Food Recommendation System

Final Project ISM6218.901F21.94685 Advanced Database Management

Team 10:

Amulya Alwala
Suryanarayana Aneesh Prayaga
Raj Phanindra Kakarla
Nicolas Cinera Pardo

Topic Area	Description	Points	Team-Members
Database Design	Logical database design with ERD and normalization to control redundancy, integrity constraints for data quality, use case identification. Defining the ERD to accommodate future updates to the project	Default: 30	Amulya, Raj Phanindra, Nicholas
Query Writing	Writing SQL queries to provide recommendations to the user based on the identified use cases, database programming for stored procedures included in some queries	Default: 25	Raj, Amulya, Nicolas, Aneesh
Performance Tuning	Performed performance optimization using non clustered indexing.	Default: 25	Amulya, Raj, Aneesh
Other Topics	We have worked on data visualization in this section	Default: 20	Raj, Nicolas, Amulya, Aneesh

Executive Summary:

Due to heavy information overloads triggered by the Internet, extracting/finding valuable information becomes increasingly difficult. In this context, recommender systems became an effective tool to extract useful information and deliver it in an efficient way. A recommender system predicts the preferences of users for unrated items and recommends new items to users. Along with the benefits of recommender systems, developing new recommendation approaches and including them in different fields rise extremely.
Following are some approaches that can be used to develop recommender systems.

Collaborative filtering recommender systems (CF):

Collaborative Filtering became one of the most widely prevalent techniques of recommender systems, Netflix recommender system uses this technique. The basic idea of CF is to use the wisdom of the crowd for making recommendations. First, a user rates some given items in an implicit or explicit fashion. Then, the recommender identifies the nearest neighbors whose tastes are like those of a given user and recommends items that the nearest neighbors have liked. CF is usually implemented based on the following approaches: user-based, item-based, model-based approaches and matrix factorization.

Content-based recommender systems (CB):

These systems can make a personalized recommendation by exploiting information about available item descriptions (e.g., genre and director of movies) and user profiles describing what the users like. The main task of a CB system is to analyze the information regarding user preferences and item descriptions consumed by the user, and then recommend items based on this information. There are different approaches applied to make recommendations to users, such as Information Retrieval or Machine Learning algorithms.

The initial idea of this project is to implement a food recommender system based on collaborative filtering, by creating a suitable database with good amount of data and implementing cosine similarity concept explained in class in python. Since this approach was time consuming, due to the lack of proper data, we had to stick to some simple use cases and content-based filtering approach. The idea is to scale this project further using the above-mentioned approach. In this project we intend to design a simple context-based food recommendation system that has the following basic use cases

- Recommending best/top rated food items in the user's location, based on cuisine preferences.
- Recommending best restaurants in the user's location based on ratings.
- Recommending the most loved food items from all cuisines to a new user

- Recommending the best restaurants to a new user based on food item ratings

- Future scope: Recommending a cuisine based on food type preferences, for instance users searching for rice item preferences will be suggested Thai, Chinese or Indian cuisines and restaurants offering them based on location.

This system enables the users to pick suitable restaurants near them based on different search criteria and individual preferences.

Food Recommender System Design and Flow:

The major entities considered to design the food recommender system to facilitate the above-mentioned use cases are User, Cuisine, Food Type, Restaurant, ratings and offers. Since it is context-based filtering, there needs to be existing data on user location and preferences of the users, which was quantified through ratings in the ratings as well as FoodType entities.

All the data related to the user like first name, last name, location, gender, and email were included in the user table. Restaurant related details like restaurant name, restaurant location which is needed to show restaurant results based on location are included in the restaurant table. Cuisine and food type details are added to separate tables to conform to normalization. The idea is to link restaurants table to cuisine table, as each restaurant can offer multiple cuisines. Each cuisine can in turn have multiple food items, which were included in the FoodType Table and linked to cuisine table. The FoodType table also has food rating column/attribute to enable users to add food item/ dish specific ratings, which can further be used to recommend food items and restaurants offering them. The ratings table has restaurant ratings which can be used to recommend best restaurants in the user's location, it is linked to the restaurant table and the user table as each restaurant can have multiple ratings and each user can rate a multiple restaurant. The offers table was included to enable another use case which we intend to scale this project in future, recommending restaurants in user location based on offers along with intended user preferences.
The entities that were identified in accordance with the afore-mentioned details and use cases are listed below.

Entities Identified

- **User**
- **Restaurant**
- **Cuisine**
- **FoodType**
- **Ratings**
- **Offers**

ENTITIES WITH ATTRIBUTES

- **User**
  - First_name
  - Last_name
  - user_location
  - gender
  - email
  - user_ID: Primary Key

- **Restaurant**
  - restaurant_name
  - restaurant_location
  - restaurant_ID: Primary key

- **Cuisine**
  - cuisine_name
  - cuisine_ID: Primary key

- **FoodType**
  - item_name
  - item_ID
  - food_rating: Primary key

- **Ratings**
  - restaurant_rating
  - rating_ID: Primary key
- **Offers**
-
  - offer_type
  - offer_validity
  - offer_ID: Primary key

Business Rules

⇒ Each user can give one rating to a single restaurant, but each user may rate multiple restaurants or may not rate any
⇒ Each user can rate multiple dishes/food items or may not rate any
⇒ Each restaurant can have multiple ratings from different users
⇒ Each restaurant can offer at least one or multiple cuisines.
⇒ Each cuisine can have at least one multiple food items/dishes

Other Assumptions:

⇒ Restaurants have only one cuisine offered, due to unavailability of data, above business rules were designed to enable future scaling of the application

