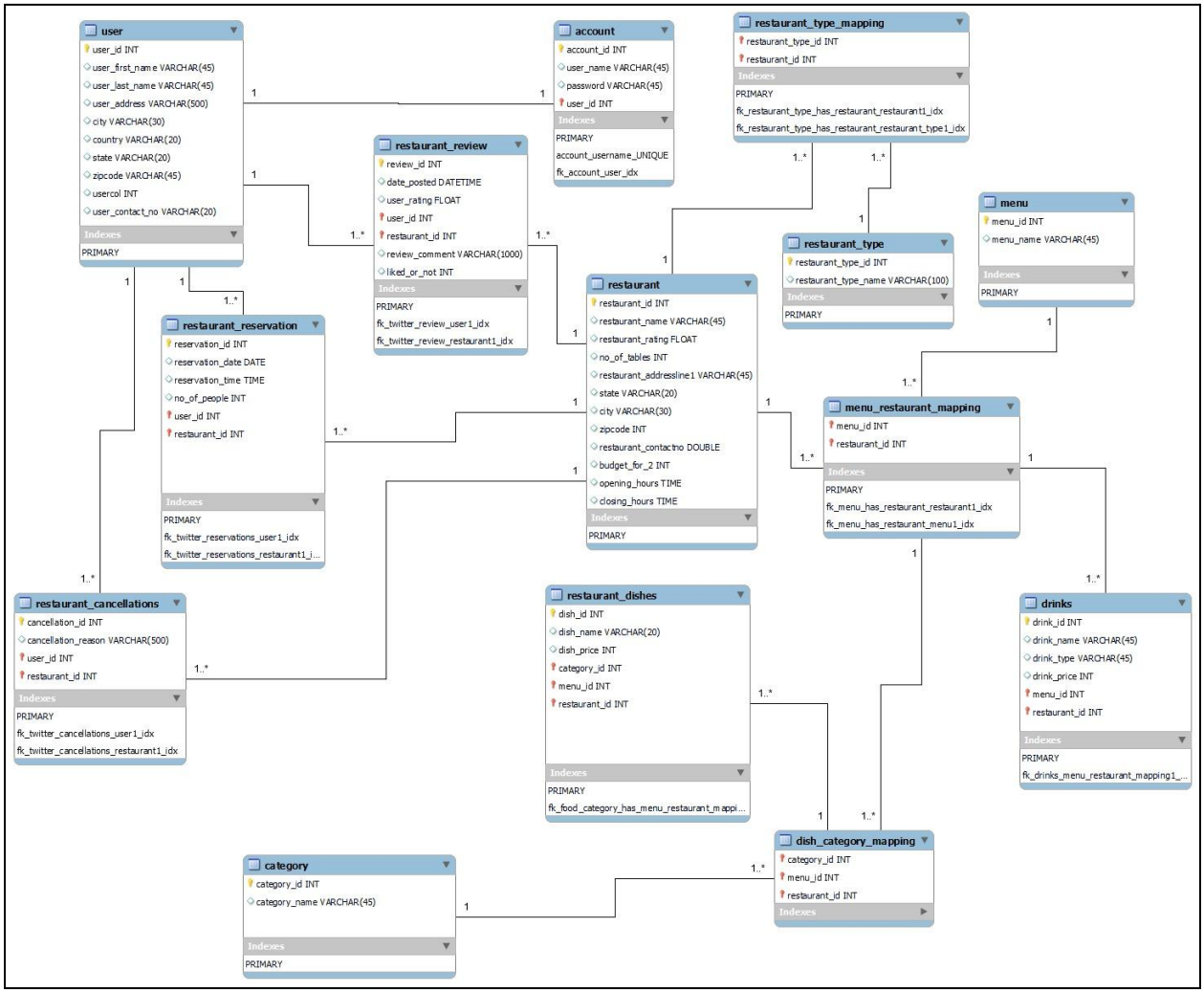
DMDD ASSIGNMENT 4 - Normalization
Topic Name: Restaurant Recommendation and Reservation System

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)

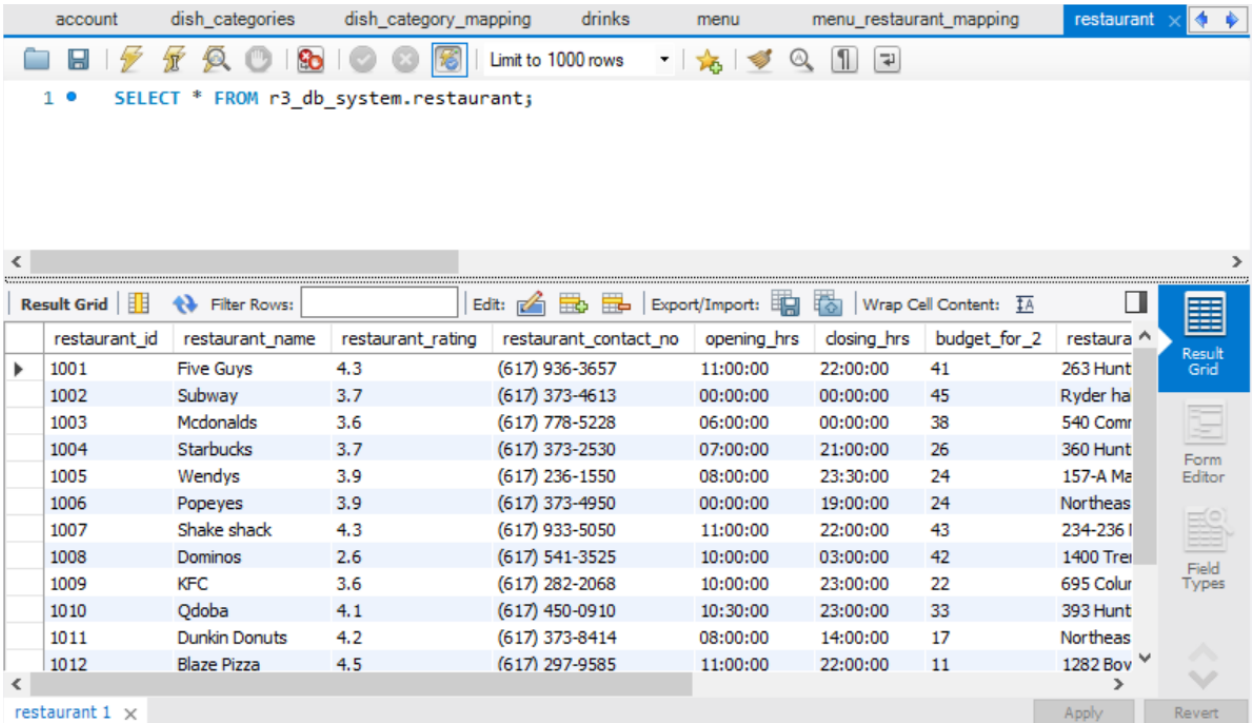
Physical Model of the R3 System



Normalization check for Restaurant recommendation

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)
);
```



1st NF check:

- restaurant_id is the primary key for the table named restaurant
- The values in each column of a restaurant table 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 above table

2nd NF check:

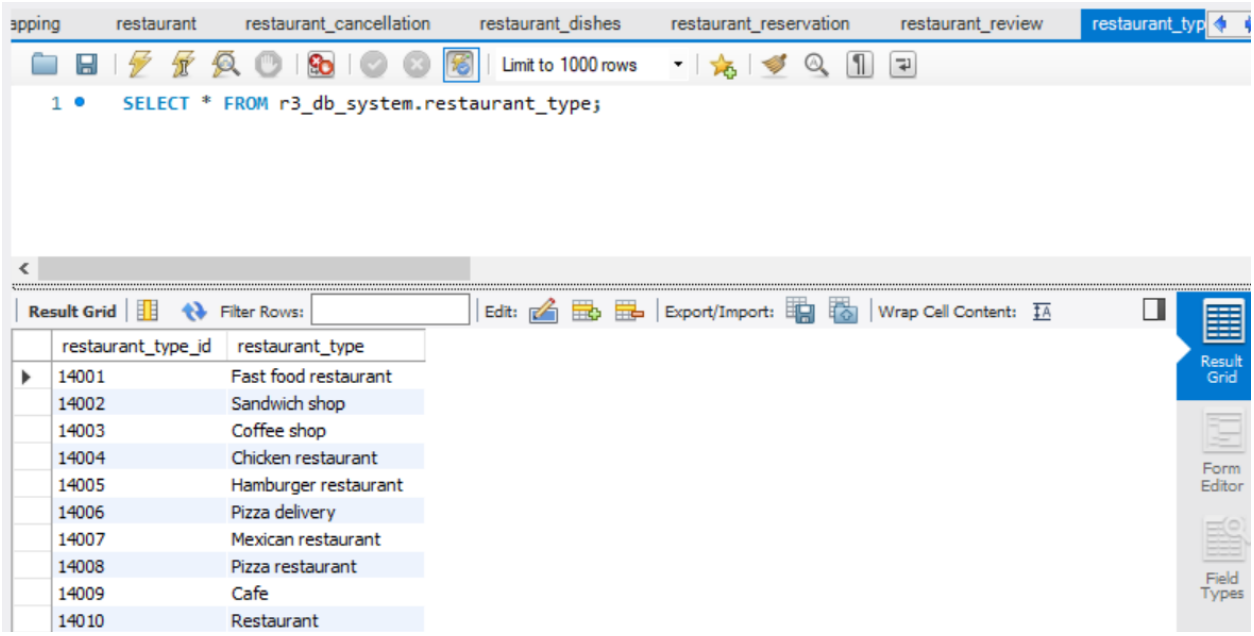
- All requirements for 1st NF are met
- No partial dependencies in the user table.
    - All non key attributes like restaurant_name, restaurant_rating, restaurant_contact_no, opening_hrs, closing_hrs, budget_for_2, restaurant_address, city, state, zip code, no_of_tables, depend on the primary key restaurant_id
    - The restaurant_type table has been created as a separate table which was earlier included into the restaurant table
- There is No calculated data in the above table

3rd NF check:

- All the requirements for 2nd NF are met
- There is no transitive dependency for non-prime attributes

Restaurant Type Table: Created when restaurant table was broken into 2 tables (restaurant, restaurant_type)

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



restaurant_type_id	restaurant_type
14001	Fast food restaurant
14002	Sandwich shop
14003	Coffee shop
14004	Chicken restaurant
14005	Hamburger restaurant
14006	Pizza delivery
14007	Mexican restaurant
14008	Pizza restaurant
14009	Cafe
14010	Restaurant

1st NF check:

- restaurant_id is the primary key for the table named restaurant_type
- The values in each column of a restaurant table are atomic
- There are no repeating groups in the above table

2nd NF check:

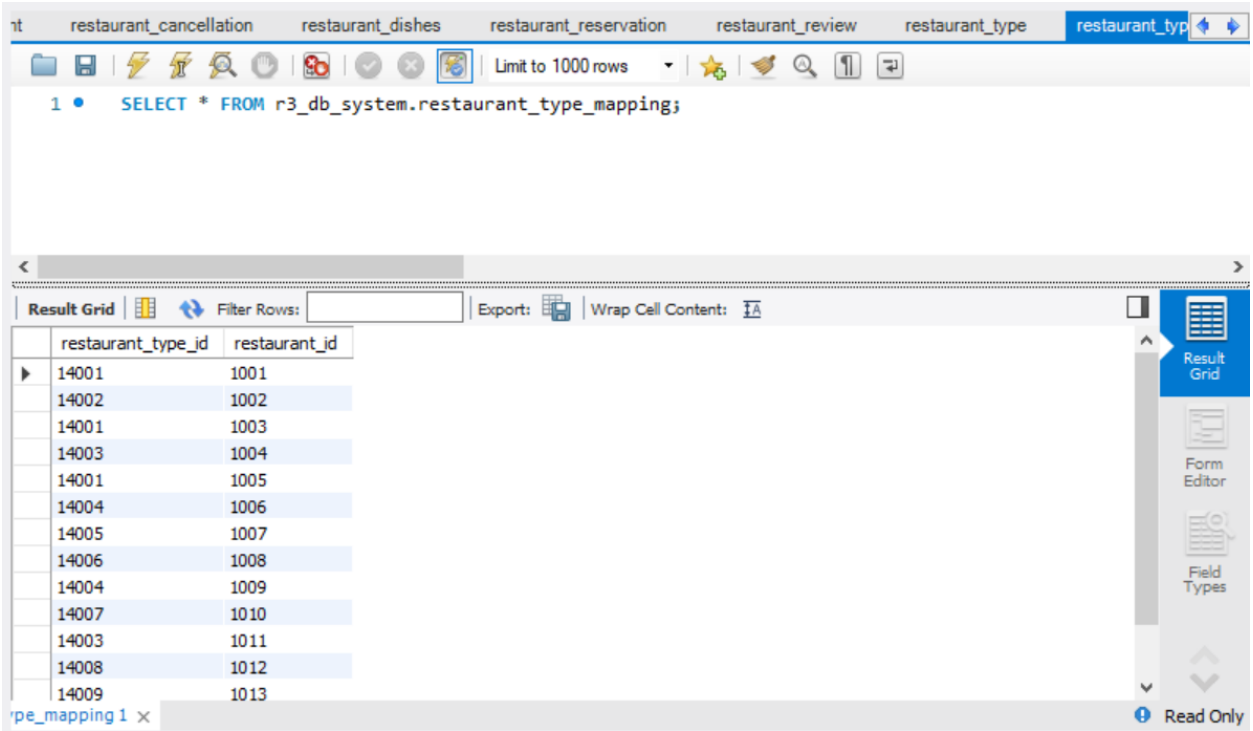
- All requirements for 1st NF are met
- No partial dependencies in the user table. All non key attributes like restaurant_type depend on the primary key restaurant_id
- There is No calculated data in the above table

3rd NF check:

- All the requirements for 2nd NF are met
- There is no transitive dependency for non-prime attributes

Restaurant Type Mapping Table: Created because of many to many mapping of restaurant, restaurant_type tables

