SQL Database [Restaurants insights]

Instructor: Jackie Abo Aleam

Prepared by: Sameh Nabil Hasan

7/1/24

DEPI-ALX_DAT_G4D

Table of Contents

Executive Summary: Insights from Restaurant Data Analysis

Overview

This summary covers the setup and analysis of a database using SQL, focusing on restaurants and customer behaviors. The script creates tables for cities, meals, customers, orders, meal types, order details, restaurants, restaurant types, members and members monthly expenses. It imports real data from CSV files to fill these tables.

Results

1- Customer Insights:

- Identified top-spending customers and their favorite meals. This helps in personalizing customer interactions and promotions.

2- Sales Trends:

- Discovered which meals sell best in different cities and how food preferences vary between genders.

3- Expense Analysis:

- Highlighted customers with the highest monthly spending in each city. This guides decisions on where to focus marketing efforts.

4- Operational Tips:

- Analyzed average spending per order and how sales happen through the day, helping in staffing and inventory decisions.

5- Specialized Analyses:

- Examines specific metrics such as price range differences for hot and cold meals, popularity of vegan options, and preferences for different serve types (starter, main, dessert) by gender.

Conclusion

By using SQL queries, a solid database was built , structured and gained valuable insights into customer behaviors and restaurant operations. These insights can drive decisions to improve menu offerings, target promotions better, and enhance overall service.

About Dataset

This dataset is taken from **SQL Server Database**, so the files (tables) are **related to each other.** The dataset includes the category, restaurant information, member information, order information, date, time, location, finance information, and more…

Data Source

This dataset was obtained from Kaggle, which provides datasets for analysis and practice.
https://www.kaggle.com/datasets/vainero/restaurants-customers-orders-dataset

Dataset Content

This dataset consists of 10 CSV files containing tables associated with restaurants in five cities within a country.

1- Cities table contains 2 columns and 5 records.
2- Meal types table contains 2 columns and 4 records.
3- Meals table contains 7 columns and 350 records.
4- Members table contains 7 columns and 200 records.
5- Monthly member totals table contains 14 columns and 1200 records.
6- Order details table contains 3 columns and 70577 records.
7- Orders table contains 6 columns and 36000 records.
8- Restaurant type table contains 2 columns and 5 records.
9- Restaurants table contains 5 columns and 30 records.
10- Serve types table contains 2 columns and 3 records.

Database and Schema Creation

1- Database created

- A new database was created, named Assignment using SQL server management studio wizard. **Figure (1)**

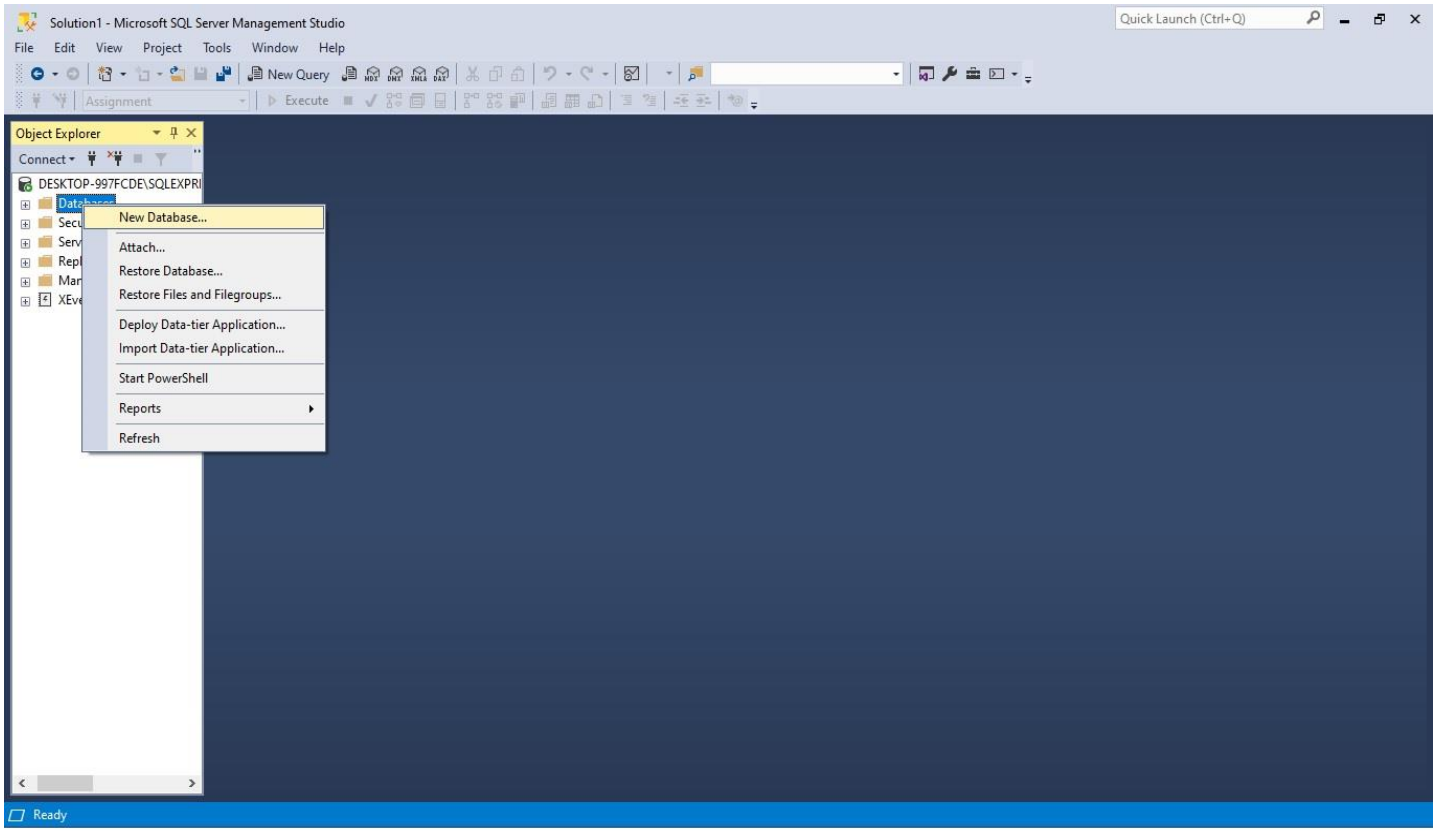


Figure 1

2- Tables Created

- **cities**: Contains information about cities.
- **meal_types**: Defines different types of meals.
- **meals**: Stores details about meals offered by restaurants.
- **members**: Information about members (customers).
- **monthly_member_totals**: Monthly aggregated data for each member.
- **order_details**: Details of orders specifying which meals were ordered.
- **orders**: Information about individual orders placed.
- **restaurant_type**: Types of restaurants.
- **restaurants**: Details of restaurants.
- **serve_types**: Types of service (e.g., Starter, Main, Dessert).

```
1. --Creating table cities contains 2 columns Id as int Primary Key, City as Vachar(50)
2. create table cities
3.     (
4.     Id int primary key,
5.     City varchar(50)
6.     );
7.
8. --Creating table Meal_Types contains 2 columns Id as int primary key, Meal_Type as
   varchar(50)
9. create table meal_types
10.    (
11.    Id int primary key,
12.    meal_type varchar(50)
13.    );
14.
15./*Creating table Meals contains 7 columns Id as int primary key, restaurant_id as int
   (forgin key for restaurant),
16.serve_type_id as int (foreign key for serve_type), meal_type_id as int (frogine key
   for meal_type),
17.hot_cold as varchar(10), meal_name as varchar(20), price as decimal*/
18.
19.create table meals
20.    (
21.    Id int primary key,
22.    restaurant_id int,
23.    serve_type_id int,
24.    meal_type_id int,
25.    hot_cold varchar(10),
```

```
    meal_name varchar(10),
    price decimal(10,2)
    );
/*creating table members contains 7 columns Id as int primary key, first_name as
varchar(40),
surname as varchar(40), sex as varchar(10), email as varchar(100), city_id as int
(foreign key for cities),
monthly_budget as int.*/

create table members
    (
    Id int primary key,
    first_name varchar(40),
    surname varchar(40),
    sex varchar (10),
    email varchar(100),
    city_id int,
    monthly_budget decimal(10,2)
    );

/*creating table monthly_member_totals contains 14 columns member_id as int primery
key, first_name as varchar(40),
surname as varchar(40), sex as varchar(10), email as varchar(100), city as
varchar(40), year as int, month as int,
order_count as int, meals_count as int, monthly_budget as decimal (10,2),
total_expense as decimal(10,2),
balance as decimal(10,2), commission as decimal(10,2).*/

create table monthly_member_totals
    (
    member_id int,
    first_name varchar(40),
    surname varchar(40),
    sex varchar(10),
    email varchar(100),
    city varchar(40),
    year int,
    month int,
    order_count int,
    meals_count int,
    monthly_budget decimal (10,2),
    total_expense decimal(10,2),
    balance decimal(10,2),
    commission decimal(10,2)
    );

--Creating table order_details contains 3 columns Id as int Primary Key, order_id as
int, meal_id as int.

```

```
create table order_details
    (
    Id int primary key,
    order_id int,
    meal_id int
    );

/*Creating table orders contains 6 columns Id as int primary key, date as date, hour as time,
member_id as int (foreign key for members), restaurant_id  as int (frogine key for restaurants),
total_order decimal(10,2).*/

create table orders
    (
    id int primary key,
    date date,
    hour time,
    member_id int,
    restaurant_id int,
    total_order decimal(10,2)
    );

--Creating table restaurant_type contains 2 columns Id as int Primary Key, restaurant_type as Vachar(50)
create table restaurant_type
    (
    id int primary key,
    restaurant_type varchar(50)
    );

/*Creating table restaurants contains 5 columns id as int primary key, restaurant_name as varchar(40),
restaurant_type_id as int (foreign key for restaurant_type ), income_persentage as decimal,
city_id as int (frogine key for Cities).*/

     create table restaurants
         (
         id int primary key,
         restaurant_name varchar(40),
         restaurant_type_id int,
         income_persentage decimal,
         city_id int
         );

     --Creating table serve_types contains 2 columns id as int Primary Key, serve_type as Vachar(40)
     create table serve_types
```

```
112.          (
113.          id int primary key,
114.          serve_type varchar(40)
115.          );
```

3- Relations and Schema

- By using SQL server management studio, the relationship and diagram was created.
- Each table in this dataset has a primary key and potentially one or more foreign keys to establish relationships with other tables. ensured that every foreign key was connected to its corresponding primary key, as shown in **figure (3)**.