Entity Relationship Diagram for Food Recommender System

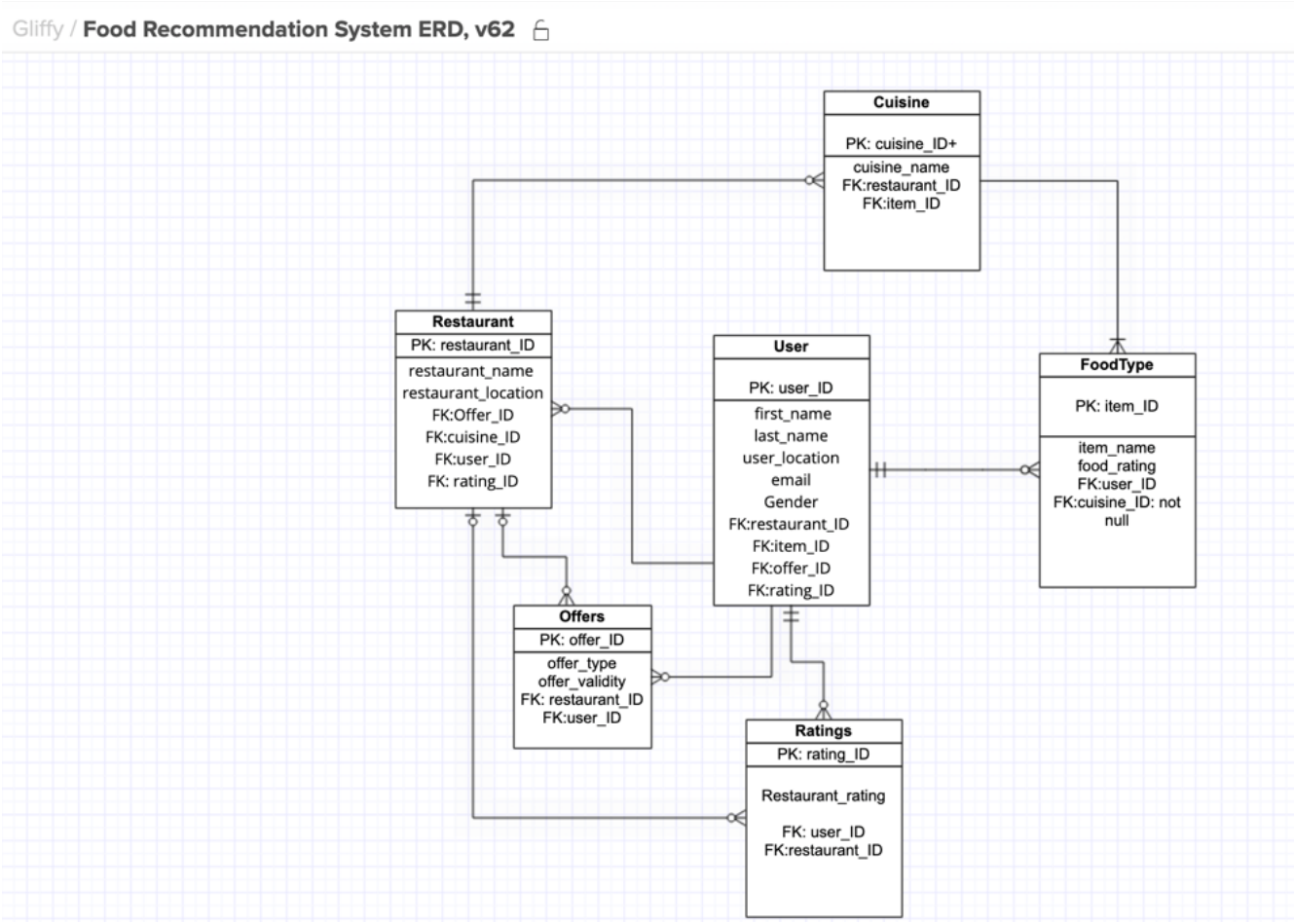
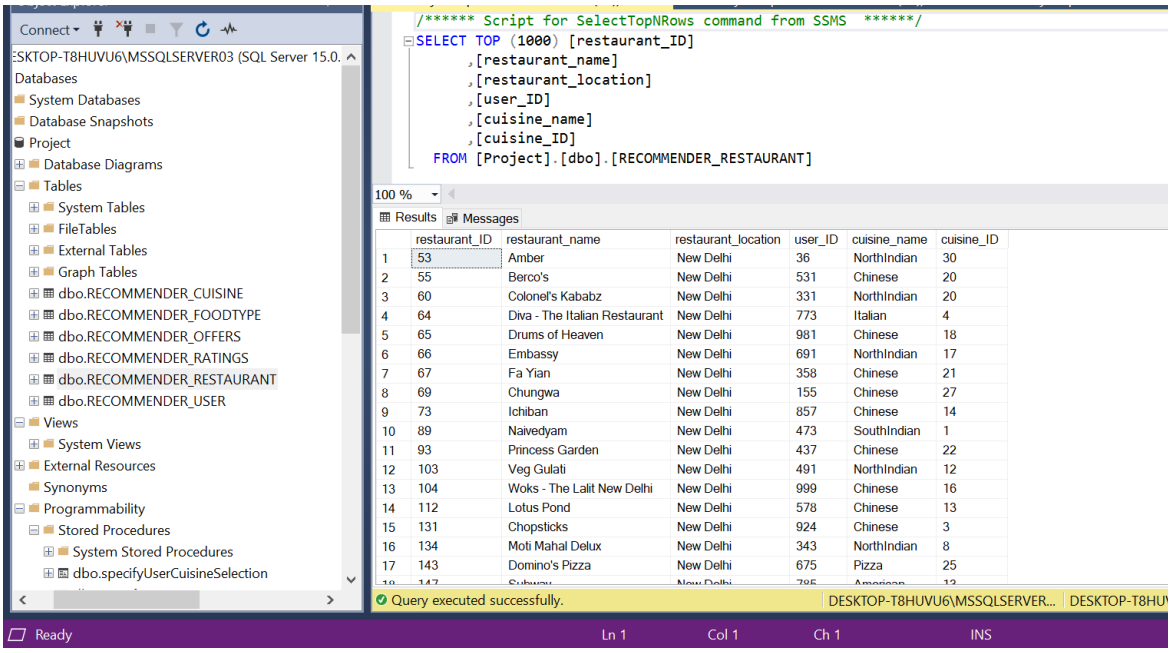


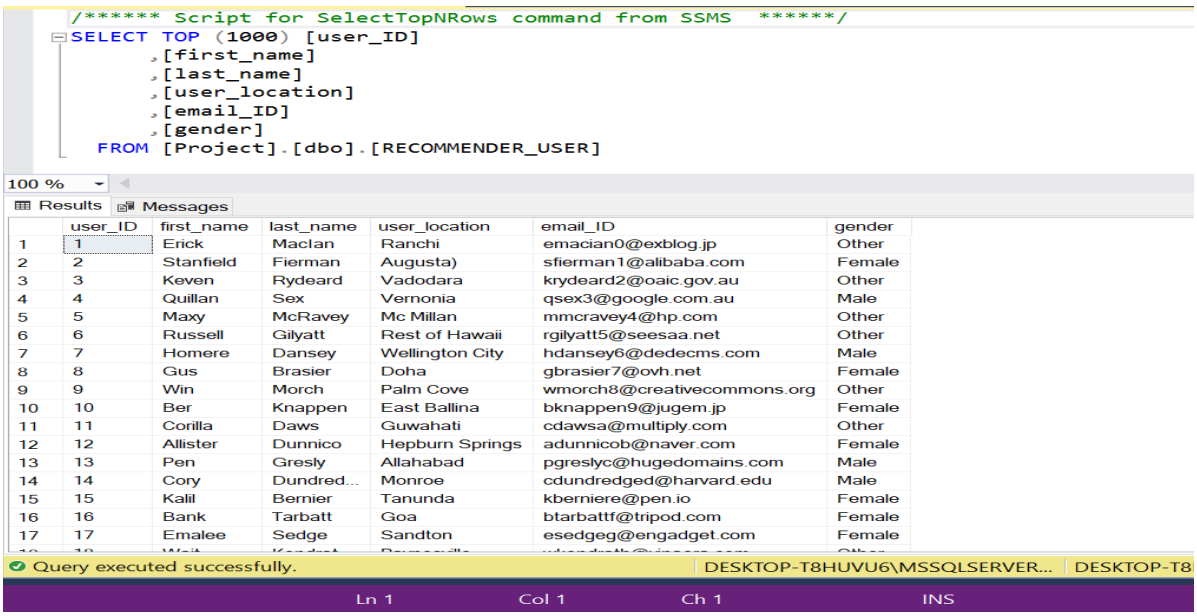
Table Views

Restaurant Table:



	restaurant_ID	restaurant_name	restaurant_location	user_ID	cuisine_name	cuisine_ID
1	53	Amber	New Delhi	36	NorthIndian	30
2	55	Berco's	New Delhi	531	Chinese	20
3	60	Colonel's Kababz	New Delhi	331	NorthIndian	20
4	64	Diva - The Italian Restaurant	New Delhi	773	Italian	4
5	65	Drums of Heaven	New Delhi	981	Chinese	18
6	66	Embassy	New Delhi	691	NorthIndian	17
7	67	Fa Yian	New Delhi	358	Chinese	21
8	69	Chungwa	New Delhi	155	Chinese	27
9	73	Ichiban	New Delhi	857	Chinese	14
10	89	Naivedyam	New Delhi	473	SouthIndian	1
11	93	Princess Garden	New Delhi	437	Chinese	22
12	103	Veg Gulati	New Delhi	491	NorthIndian	12
13	104	Woks - The Lalit New Delhi	New Delhi	999	Chinese	16
14	112	Lotus Pond	New Delhi	578	Chinese	13
15	131	Chopsticks	New Delhi	924	Chinese	3
16	134	Moti Mahal Delux	New Delhi	343	NorthIndian	8
17	143	Domino's Pizza	New Delhi	675	Pizza	25

User Table:



	user_ID	first_name	last_name	user_location	email_ID	gender
1	1	Erick	MacIan	Ranchi	emacian0@exblog.jp	Other
2	2	Stanfield	Fierman	Augusta)	sfierman1@alibaba.com	Female
3	3	Keven	Rydeard	Vadodara	krydeard2@oaic.gov.au	Other
4	4	Quillan	Sex	Vernonia	qsex3@google.com.au	Male
5	5	Maxy	McRavey	Mc Millan	mmcravey4@hp.com	Other
6	6	Russell	Gilyatt	Rest of Hawaii	rgilyatt5@seesaa.net	Other
7	7	Homere	Dansey	Wellington City	hdansey6@dedecms.com	Male
8	8	Gus	Brasier	Doha	gbrasier7@ovh.net	Female
9	9	Win	Morch	Palm Cove	wmorch8@creativecommons.org	Other
10	10	Ber	Knappen	East Ballina	bknappen9@jugem.jp	Female
11	11	Corilla	Daws	Guwahati	cdawsa@multiply.com	Other
12	12	Allister	Dunnico	Hepburn Springs	adunnicob@naver.com	Female
13	13	Pen	Gresly	Allahabad	pgreslyc@hugedomains.com	Male
14	14	Cory	Dundred...	Monroe	cdundredged@harvard.edu	Male
15	15	Kalil	Bernier	Tanunda	kberniere@pen.io	Female
16	16	Bank	Tarbatt	Goa	btarbattf@tripod.com	Female
17	17	Emalee	Sedge	Sandton	esedgeg@engadget.com	Female