```
CREATE TABLE restaurant_type_mapping (
restaurant_type_id INT ,
restaurant_id INT,
restaurant_type VARCHAR(100),
FOREIGN KEY (restaurant_type_id ) REFERENCES restaurant_type_mapping(
(restaurant_type_id ),
FOREIGN KEY (restaurant_id ) REFERENCES restaurant_type_mapping(restaurant_id )
);
```



restaurant_type_id	restaurant_id
14001	1001
14002	1002
14001	1003
14003	1004
14001	1005
14004	1006
14005	1007
14006	1008
14004	1009
14007	1010
14003	1011
14008	1012
14009	1013

1st NF check:

- restaurant_id is the primary key for the table named restaurant_type
- The values in each column of a restaurant table are atomic
- There are no repeating groups in the above table

2nd NF check:

- All requirements for 1st NF are met
- No partial dependencies in the user table. All non key attributes like restaurant_type depend on the primary key restaurant_id
- There is No calculated data in the above table

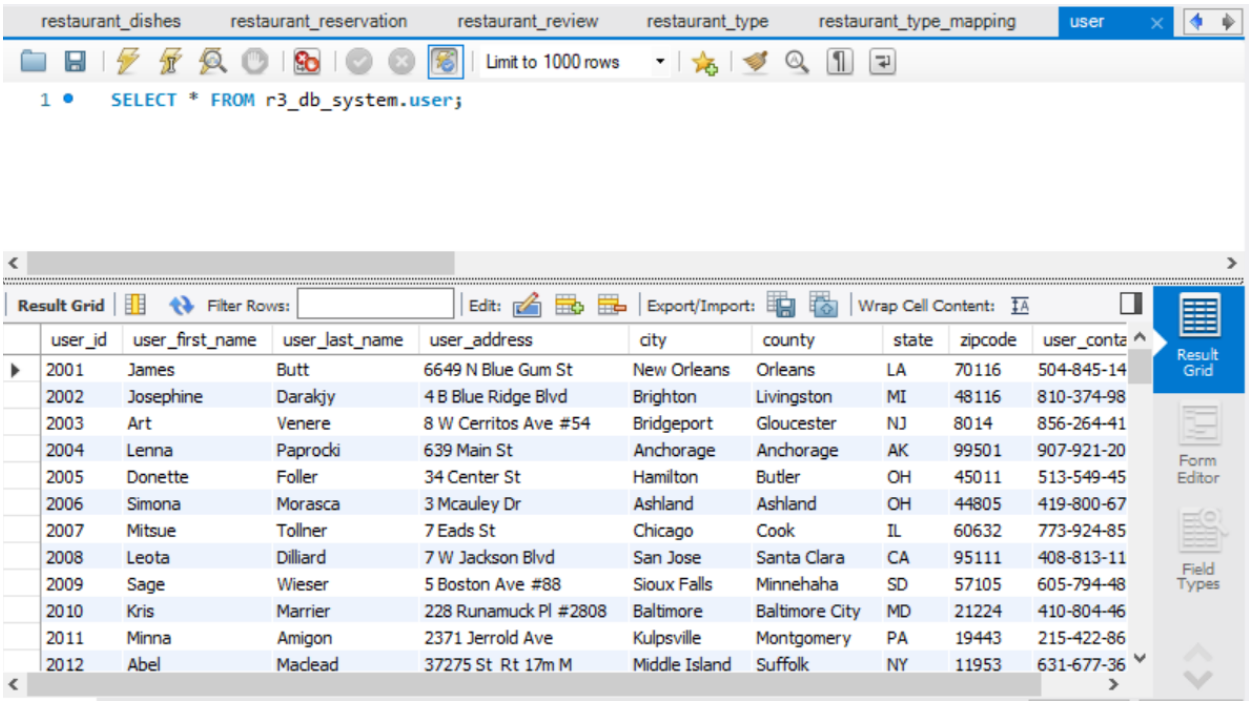
3rd NF check:

- All the requirements for 2nd NF are met
- There is no transitive dependency for non-prime attributes

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)
);



user_id	user_first_name	user_last_name	user_address	city	county	state	zipcode	user_conta
2001	James	Butt	6649 N Blue Gum St	New Orleans	Orleans	LA	70116	504-845-14
2002	Josephine	Darakjy	4 B Blue Ridge Blvd	Brighton	Livingston	MI	48116	810-374-98
2003	Art	Venere	8 W Cerritos Ave #54	Bridgeport	Gloucester	NJ	8014	856-264-41
2004	Lenna	Paprocki	639 Main St	Anchorage	Anchorage	AK	99501	907-921-20
2005	Donette	Foller	34 Center St	Hamilton	Butler	OH	45011	513-549-45
2006	Simona	Morasca	3 Mcauley Dr	Ashland	Ashland	OH	44805	419-800-67
2007	Mitsue	Tollner	7 Eads St	Chicago	Cook	IL	60632	773-924-85
2008	Leota	Dilliard	7 W Jackson Blvd	San Jose	Santa Clara	CA	95111	408-813-11
2009	Sage	Wieser	5 Boston Ave #88	Sioux Falls	Minnehaha	SD	57105	605-794-48
2010	Kris	Marrier	228 Runamuck Pl #2808	Baltimore	Baltimore City	MD	21224	410-804-46
2011	Minna	Amigon	2371 Jerrold Ave	Kulpsville	Montgomery	PA	19443	215-422-86
2012	Abel	Maclead	37275 St Rt 17m M	Middle Island	Suffolk	NY	11953	631-677-36

1st NF check:

- user_id is the primary key for the table named user
- The values in each column of a restaurant table are atomic
- There are no repeating groups in the above table

2nd NF check:

- All requirements for 1st NF are met
- No partial dependencies in the user table.
    - All non key attributes like user_first_name, user_last_name, user_address, city, county, state, zipcode, user_contact_no, depend on the primary key user_id
- There is No calculated data in the above table

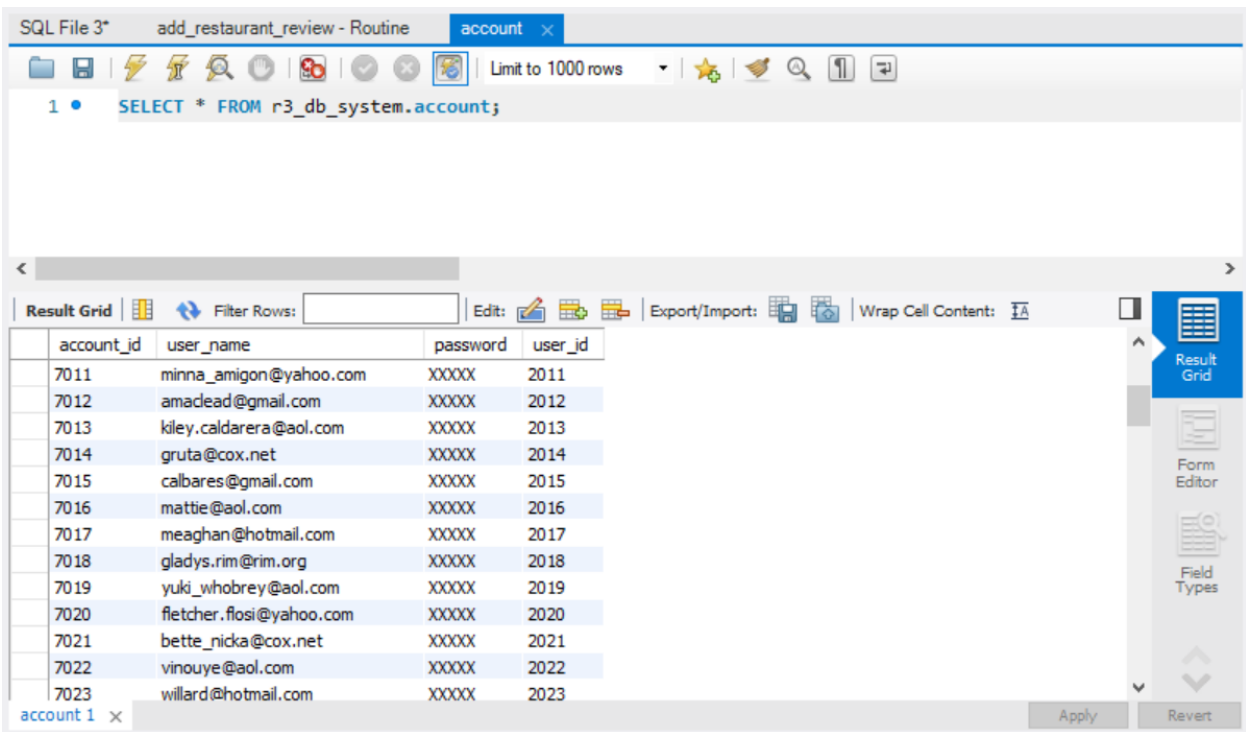
3rd NF check:

- 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.

Account Table: **Created when user table was broken into 2 tables (user, account)**

```
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)
);
```



account_id	user_name	password	user_id
7011	minna_amigon@yahoo.com	XXXXX	2011
7012	amaclead@gmail.com	XXXXX	2012
7013	kiley.caldarera@aol.com	XXXXX	2013
7014	gruta@cox.net	XXXXX	2014
7015	calbares@gmail.com	XXXXX	2015
7016	mattie@aol.com	XXXXX	2016
7017	meaghan@hotmail.com	XXXXX	2017
7018	gladys.rim@rim.org	XXXXX	2018
7019	yuki_whobrey@aol.com	XXXXX	2019
7020	fletcher.flosi@yahoo.com	XXXXX	2020
7021	bette_nicka@cox.net	XXXXX	2021
7022	vinouye@aol.com	XXXXX	2022
7023	willard@hotmail.com	XXXXX	2023

1st NF check:

- account_id is the primary key for the table named account
- The values in each column of a restaurant table are atomic
- There are no repeating groups in the above table

2nd NF check:

- All requirements for 1st NF are met
- No partial dependencies in the user table.
    - All non key attributes like user_name, and password depend on the primary key account_id

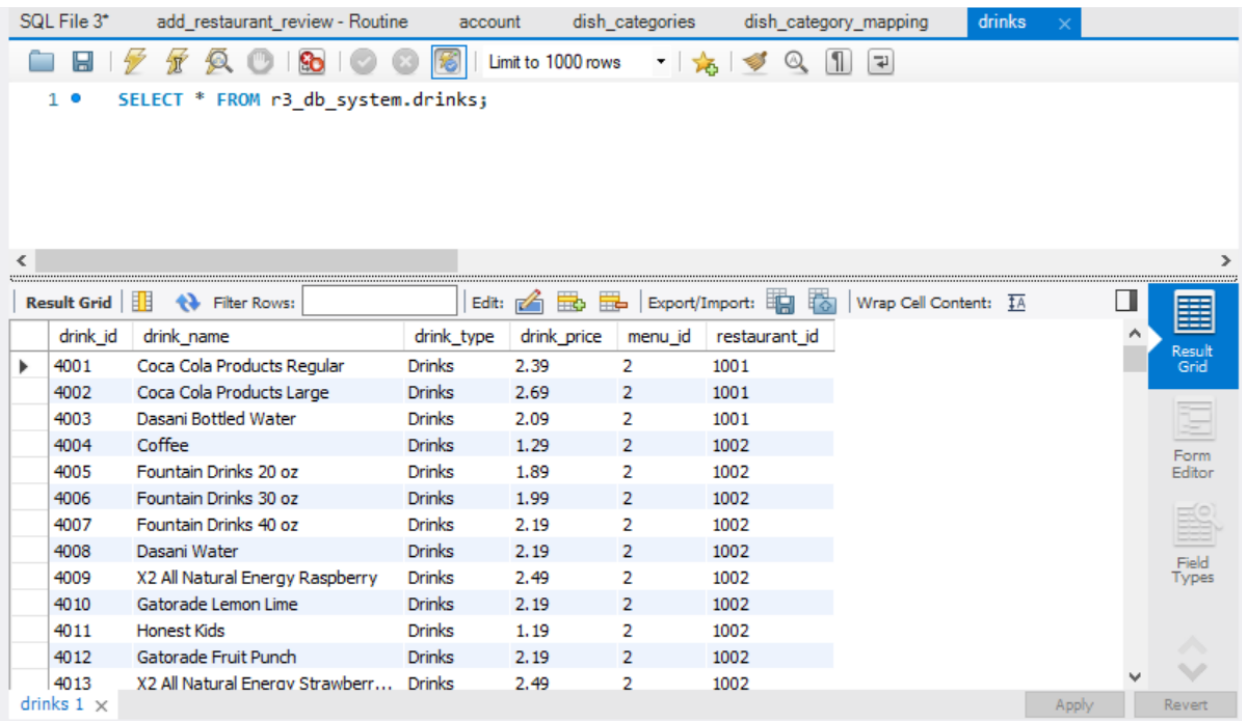
- There is No calculated data in the above table

3rd NF check:

- All the requirements for 2nd NF are met
- There is no transitive dependency for non-prime attributes

Menu Table: Created when restaurant table had non atomic values for menu

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



drink_id	drink_name	drink_type	drink_price	menu_id	restaurant_id
4001	Coca Cola Products Regular	Drinks	2.39	2	1001
4002	Coca Cola Products Large	Drinks	2.69	2	1001
4003	Dasani Bottled Water	Drinks	2.09	2	1001
4004	Coffee	Drinks	1.29	2	1002
4005	Fountain Drinks 20 oz	Drinks	1.89	2	1002
4006	Fountain Drinks 30 oz	Drinks	1.99	2	1002
4007	Fountain Drinks 40 oz	Drinks	2.19	2	1002
4008	Dasani Water	Drinks	2.19	2	1002
4009	X2 All Natural Energy Raspberry	Drinks	2.49	2	1002
4010	Gatorade Lemon Lime	Drinks	2.19	2	1002
4011	Honest Kids	Drinks	1.19	2	1002
4012	Gatorade Fruit Punch	Drinks	2.19	2	1002
4013	X2 All Natural Energy Strawberr...	Drinks	2.49	2	1002

1st NF check:

- menu_id is the primary key for the table named menu
- The values in each column of a restaurant table are atomic
- There are no repeating groups in the above table

2nd NF check:

- All requirements for 1st NF are met
- No partial dependencies in the user table.
    - Non key attribute like menu_name depend on the primary key menu_id

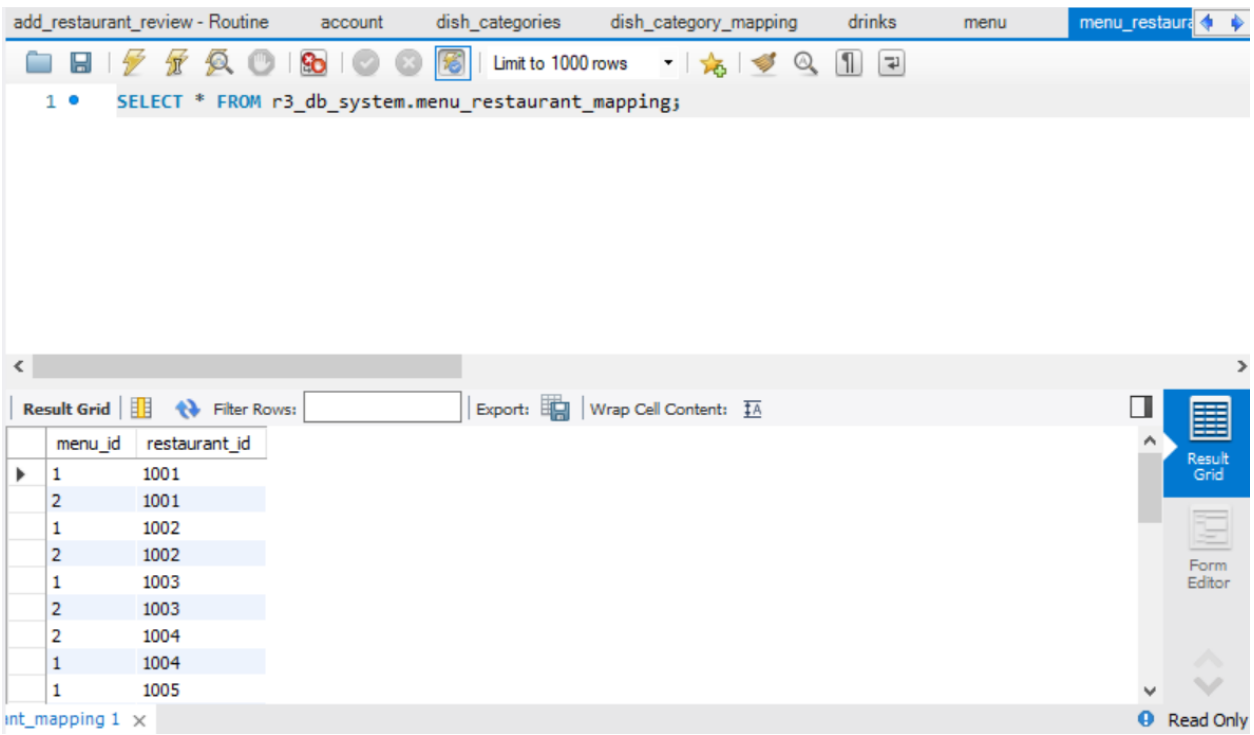
- There is No calculated data in the above table

3rd NF check:

- All the requirements for 2nd NF are met
- There is no transitive dependency for non-prime attributes

Menu Restaurant Mapping Table: Created because of many to many mapping of restaurant and menu tables