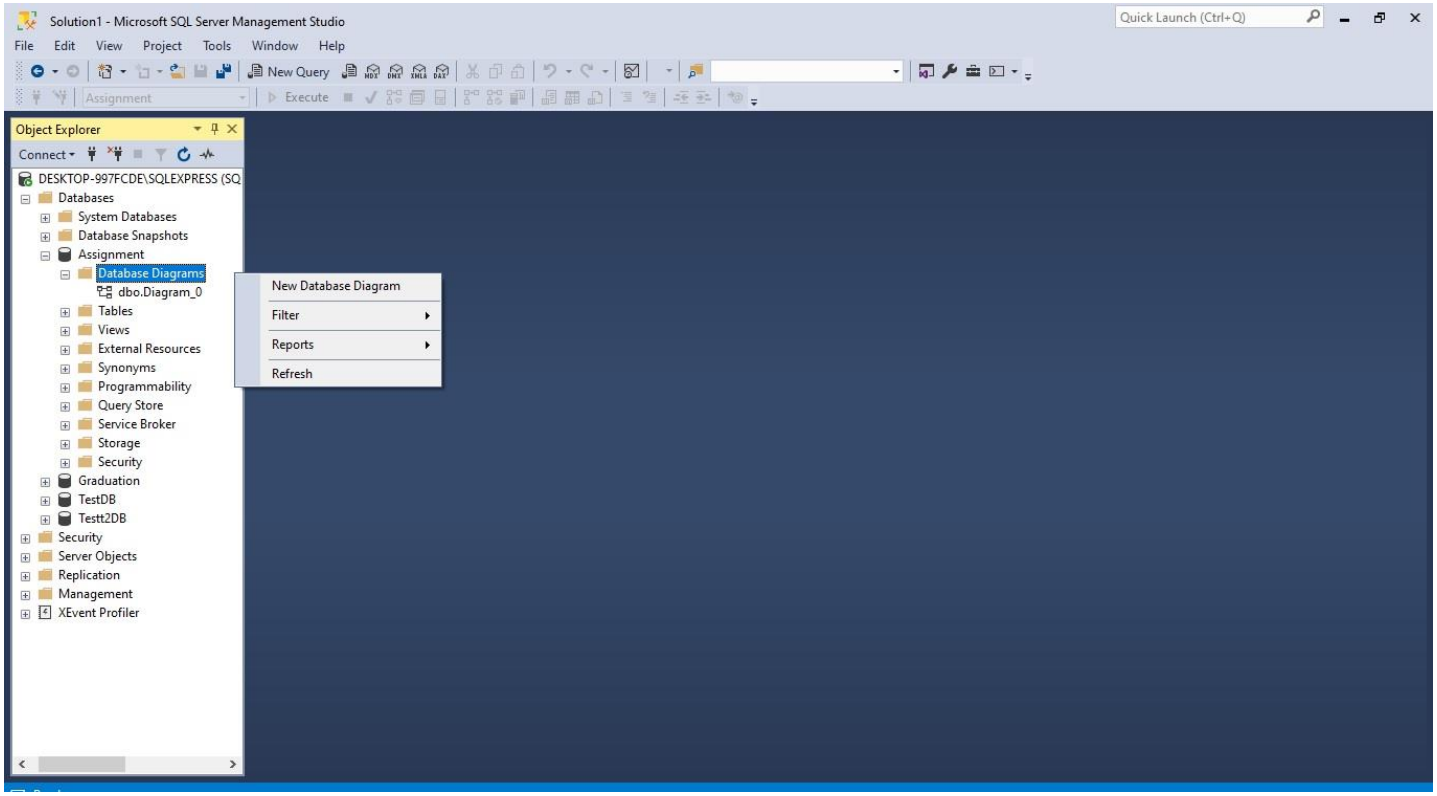


Figure 2

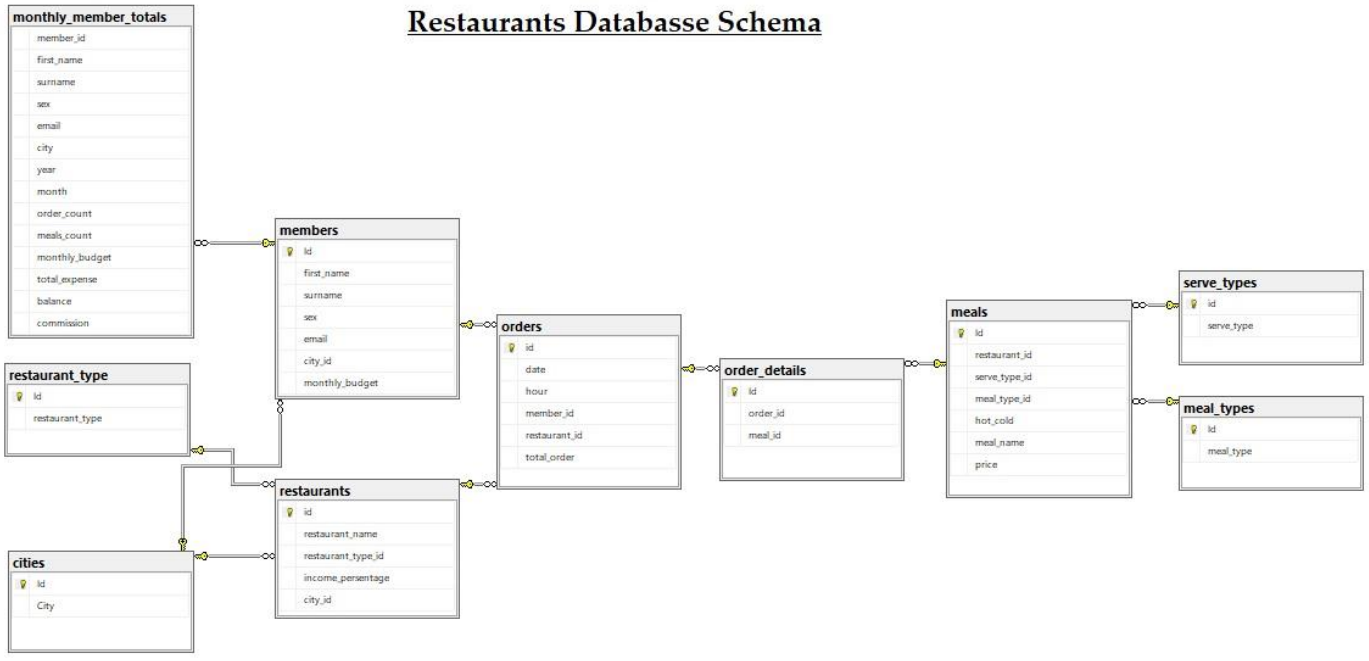


Figure 3

Data insert

- When inserting data into tables, particularly with 70,577 records in the order details table, it was found to be more effective to use a single statement rather than the traditional 'insert into' method for better data accuracy. After conducting research, a simpler and more efficient method was discovered for handling large datasets: 'Bulk insert'.

Resource: https://www.mssqltips.com/sqlservertip/6109/bulk-insert-data-into-sql-server/

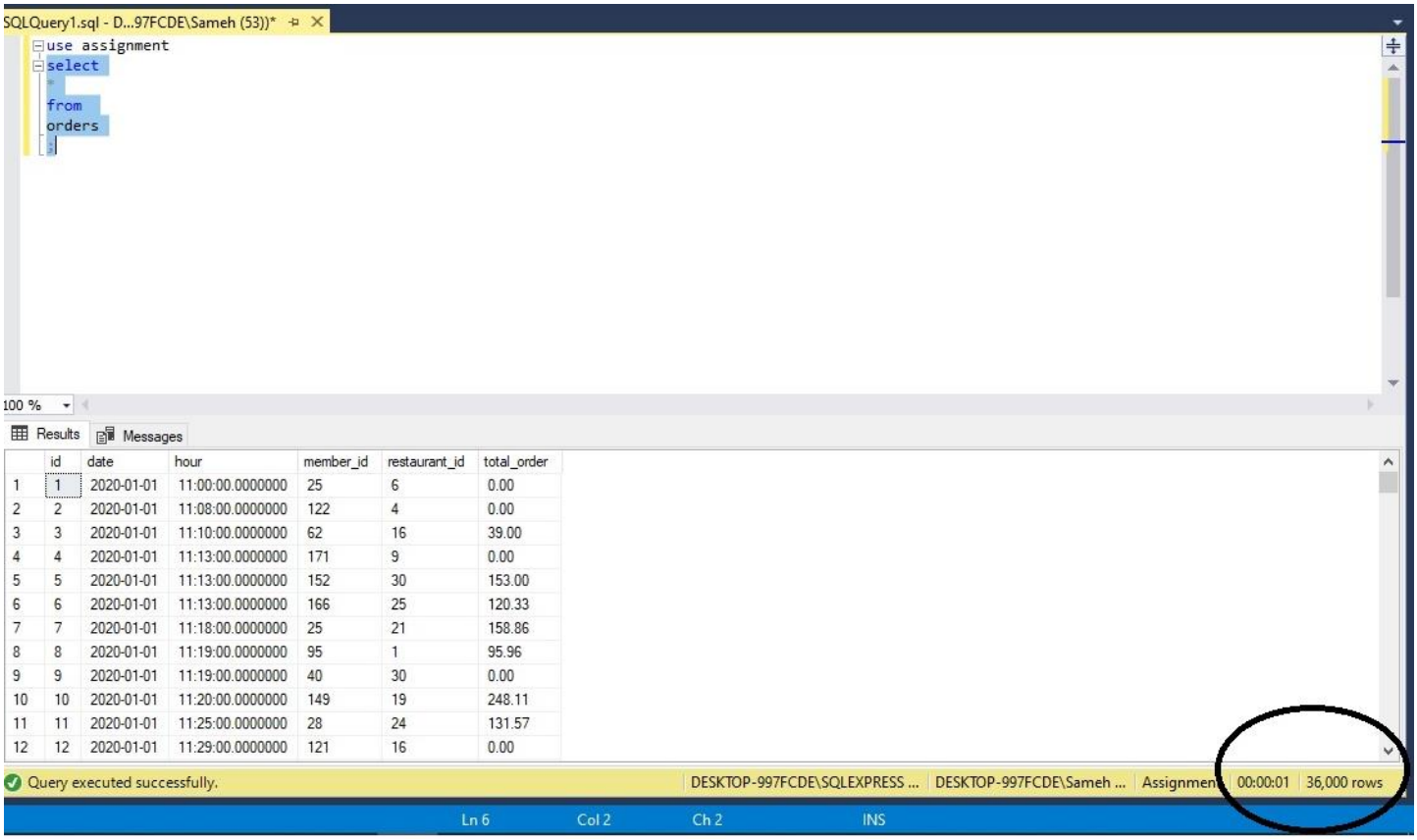
```
--importing data from cities.csv files
bulk insert dbo.cities
from 'I:/Courses/Data analysis/Technical/Assignment/archive/cities.csv'
with (
    fieldterminator = ',', --columns separator
    rowterminator = '\n', -- row separator
    firstrow = 2 --to avoid importing first row
    );

--importing data from meal_types.csv files
bulk insert dbo.meal_types
from 'I:/Courses/Data analysis/Technical/Assignment/archive/meal_types.csv'
with (
    fieldterminator = ',',
    rowterminator = '\n',
    firstrow = 2 --to avoid importing first row
    );


--importing data from meals.csv files
bulk insert dbo.meals
from 'I:/Courses/Data analysis/Technical/Assignment/archive/meals.csv'
with (
    fieldterminator = ',',
    rowterminator = '\n',
    firstrow = 2 --to avoid importing first row
    );

--importing data from members.csv files
bulk insert dbo.members
from 'I:/Courses/Data analysis/Technical/Assignment/archive/members.csv'
with (
    fieldterminator = ',',
    rowterminator = '\n',
    firstrow = 2 --to avoid importing first row
    );

--importing data from monthly_member_totals.csv files
bulk insert dbo.monthly_member_totals
from 'I:/Courses/Data analysis/Technical/Assignment/archive/monthly_member_totals.csv'
```

```
with (
    fieldterminator = ',',
    rowterminator = '\n',
    firstrow = 2 --to avoid importing first row
    );

--importing data from order_details.csv files
bulk insert dbo.order_details
from 'I:/Courses/Data analysis/Technical/Assignment/archive/order_details.csv'
with (
    fieldterminator = ',',
    rowterminator = '\n',
    firstrow = 2 --to avoid importing first row
    );

--importing data from orders.csv files
BULK insert dbo.orders
from 'I:/Courses/Data analysis/Technical/Assignment/archive/orders.csv'
with (
    fieldterminator = ',',
    rowterminator = '\n',
    firstrow = 2 --to avoid importing first row
    );

--importing data from restaurant_type.csv files
bulk insert dbo.restaurant_type
from 'I:/Courses/Data analysis/Technical/Assignment/archive/restaurant_types.csv'
with (
    fieldterminator = ',',
    rowterminator = '\n',
    firstrow = 2 --to avoid importing first row
    );

--importing data from restaurants.csv files
bulk insert dbo.restaurants
from 'I:/Courses/Data analysis/Technical/Assignment/archive/restaurants.csv'
with (
    fieldterminator = ',',
    rowterminator = '\n',
    firstrow = 2 --to avoid importing first row
    );

--importing data from serve_types.csv files
bulk insert dbo.serve_types
from 'I:/Courses/Data analysis/Technical/Assignment/archive/serve_types.csv'
with (
    fieldterminator = ',',
    rowterminator = '\n',
    firstrow = 2 --to avoid importing first row
```

```
);
```