FoodType Table:

```
/****** Script for SelectTopNRows command from SSMS  ******/
SELECT TOP (1000) [item_ID]
      ,[item_name]
      ,[food_rating]
      ,[user_ID]
      ,[cuisine_ID]
      ,[cuisine_name]
  FROM [Project].[dbo].[RECOMMENDER_FOODTYPE]
```

100 %

Results | Messages

	item_ID	item_name	food_rating	user_ID	cuisine_ID	cuisine_name
1	1	twisted american chop suey	4.8	58	8	American
2	2	addy s pound cake american girl collection	4.5	672	8	American
3	3	all american apple pie	4.4	324	8	American
4	4	all american burgers	4.9	860	8	American
5	5	all american cheeseburger pizza	4.8	461	8	American
6	6	all american cheeseburgers	4.4	82	8	American
7	7	all american chicken fried steak with cream gravy	4	776	8	American
8	8	all american chili	4.2	588	8	American
9	9	all american chili mccormick	4.9	364	8	American
10	10	all american club	4.8	38	8	American
11	11	all american county fair prize winning chili	4.9	562	8	American
12	12	all american fruit pie	4.2	138	8	American
13	13	all american hamburgers	4.8	377	8	American
14	14	all american jambalaya	4.2	344	8	American
15	15	all american jello salad	4.3	769	8	American
16	16	all american macaroni and cheese	3.6	242	8	American
17	17	all american molasses baked beans	4	648	8	American
18	18	all american olive burgers	4.5	886	8	American

Query executed successfully. DESKTOP-T8HUVU6\MSSQLSERVER... DESKTOP-T8HUVU6

Ln 1 Col 1 Ch 1 INS

Ratings Table:

```
/****** Script for SelectTopNRows command from SSMS  ******/
SELECT TOP (1000) [rating_ID]
      ,[restaurant_rating]
      ,[user_ID]
      ,[restaurant_ID]
  FROM [Project].[dbo].[RECOMMENDER_RATINGS]
```

100 %

Results | Messages

	rating_ID	restaurant_rating	user_ID	restaurant_ID
1	1	4.8	NULL	6317637
2	2	4.5	NULL	6304287
3	3	4.4	NULL	6300002
4	4	4.9	NULL	6318506
5	5	4.8	NULL	6314302
6	6	4.4	NULL	18189371
7	7	4	NULL	6300781
8	8	4.2	NULL	6301290
9	9	4.9	NULL	6300010
10	10	4.8	NULL	6314987
11	11	4.9	NULL	6309903
12	12	4.2	NULL	6309455
13	13	4.8	NULL	6318433
14	14	4.2	NULL	6310470
15	15	4.3	NULL	6314605
16	16	3.6	NULL	18185059
17	17	4	NULL	18182702
18	18	4.5	NULL	6318213
19	19	4.5	NULL	18255654

Query executed successfully. DESKTOP-T8HUVU6\MSSQLSERVER...

Ln 1 Col 1 Ch 1 INS

Data Synthesis

The data for the project has been synthesized using a combination of data from the Zomato dataset, some random data generated through Python and Microsoft Excel. Some of the prominent functions that were used in Excel include,

- VLOOKUP
- INDEX
- ROWS
- RAND
- RANDBETWEEN

The table below shows the data stored in all the important tables used in this phase of the project.

Table Name	Columns	No of constraints	No of Records
User	**6**	**4**	**1000**
Restaurant	**6**	**6**	**9551**
Cuisine	**2**	**2**	**30**
FoodType	**6**	**5**	**3667**
Ratings	**4**	**4**	**9551**

Data Integrity

Data Integrity refers to the consistency and maintenance of the data through the life cycle of the database. In a database, data integrity can be ensured through the implementation of Integrity Constraints in a table. Integrity constraints help apply business rules to the database tables. The constraints can either be at a column level or a table level. Some of the most common constraints are

- NOT NULL: – Prevents a column from having a NULL value
- PRIMARY KEY: – Uniquely identifies each row or record in table
- FOREIGN KEY: – Uniquely identifies a column that references a PRIMARY KEY in another table
- UNIQUE: – Prevents a column from having duplicate values.

Following code shows some constraints that were applied to the tables in our project

Scripts used in designing the database:

Cuisine table:

```
CREATE TABLE [dbo].[RECOMMENDER_CUISINE](
[cuisine_ID] [int] NOT NULL,
[cuisine_name] [nvarchar](100) NOT NULL,
```

```
CONSTRAINT [PK_RECOMMENDER_CUISINE_1] PRIMARY KEY CLUSTERED
(
[cuisine_ID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY
= OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
```

FoodType Table:

```
CREATE TABLE [dbo].[RECOMMENDER_FOODTYPE](
[item_ID] [int] NOT NULL,
[item_name] [nvarchar](100) NOT NULL,
[food_rating] [float] NULL,
[user_ID] [int] NULL,
[cuisine_ID] [int] NOT NULL,
[cuisine_name] [nvarchar](100) NULL,
CONSTRAINT [PK__RECOMMEN__52030845B9570FDA] PRIMARY KEY CLUSTERED
(
[item_ID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY
= OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[RECOMMENDER_FOODTYPE] WITH CHECK ADD
CONSTRAINT [FK__RECOMMEND__user___32E0915F] FOREIGN KEY([user_ID])
REFERENCES [dbo].[RECOMMENDER_USER] ([user_ID])
GO

ALTER TABLE [dbo].[RECOMMENDER_FOODTYPE] CHECK CONSTRAINT
[FK__RECOMMEND__user___32E0915F]
GO
```

Offers table:

```
CREATE TABLE [dbo].[RECOMMENDER_OFFERS](
[offer_ID] [int] NOT NULL,
[offer_type] [varchar](30) NULL,
[offer_validity] [date] NULL,
[user_ID] [int] NULL,
[restaurant_ID] [int] NULL,
```

```
PRIMARY KEY CLUSTERED
(
[offer_ID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY
= OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO


ALTER TABLE [dbo].[RECOMMENDER_OFFERS] WITH CHECK ADD CONSTRAINT
[FK__RECOMMEND__resta__37A5467C] FOREIGN KEY([restaurant_ID])
REFERENCES [dbo].[RECOMMENDER_RESTAURANT] ([restaurant_ID])
GO


ALTER TABLE [dbo].[RECOMMENDER_OFFERS] CHECK CONSTRAINT
[FK__RECOMMEND__resta__37A5467C]
GO


ALTER TABLE [dbo].[RECOMMENDER_OFFERS] WITH CHECK ADD CONSTRAINT
[FK__RECOMMEND__user___36B12243] FOREIGN KEY([user_ID])
REFERENCES [dbo].[RECOMMENDER_USER] ([user_ID])
GO


ALTER TABLE [dbo].[RECOMMENDER_OFFERS] CHECK CONSTRAINT
[FK__RECOMMEND__user___36B12243]
GO
```

Ratings Table:

```
CREATE TABLE [dbo].[RECOMMENDER_RATINGS](
[rating_ID] [int] NOT NULL,
[restaurant_rating] [float] NULL,
[user_ID] [int] NULL,
[restaurant_ID] [int] NULL,
PRIMARY KEY CLUSTERED
(
[rating_ID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY
```

```
= OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[RECOMMENDER_RATINGS] WITH CHECK ADD CONSTRAINT
[FK__RECOMMEND__resta__300424B4] FOREIGN KEY([restaurant_ID])
REFERENCES [dbo].[RECOMMENDER_RESTAURANT] ([restaurant_ID])
GO

ALTER TABLE [dbo].[RECOMMENDER_RATINGS] CHECK CONSTRAINT
[FK__RECOMMEND__resta__300424B4]
GO

ALTER TABLE [dbo].[RECOMMENDER_RATINGS] WITH CHECK ADD CONSTRAINT
[FK__RECOMMEND__user___2F10007B] FOREIGN KEY([user_ID])
REFERENCES [dbo].[RECOMMENDER_USER] ([user_ID])
GO

ALTER TABLE [dbo].[RECOMMENDER_RATINGS] CHECK CONSTRAINT
[FK__RECOMMEND__user___2F10007B]
GO
```