```
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)
);
```



menu_id	restaurant_id
1	1001
2	1001
1	1002
2	1002
1	1003
2	1003
2	1004
1	1004
1	1005

1st NF check:

- The values in each column of a restaurant table are atomic
- There are no repeating groups in the above table

2nd NF check:

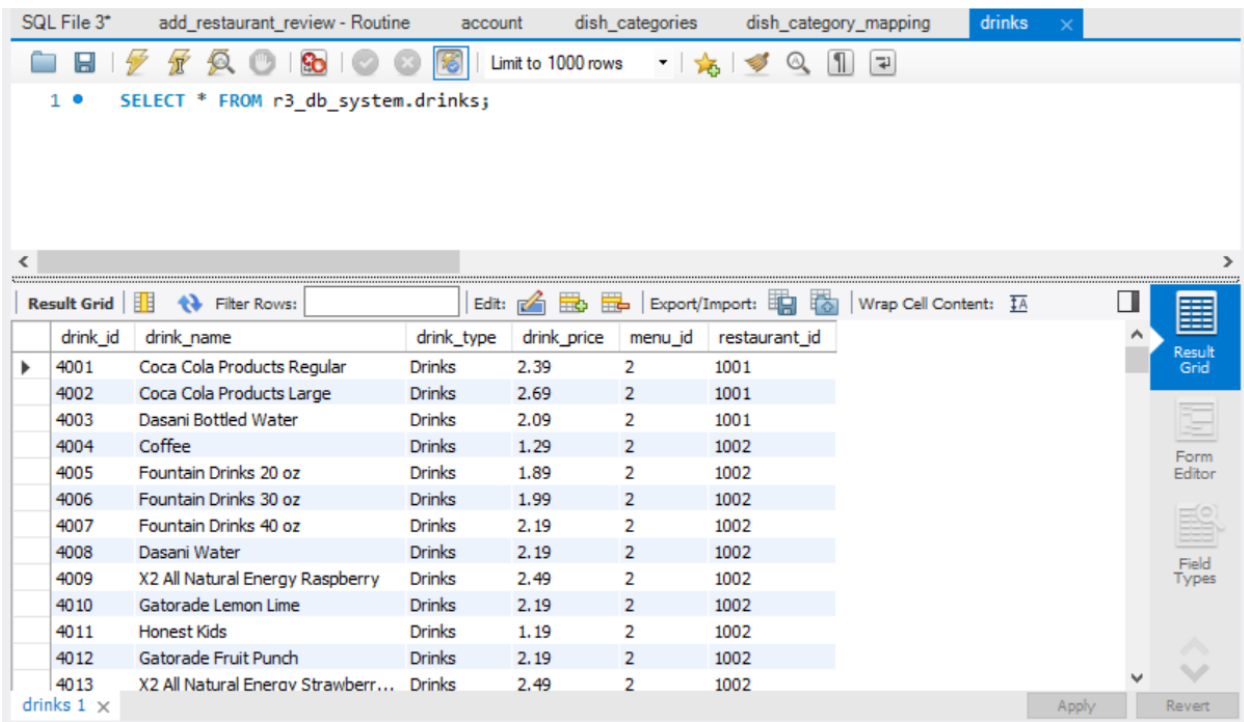
- All requirements for 1st NF are met
- No partial dependencies in the user table.
- There is No calculated data in the above table

3rd NF check:

- All the requirements for 2nd NF are met
- There is no transitive dependency for non-prime attributes

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)
);
```



drink_id	drink_name	drink_type	drink_price	menu_id	restaurant_id
4001	Coca Cola Products Regular	Drinks	2.39	2	1001
4002	Coca Cola Products Large	Drinks	2.69	2	1001
4003	Dasani Bottled Water	Drinks	2.09	2	1001
4004	Coffee	Drinks	1.29	2	1002
4005	Fountain Drinks 20 oz	Drinks	1.89	2	1002
4006	Fountain Drinks 30 oz	Drinks	1.99	2	1002
4007	Fountain Drinks 40 oz	Drinks	2.19	2	1002
4008	Dasani Water	Drinks	2.19	2	1002
4009	X2 All Natural Energy Raspberry	Drinks	2.49	2	1002
4010	Gatorade Lemon Lime	Drinks	2.19	2	1002
4011	Honest Kids	Drinks	1.19	2	1002
4012	Gatorade Fruit Punch	Drinks	2.19	2	1002
4013	X2 All Natural Energy Strawberr...	Drinks	2.49	2	1002

1st NF check:

- drink_id is the primary key for the table named drinks
- The values in each column of a restaurant table are atomic
- There are no repeating groups in the above table

2nd NF check:

- All requirements for 1st NF are met

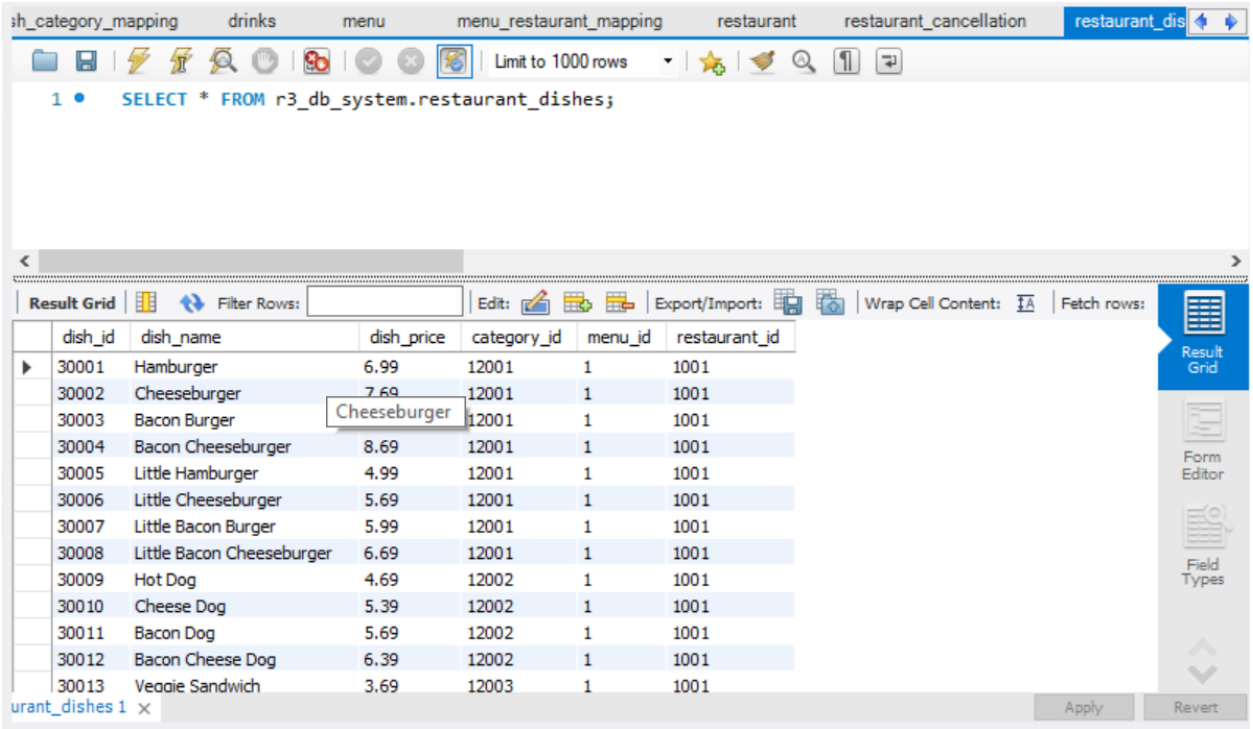
- No partial dependencies in the drinks table.
    - Non key attribute like drink_name, drink_type, drink_price depend on the primary key drink_id
- There is No calculated data in the above table

3rd NF check:

- All the requirements for 2nd NF are met
- There is no transitive dependency for non-prime attributes

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



dish_id	dish_name	dish_price	category_id	menu_id	restaurant_id
30001	Hamburger	6.99	12001	1	1001
30002	Cheeseburger	7.69	12001	1	1001
30003	Bacon Burger	[illegible]	12001	1	1001
30004	Bacon Cheeseburger	8.69	12001	1	1001
30005	Little Hamburger	4.99	12001	1	1001
30006	Little Cheeseburger	5.69	12001	1	1001
30007	Little Bacon Burger	5.99	12001	1	1001
30008	Little Bacon Cheeseburger	6.69	12001	1	1001
30009	Hot Dog	4.69	12002	1	1001
30010	Cheese Dog	5.39	12002	1	1001
30011	Bacon Dog	5.69	12002	1	1001
30012	Bacon Cheese Dog	6.39	12002	1	1001
30013	Veggie Sandwich	3.69	12003	1	1001

1st NF check:

- dish_id is the primary key for the table named restaurant_dishes
- The values in each column of a restaurant table are atomic.
- There are no repeating groups in the above table.

2nd NF check:

- All requirements for 1st NF are met
- No partial dependencies in the drinks table.
    - Non key attribute like dish_name, dish_price, dish_price depend on the primary key dish_id
    - 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 table

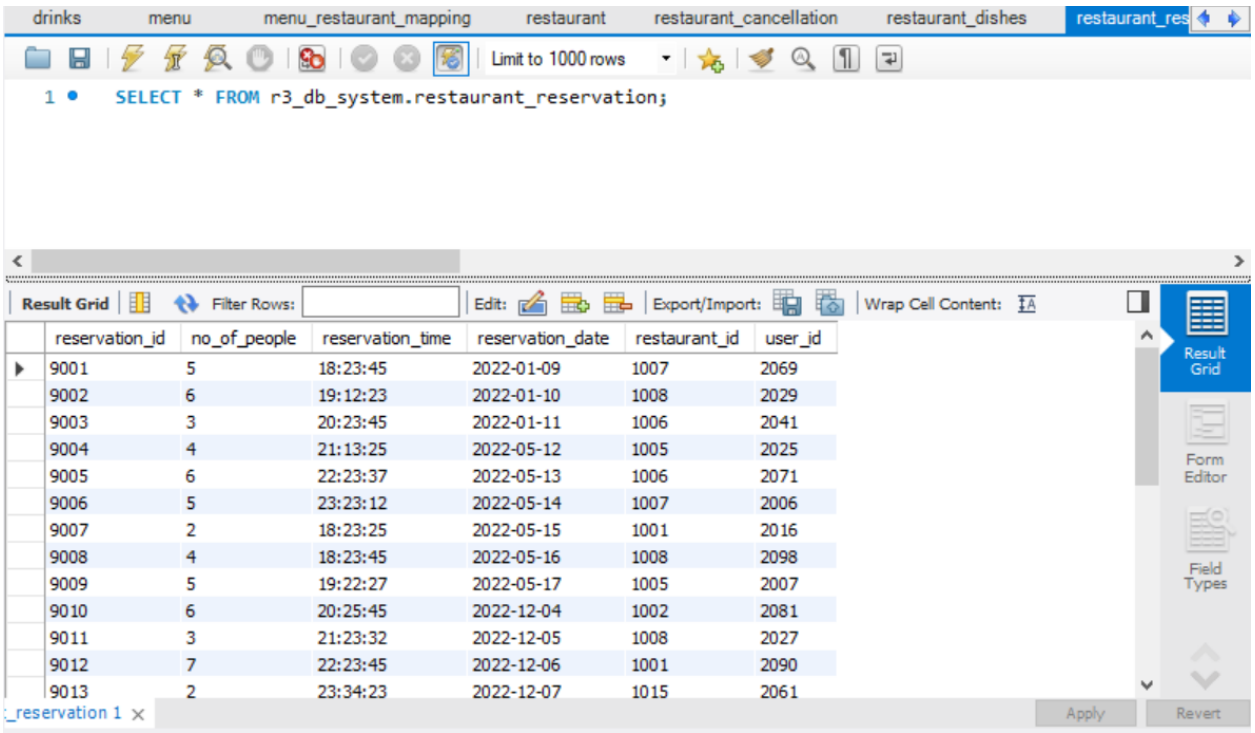
3rd NF check:

- All the requirements for 2nd NF are met
- There is no transitive dependency for non-prime attributes

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)
);



reservation_id	no_of_people	reservation_time	reservation_date	restaurant_id	user_id
9001	5	18:23:45	2022-01-09	1007	2069
9002	6	19:12:23	2022-01-10	1008	2029
9003	3	20:23:45	2022-01-11	1006	2041
9004	4	21:13:25	2022-05-12	1005	2025
9005	6	22:23:37	2022-05-13	1006	2071
9006	5	23:23:12	2022-05-14	1007	2006
9007	2	18:23:25	2022-05-15	1001	2016
9008	4	18:23:45	2022-05-16	1008	2098
9009	5	19:22:27	2022-05-17	1005	2007
9010	6	20:25:45	2022-12-04	1002	2081
9011	3	21:23:32	2022-12-05	1008	2027
9012	7	22:23:45	2022-12-06	1001	2090
9013	2	23:34:23	2022-12-07	1015	2061

1st NF check:

- reservation_id is the primary key for the table named restaurant_reservation.
- The values in each column of a restaurant table are atomic.
- There are no repeating groups in the above table.