Test query result

- After inserting data into the database tables, it was necessary to test the content of this tables before starting to answer the questions, so the first test query was run. **Figure (4)**



	id	date	hour	member_id	restaurant_id	total_order
1	1	2020-01-01	11:00:00.0000000	25	6	0.00
2	2	2020-01-01	11:08:00.0000000	122	4	0.00
3	3	2020-01-01	11:10:00.0000000	62	16	39.00
4	4	2020-01-01	11:13:00.0000000	171	9	0.00
5	5	2020-01-01	11:13:00.0000000	152	30	153.00
6	6	2020-01-01	11:13:00.0000000	166	25	120.33
7	7	2020-01-01	11:18:00.0000000	25	21	158.86
8	8	2020-01-01	11:19:00.0000000	95	1	95.96
9	9	2020-01-01	11:19:00.0000000	40	30	0.00
10	10	2020-01-01	11:20:00.0000000	149	19	248.11
11	11	2020-01-01	11:25:00.0000000	28	24	131.57
12	12	2020-01-01	11:29:00.0000000	121	16	0.00

Figure 4

Answering Questions and finding insights

1. City Sales During Working Hours

- Summarizes order totals during morning, evening, and night periods across cities using orders and cities. **Figure (5)**

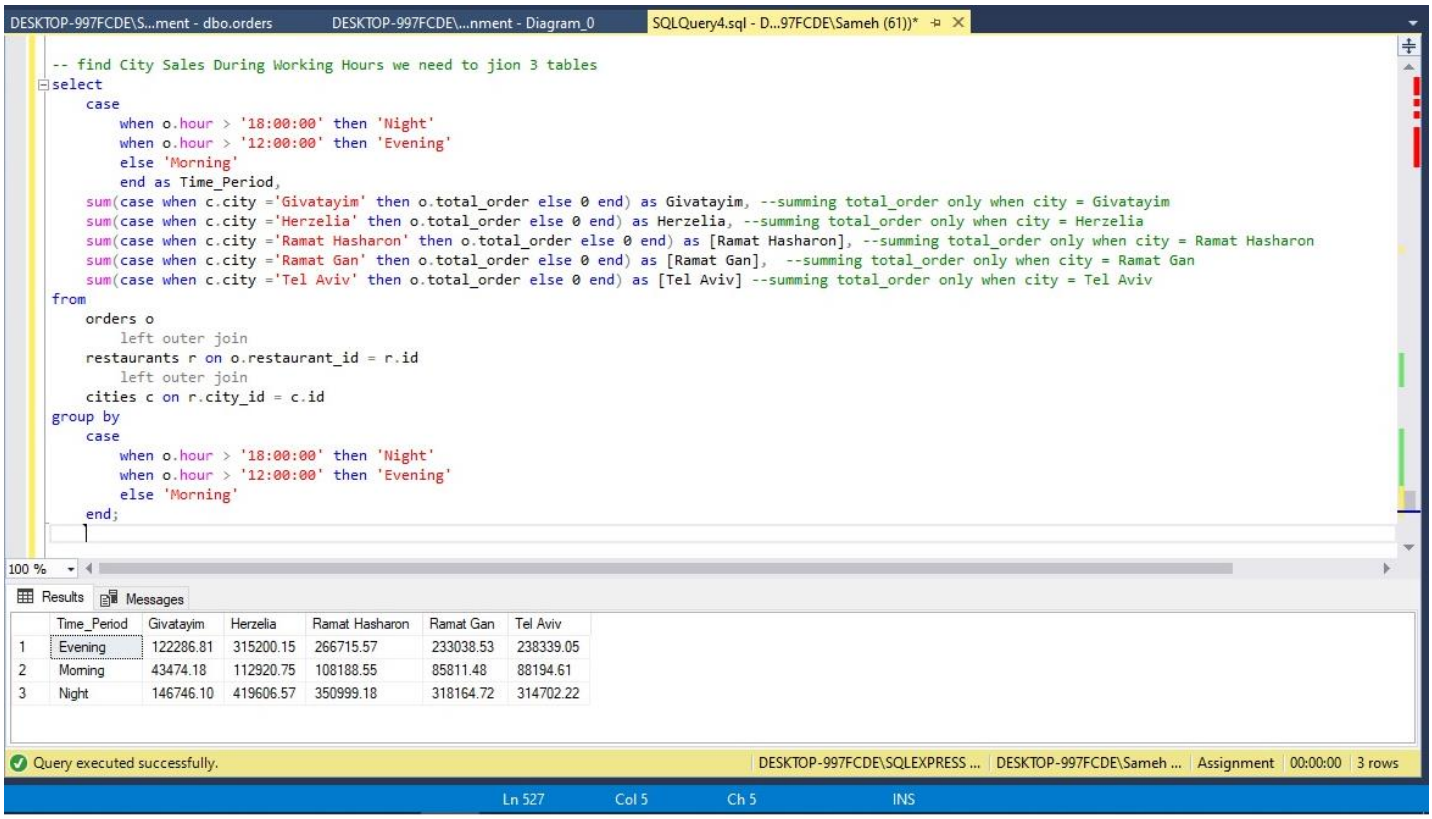


Figure 5

```
-- find City Sales During Working Hours it’s necessary to join 3 tables
select
    case
        when o.hour > '18:00:00' then 'Night'
        when o.hour > '12:00:00' then 'Evening'
        else 'Morning'
        end as Time_Period,
    sum(case when c.city ='Givatayim' then o.total_order else 0 end) as Givatayim, --summing total_order
only when city = Givatayim
    sum(case when c.city ='Herzelia' then o.total_order else 0 end) as Herzelia, --summing total_order only
when city = Herzelia
    sum(case when c.city ='Ramat Hasharon' then o.total_order else 0 end) as [Ramat Hasharon], --summing
total_order only when city = Ramat Hasharon
    sum(case when c.city ='Ramat Gan' then o.total_order else 0 end) as [Ramat Gan],  --summing total_order
only when city = Ramat Gan
    sum(case when c.city ='Tel Aviv' then o.total_order else 0 end) as [Tel Aviv] --summing total_order only
when city = Tel Aviv
```

```
from
    orders o
        left outer join
    restaurants r on o.restaurant_id = r.id
        left outer join
    cities c on r.city_id = c.id
group by
    case
        when o.hour > '18:00:00' then 'Night'
        when o.hour > '12:00:00' then 'Evening'
        else 'Morning'
    end;
```

2. Meal Type Sales by City

- Analyzes meal type sales across cities using meal_types, meals, order_details, orders, restaurants, and cities. **Figure (6)**

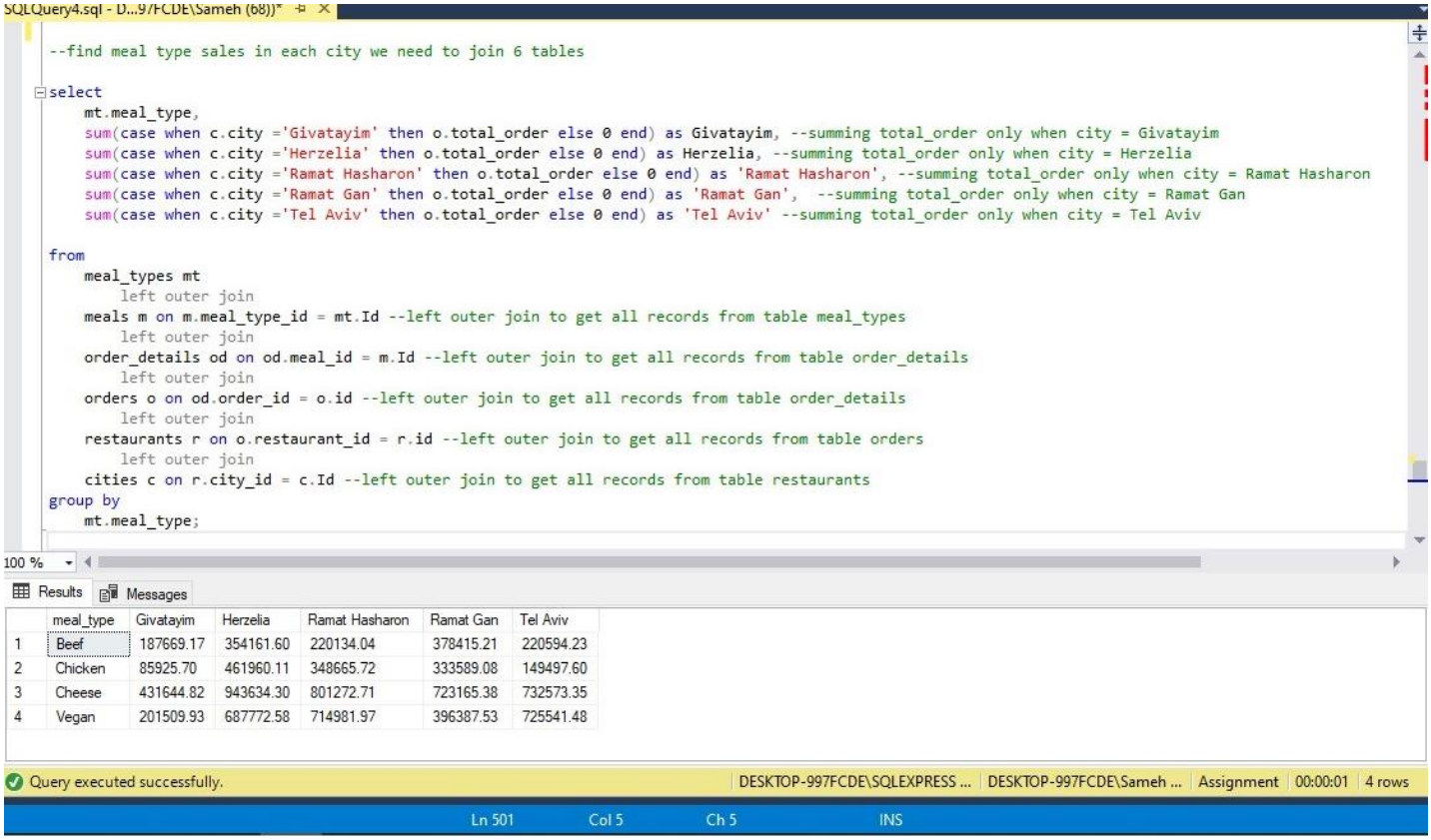


Figure 6

```
--find meal type sales in each city it's necessary to join 6 tables

select
    mt.meal_type,
    sum(case when c.city ='Givatayim' then o.total_order else 0 end) as Givatayim, --summing
total_order only when city = Givatayim
    sum(case when c.city ='Herzelia' then o.total_order else 0 end) as Herzelia, --summing
total_order only when city = Herzelia
    sum(case when c.city ='Ramat Hasharon' then o.total_order else 0 end) as 'Ramat
Hasharon', --summing total_order only when city = Ramat Hasharon
    sum(case when c.city ='Ramat Gan' then o.total_order else 0 end) as 'Ramat Gan',  --
summing total_order only when city = Ramat Gan
    sum(case when c.city ='Tel Aviv' then o.total_order else 0 end) as 'Tel Aviv' --summing
total_order only when city = Tel Aviv

from
    meal_types mt
        left outer join
```

```
    meals m on m.meal_type_id = mt.Id --left outer join to get all records from table
meal_types
        left outer join
    order_details od on od.meal_id = m.Id --left outer join to get all records from table
order_details
        left outer join
    orders o on od.order_id = o.id --left outer join to get all records from table
order_details
        left outer join
    restaurants r on o.restaurant_id = r.id --left outer join to get all records from table
orders
        left outer join
    cities c on r.city_id = c.Id --left outer join to get all records from table restaurants
group by
    mt.meal_type;
```

3. Italian Restaurants Orders by City

- Calculates total orders from Italian restaurants by city using orders, restaurants, restaurant_type, and cities. Figure (7)