Restaurant Table:

```
CREATE TABLE [dbo].[RECOMMENDER_RESTAURANT](
[restaurant_ID] [int] NOT NULL,
[restaurant_name] [nvarchar](100) NULL,
[restaurant_location] [nvarchar](100) NOT NULL,
[user_ID] [int] NULL,
[cuisine_name] [nvarchar](100) NULL,
[cuisine_ID] [int] NULL,
CONSTRAINT [PK__RECOMMEN__3B0EA699E5C56E40] PRIMARY KEY CLUSTERED
(
[restaurant_ID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY
= OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[RECOMMENDER_RESTAURANT] WITH CHECK ADD
CONSTRAINT [FK__RECOMMEND__user___2B3F6F97] FOREIGN KEY([user_ID])
```

```
REFERENCES [dbo].[RECOMMENDER_USER] ([user_ID])
GO

ALTER TABLE [dbo].[RECOMMENDER_RESTAURANT] CHECK CONSTRAINT
[FK__RECOMMEND__user___2B3F6F97]
GO

ALTER TABLE [dbo].[RECOMMENDER_RESTAURANT] WITH CHECK ADD
CONSTRAINT [FK_RECOMMENDER_RESTAURANT_RECOMMENDER_CUISINE]
FOREIGN KEY([cuisine_ID])
REFERENCES [dbo].[RECOMMENDER_CUISINE] ([cuisine_ID])
GO

ALTER TABLE [dbo].[RECOMMENDER_RESTAURANT] CHECK CONSTRAINT
[FK_RECOMMENDER_RESTAURANT_RECOMMENDER_CUISINE]
GO
```

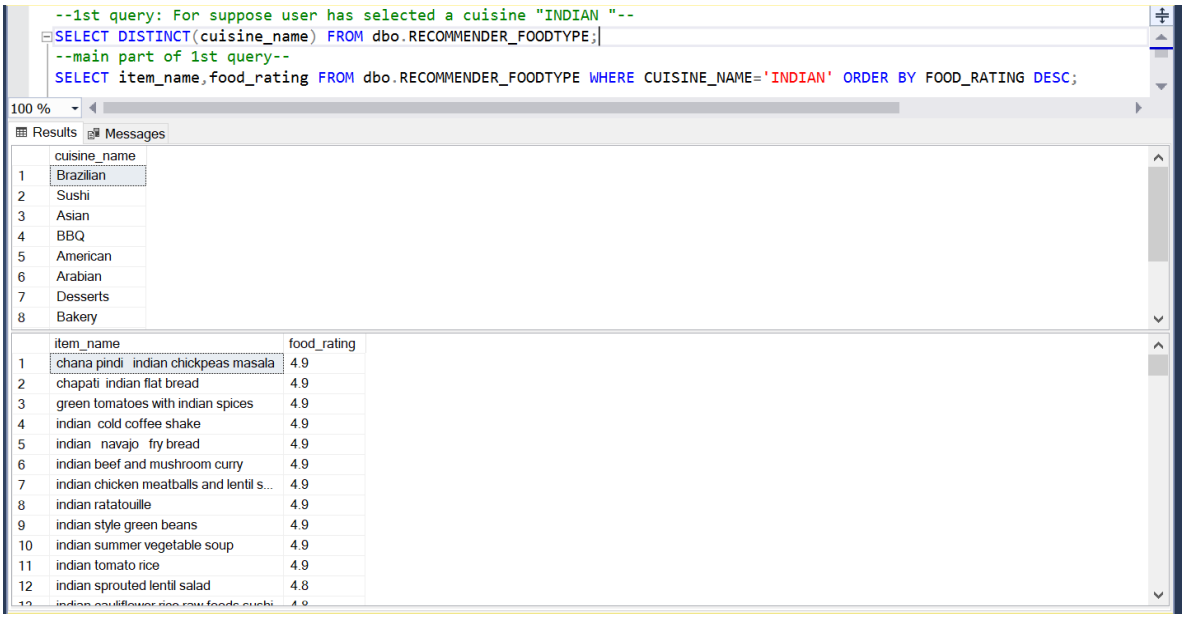
User Table:

```
CREATE TABLE [dbo].[RECOMMENDER_USER](
[user_ID] [int] NOT NULL,
[first_name] [varchar](50) NOT NULL,
[last_name] [varchar](50) NOT NULL,
[user_location] [varchar](60) NULL,
[email_ID] [varchar](50) NULL,
[gender] [varchar](10) NULL,
CONSTRAINT [PK__RECOMMEN__B9BF33074AC11651] PRIMARY KEY CLUSTERED
(
[user_ID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY
= OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
```

Queries for Recommendation

Usecase-1:

Recommending the top 5 food items with highest ratings based on cuisine selection from the user



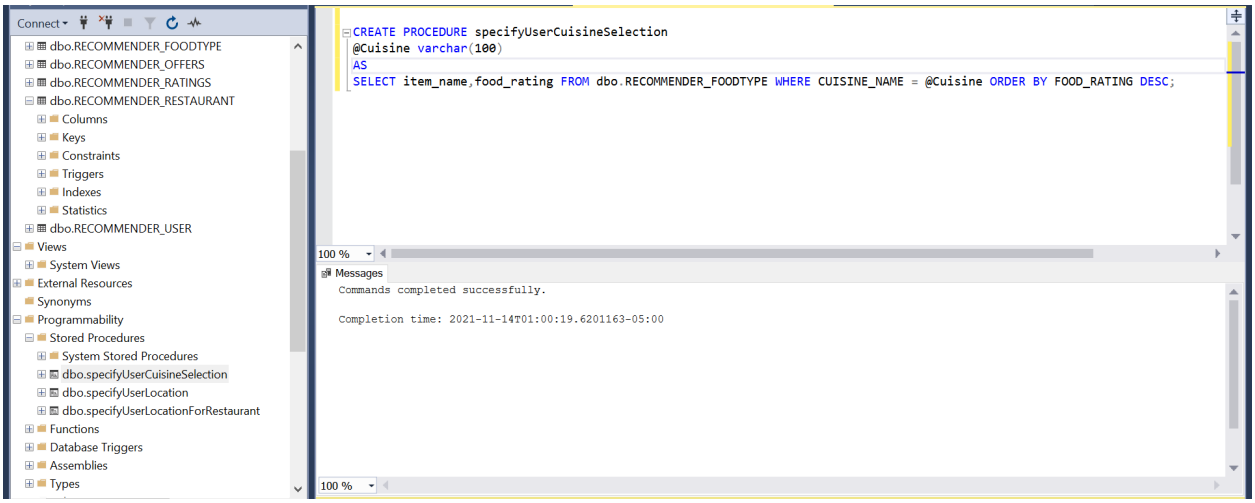
```
--1st query: For suppose user has selected a cuisine "INDIAN "--
SELECT DISTINCT(cuisine_name) FROM dbo.RECOMMENDER_FOODTYPE;
--main part of 1st query--
SELECT item_name,food_rating FROM dbo.RECOMMENDER_FOODTYPE WHERE CUISINE_NAME='INDIAN' ORDER BY FOOD_RATING DESC;
```

Results | Messages

	cuisine_name
1	Brazilian
2	Sushi
3	Asian
4	BBQ
5	American
6	Arabian
7	Desserts
8	Bakery

	item_name	food_rating
1	chana pindi indian chickpeas masala	4.9
2	chapati indian flat bread	4.9
3	green tomatoes with indian spices	4.9
4	indian cold coffee shake	4.9
5	indian navajo fry bread	4.9
6	indian beef and mushroom curry	4.9
7	indian chicken meatballs and lentil s...	4.9
8	indian ratatouille	4.9
9	indian style green beans	4.9
10	indian summer vegetable soup	4.9
11	indian tomato rice	4.9
12	indian sprouted lentil salad	4.8

Stored procedure for cuisine selection:



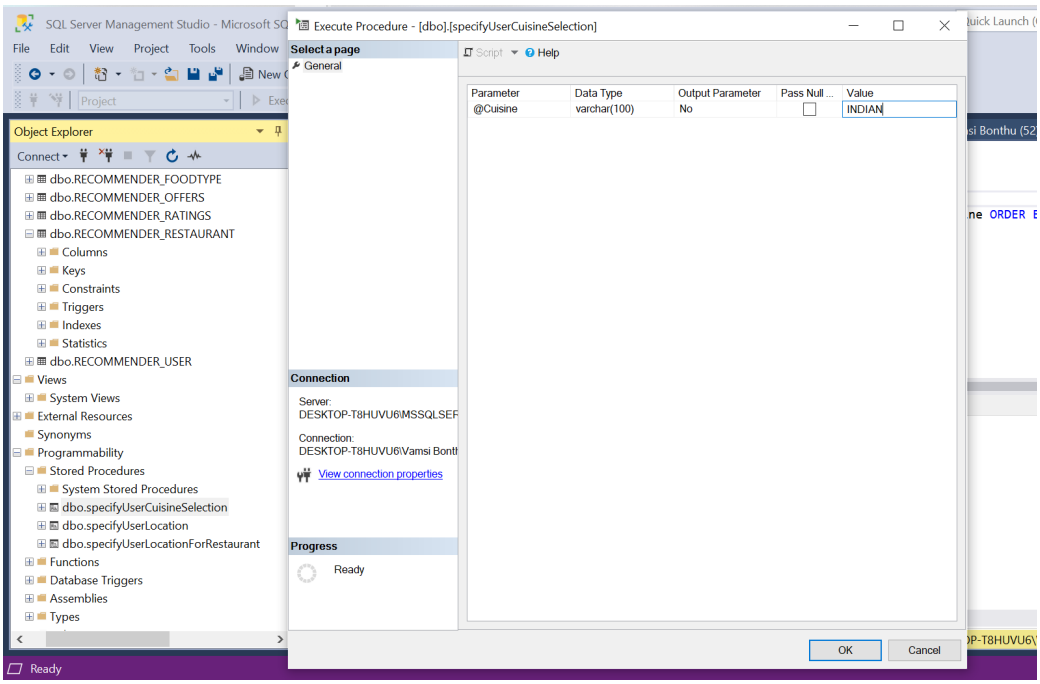
```
CREATE PROCEDURE specifyUserCuisineSelection
@Cuisine varchar(100)
AS
SELECT item_name,food_rating FROM dbo.RECOMMENDER_FOODTYPE WHERE CUISINE_NAME = @Cuisine ORDER BY FOOD_RATING DESC;
```

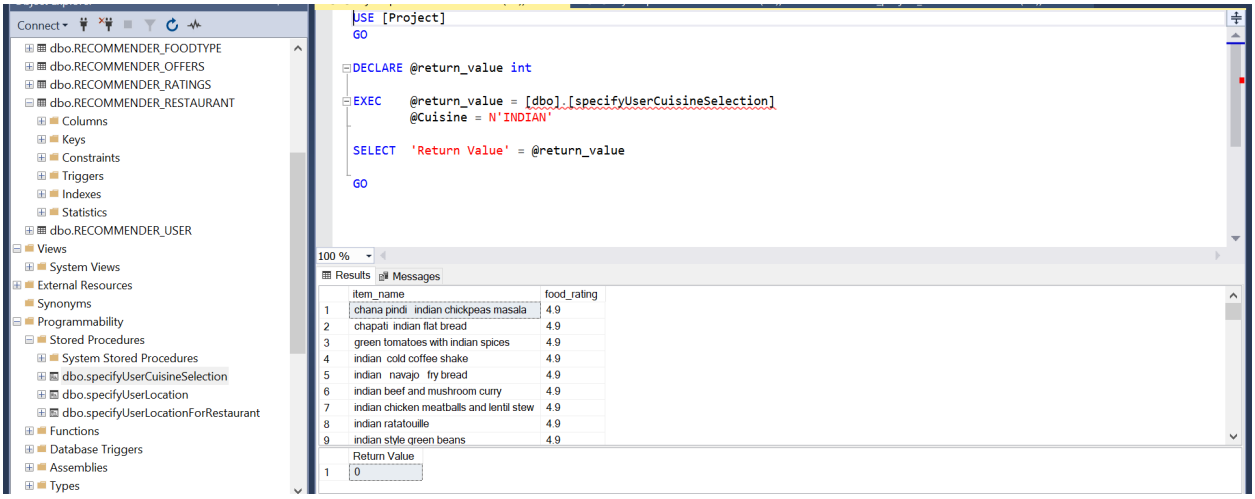
Connect

- dbo.RECOMMENDER_FOODTYPE
- dbo.RECOMMENDER_OFFERS
- dbo.RECOMMENDER_RATINGS
- dbo.RECOMMENDER_RESTAURANT
  - Columns
  - Keys
  - Constraints
  - Triggers
  - Indexes
  - Statistics
- dbo.RECOMMENDER_USER
- Views
  - System Views
- External Resources
- Synonyms
- Programmability
  - Stored Procedures
    - System Stored Procedures
    - dbo.specifyUserCuisineSelection
    - dbo.specifyUserLocation
    - dbo.specifyUserLocationForRestaurant
  - Functions
  - Database Triggers
  - Assemblies
  - Types

Messages

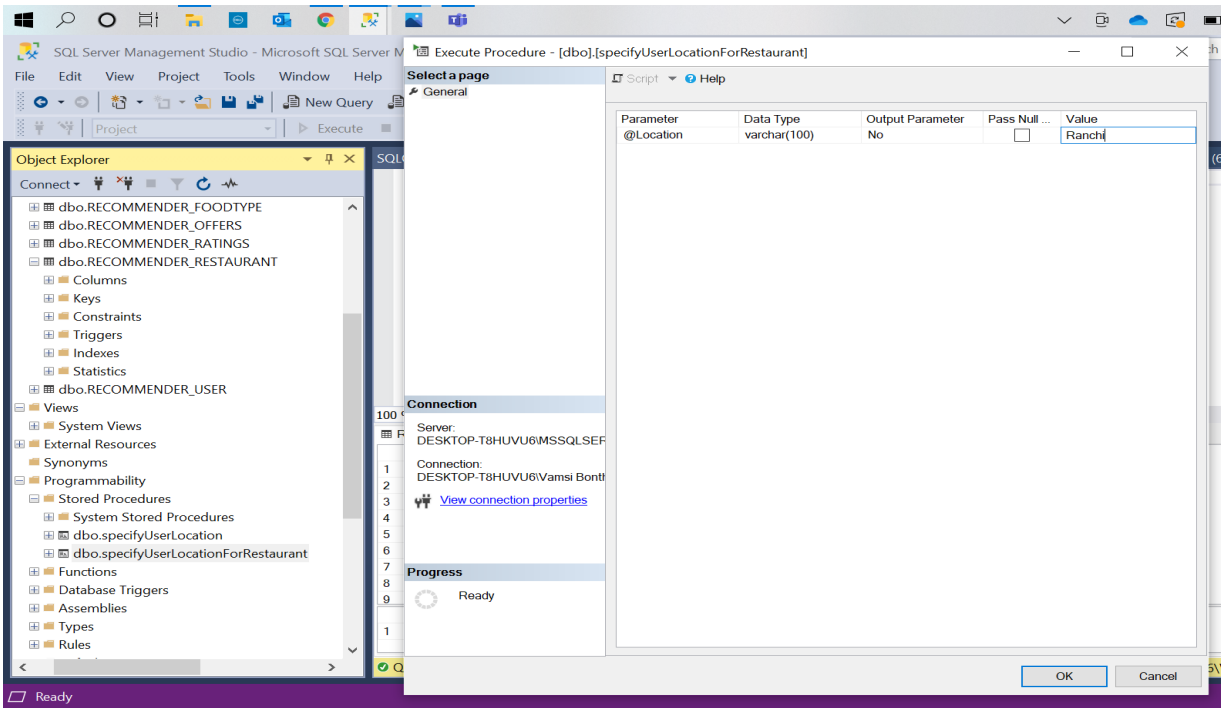
```
Commands completed successfully.

Completion time: 2021-11-14T01:00:19.6201163-05:00
```