2nd NF check:

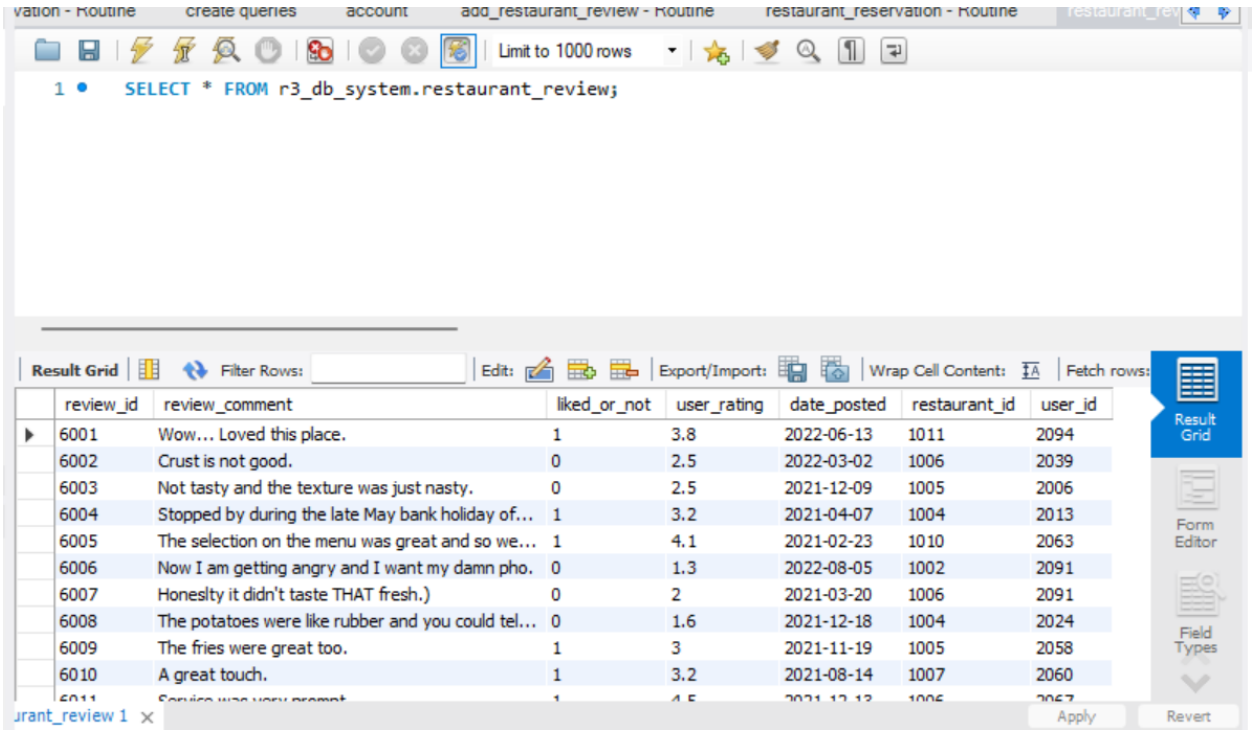
- All requirements for 1st NF are met
- No partial dependencies in the drinks table. Non key attribute like no_of_people depend on the primary key reservation_id
- There is No calculated data in the above table

3rd NF check:

- All the requirements for 2nd NF are met
- There is no transitive dependency for non-prime attributes

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)
);
```



review_id	review_comment	liked_or_not	user_rating	date_posted	restaurant_id	user_id
6001	Wow... Loved this place.	1	3.8	2022-06-13	1011	2094
6002	Crust is not good.	0	2.5	2022-03-02	1006	2039
6003	Not tasty and the texture was just nasty.	0	2.5	2021-12-09	1005	2006
6004	Stopped by during the late May bank holiday of...	1	3.2	2021-04-07	1004	2013
6005	The selection on the menu was great and so we...	1	4.1	2021-02-23	1010	2063
6006	Now I am getting angry and I want my damn pho.	0	1.3	2022-08-05	1002	2091
6007	Honeslty it didn't taste THAT fresh.)	0	2	2021-03-20	1006	2091
6008	The potatoes were like rubber and you could tel...	0	1.6	2021-12-18	1004	2024
6009	The fries were great too.	1	3	2021-11-19	1005	2058
6010	A great touch.	1	3.2	2021-08-14	1007	2060

1st NF check:

- review_id is the primary key for the table named restaurant_review.
- The values in each column of a restaurant table are atomic.
- There are no repeating groups in the above table.

2nd NF check:

- All requirements for 1st NF are met

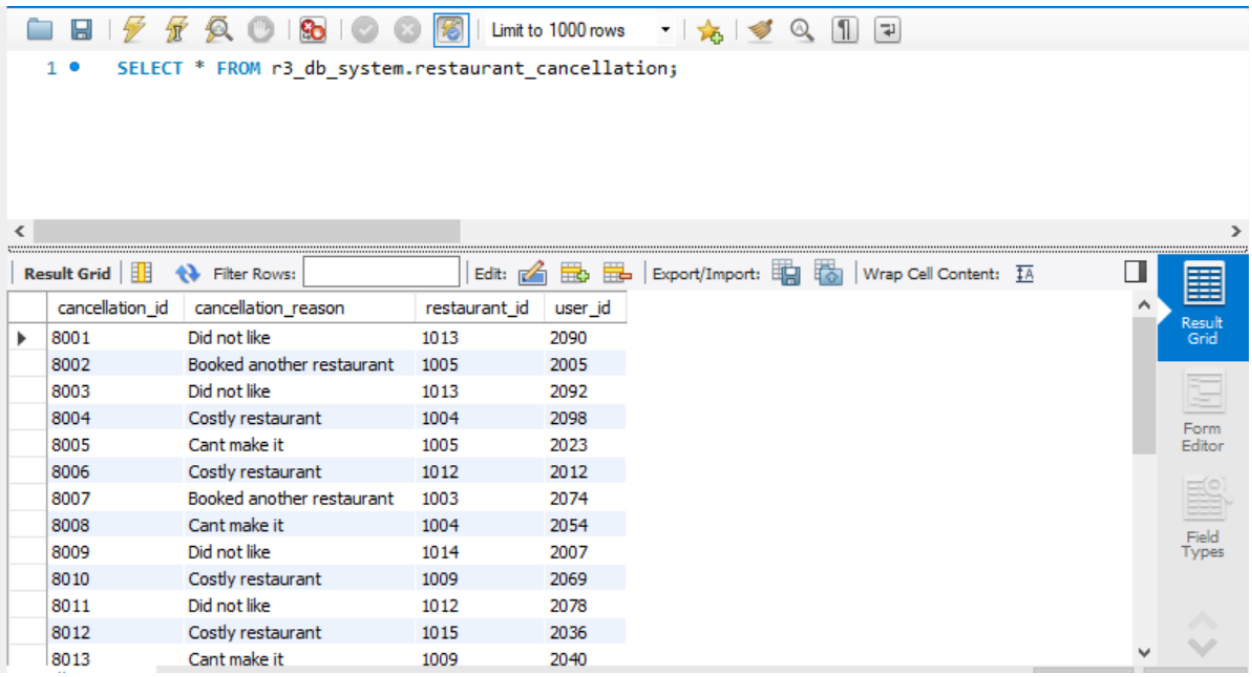
- No partial dependencies in the drinks table. Non key attribute like review_comment, review_like, user_rating, date_posted depend on the primary key review_id
- There is No calculated data in the above table

3rd NF check:

- All the requirements for 2nd NF are met
- There is no transitive dependency for non-prime attributes

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)
);
```



cancellation_id	cancellation_reason	restaurant_id	user_id
8001	Did not like	1013	2090
8002	Booked another restaurant	1005	2005
8003	Did not like	1013	2092
8004	Costly restaurant	1004	2098
8005	Cant make it	1005	2023
8006	Costly restaurant	1012	2012
8007	Booked another restaurant	1003	2074
8008	Cant make it	1004	2054
8009	Did not like	1014	2007
8010	Costly restaurant	1009	2069
8011	Did not like	1012	2078
8012	Costly restaurant	1015	2036
8013	Cant make it	1009	2040

1st NF check:

- cancellation_id is the primary key for the table named restaurant_cancellation.
- The values in each column of a restaurant table are atomic.

- There are no repeating groups in the above table.

2nd NF check:

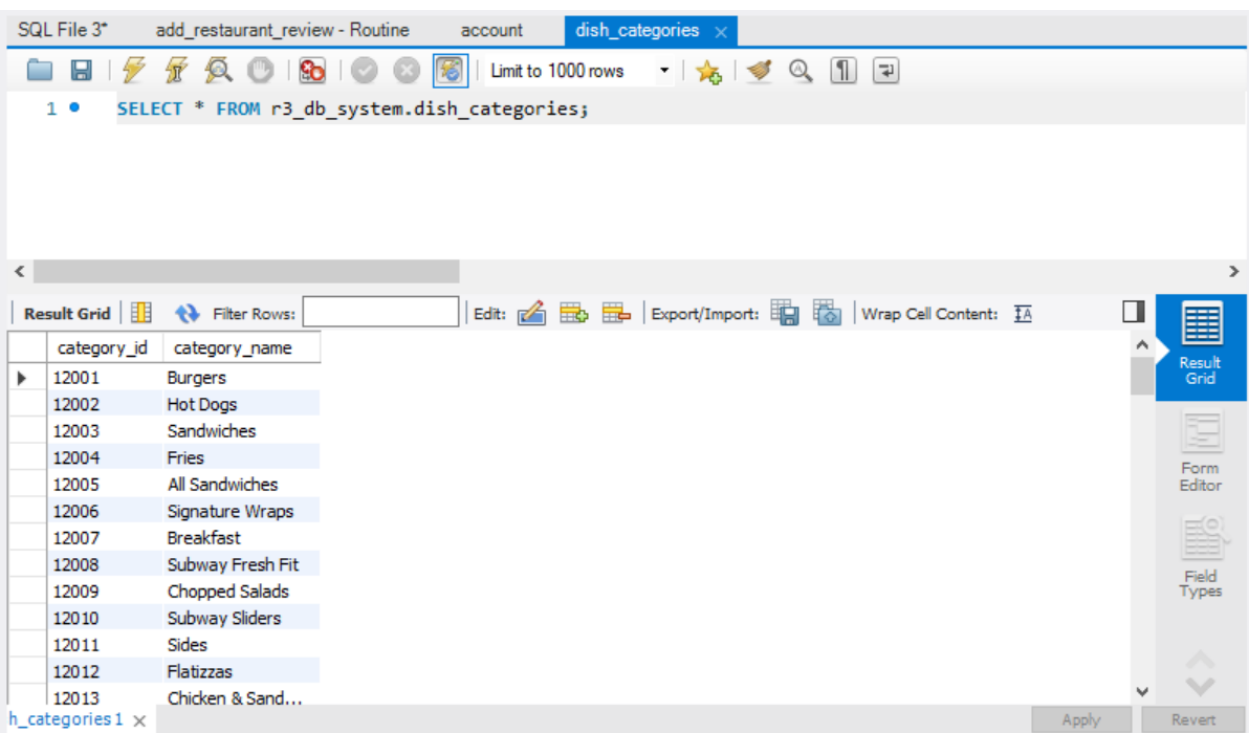
- All requirements for 1st NF are met
- No partial dependencies in the drinks table. Non key attribute like cancellation_reason depend on the primary key cancellation_id
- There is No calculated data in the above table

3rd NF check:

- All the requirements for 2nd NF are met
- There is no transitive dependency for non-prime attributes

Dish Categories Table: Created when restaurant_dishes table was broken into 2 tables (restaurant_dishes, dish_categories)

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



1st NF check:

- categories_id is the primary key for the table named dish_categories

- The values in each column of a restaurant table are atomic
- There are no repeating groups in the above table

2nd NF check:

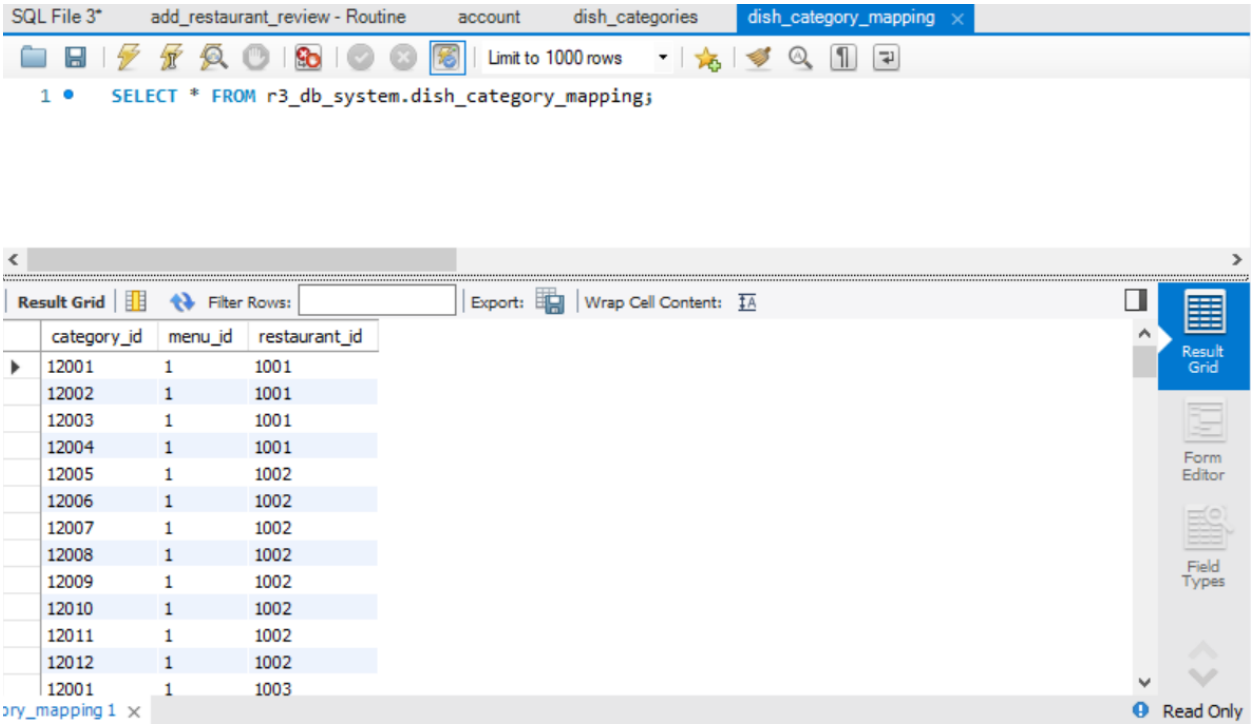
- All requirements for 1st NF are met
- No partial dependencies in the user table.
    - Non key attribute like category_name depend on the primary key category_id
- There is No calculated data in the above table

3rd NF check:

- All the requirements for 2nd NF are met
- There is no transitive dependency for non-prime attributes

Dish Categories Mapping Table: Created because of many to many mapping between dish_categories and restaurant_dishes table

```
CREATE TABLE dish_categories_mapping(
categories_id INT,
menu_id INT,
restaurant_id INT,
FOREIGN KEY(categories_id) REFERENCES dish_categories(categories_id),
FOREIGN KEY(restaurant_id) REFERENCES restaurant (restaurant_id),
FOREIGN KEY(menu_id) REFERENCES menu(menu_id)
);
```



category_id	menu_id	restaurant_id
12001	1	1001
12002	1	1001
12003	1	1001
12004	1	1001
12005	1	1002
12006	1	1002
12007	1	1002
12008	1	1002
12009	1	1002
12010	1	1002
12011	1	1002
12012	1	1002
12001	1	1003

1st NF check:

- The values in each column of a restaurant table are atomic
- There are no repeating groups in the above table

2nd NF check:

- All requirements for 1st NF are met
- No partial dependencies in the user table.
- There is No calculated data in the above table

3rd NF check:

- All the requirements for 2nd NF are met
- There is no transitive dependency for non-prime attributes

Views for all of your use-cases

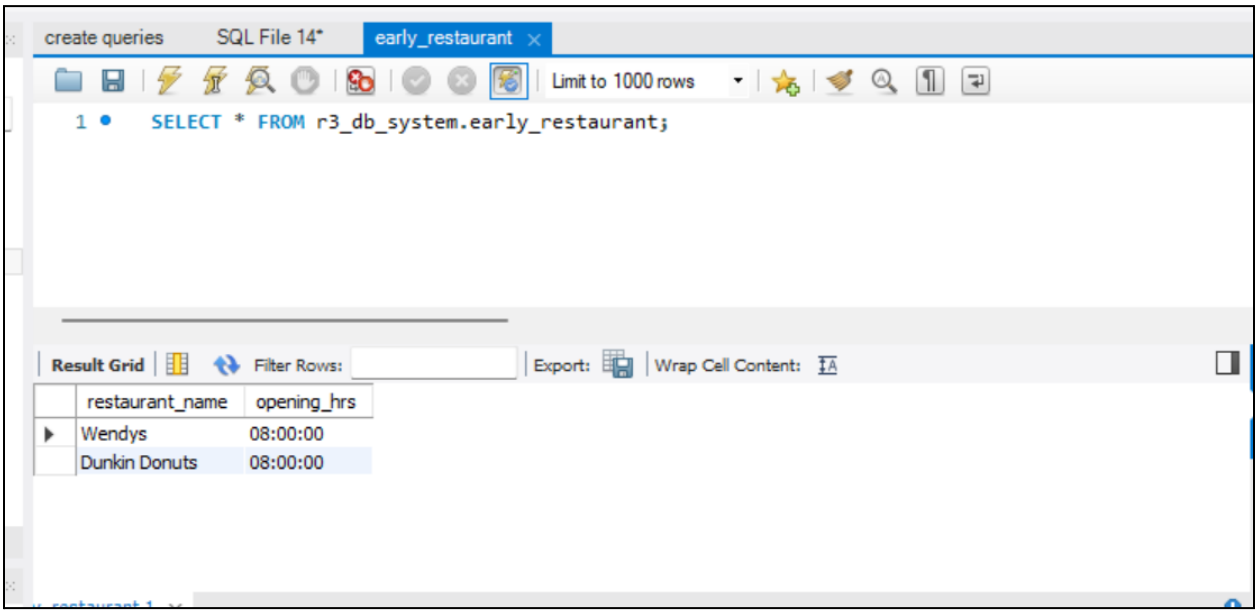
Use Case 1: Search for a Restaurant that opens at 8 am with a rating above 4.

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 View:-

```
CREATE VIEW Early_restaurant AS
SELECT distinct(restaurant_name), r.opening_hrs
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;
```

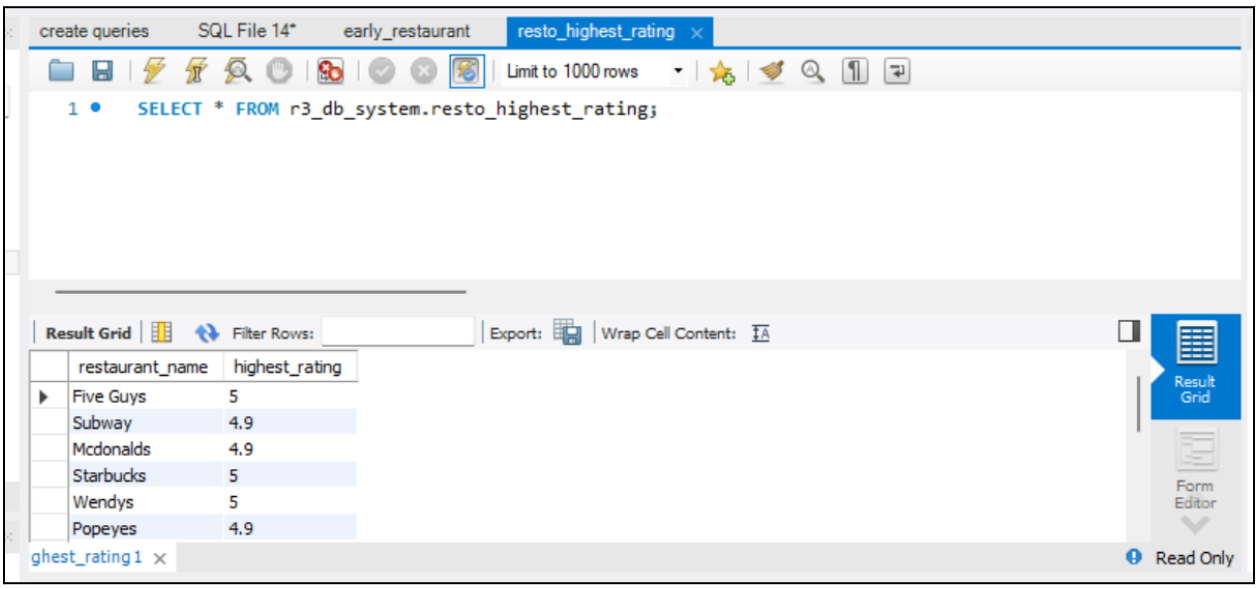


Use Case 2: Check the restaurant with the best User Rating.

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 View:-

```
CREATE VIEW Resto_highest_rating AS
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;
```

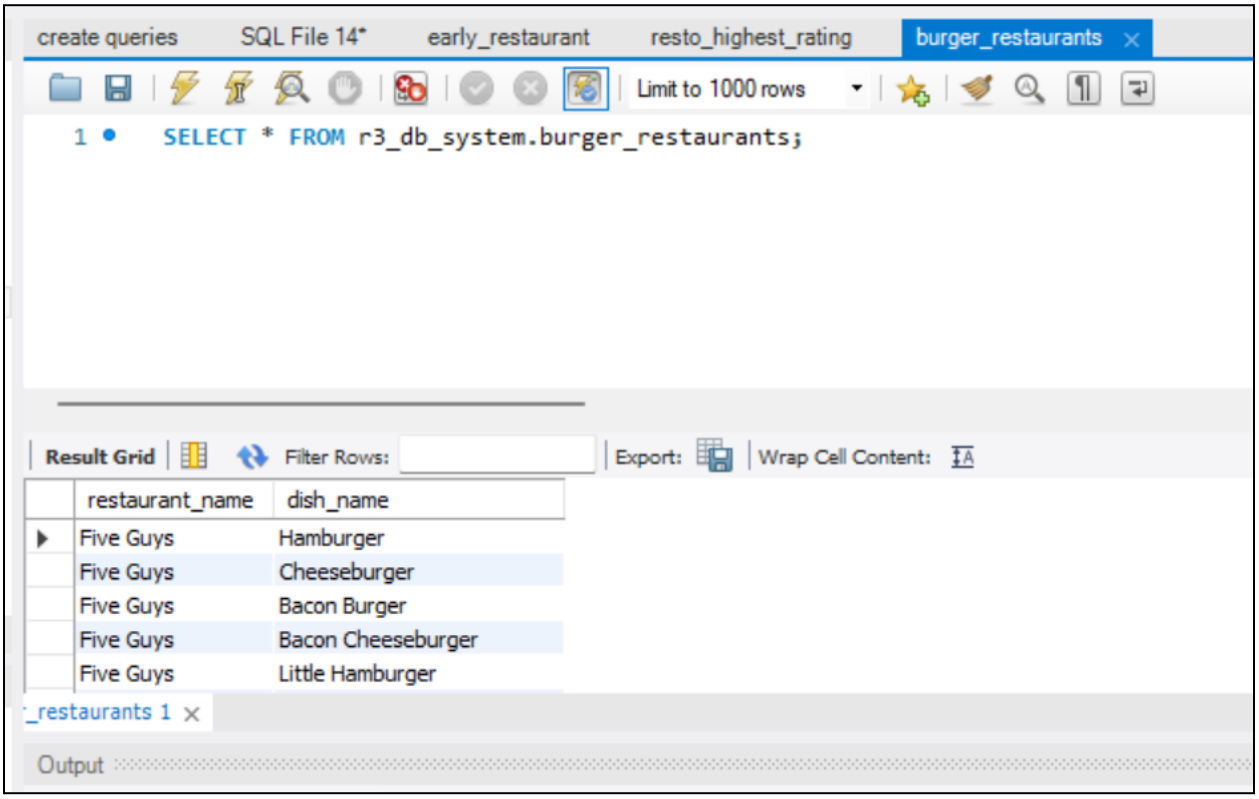


Use Case 3: Search for restaurants offering Different Burgers in the zip code 2115

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 Burgers
System Responses – Displays all the restaurants offering Burgers 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 View:

```
CREATE VIEW Burger_restaurants AS
SELECT distinct(r.restaurant_name), d.dish_name
FROM restaurant r INNER JOIN restaurant_dishes d
ON r.restaurant_id=d.restaurant_id
WHERE r.zipcode = '2115' AND d.category_id in (select category_id from dish_categories
                                                where category_name like '%Burger%');
```



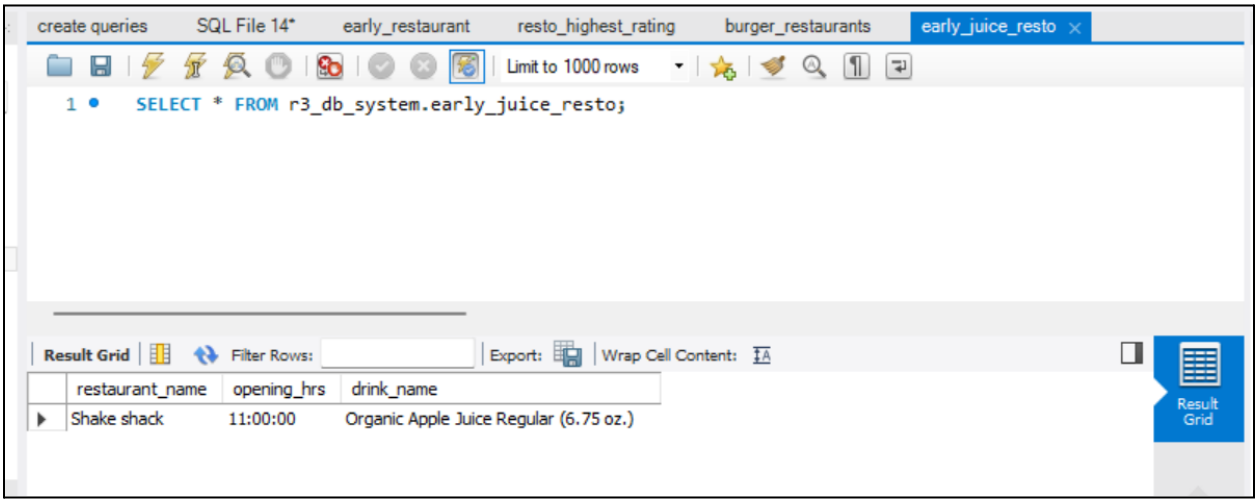
Use Case 4: View a restaurant that offers Organic Apple Juice Regular (6.75 oz.) and opens at 11 am