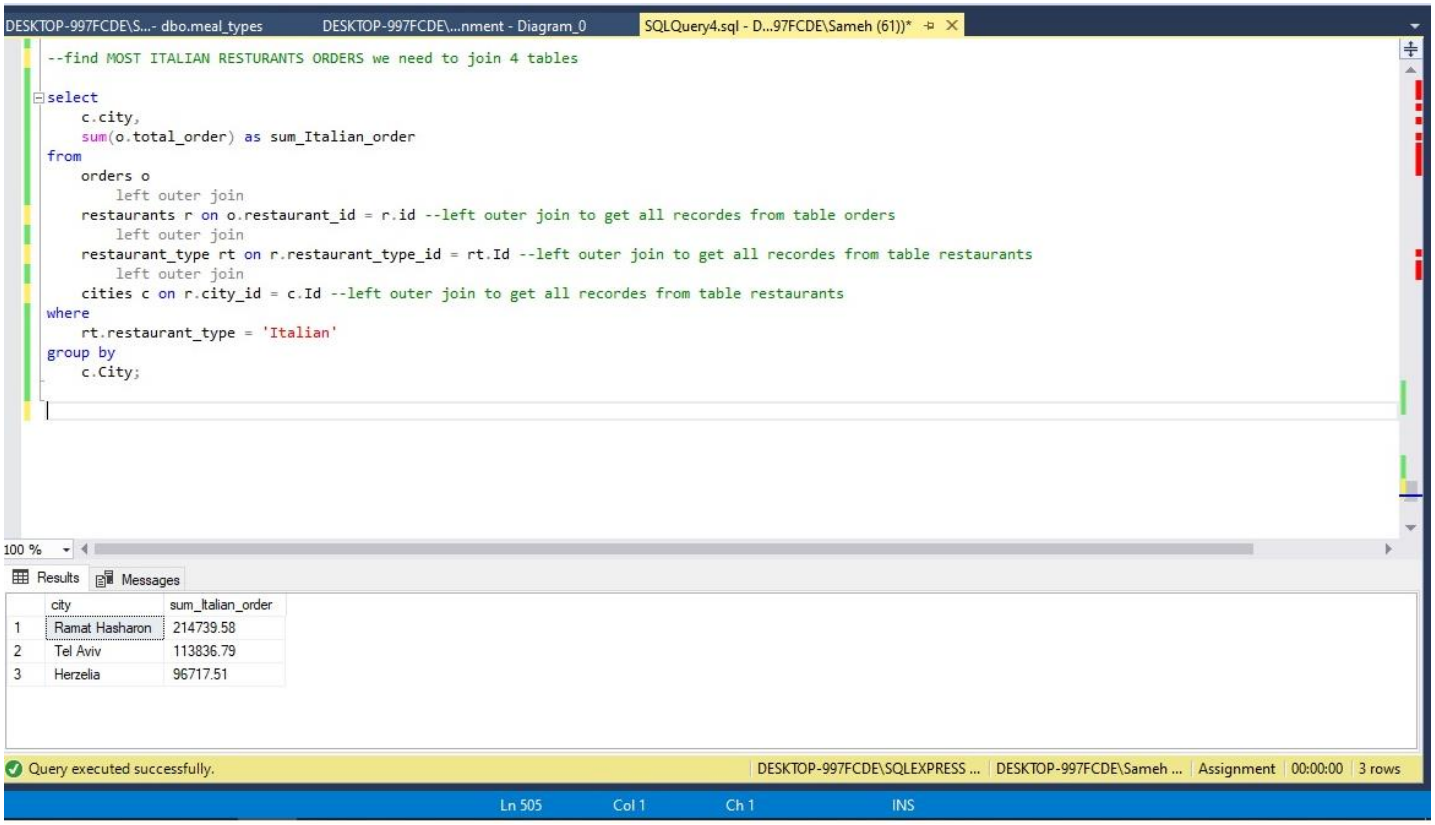


Figure 7

```sql
--find MOST ITALIAN RESTURANTS ORDERS it's necessary to join 4 tables

select
    c.city,
    sum(o.total_order) as sum_Italian_order
from
    orders o
        left outer join
    restaurants r on o.restaurant_id = r.id --left outer join to get all records from table
orders
        left outer join
    restaurant_type rt on r.restaurant_type_id = rt.Id --left outer join to get all records
from table restaurants
        left outer join
    cities c on r.city_id = c.Id --left outer join to get all records from table restaurants
where
    rt.restaurant_type = 'Italian'
group by
    c.City;
```

4. Vegan Meal Orders in Each City:

- Counts and sums vegan meal orders by city using meal_types, meals, order_details, orders, restaurants, and cities. **Figure (8)**

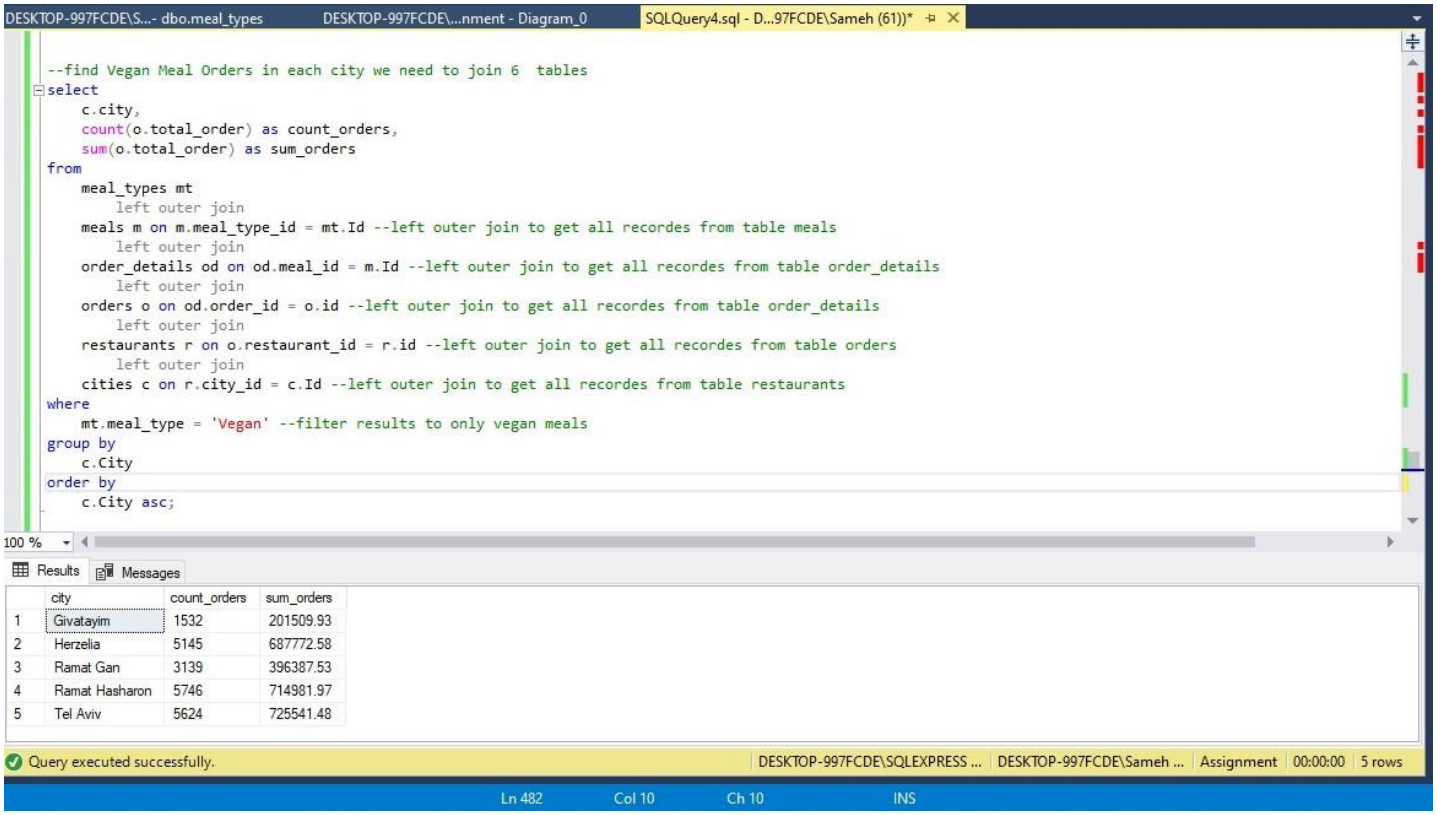


Figure 8

```
--find Vegan Meal Orders in each city it's necessary to join 6 tables
select
    c.city,
    count(o.total_order) as count_orders,
    sum(o.total_order) as sum_orders
from
    meal_types mt
        left outer join
    meals m on m.meal_type_id = mt.Id --left outer join to get all records from table meals
        left outer join
    order_details od on od.meal_id = m.Id --left outer join to get all records from table
order_details
        left outer join
    orders o on od.order_id = o.id --left outer join to get all records from table
order_details
        left outer join
    restaurants r on o.restaurant_id = r.id --left outer join to get all records from table
orders
        left outer join
    cities c on r.city_id = c.Id --left outer join to get all records from table restaurants
```

```
where
    mt.meal_type = 'Vegan' --filter results to only vegan meals
group by
    c.City
order by
    c.City asc;
```

5. Price Range Difference for Hot and Cold Meals

- Calculates price range differences between Hot and Cold meals using meals. **Figure (9)**

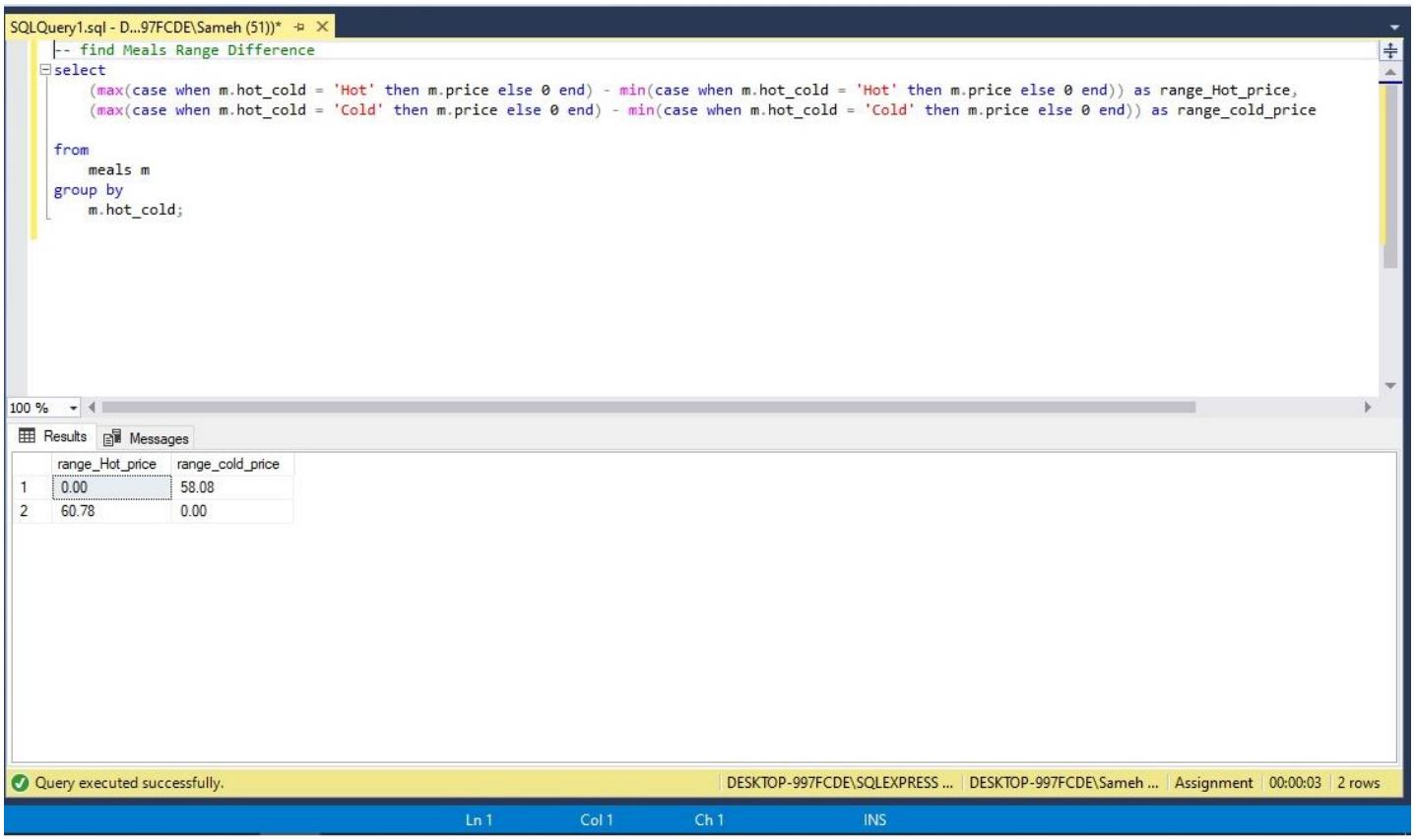


Figure 9

```sql
-- find Meals Range Difference
select
    (max(case when m.hot_cold = 'Hot' then m.price else 0 end) - min(case when m.hot_cold =
'Hot' then m.price else 0 end)) as range_Hot_price,
    (max(case when m.hot_cold = 'Cold' then m.price else 0 end) - min(case when m.hot_cold =
'Cold' then m.price else 0 end)) as range_cold_price

from
    meals m
group by
    m.hot_cold;
```