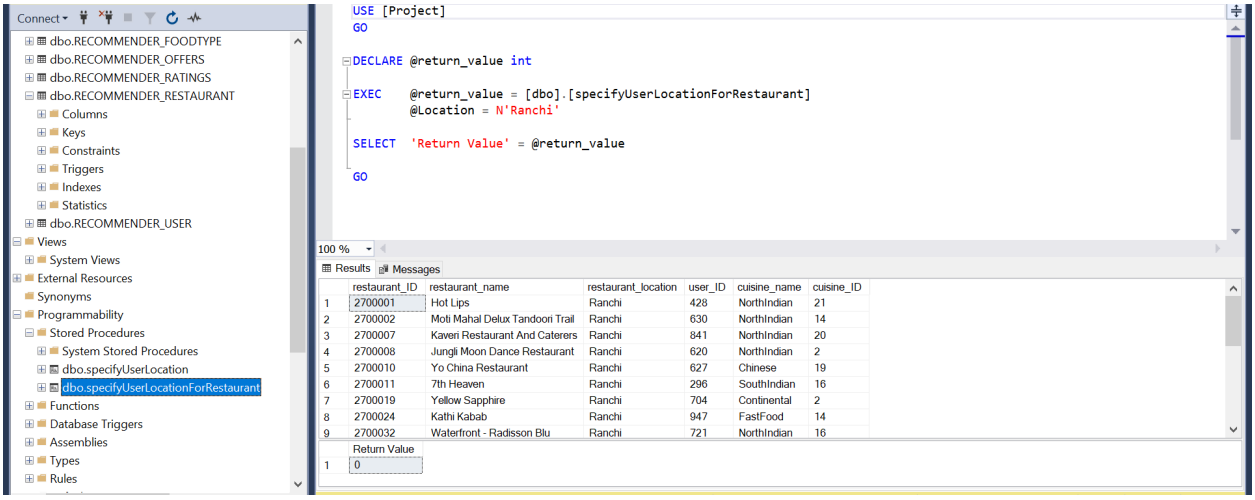


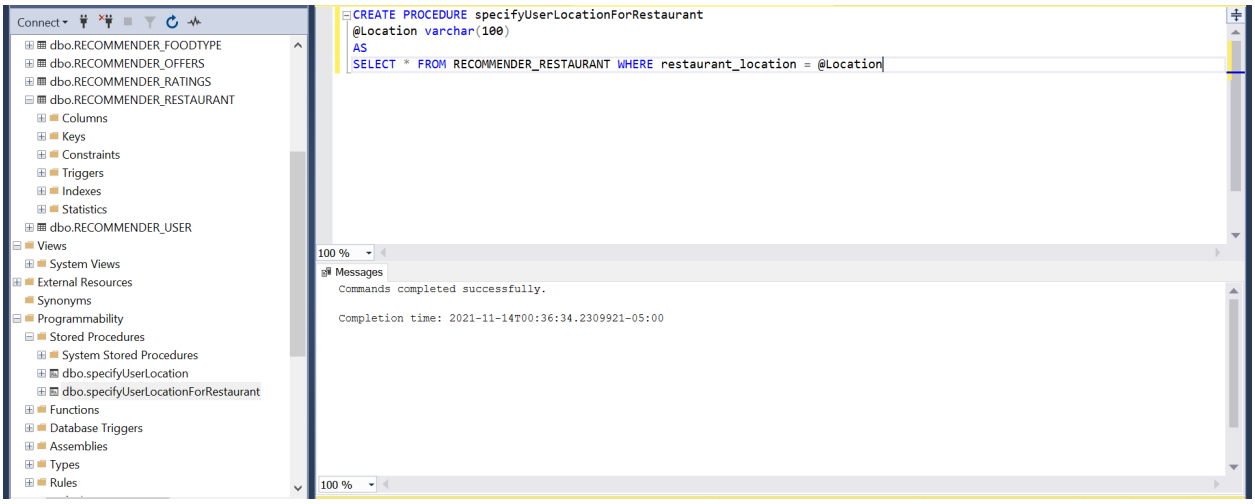


Usecase-2:

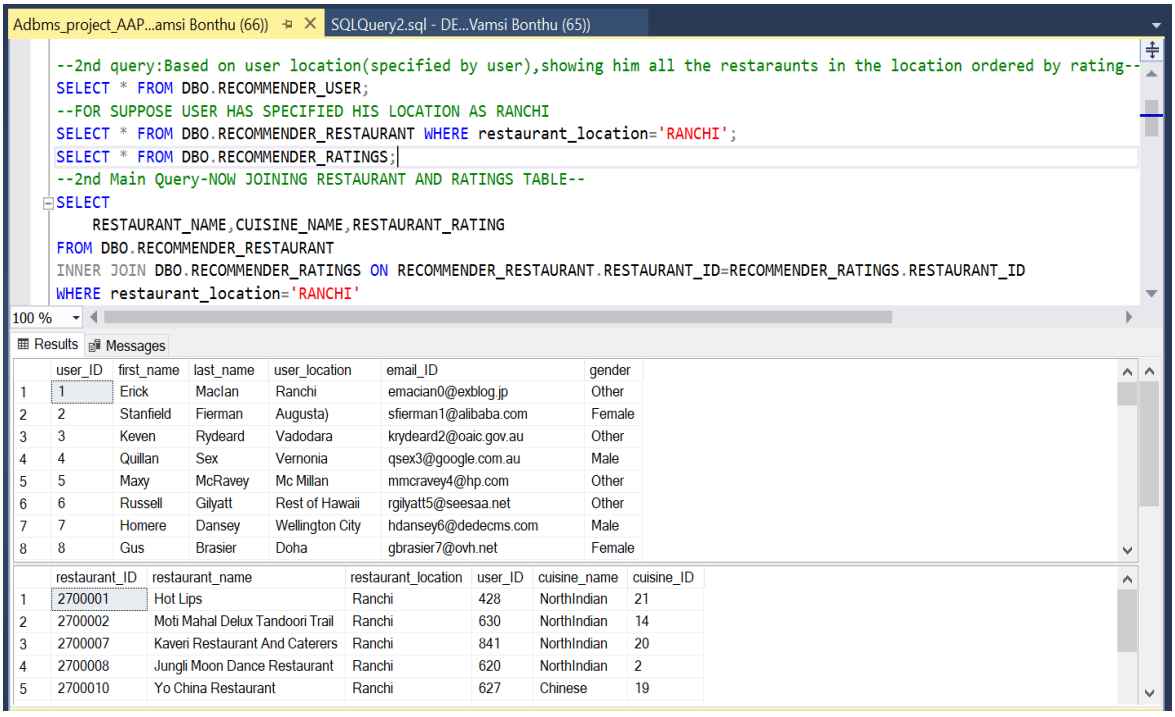
Recommending the best restaurants in the user location.
Following is the stored procedure for the user to input the location for recommendations.


SQL Server Management Studio - Microsoft SQL Server M
File Edit View Project Tools Window Help
New Query
Execute
Object Explorer
Connect
dbo.RECOMMENDER_FOODTYPE
dbo.RECOMMENDER_OFFERS
dbo.RECOMMENDER_RATINGS
dbo.RECOMMENDER_RESTAURANT
Columns
Keys
Constraints
Triggers
Indexes
Statistics
dbo.RECOMMENDER_USER
Views
System Views
External Resources
Synonyms
Programmability
Stored Procedures
System Stored Procedures
dbo.specifyUserLocation
dbo.specifyUserLocationForRestaurant
Functions
Database Triggers
Assemblies
Types
Rules
Execute Procedure - [dbo].[specifyUserLocationForRestaurant]
Select a page
General
Script
Help
Parameter
Data Type
Output Parameter
Pass Null ...
Value
@Location
varchar(100)
No
Ranchi
Connection
Server:
DESKTOP-T8HUVU6\MSSQLSEF
Connection:
DESKTOP-T8HUVU6\Vamsi Bonth
View connection properties
Progress
Ready
OK
Cancel
Ready


