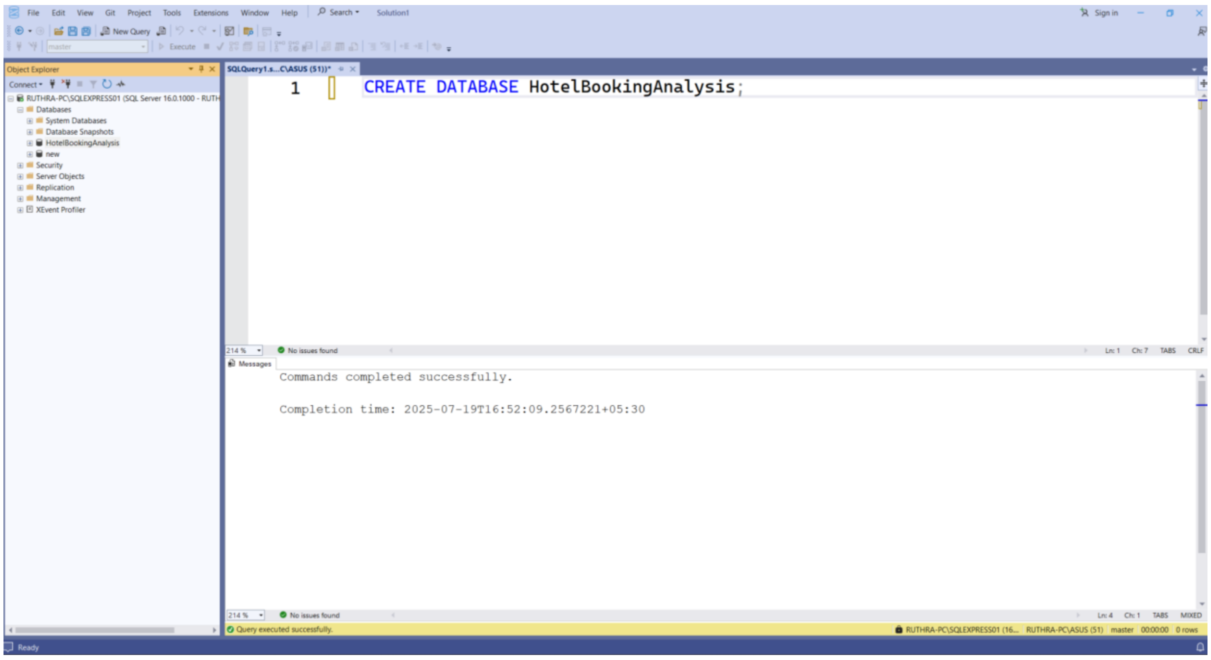
OYO Booking Analysis Case Study

Step 1: Database Setup

First, we'll create a database.



```sql
CREATE DATABASE HotelBookingAnalysis;
```

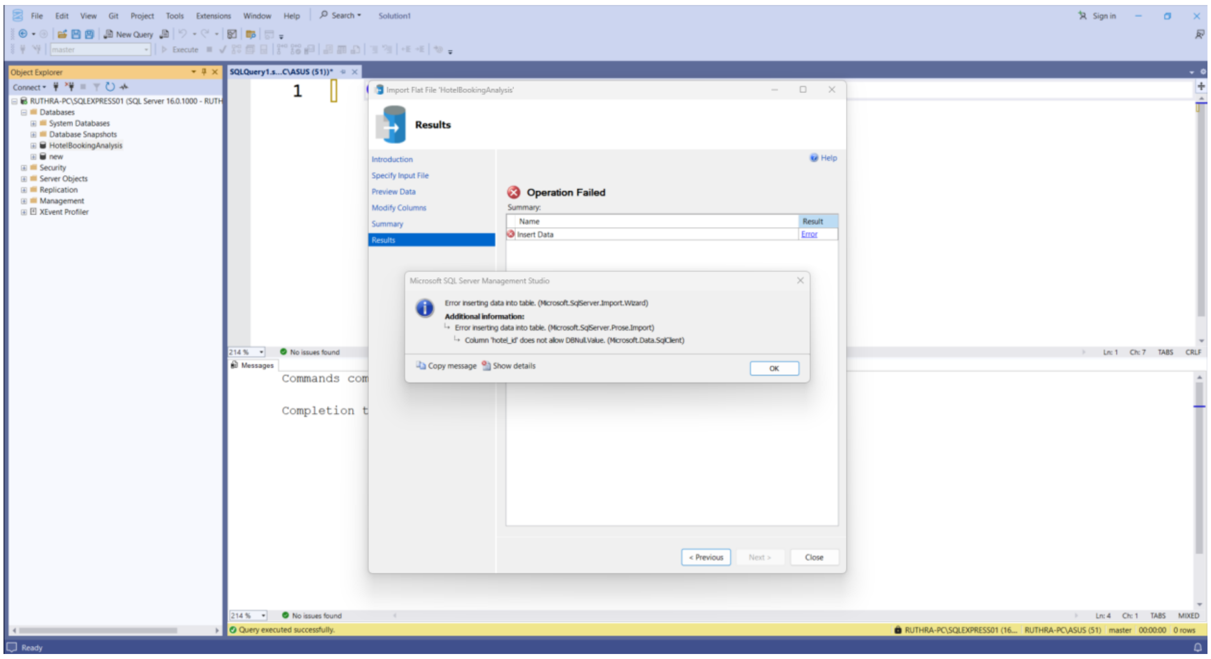
```
Commands completed successfully.

Completion time: 2025-07-19T16:52:09.2567221+05:30
```

Step 2: Import Data

Since you have data in Excel, you can:

1. Save your Excel files as CSV
2. In SSMS, right-click your database → Tasks → Import Data
3. Follow the wizard to import your CSV files to the respective tables



Import Flat File 'HotelBookingAnalysis'

Results

Operation Failed

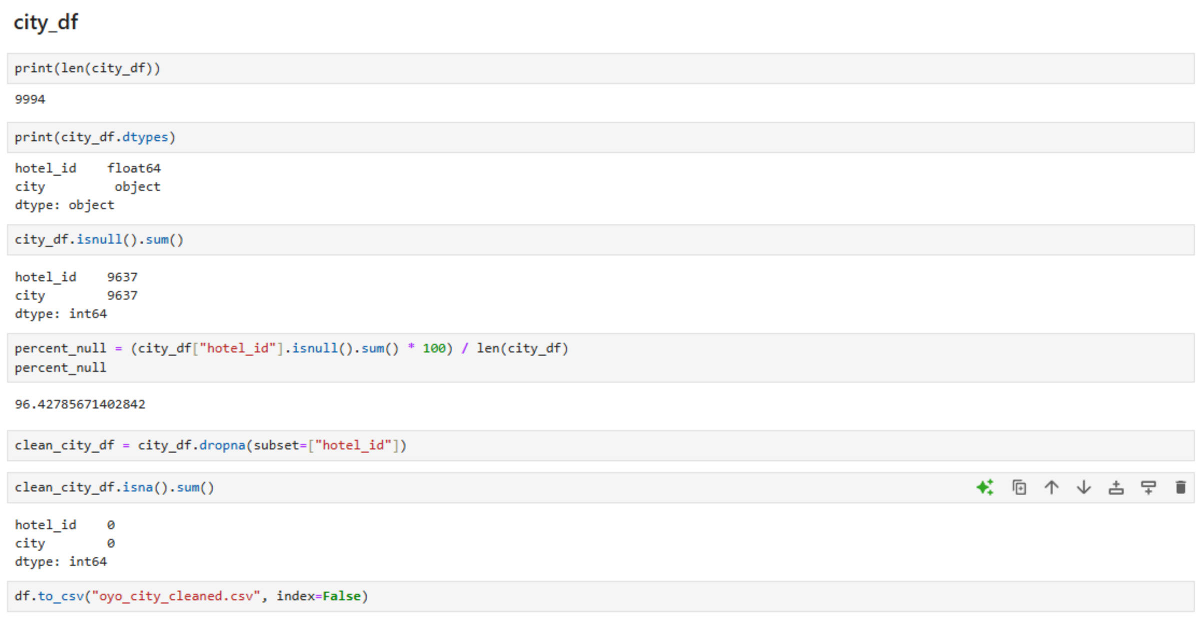
Summary:

Name	Result
Insert Data	Error

```
Error inserting data into table. (Microsoft.SqlServer.Import.Wizard)
Additional information:
  ↳ Error inserting data into table. (Microsoft.SqlServer.Prose.Import)
    ↳ Column 'hotel_id' does not allow DBNull.Value. (Microsoft.Data.SqlClient)
```

There are some empty fields in the data, so the data was not imported properly. Therefore, we have to clean the data, and I chose to clean the data using pandas.`

Data Cleaning:



city_df

```
print(len(city_df))
```

```
9994
```

```
print(city_df.dtypes)
```

```
hotel_id    float64
city         object
dtype: object
```

```
city_df.isnull().sum()
```

```
hotel_id    9637
city        9637
dtype: int64
```

```
percent_null = (city_df["hotel_id"].isnull().sum() * 100) / len(city_df)
percent_null
```

```
96.42785671402842
```

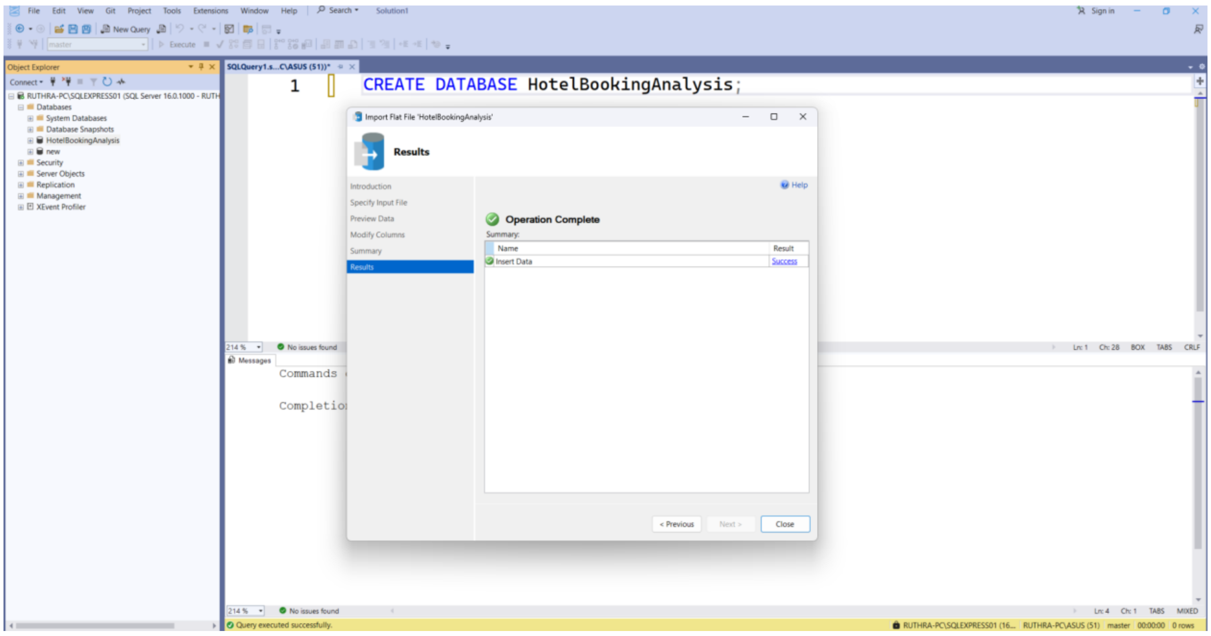
```
clean_city_df = city_df.dropna(subset=["hotel_id"])
```

```
clean_city_df.isna().sum()
```

```
hotel_id    0
city        0
dtype: int64
```

```
df.to_csv("oyo_city_cleaned.csv", index=False)
```