6. Serve Type Preferences by Gender

- Analyzes the preference for starter, main, and dessert types among different genders. **Figure (10)**

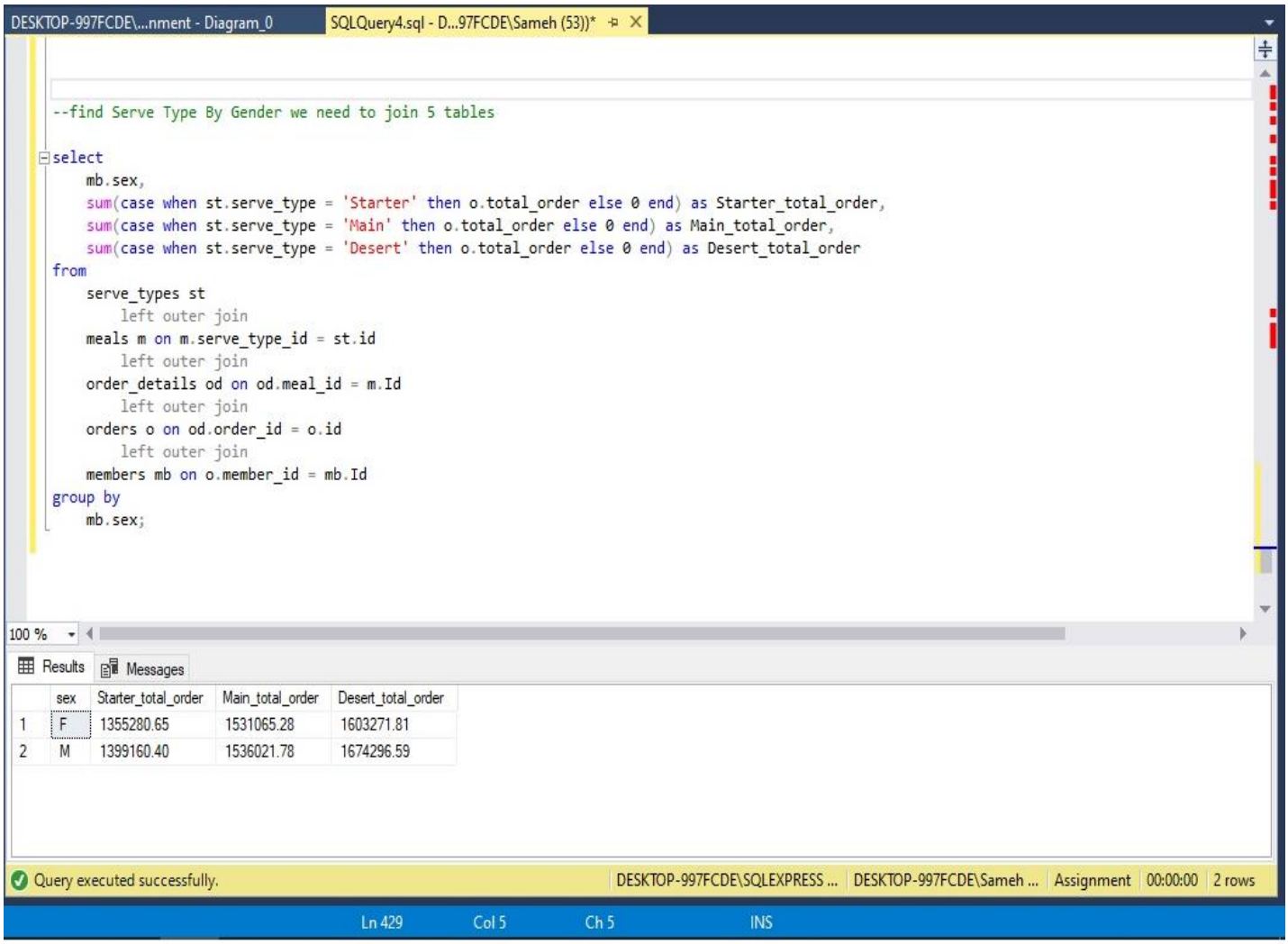


Figure 10

```
--find Serve Type by Gender it’s necessary to join 5 tables

select
    mb.sex,
    sum(case when st.serve_type = 'Starter' then o.total_order else 0 end) as
Starter_total_order,
    sum(case when st.serve_type = 'Main' then o.total_order else 0 end) as Main_total_order,
    sum(case when st.serve_type = 'Desert' then o.total_order else 0 end) as
Desert_total_order
from
    serve_types st
        left outer join
    meals m on m.serve_type_id = st.id
        left outer join
    order_details od on od.meal_id = m.Id
```

```
        left outer join
    orders o on od.order_id = o.id
        left outer join
    members mb on o.member_id = mb.Id
group by
    mb.sex;
```

7. Most Favored Food Types Through Cities

- Highlights preferences for different restaurant types (e.g., Fast Food, Italian) across cities. **Figure (11)**

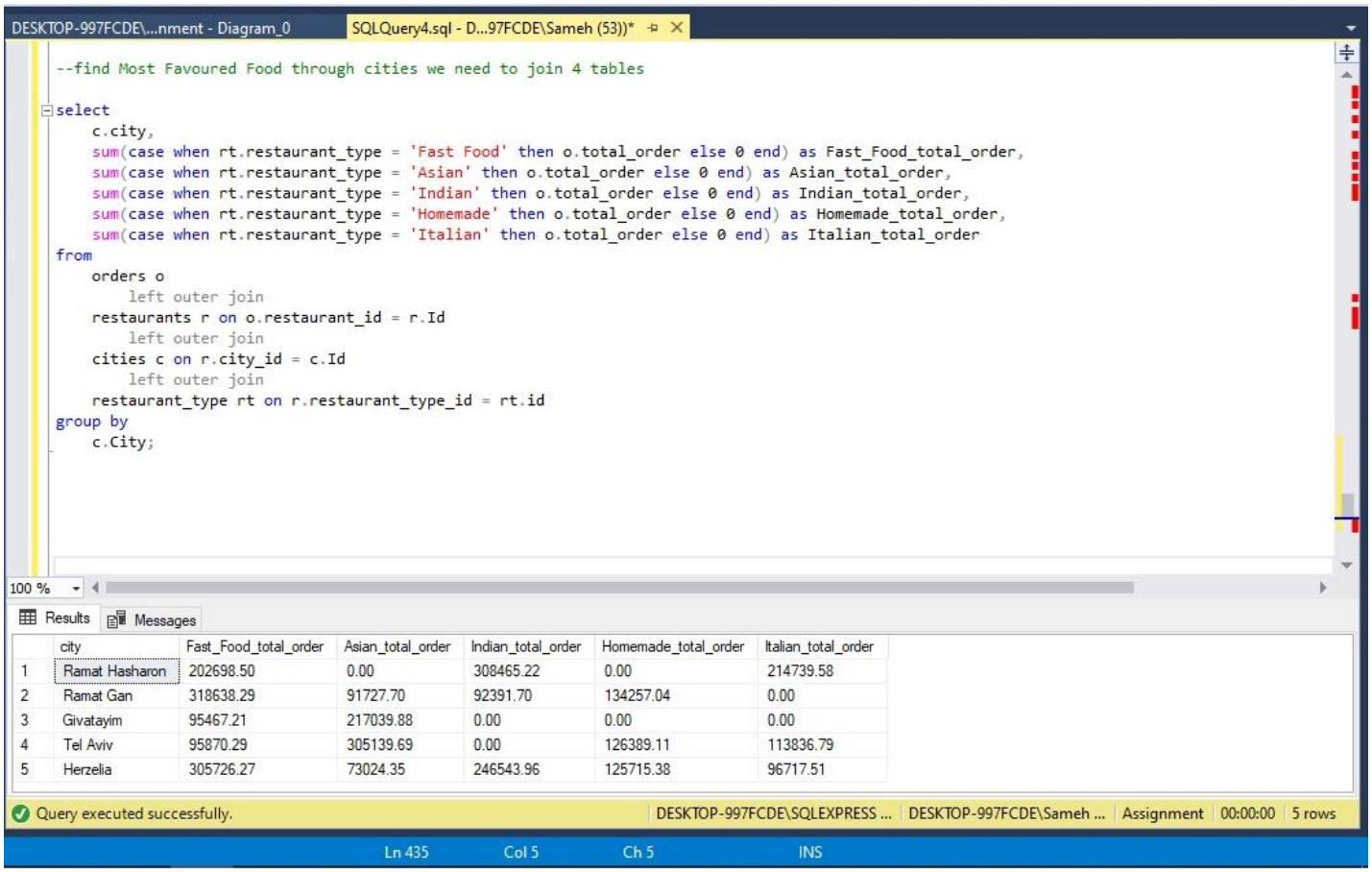


Figure 11

```
--find Most Favored Food through cities it's necessary to join 4 tables
select
    c.city,
    sum(case when rt.restaurant_type = 'Fast Food' then o.total_order else 0 end) as
Fast_Food_total_order,
    sum(case when rt.restaurant_type = 'Asian' then o.total_order else 0 end) as
Asian_total_order,
    sum(case when rt.restaurant_type = 'Indian' then o.total_order else 0 end) as
Indian_total_order,
    sum(case when rt.restaurant_type = 'Homemade' then o.total_order else 0 end) as
Homemade_total_order,
    sum(case when rt.restaurant_type = 'Italian' then o.total_order else 0 end) as
Italian_total_order
from
    orders o
        left outer join
    restaurants r on o.restaurant_id = r.Id
        left outer join
```

```
    cities c on r.city_id = c.Id
        left outer join
    restaurant_type rt on r.restaurant_type_id = rt.id
group by
    c.City;
```

