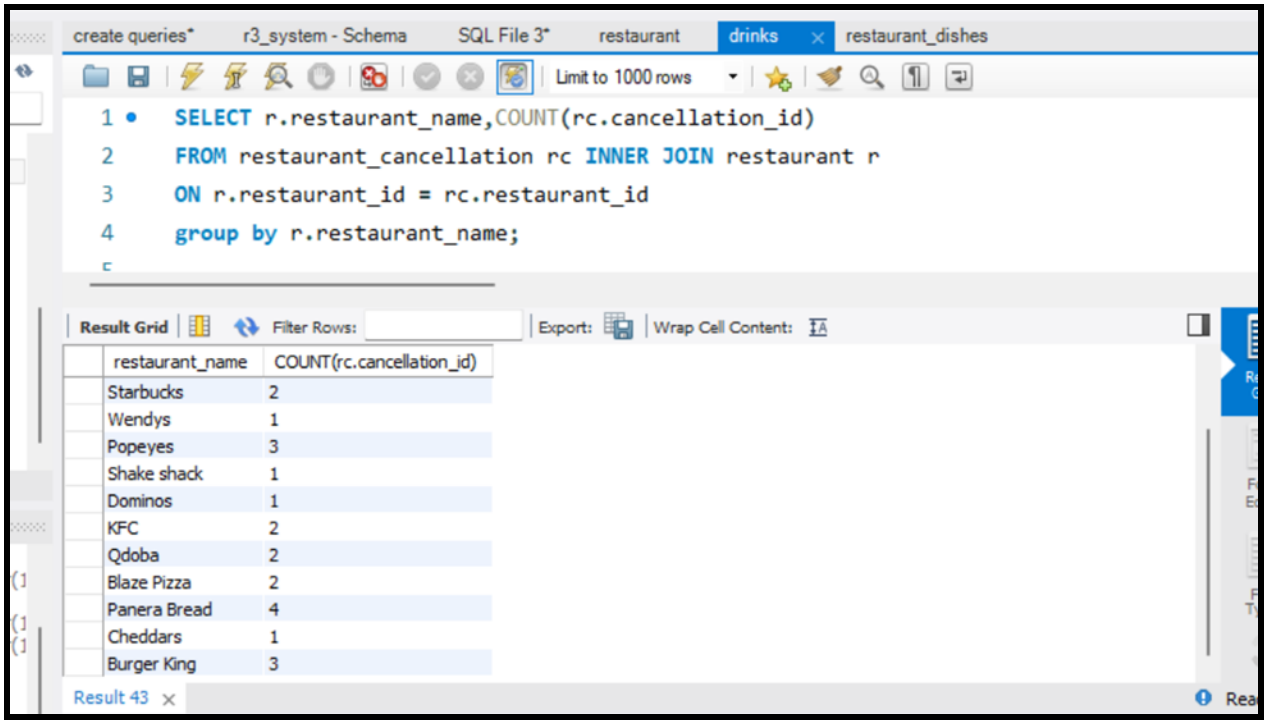


Use Case 20: How many cancellations has a restaurant received till date?

Description: The user searches for cancellations for a restaurant received till 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 till date
System Responses: Displays cancellations a restaurant received till date
Post Condition: The user will be able to see all cancellations till 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;
```



9. Assignment 2 Feedback:

No feedback. (Assignment score 100/100)