Data Insertion is successful after cleaning the Data.



```
CREATE DATABASE HotelBookingAnalysis;
```

Import Flat File 'HotelBookingAnalysis'

Results

Operation Complete

Summary:

Name	Result
Insert Data	Success

Commands

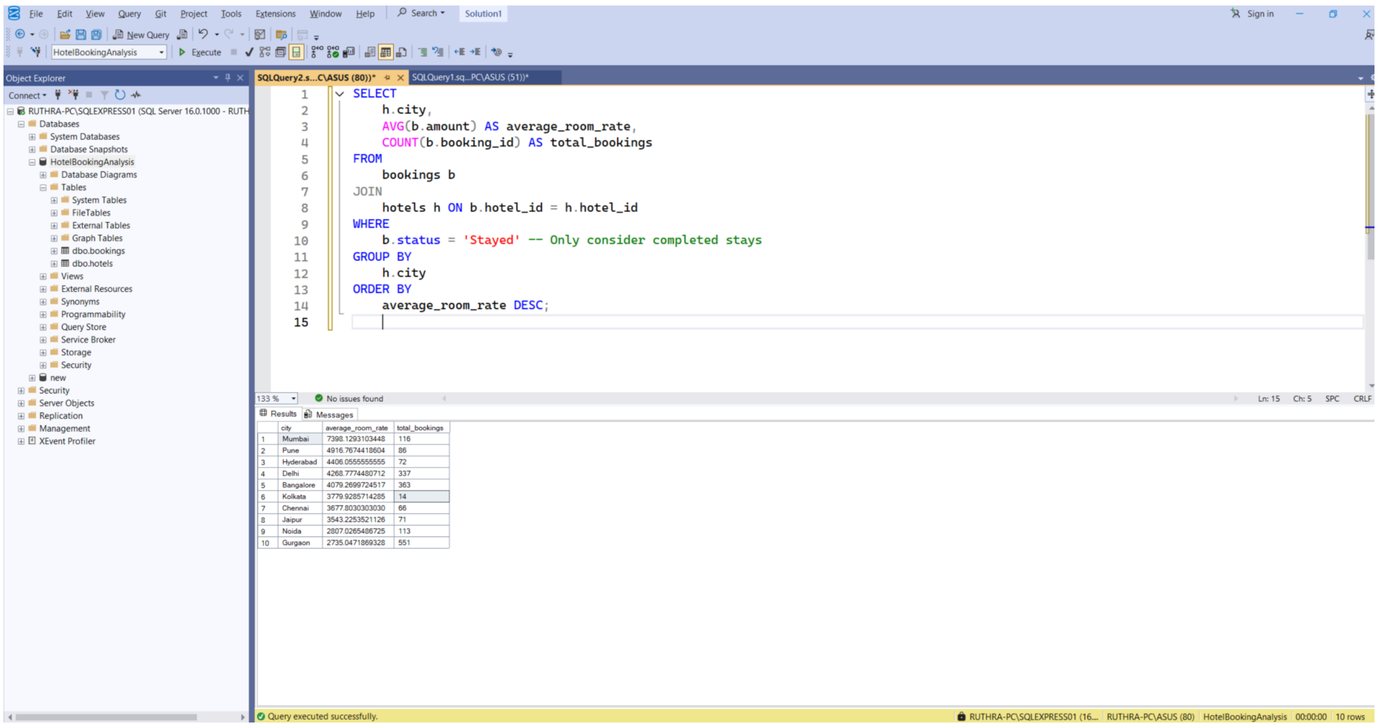
Completio

Query executed successfully.

Data Base is Populated **Successfully** by importing the data from the csv file.