8. Members' Monthly Expenses by Gender

- Shows the monthly expenses split by gender for each city. **Figure (12)**

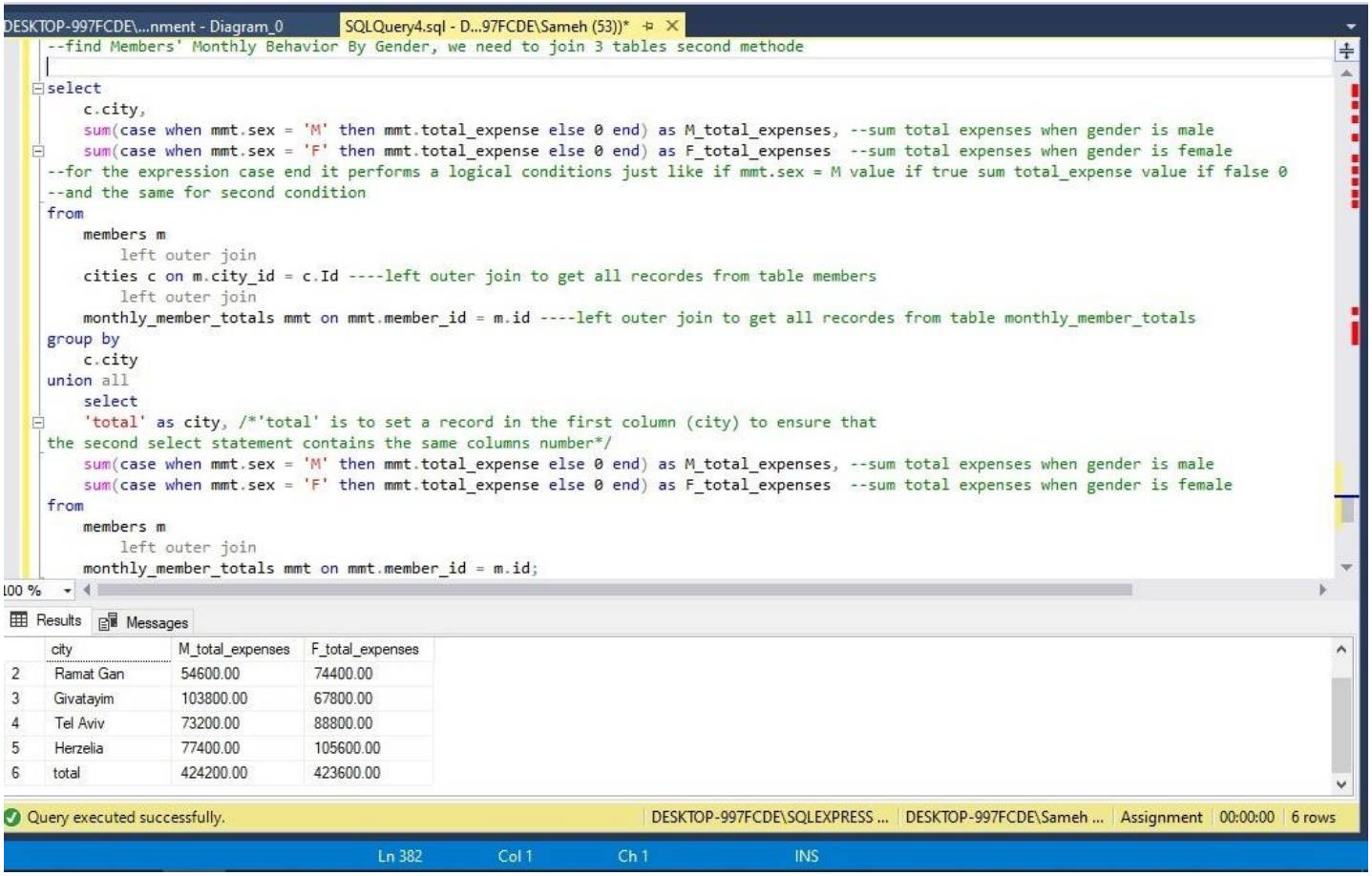


Figure 12

```
--find Members' Monthly Behavior by Gender, it's necessary to join 3 tables

select
    c.city,
    sum(case when mmt.sex = 'M' then mmt.total_expense else 0 end) as M_total_expenses, --
sum total expenses when gender is male
    sum(case when mmt.sex = 'F' then mmt.total_expense else 0 end) as F_total_expenses  --
sum total expenses when gender is female
from
    members m
        left outer join
    cities c on m.city_id = c.Id ----left outer join to get all records from table members
        left outer join
    monthly_member_totals mmt on mmt.member_id = m.id ----left outer join to get all records
from table monthly_member_totals
--WHERE
    --mmt.sex IN ('M', 'F') i tried to it with this command but i got 1 column contains M
and F
```

```sql
group by
    c.city;
--find Members' Monthly Behavior by Gender, it's necessary to join 3 tables second method

select
    c.city,
    sum(case when mmt.sex = 'M' then mmt.total_expense else 0 end) as M_total_expenses, --
sum total expenses when gender is male
    sum(case when mmt.sex = 'F' then mmt.total_expense else 0 end) as F_total_expenses  --
sum total expenses when gender is female
--for the expression case end it performs a logical condition just like if mmt.sex = M value
if true sum total_expense value if false 0
--and the same for second condition
from
    members m
        left outer join
    cities c on m.city_id = c.Id ----left outer join to get all records from table members
        left outer join
    monthly_member_totals mmt on mmt.member_id = m.id ----left outer join to get all records
from table monthly_member_totals
group by
    c.city
union all
    select
    'total' as city, /*'total' is to set a record in the first column (city) to ensure that
the second select statement contains the same columns number*/
    sum(case when mmt.sex = 'M' then mmt.total_expense else 0 end) as M_total_expenses, --
sum total expenses when gender is male
    sum(case when mmt.sex = 'F' then mmt.total_expense else 0 end) as F_total_expenses  --
sum total expenses when gender is female
from
    members m
        left outer join
    monthly_member_totals mmt on mmt.member_id = m.id;
```

9. Average Sales in Each City

- Provides insights into the average sales across different cities. **Figure (13)**

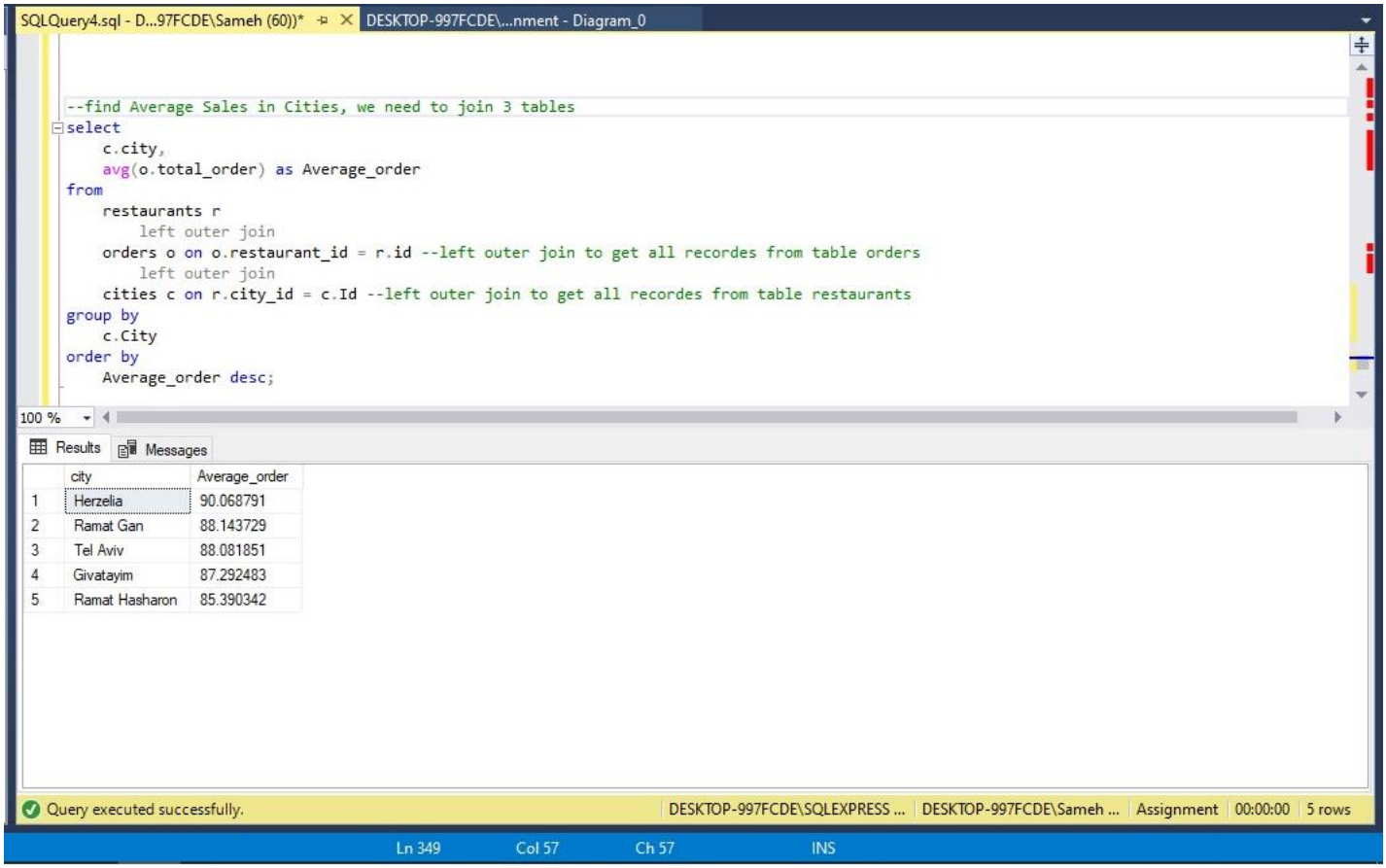


Figure 13

```
--find Average Sales in Cities, it's necessary to join 3 tables
select
    c.city,
    avg(o.total_order) as Average_order
from
    restaurants r
        left outer join
    orders o on o.restaurant_id = r.id --left outer join to get all records from table
orders
        left outer join
    cities c on r.city_id = c.Id --left outer join to get all records from table restaurants
group by
    c.City
order by
    Average_order desc;
```