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 View:

```
 CREATE VIEW Early_juice_resto AS
 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.) ';
```

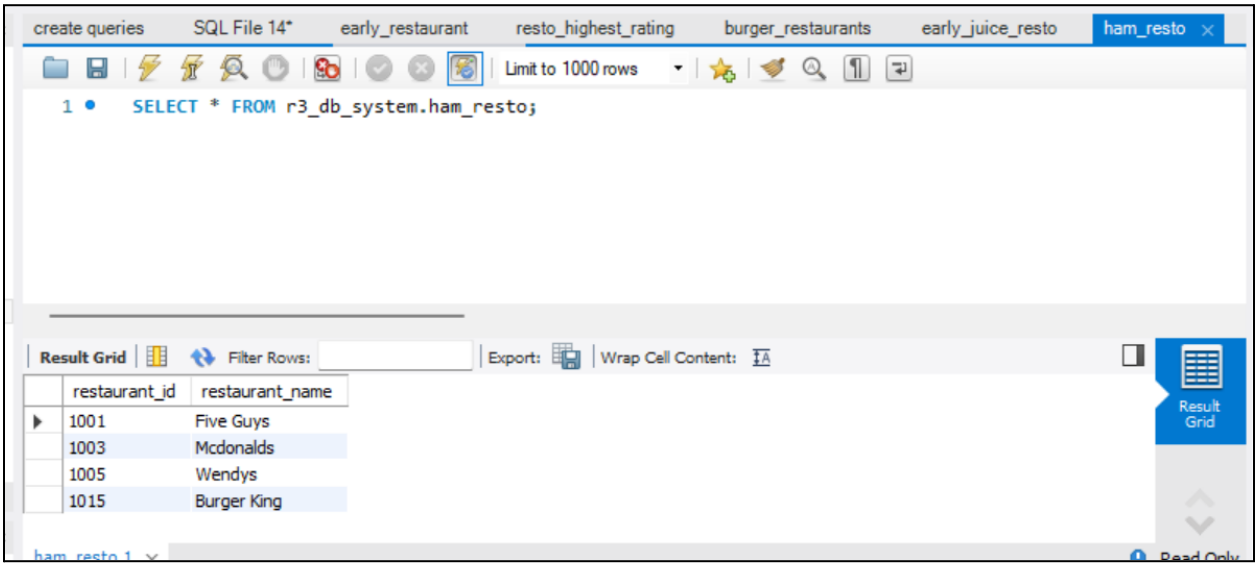


Use Case 5: Search for a restaurant that serves Hamburgers.

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 View:-

```
CREATE VIEW Ham_resto AS
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 ';
```



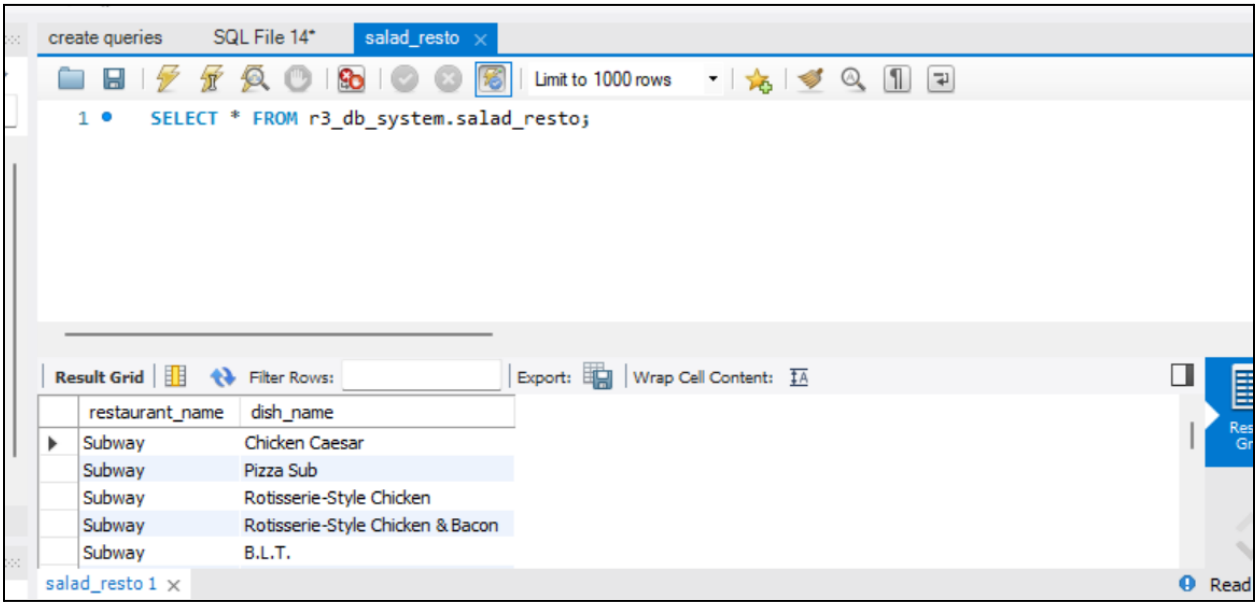
Use case 6: Search for a restaurant offering Salads.

Description: The user searches for different restaurants offering salads
Actor: User
Precondition: The user needs to log in to his account
Steps:
Actor action: The user requests the Details of Restaurants offering salads
System Responses: Details of all the Restaurants offering salads 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 View:-

```
CREATE VIEW Salad_resto AS
```

```
SELECT distinct(r.restaurant_name), d.dish_name
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 '%Salad%');
```



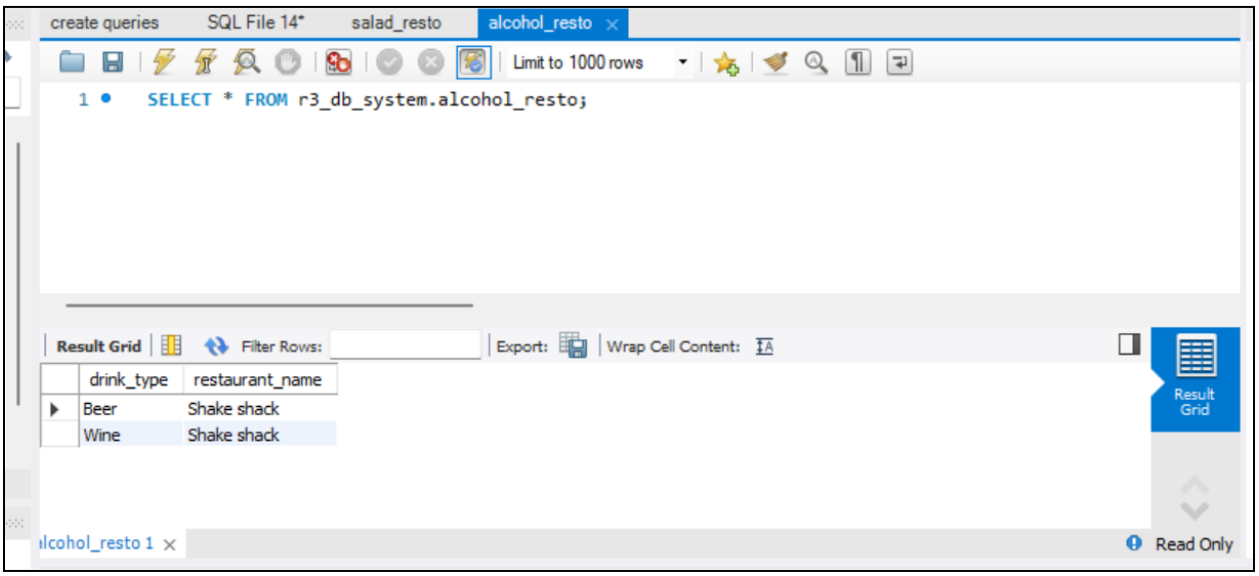
Use Case 7: View the restaurant which serves alcoholic (Beer, wine) drinks.

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 View:

```
CREATE VIEW alcohol_resto AS
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');
```

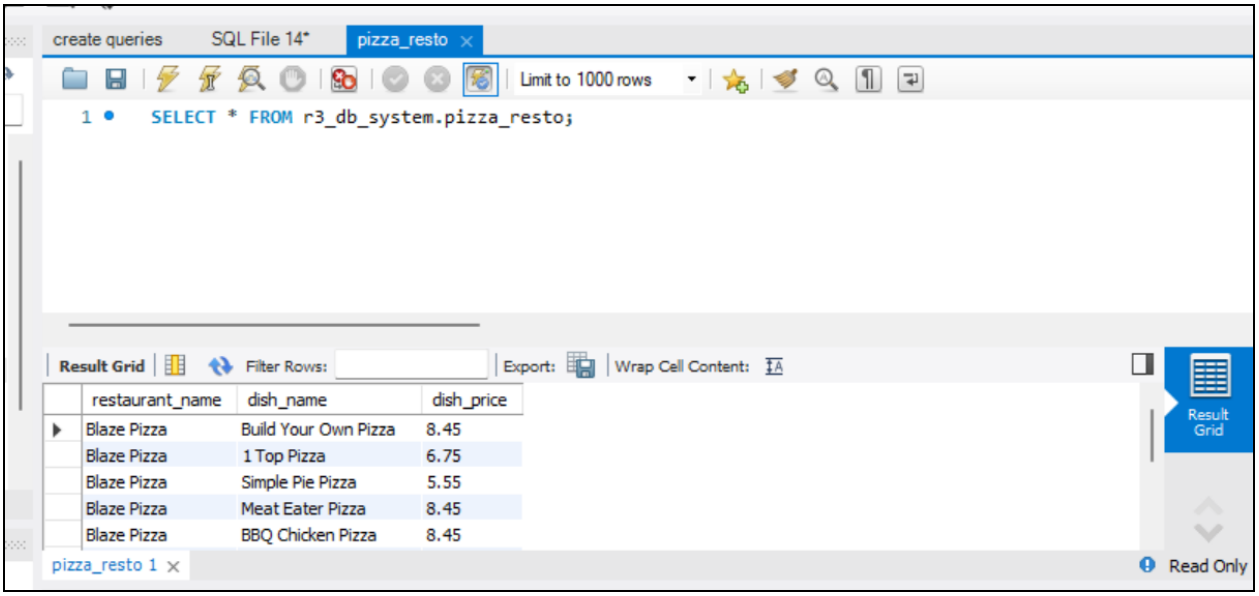


Use Case 8: View a restaurant offering Pizza with a specific budget (say less than $20)

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 View:-

```
CREATE VIEW Pizza_resto AS
SELECT distinct(r.restaurant_name), d.dish_name, d.dish_price
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 '%Pizza%') and r.budget_for_2<=20;
```

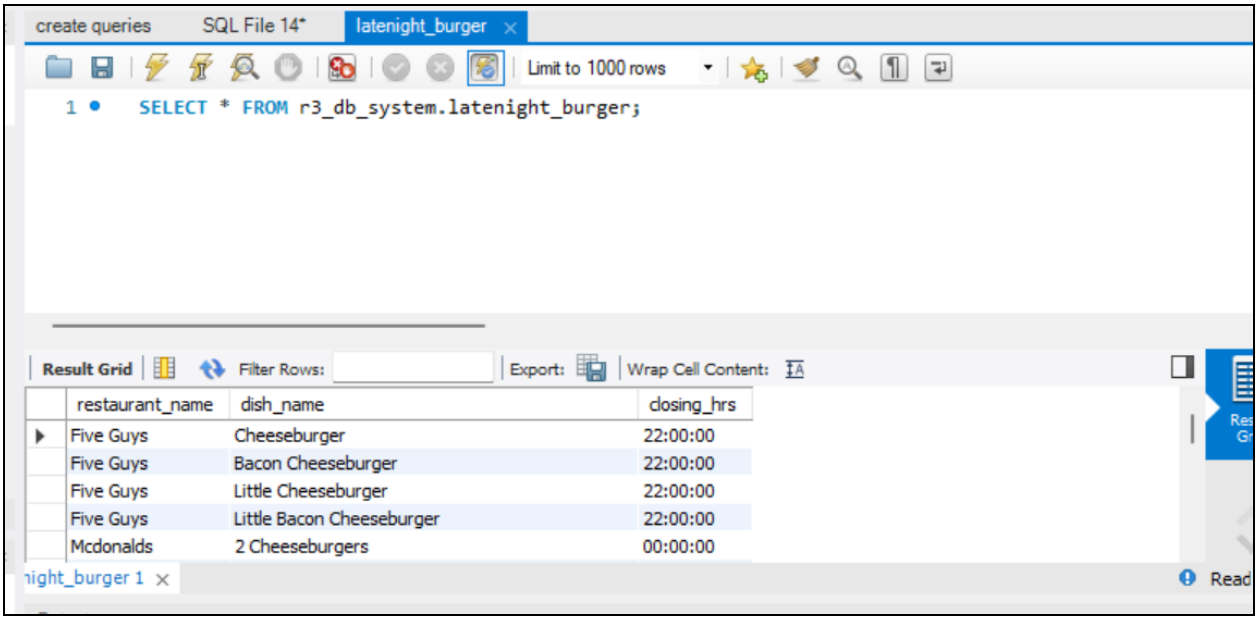


Use Case 9: View a restaurant that serves Cheeseburger and is open till 10 pm

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 View:

```
CREATE VIEW Latenight_burger AS
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'
```