Step 3: Analysis Queries

1. Average Room Rates by City

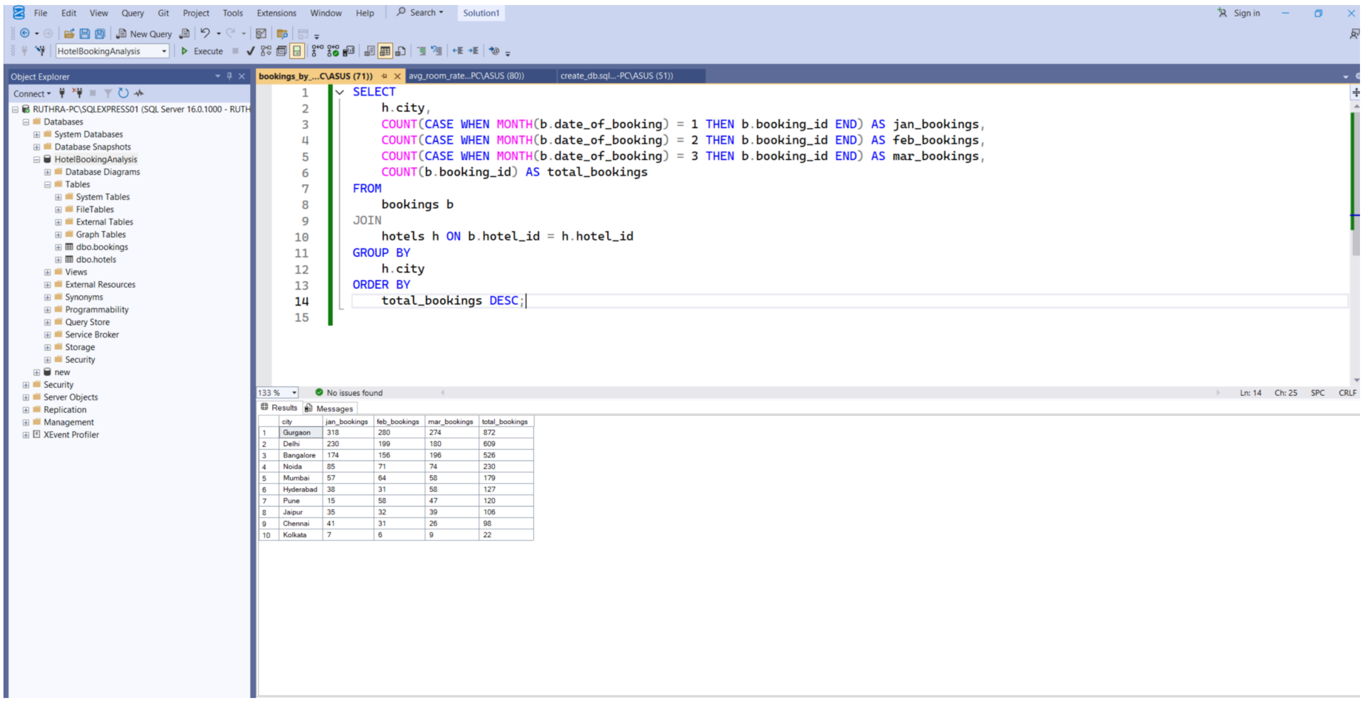


```sql
SELECT
    h.city,
    AVG(b.amount) AS average_room_rate,
    COUNT(b.booking_id) AS total_bookings
FROM
    bookings b
JOIN
    hotels h ON b.hotel_id = h.hotel_id
WHERE
    b.status = 'Stayed' -- Only consider completed stays
GROUP BY
    h.city
ORDER BY
    average_room_rate DESC;
```

	city	average_room_rate	total_bookings
1	Mumbai	7398.1293103448	116
2	Pune	4916.7674418604	86
3	Hyderabad	4406.0555555555	72
4	Delhi	4268.7774480712	337
5	Bangalore	4079.2699724517	363
6	Kolkata	3779.9285714285	14
7	Chennai	3677.8030303030	66
8	Jaipur	3543.2253521126	71
9	Noida	2807.0265486725	113
10	Gurgaon	2735.0471869328	551

Query executed successfully.

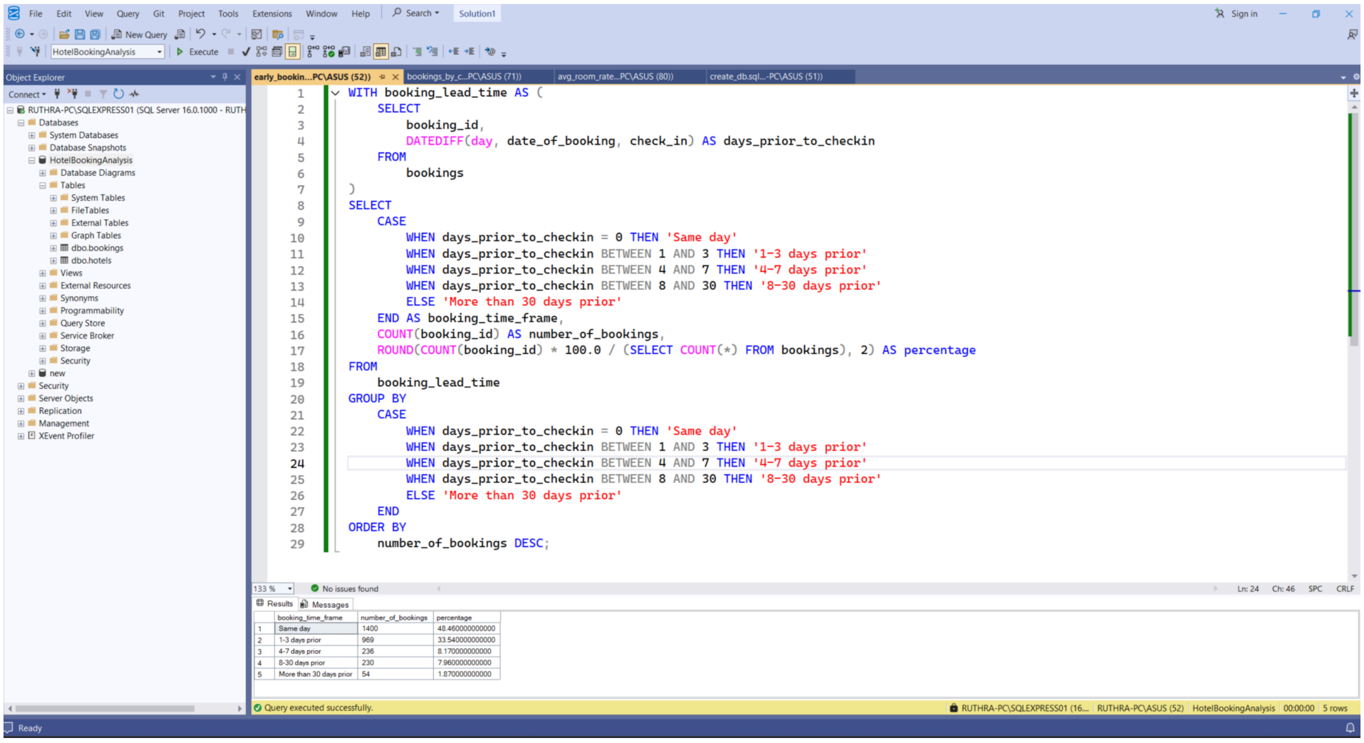
2. Bookings by City (Jan-Feb-Mar)



```sql
SELECT
    h.city,
    COUNT(CASE WHEN MONTH(b.date_of_booking) = 1 THEN b.booking_id END) AS jan_bookings,
    COUNT(CASE WHEN MONTH(b.date_of_booking) = 2 THEN b.booking_id END) AS feb_bookings,
    COUNT(CASE WHEN MONTH(b.date_of_booking) = 3 THEN b.booking_id END) AS mar_bookings,
    COUNT(b.booking_id) AS total_bookings
FROM
    bookings b
JOIN
    hotels h ON b.hotel_id = h.hotel_id
GROUP BY
    h.city
ORDER BY
    total_bookings DESC;
```