10. Relationship Between Month and Orders' Volume

- Analyzes the order volume across months for each city. **Figure (14)**

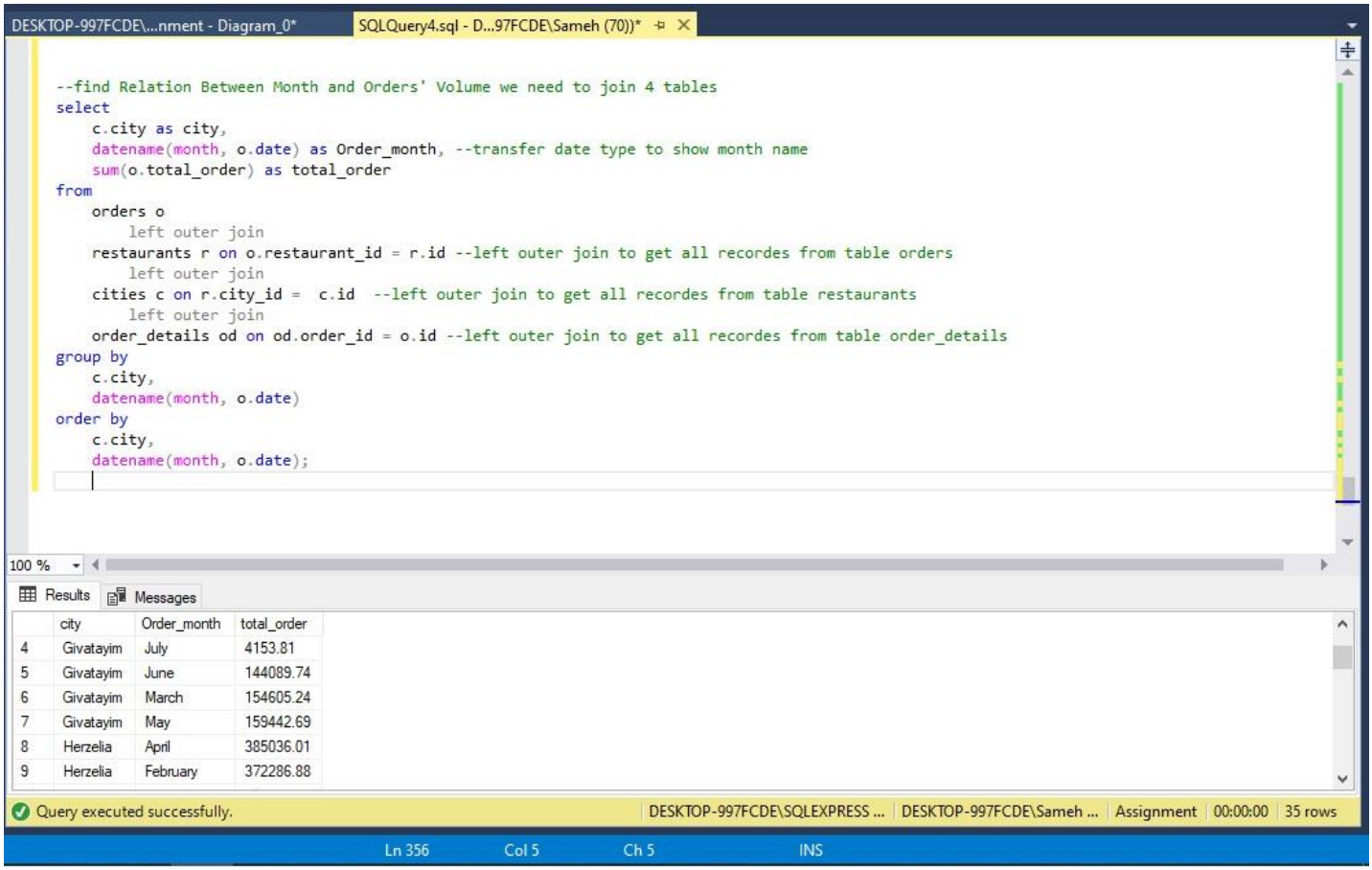


Figure 14

```
--find Relation Between Month and Orders' Volume it's necessary to join 4 tables
select
    c.city as city,
    datename(month, o.date) as Order_month, --transfer date type to show month name
    sum(o.total_order) as total_order
from
    orders o
        left outer join
    restaurants r on o.restaurant_id = r.id --left outer join to get all records from table
orders
        left outer join
    cities c on r.city_id =  c.id  --left outer join to get all records from table
restaurants
        left outer join
    order_details od on od.order_id = o.id --left outer join to get all records from table
order_details
group by
    c.city,
```

```
    datename(month, o.date)
order by
    c.city,
    datename(month, o.date);
```

11. Average Order Value by Gender

- Calculates the average order value for male and female members. **Figure (15)**

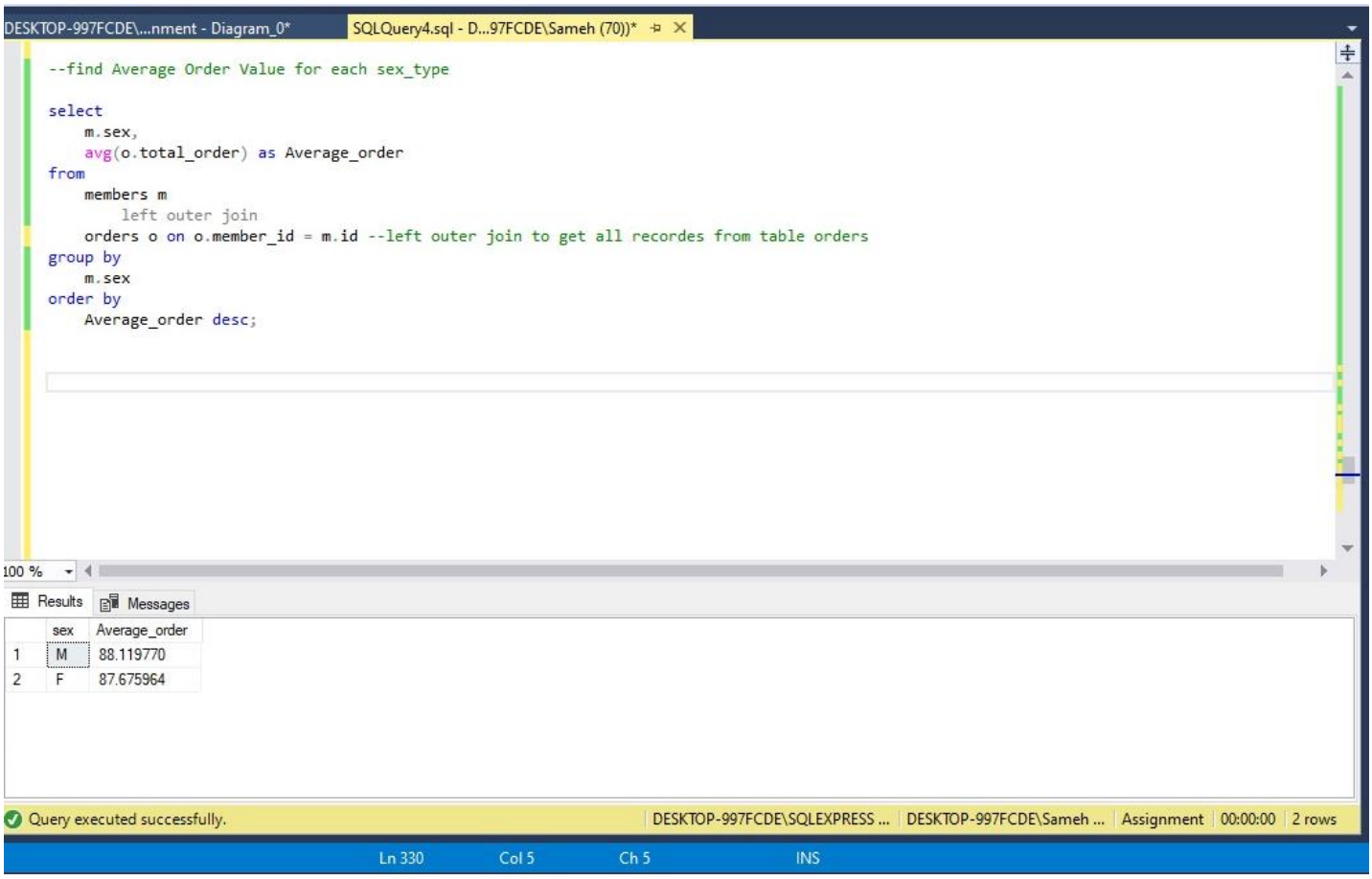


Figure 15

```sql
--find Average Order Value for each sex_type

select
    m.sex,
    avg(o.total_order) as Average_order
from
    members m
        left outer join
    orders o on o.member_id = m.id --left outer join to get all records from table orders
group by
    m.sex
order by
    Average_order desc;
```

12. Top 20 Members with Most Expenses in Each City

- Lists the top spending members in each city based on their total monthly expenses. **Figure (16)**

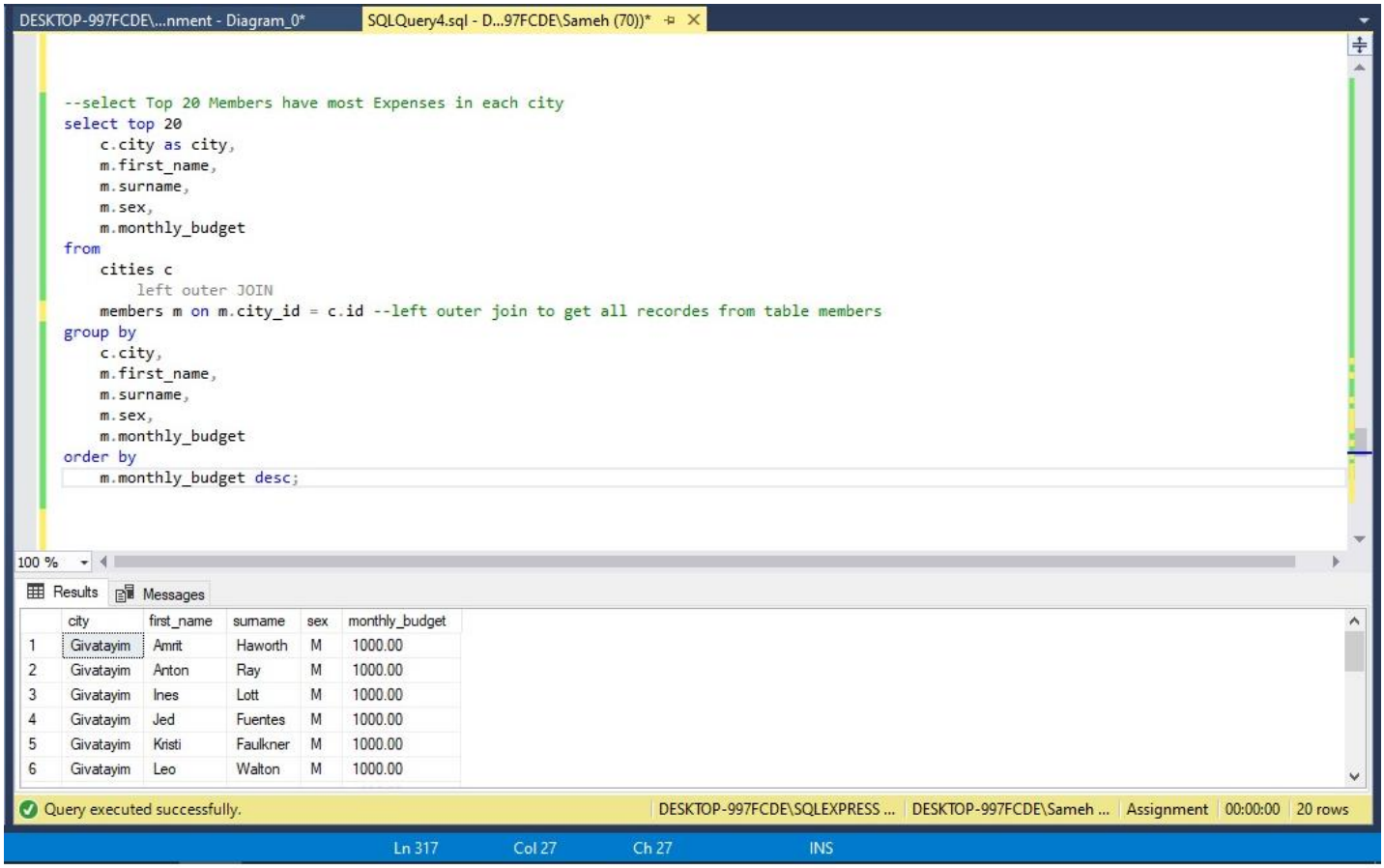


Figure 16

```
--select Top 20 Members have most Expenses in each city
select top 20
    c.city as city,
    m.first_name,
    m.surname,
    m.sex,
    m.monthly_budget
from
    cities c
        left outer JOIN
    members m on m.city_id = c.id --left outer join to get all records from table members
group by
    c.city,
    m.first_name,
    m.surname,
    m.sex,
    m.monthly_budget
order by
    m.monthly_budget desc;
```

13. Most Popular Meal Choices by Gender

- Shows the top meal choices for each gender based on total orders. **Figure (17)**

