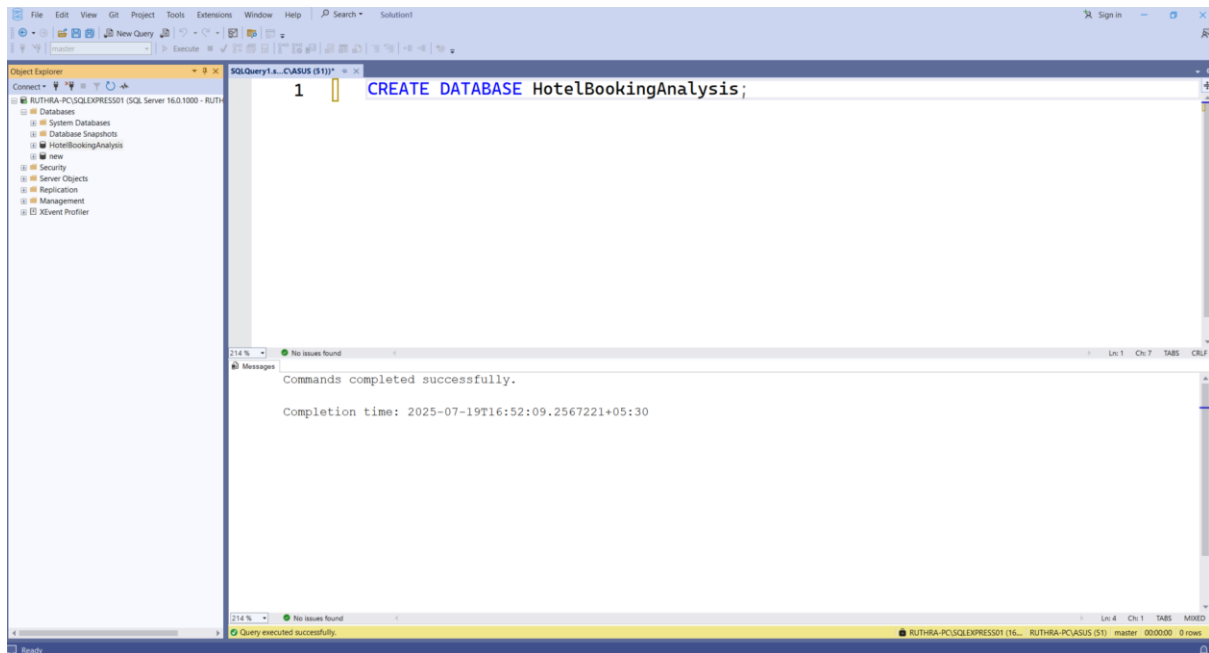


OYO Booking Analysis Case Study

Step 1: Database Setup

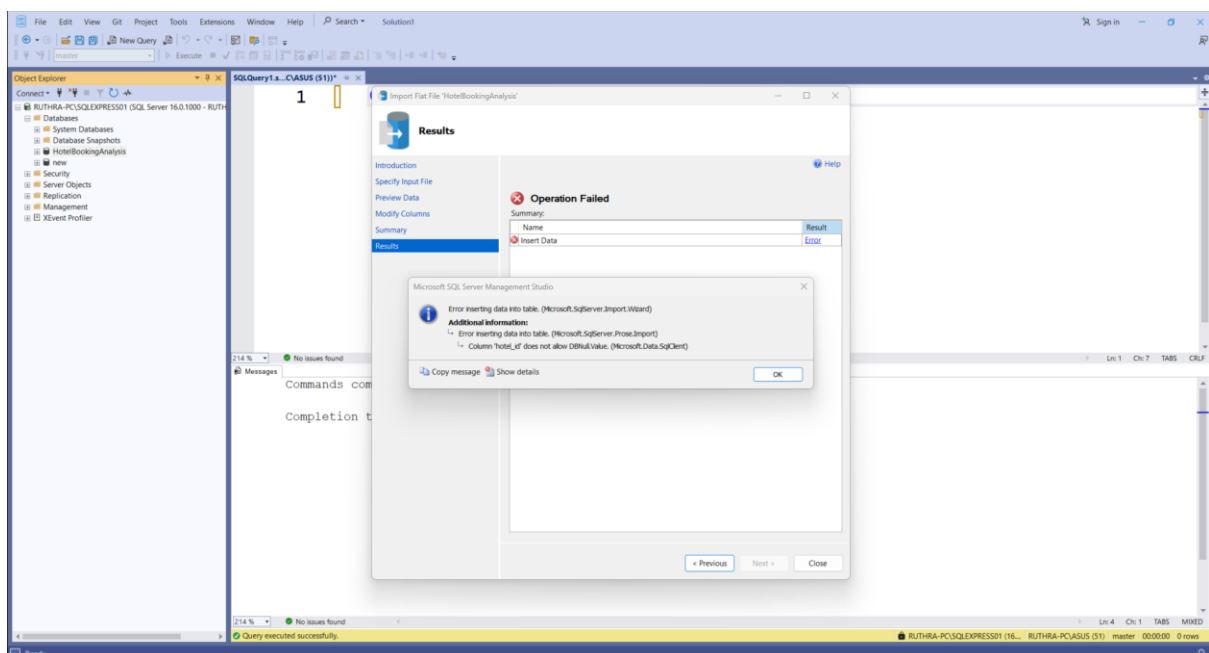
First, we'll create a database.