	city	jan_bookings	feb_bookings	mar_bookings	total_bookings
1	Gurgaon	318	280	274	872
2	Delhi	230	199	180	609
3	Bangalore	174	156	196	526
4	Noida	85	71	74	230
5	Mumbai	57	64	58	179
6	Hyderabad	38	31	58	127
7	Pune	15	58	47	120
8	Jaipur	35	32	39	106
9	Chennai	41	31	26	98
10	Kolkata	7	6	9	22

3. Frequency of Early Bookings (Days Prior to Check-in)

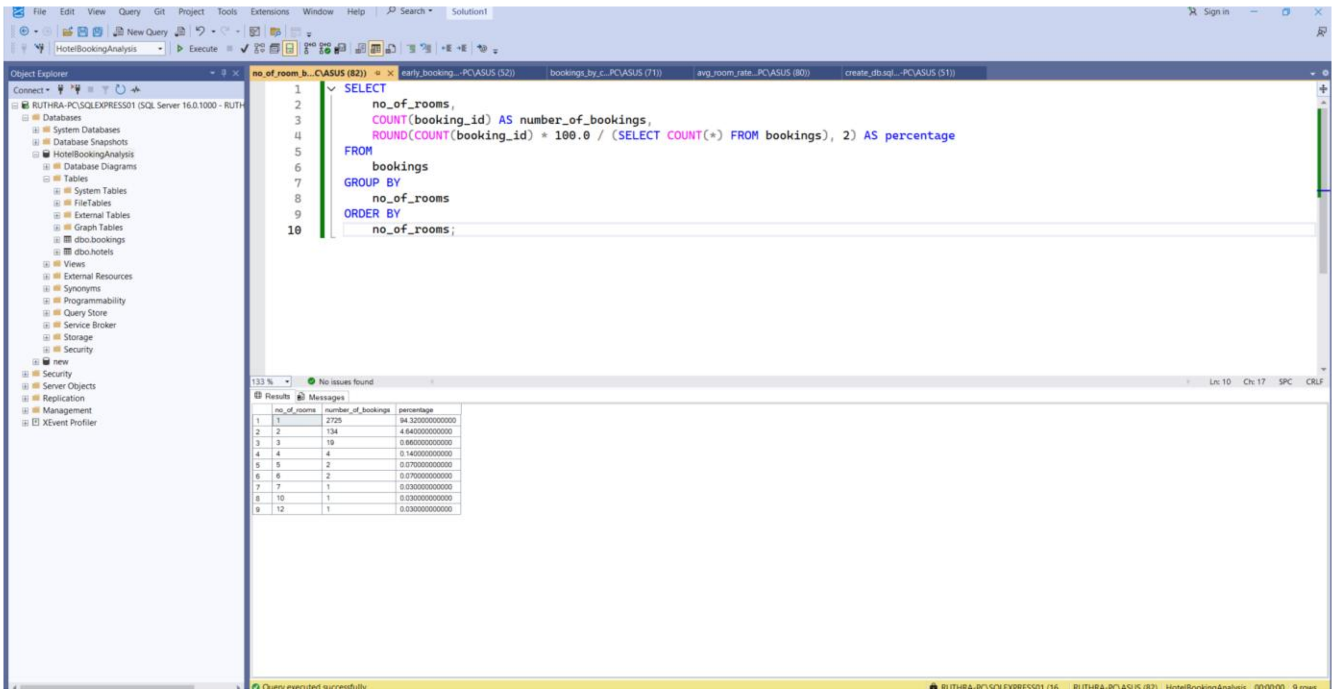


```sql
WITH booking_lead_time AS (
    SELECT
        booking_id,
        DATEDIFF(day, date_of_booking, check_in) AS days_prior_to_checkin
    FROM
        bookings
)
SELECT
    CASE
        WHEN days_prior_to_checkin = 0 THEN 'Same day'
        WHEN days_prior_to_checkin BETWEEN 1 AND 3 THEN '1-3 days prior'
        WHEN days_prior_to_checkin BETWEEN 4 AND 7 THEN '4-7 days prior'
        WHEN days_prior_to_checkin BETWEEN 8 AND 30 THEN '8-30 days prior'
        ELSE 'More than 30 days prior'
    END AS booking_time_frame,
    COUNT(booking_id) AS number_of_bookings,
    ROUND(COUNT(booking_id) * 100.0 / (SELECT COUNT(*) FROM bookings), 2) AS percentage
FROM
    booking_lead_time
GROUP BY
    CASE
        WHEN days_prior_to_checkin = 0 THEN 'Same day'
        WHEN days_prior_to_checkin BETWEEN 1 AND 3 THEN '1-3 days prior'
        WHEN days_prior_to_checkin BETWEEN 4 AND 7 THEN '4-7 days prior'
        WHEN days_prior_to_checkin BETWEEN 8 AND 30 THEN '8-30 days prior'
        ELSE 'More than 30 days prior'
    END
ORDER BY
    number_of_bookings DESC;
```

	booking_time_frame	number_of_bookings	percentage
1	Same day	1400	48.460000000000
2	1-3 days prior	969	33.540000000000
3	4-7 days prior	236	8.170000000000
4	8-30 days prior	230	7.960000000000
5	More than 30 days prior	54	1.870000000000

Query executed successfully.