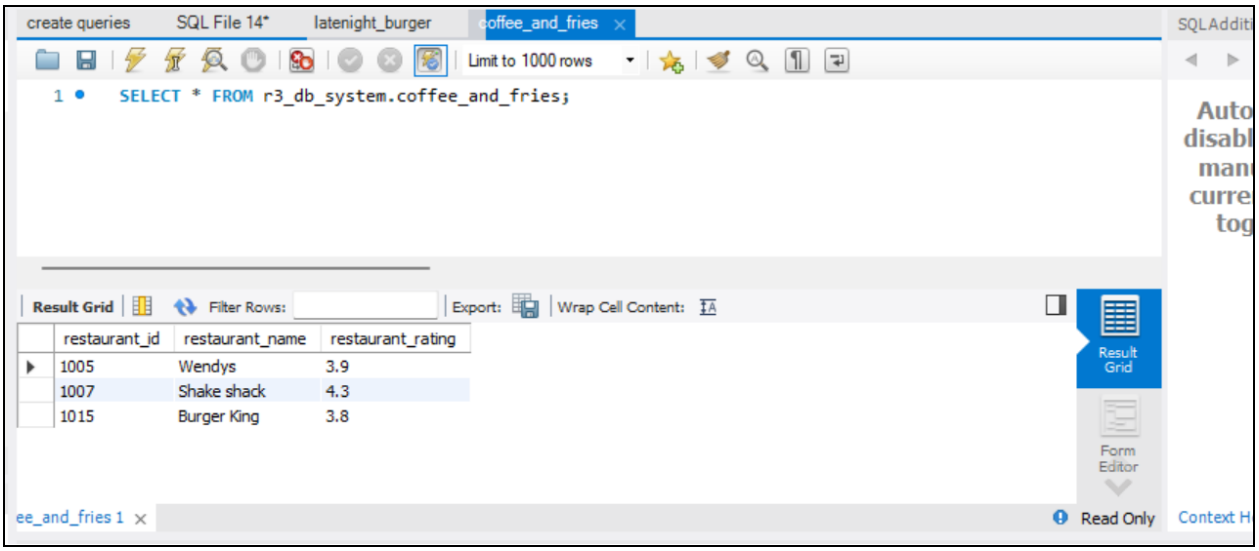


Use case 10: View the restaurant's offerings of Coffee and Fries

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 View:-

```
CREATE VIEW coffee_and_fries AS
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%';
```

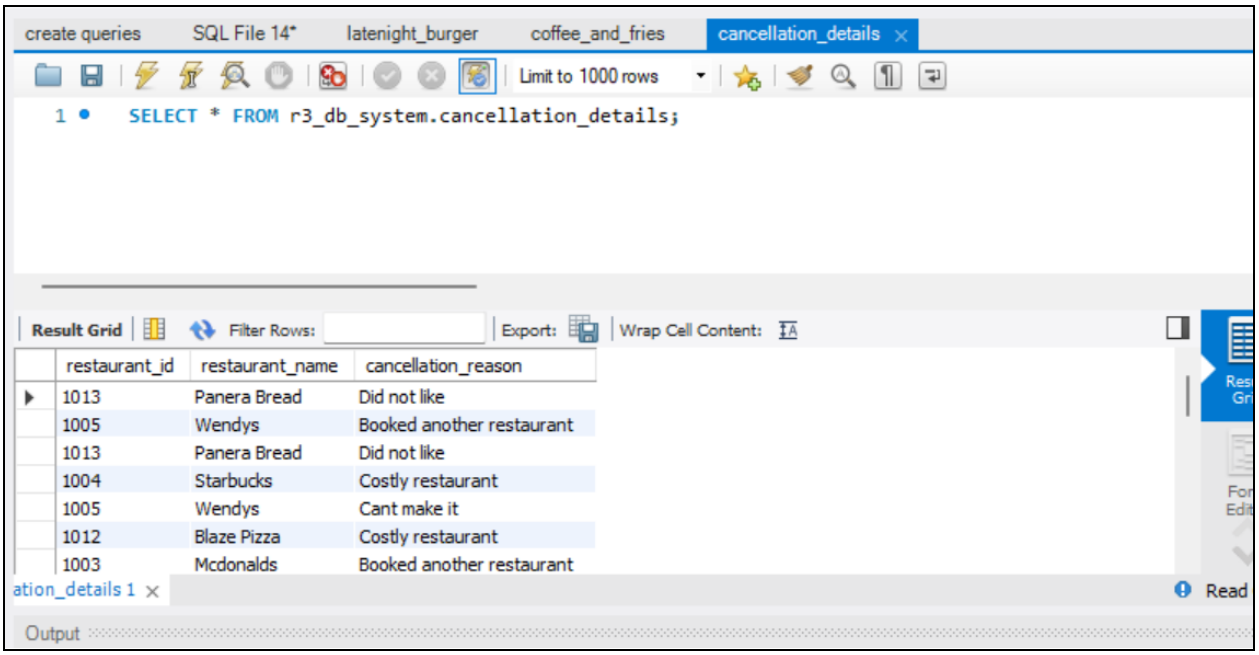


Use case 11: View cancellations done for a restaurant and their reasons

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 View:-

```
CREATE VIEW Cancellation_details AS
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;
```



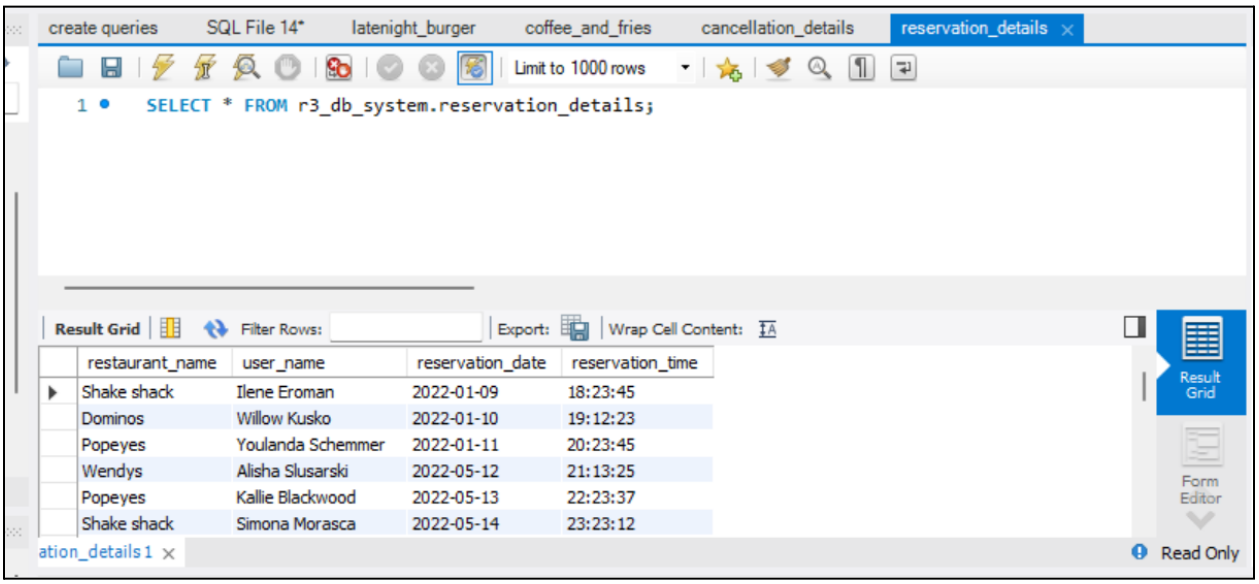
Use Case 12: What are the restaurant details, user details, and reservation details where the users reserved a table?

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 View:-

```
CREATE VIEW Reservation_details AS
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;

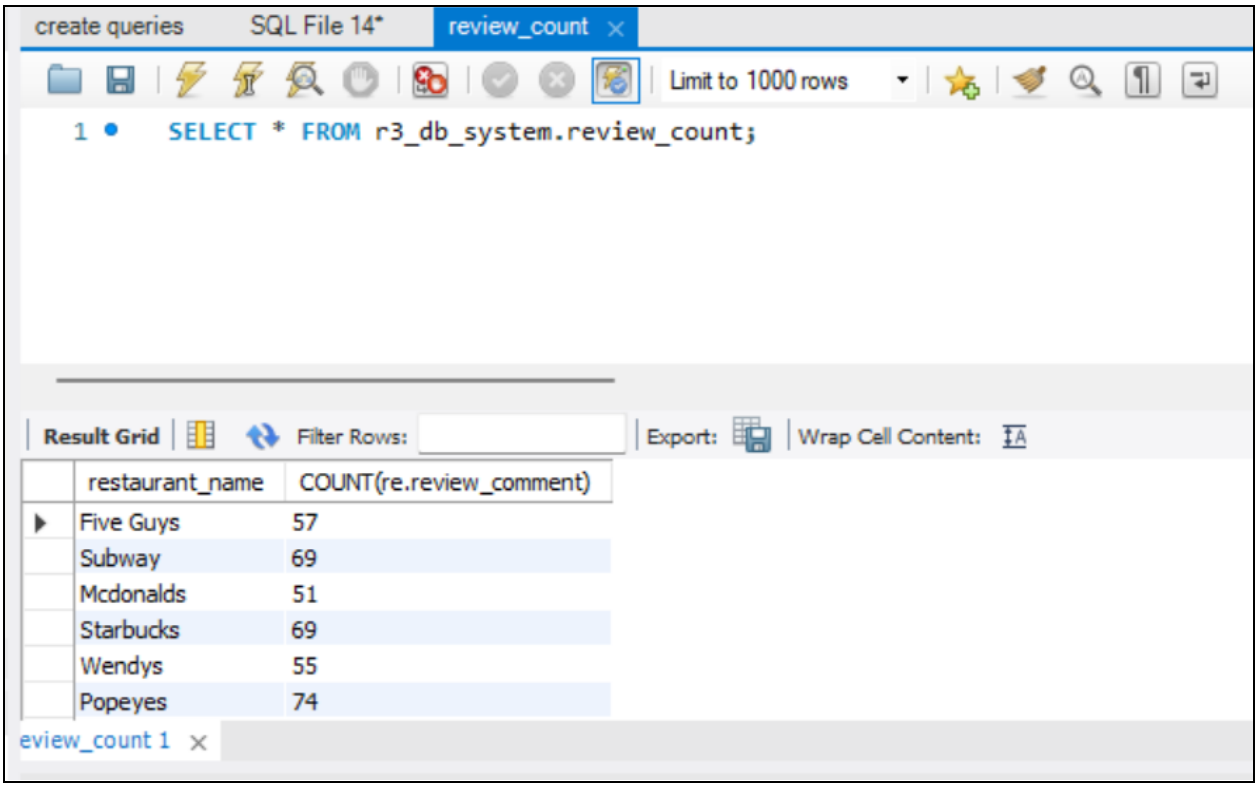


Use Case 13: How many reviews have a restaurant received to date?

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 View:-

```
CREATE VIEW review_count AS
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;
```

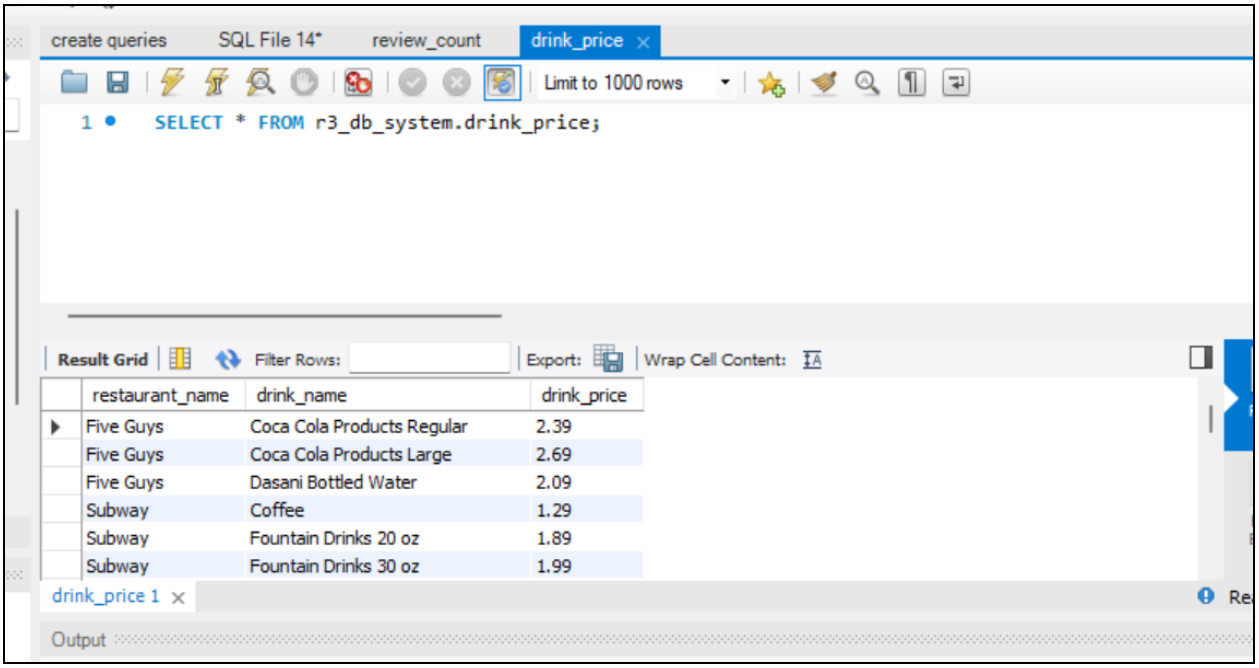


Use Case 14: Restaurants with drinks below 3$?

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 View:-