Connect
dbo.RECOMMENDER_FOODTYPE
dbo.RECOMMENDER_OFFERS
dbo.RECOMMENDER_RATINGS
dbo.RECOMMENDER_RESTAURANT
Columns
Keys
Constraints
Triggers
Indexes
Statistics
dbo.RECOMMENDER_USER
Views
System Views
External Resources
Synonyms
Programmability
Stored Procedures
System Stored Procedures
dbo.specifyUserLocation
dbo.specifyUserLocationForRestaurant
Functions
Database Triggers
Assemblies
Types
Rules
USE [Project]
GO
DECLARE @return_value int
EXEC @return_value = [dbo].[specifyUserLocationForRestaurant]
@Location = N'Ranchi'
SELECT 'Return Value' = @return_value
GO
100 %
Results
Messages
restaurant_ID
restaurant_name
restaurant_location
user_ID
cuisine_name
cuisine_ID
1 2700001 Hot Lips Ranchi 428 NorthIndian 21
2 2700002 Moti Mahal Delux Tandoori Trail Ranchi 630 NorthIndian 14
3 2700007 Kaveri Restaurant And Caterers Ranchi 841 NorthIndian 20
4 2700008 Jungli Moon Dance Restaurant Ranchi 620 NorthIndian 2
5 2700010 Yo China Restaurant Ranchi 627 Chinese 19
6 2700011 7th Heaven Ranchi 296 SouthIndian 16
7 2700019 Yellow Sapphire Ranchi 704 Continental 2
8 2700024 Kathi Kabab Ranchi 947 FastFood 14
9 2700032 Waterfront - Radisson Blu Ranchi 721 NorthIndian 16
Return Value
1 0



Displaying the top restaurants in the user location based on rating from the ratings table



Usecase-3:

Recommending the most loved food items/ dishes from all cuisines to a new user

```sql
--3rd query-For a new user(show him the most loved food from all the cuisines.--
SELECT * FROM DBO.RECOMMENDER_FOODTYPE;
SELECT DISTINCT(CUISINE_NAME),ITEM_NAME,FOOD_RATING FROM RECOMMENDER_FOODTYPE ORDER BY FOOD_RATING DESC;

--SELECT TOP 5 * FROM (SELECT DISTINCT(CUISINE_NAME),ITEM_NAME,FOOD_RATING FROM RECOMMENDER_FOODTYPE ORDER BY FOOD_RATING

--3rd Main query--
select a.cuisine_name, a.item_name, a.food_rating from (
SELECT cuisine_name, item_name, food_rating,
RANK() OVER(partition by cuisine_name order by food_rating desc) as Rank
FROM RECOMMENDER_FOODTYPE ) a
where Rank <= 5
```

100 %

Results | Messages

	item_ID	item_name	food_rating	user_ID	cuisine_ID	cuisine_name
1	1	twisted american chop suey	4.8	58	8	American
2	2	addy s pound cake american girl collection	4.5	672	8	American
3	3	all american apple pie	4.4	324	8	American
4	4	all american burgers	4.9	860	8	American
5	5	all american cheeseburger pizza	4.8	461	8	American
6	6	all american cheeseburgers	4.4	82	8	American
7	7	all american chicken fried steak with cream gravy	4	776	8	American
8	8	all american chili	4.2	588	8	American

	CUISINE_NAME	ITEM_NAME	FOOD_RATING
1	American	all american burgers	4.9
2	American	all american chili mccormick	4.9
3	American	all american county fair prize winning chili	4.9
4	American	all american pancakes	4.9

Usecase-4:

Recommending the top-rated restaurants for a new user based on the best rated dishes/food items above

```sql
--4th query:showing the restaurant names for above query--
SELECT RF.cuisine_name, RF.ITEM_NAME, rf.FOOD_RATING, rr.restaurant_name
FROM DBO.RECOMMENDER_FOODTYPE RF JOIN  DBO.RECOMMENDER_RESTAURANT RR
ON rf.cuisine_name = rf.cuisine_name  order by rf.food_rating desc;
--4th main query--
select a.cuisine_name, a.item_name, a.food_rating, a.restaurant_name from (
SELECT b.cuisine_name, b.item_name, b.food_rating, c.restaurant_name,
RANK() OVER(partition by b.cuisine_name order by b.food_rating desc,b.cuisine_name,b.item_name,c.restaurant_name )
as Rank
FROM RECOMMENDER_FOODTYPE b join RECOMMENDER_RESTAURANT c on c.cuisine_name = b.cuisine_name) a
where a.Rank <= 1;
```

100 %

Results | Messages

	cuisine_name	ITEM_NAME	FOOD_RATING	restaurant_name
1	American	all american burgers	4.9	Amber
2	American	all american burgers	4.9	Berco's
3	American	all american burgers	4.9	Colonel's Kababz
4	American	all american burgers	4.9	Diva - The Italian Restaurant
5	American	all american burgers	4.9	Drums of Heaven
6	American	all american burgers	4.9	Embassy
7	American	all american burgers	4.9	Fa Yian
8	American	all american burgers	4.9	Chungwa
9	American	all american burgers	4.9	Ichiban
10	American	all american burgers	4.9	Naivedyam
11	American	all american burgers	4.9	Princess Garden
12	American	all american burgers	4.9	Veg Gulati
13	American	all american burgers	4.9	Woks - The Lalit New Delhi
14	American	all american burgers	4.9	Lotus Pond
15	American	all american burgers	4.9	Chopsticks

Performance Optimization

Index:

Indexes are used to increase the overall performance in SQL server by speeding up the query process. This is achieved by reducing the data pages that must be visited or scanned every time a query is run. Without indexing, a DBMS must scan through all the records in the table to retrieve results.

Clustered Indexing:

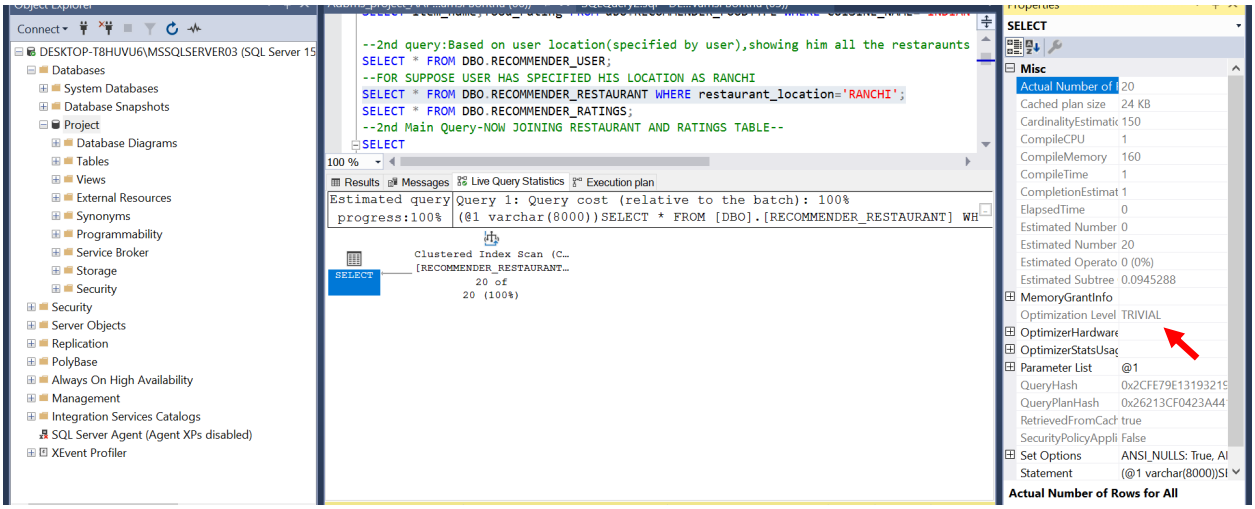
The primary key constraint automatically creates a clustered index on that column. In SQL Server, a clustered index determines the physical order of data in a table. There can be only one clustered index per table. Since we have primary key constraints added to a column in each table, clustered indexes are already applied, making the performance a little better.