4. Frequency of Number of Rooms Booked

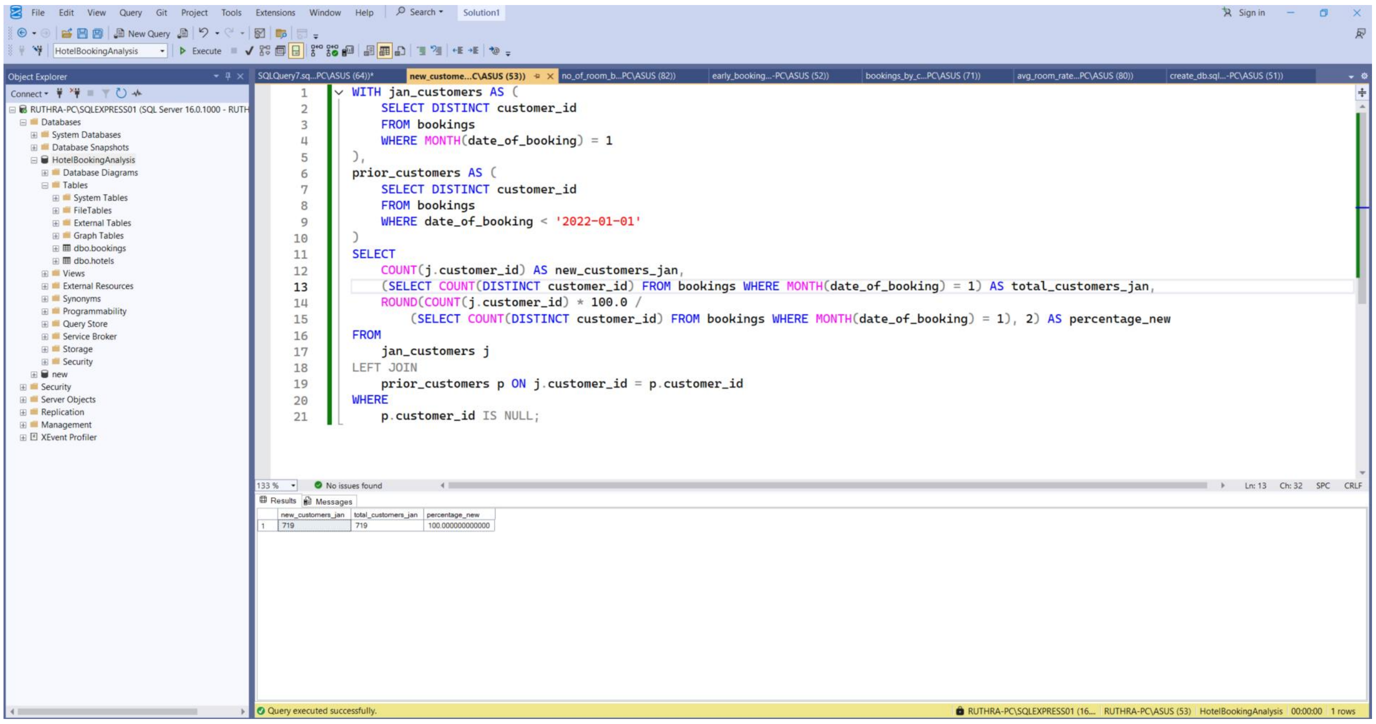


```sql
SELECT
    no_of_rooms,
    COUNT(booking_id) AS number_of_bookings,
    ROUND(COUNT(booking_id) * 100.0 / (SELECT COUNT(*) FROM bookings), 2) AS percentage
FROM
    bookings
GROUP BY
    no_of_rooms
ORDER BY
    no_of_rooms;
```

	no_of_rooms	number_of_bookings	percentage
1	1	2725	94.320000000000
2	2	134	4.640000000000
3	3	19	0.660000000000
4	4	4	0.140000000000
5	5	2	0.070000000000
6	6	2	0.070000000000
7	7	1	0.030000000000
8	10	1	0.030000000000
9	12	1	0.030000000000

Query executed successfully.