```
CREATE VIEW drink_price AS
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;
```

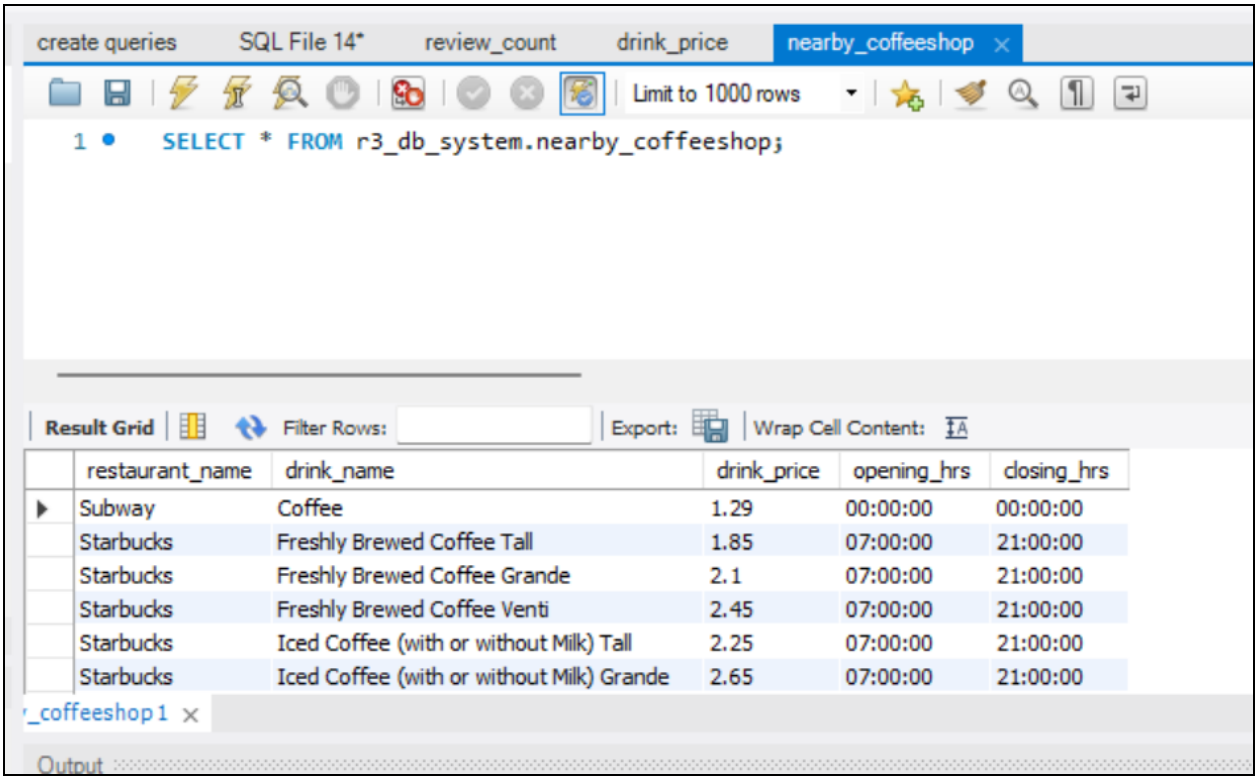


Use Case 15: Restaurants that serve coffee in zip code 02115?

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 View:-

```
CREATE VIEW nearby_CoffeeShop AS
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';
```



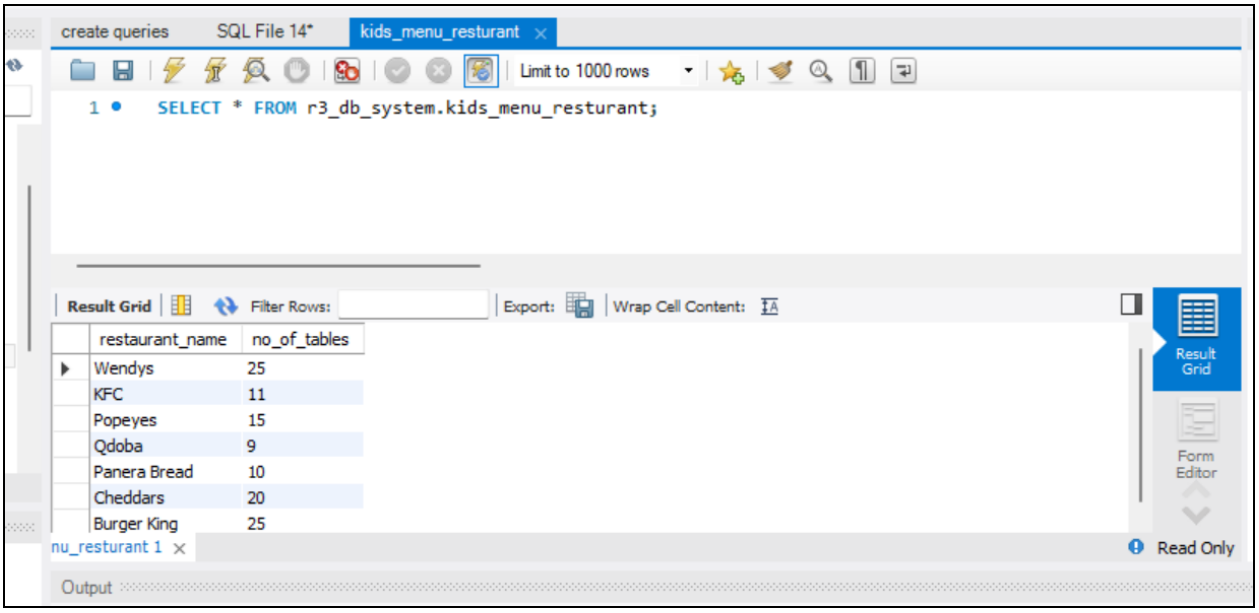
restaurant_name	drink_name	drink_price	opening_hrs	closing_hrs
Subway	Coffee	1.29	00:00:00	00:00:00
Starbucks	Freshly Brewed Coffee Tall	1.85	07:00:00	21:00:00
Starbucks	Freshly Brewed Coffee Grande	2.1	07:00:00	21:00:00
Starbucks	Freshly Brewed Coffee Venti	2.45	07:00:00	21:00:00
Starbucks	Iced Coffee (with or without Milk) Tall	2.25	07:00:00	21:00:00
Starbucks	Iced Coffee (with or without Milk) Grande	2.65	07:00:00	21:00:00

Use Case 16: Check if the table is available for Reservation in a restaurant with kids menu

Description:- The user checks if a table is available for reservation in a restaurant with kids menu
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 View:-

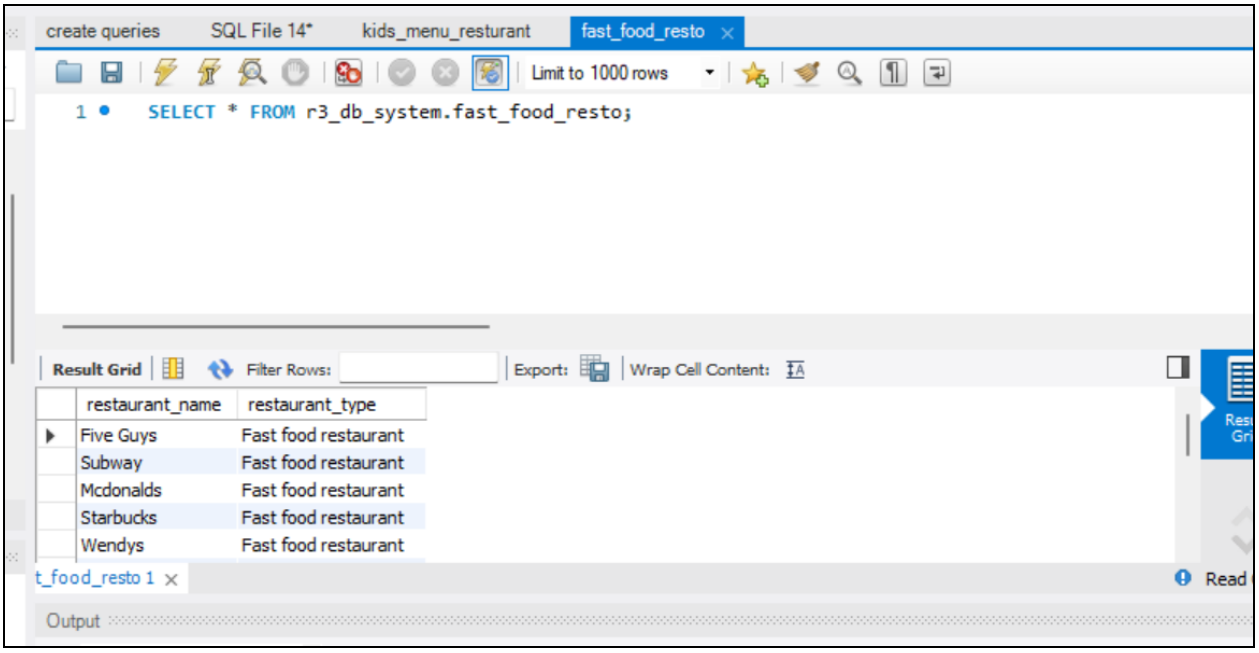
```
CREATE VIEW Kids_menu_resturant AS
SELECT distinct(r.restaurant_name),r.no_of_tables
FROM restaurant r INNER JOIN restaurant_dishes d
ON r.restaurant_id=d.restaurant_id
WHERE d.category_id in (select category_id from dish_categories
                        Where category_name like "%kid%");
```



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

SQL View:

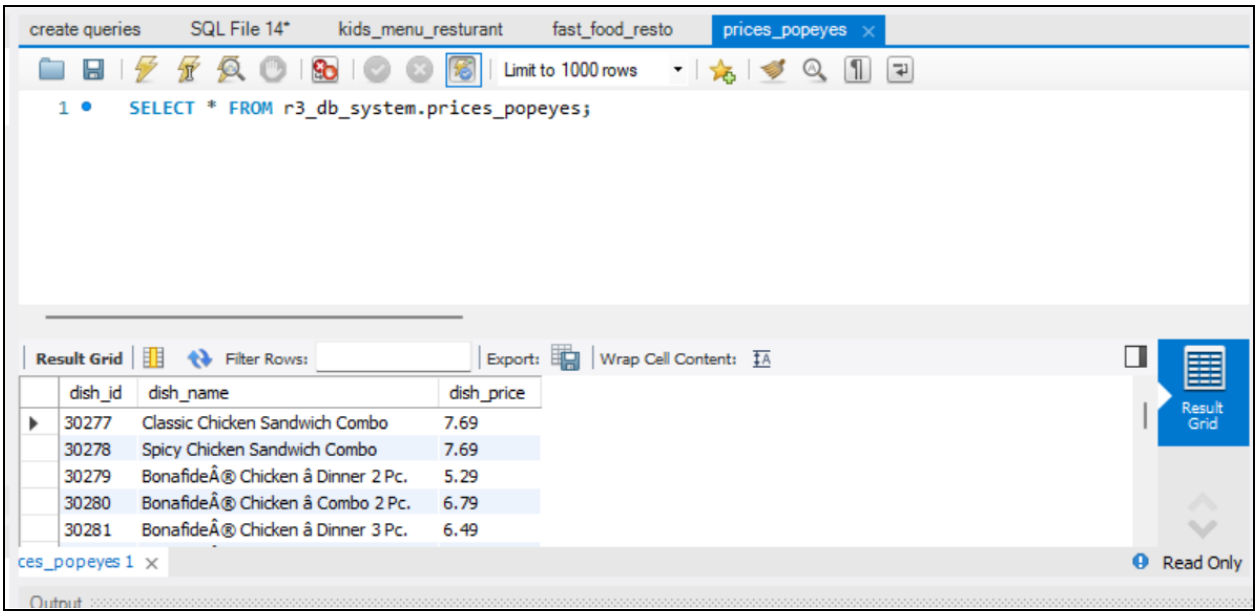
```
CREATE VIEW fast_food_resto AS
SELECT r.restaurant_name, rt.restaurant_type
FROM restaurant r INNER JOIN restaurant_type_mapping rtm inner join restaurant_type rt
on r.restaurant_id=rtm.restaurant_id
where restaurant_type='Fast food restaurant';
```



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 View:

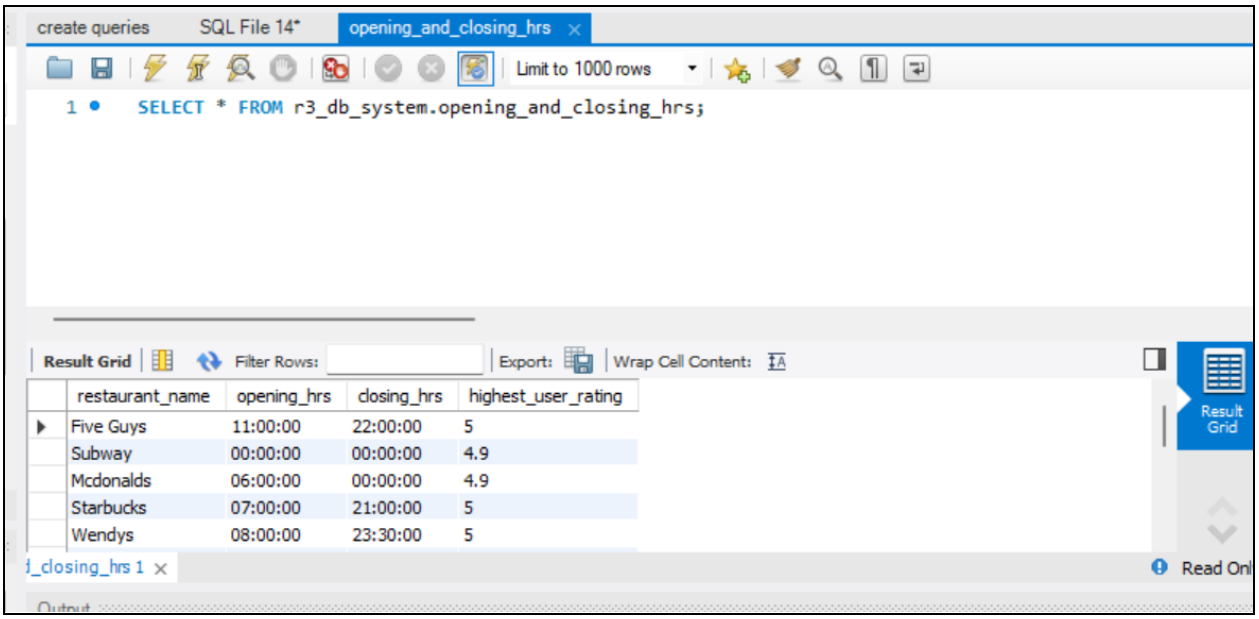
```
CREATE VIEW Prices_Popeyes AS
SELECT dt.dish_id,dt.dish_name, dt.dish_price
FROM restaurant_dishes dt INNER JOIN restaurant r
ON  dt.restaurant_id = r.restaurant_id
WHERE  r.restaurant_name='Popeyes';
```



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 View:

```
CREATE VIEW Opening_and_Closing_hrs AS
SELECT distinct r.restaurant_name, r.opening_hrs, r.closing_hrs, max(rv.user_rating) as
highest_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 View:-

```
CREATE VIEW Cancellation_details AS
SELECT r.restaurant_id, r.restaurant_name, count(c.cancellation_id)
FROM restaurant r
inner join restaurant_cancellation c
on r.restaurant_id = c.restaurant_id
group by c.restaurant_id;
```