Non-Clustered Indexing:

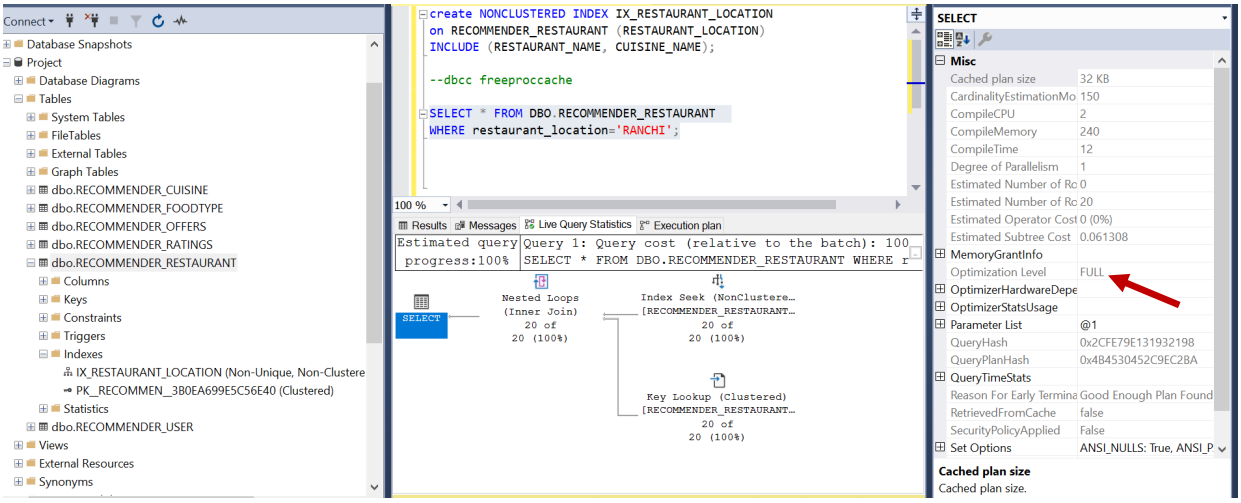
A table can only be ordered in one way, therefore there can only be one clustered index per table. Since some of our use cases show cuisines and restaurants based on restaurant location, we have created non clustered indexing on the restaurant location column of the restaurant table to improve the performance of our queries further.
A non-clustered index doesn't sort the physical data inside the table, it is stored in a different location than the table data. This index contains column values on which the index is created and the address of the record corresponding to the column value. Upon querying the database checks the index and looks for the address of the corresponding record/row in the table. The other column values in that row address are now fetched.

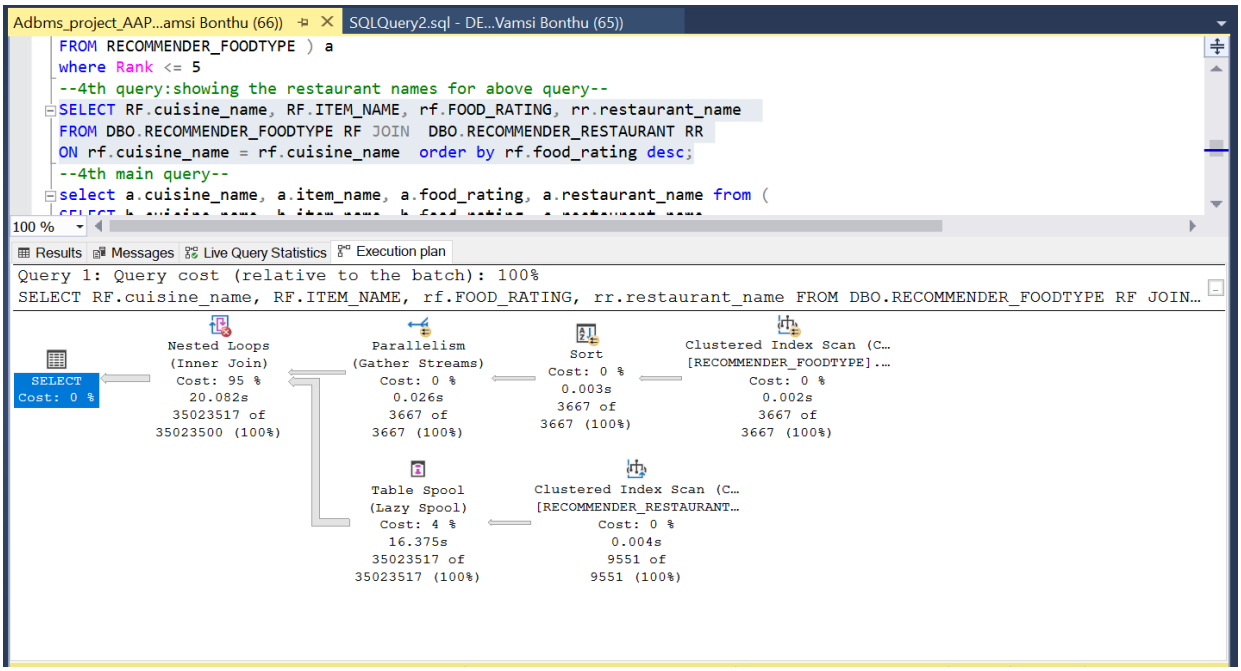
In this case we have searched for restaurant location-Ranchi, without the non-clustered index, it can be noticed that the optimization is TRIVIAL. Upon applying non-clustered indexing to this column, the optimization is changed to FULL.

Before non-clustered indexing on restaurant_location column



After non-clustered indexing on restaurant_location column



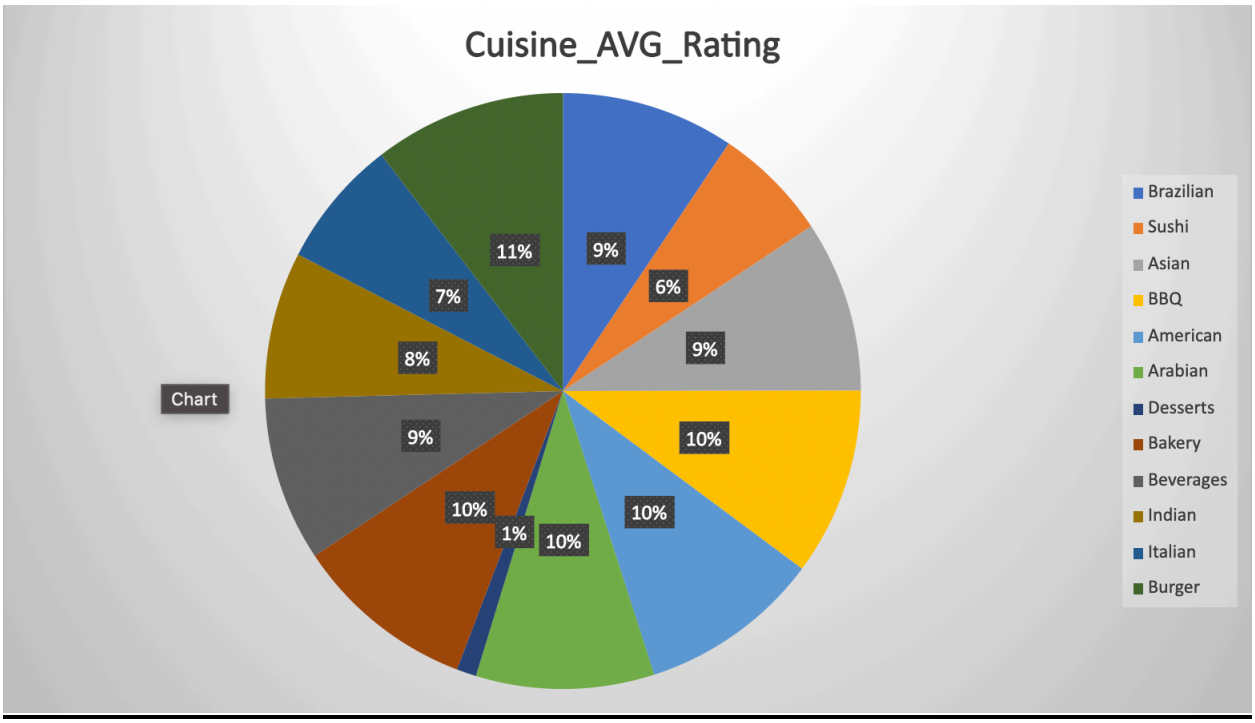


OBSERVATIONS

1. From the above we can see that running a simple count on the table with clustered Index is substantially more expensive than running it on the table with only non-clustered index
2. The non-clustered indexes tend to perform better when the index is entirely covered by the select statement.
3. Clustered indexes seem to work better in retrieving the entire row specific values, this is because when such a query is executed, the optimizer searches it using index key columns and in comparison, is faster with the clustered indexes since the records are stored physically.
4. Clustered indexes seems to be slow while performing insertion or updating operations if there is a change in one of the columns because the record must be physically relocated.

DATA VISUALIZATION

We have visualized the cuisine and ratings data to represent the average rating given by users to each cuisine in a restaurant. The following pie chart depicts the average rating for each cuisine



Future scope:

The next phase of this project is to implement collaborative filtering using cosine similarity concept by integrating the existing database with python.
We also identified some complex use cases like the one mentioned in the use cases section of this report.
We also intend to extend this project to some use cases and update functionality based on Offers table, which we could not achieve due to lack of proper data and time constraints.