5. New Customers in January

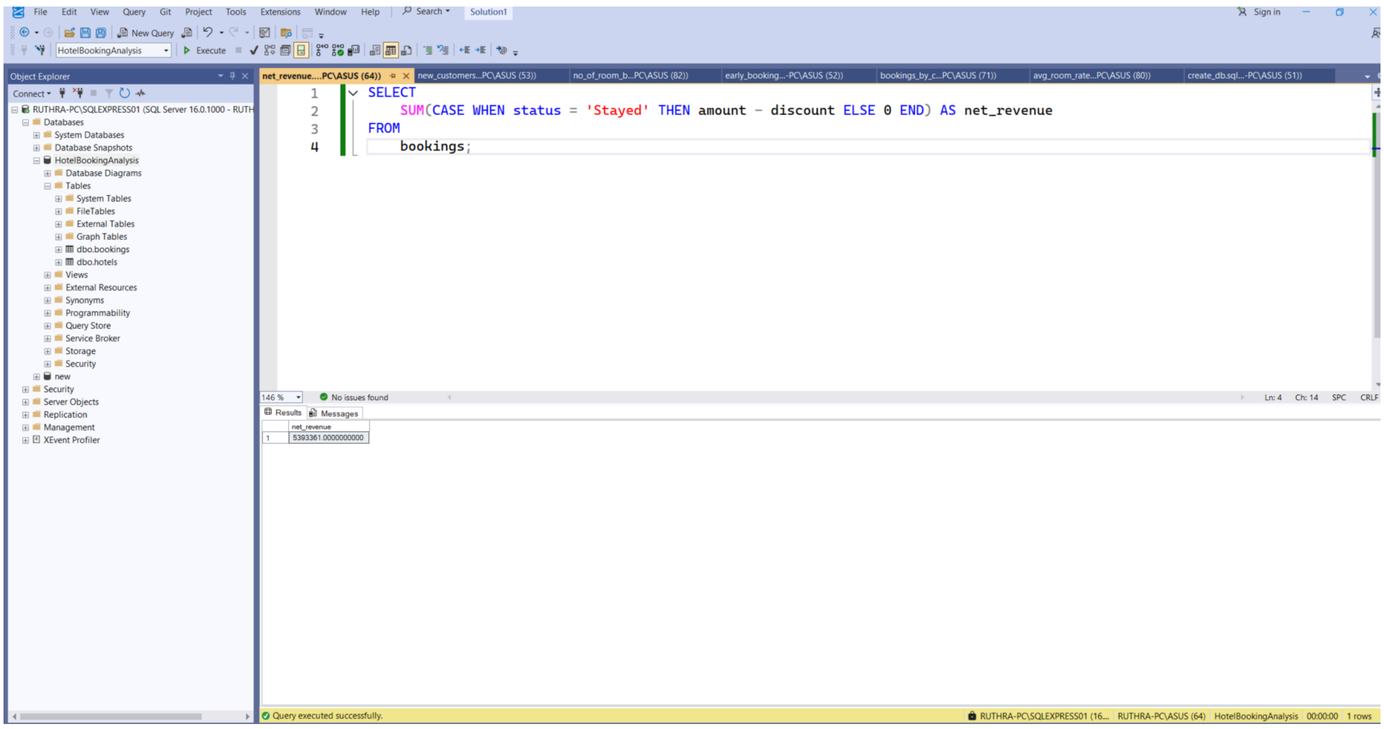


```sql
WITH jan_customers AS (
    SELECT DISTINCT customer_id
    FROM bookings
    WHERE MONTH(date_of_booking) = 1
),
prior_customers AS (
    SELECT DISTINCT customer_id
    FROM bookings
    WHERE date_of_booking < '2022-01-01'
)
SELECT
    COUNT(j.customer_id) AS new_customers_jan,
    (SELECT COUNT(DISTINCT customer_id) FROM bookings WHERE MONTH(date_of_booking) = 1) AS total_customers_jan,
    ROUND(COUNT(j.customer_id) * 100.0 /
        (SELECT COUNT(DISTINCT customer_id) FROM bookings WHERE MONTH(date_of_booking) = 1), 2) AS percentage_new
FROM
    jan_customers j
LEFT JOIN
    prior_customers p ON j.customer_id = p.customer_id
WHERE
    p.customer_id IS NULL;
```

	new_customers_jan	total_customers_jan	percentage_new
1	719	719	100.000000000000

Query executed successfully.

6. Net Revenue (After Cancellations)

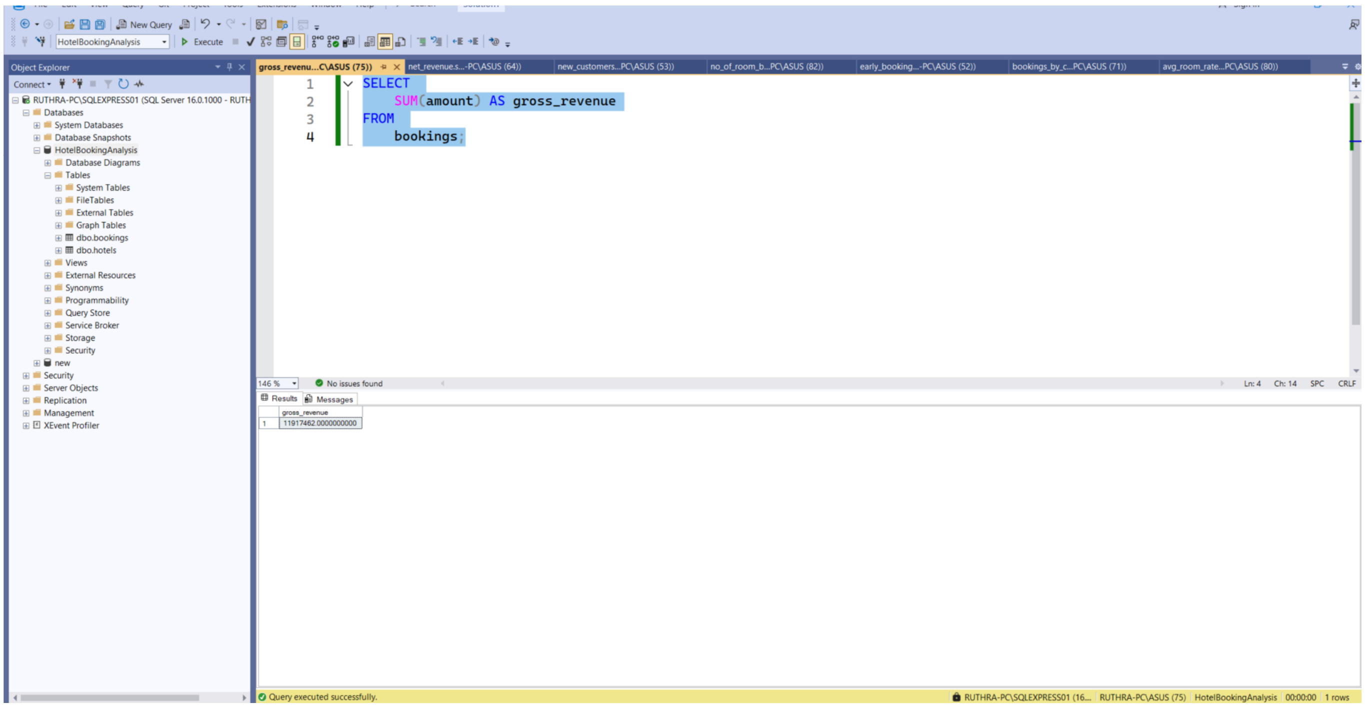


```sql
SELECT
    SUM(CASE WHEN status = 'Stayed' THEN amount - discount ELSE 0 END) AS net_revenue
FROM
    bookings;
```