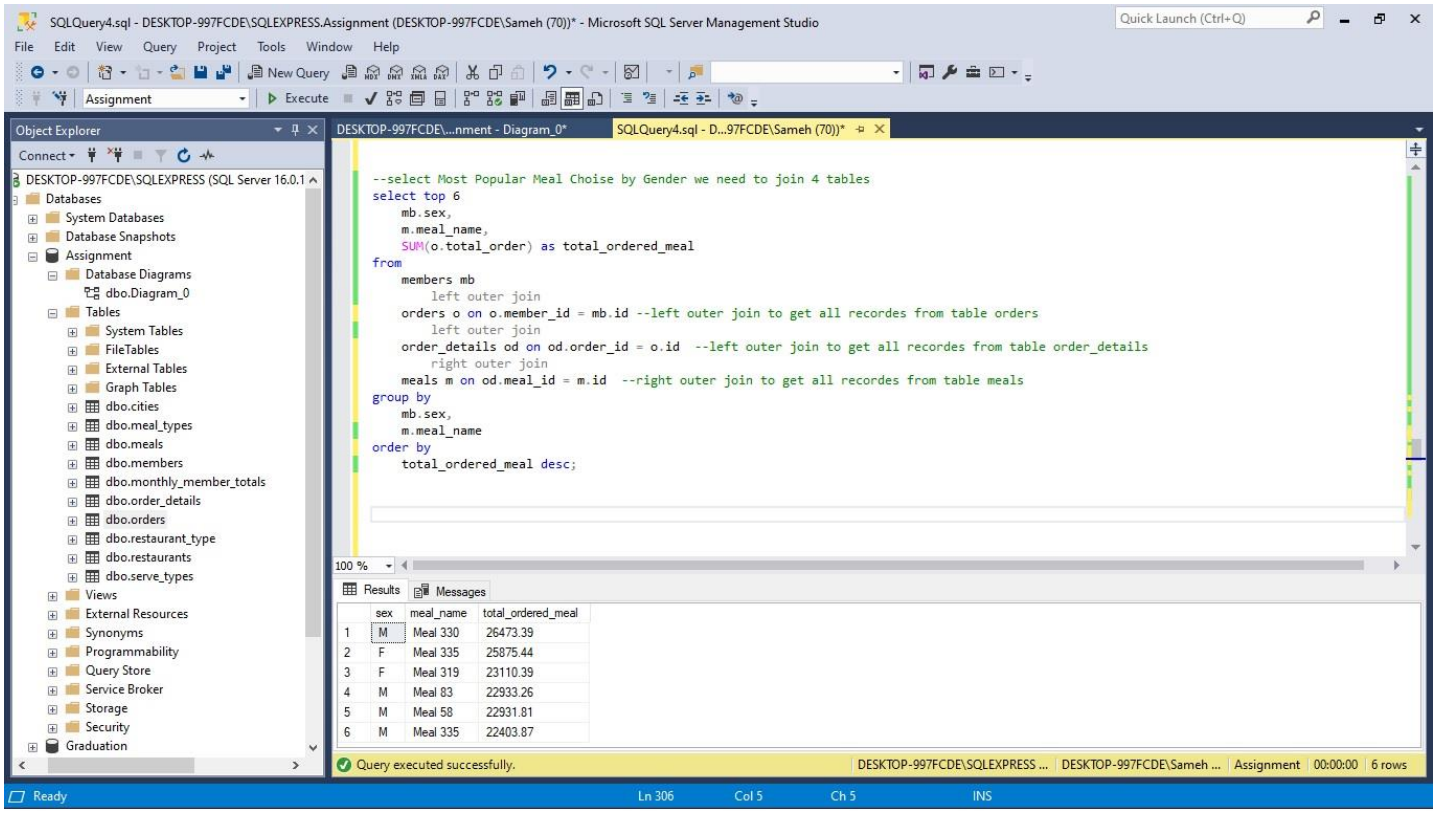


Figure 17

```
--select Most Popular Meal Choice by Gender it's necessary to join 4 tables
select top 6
    mb.sex,
    m.meal_name,
    SUM(o.total_order) as total_ordered_meal
from
    members mb
        left outer join
    orders o on o.member_id = mb.id --left outer join to get all records from table orders
        left outer join
    order_details od on od.order_id = o.id  --left outer join to get all records from table
order_details
        right outer join
    meals m on od.meal_id = m.id  --right outer join to get all records from table meals
group by
    mb.sex,
    m.meal_name
order by
    total_ordered_meal desc;
```

14. Top 5 Meals with Most Sales in Each City

- Highlights the most popular meals in each city based on total orders. **Figure (18)**

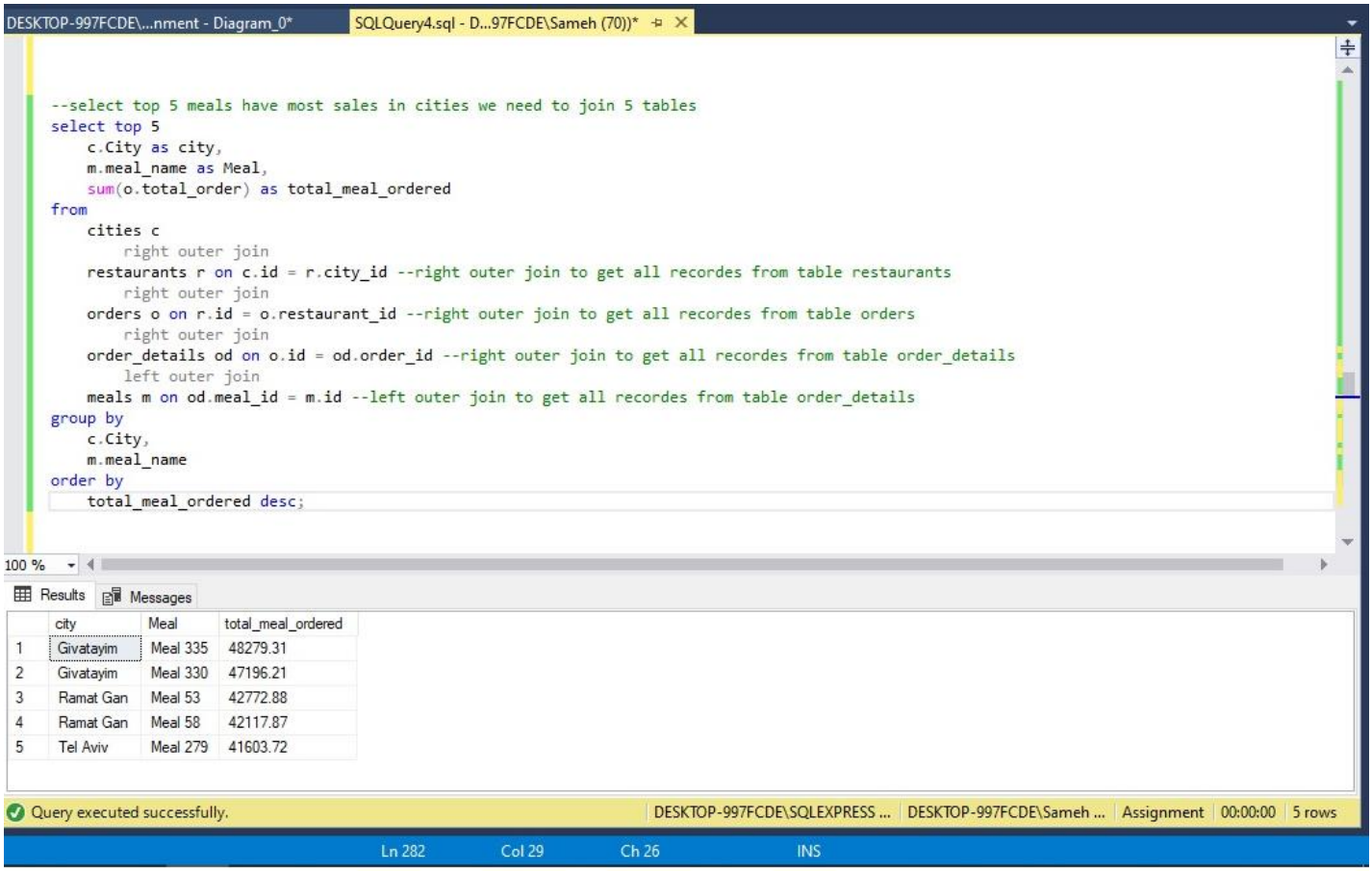


Figure 18

```
--select top 5 meals have most sales in cities it's necessary to join 5 tables

select top 5
    c.City as city,
    m.meal_name as Meal,
    sum(o.total_order) as total_meal_ordered
from
    cities c
        right outer join
    restaurants r on c.id = r.city_id

--right outer join to get all records from table restaurants
        right outer join
    orders o on r.id = o.restaurant_id

 --right outer join to get all records from table orders
```

```
        right outer join
    order_details od on o.id = od.order_id

 --right outer join to get all records from table order_details

        left outer join

    meals m on od.meal_id = m.id

--left outer join to get all records from table order_details
group by
    c.City,
    m.meal_name
order by
    total_meal_ordered desc;
```

15. Top 10 Customers with Highest Monthly Budgets and Total Expenses

- Identifies the top spending customers based on their monthly budgets and total expenses. **Figure (19)**

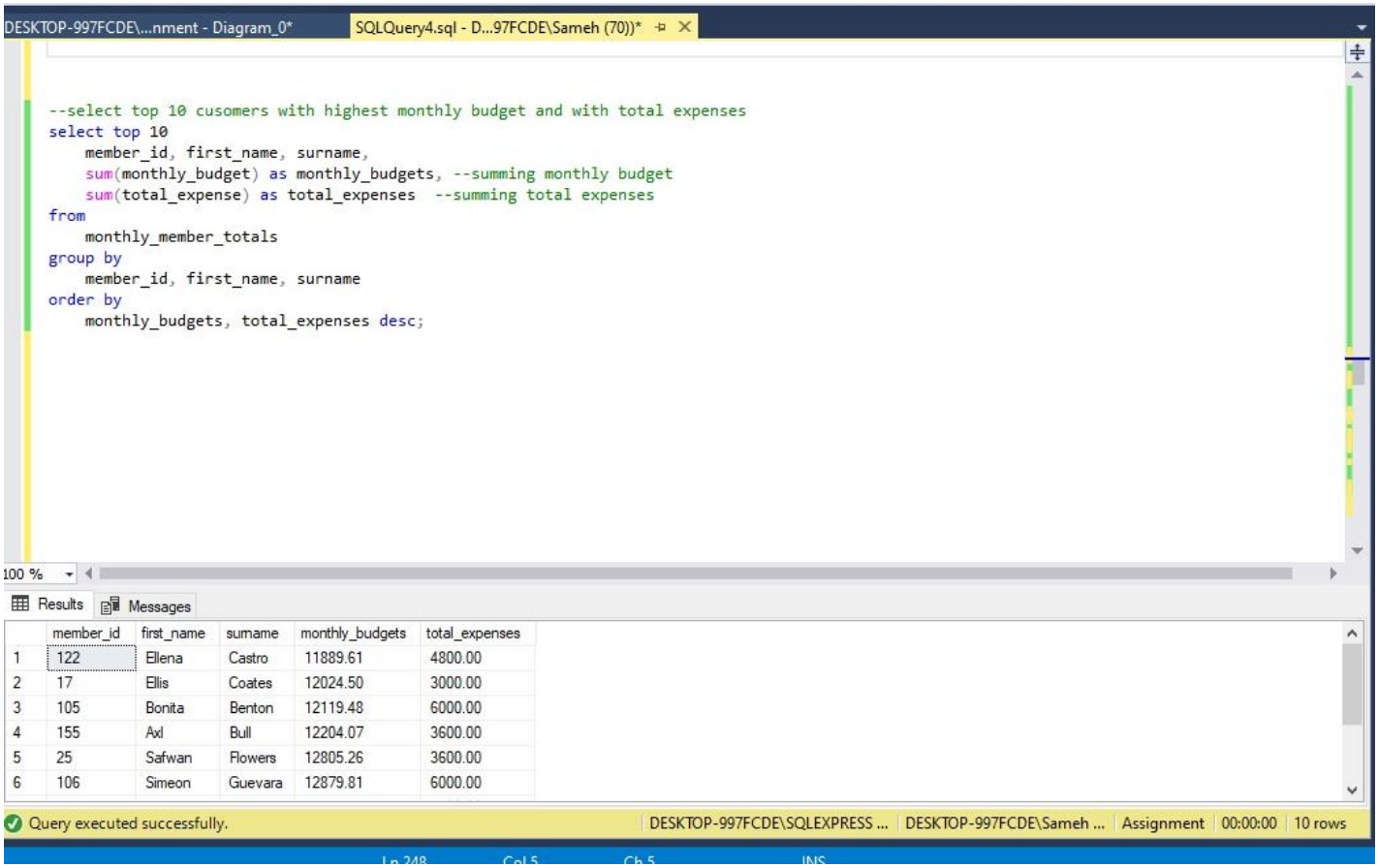


Figure 19

```
--select top 10 cusomers with highest monthly budget and with total expenses
select top 10
    member_id, first_name, surname,
    sum(monthly_budget) as monthly_budgets, --summing monthly budget
    sum(total_expense) as total_expenses  --summing total expenses
from
    monthly_member_totals
group by
    member_id, first_name, surname
order by
    monthly_budgets, total_expenses desc;
```