	net_revenue
1	5393361.0000000000

Query executed successfully.

7. Gross Revenue (Total Booked Amount)

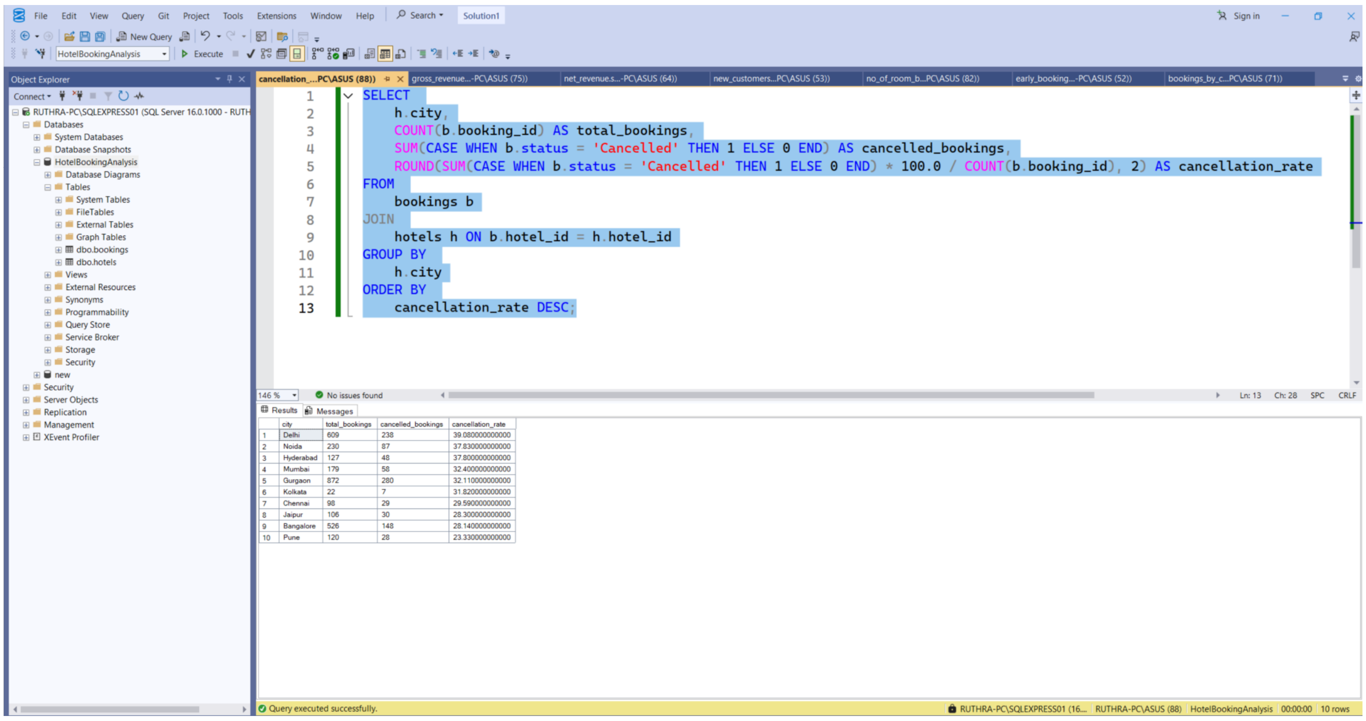


```sql
SELECT
    SUM(amount) AS gross_revenue
FROM
    bookings;
```

	gross_revenue
1	11917462.0000000000

Query executed successfully.

8. Cancellation Rate by City



```sql
SELECT
    h.city,
    COUNT(b.booking_id) AS total_bookings,
    SUM(CASE WHEN b.status = 'Cancelled' THEN 1 ELSE 0 END) AS cancelled_bookings,
    ROUND(SUM(CASE WHEN b.status = 'Cancelled' THEN 1 ELSE 0 END) * 100.0 / COUNT(b.booking_id), 2) AS cancellation_rate
FROM
    bookings b
JOIN
    hotels h ON b.hotel_id = h.hotel_id
GROUP BY
    h.city
ORDER BY
    cancellation_rate DESC;
```

	city	total_bookings	cancelled_bookings	cancellation_rate
1	Delhi	609	238	39.080000000000
2	Noida	230	87	37.830000000000
3	Hyderabad	127	48	37.800000000000
4	Mumbai	179	58	32.400000000000
5	Gurgaon	872	280	32.110000000000
6	Kolkata	22	7	31.820000000000
7	Chennai	98	29	29.590000000000
8	Jaipur	106	30	28.300000000000
9	Bangalore	526	148	28.140000000000
10	Pune	120	28	23.330000000000

Query executed successfully.