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



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

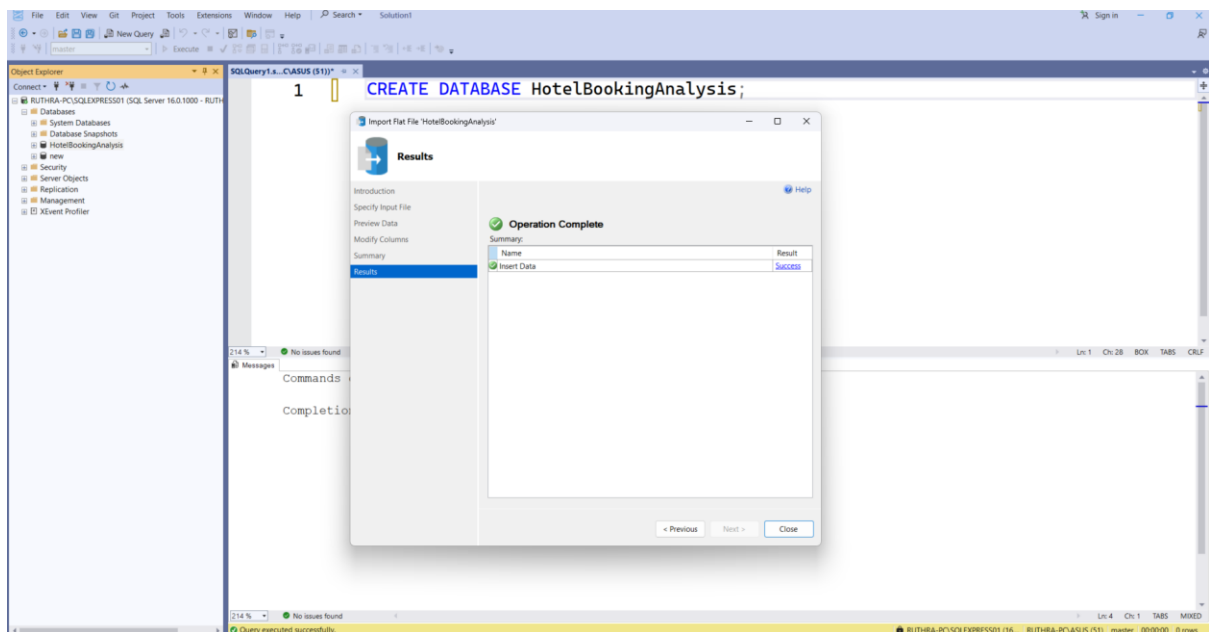
```
clean_city_df = city_df.dropna(subset=["hotel_id"])
```

```
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.



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

Step 3: Analysis Queries

1. Average Room Rates by City

The screenshot shows the SQL Server Enterprise Manager interface. The Object Explorer on the left displays the database structure for 'RUTHRA-PC\SQLEXPRESS01 (SQL Server 16.0.1000 - RUTHRA-PC\SQLEXPRESS01)'. The central pane shows a query window with the following SQL code:

```
1 SELECT
2     h.city,
3     AVG(b.amount) AS average_room_rate,
4     COUNT(b.booking_id) AS total_bookings
5 FROM
6     bookings b
7 JOIN
8     hotels h ON b.hotel_id = h.hotel_id
9 WHERE
10    b.status = 'Stayed' -- Only consider completed stays
11 GROUP BY
12     h.city
13 ORDER BY
14     average_room_rate DESC;
```

The Results pane at the bottom displays the following data:

city	average_room_rate	total_bookings
1 Mumbai	7396.1293103440	116
2 Pune	4916.7674418004	86
3 Hyderabad	4406.0555555555	72
4 Delhi	4266.7774407712	337
5 Bangalore	4079.2699724517	363
6 Kolkata	3779.9280714285	14
7 Chennai	3677.8533333333	66
8 Jaipur	3543.2253521126	71
9 Noida	2807.0265486725	113
10 Gurgaon	2736.0471889328	551

The status bar at the bottom indicates 'Query executed successfully.' and 'RUTHRA-PC\SQLEXPRESS01 (16.0.1000 - RUTHRA-PC\SQLEXPRESS01) HotelBookingAnalysis 00:00:00 10 rows'.

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

The screenshot shows the SQL Server Enterprise Manager interface. The Object Explorer on the left displays the database structure for 'RUTHRA-PC\SQLEXPRESS01 (SQL Server 16.0.1000 - RUTHRA-PC\SQLEXPRESS01)'. The central pane shows a query window with the following SQL code:

```
1 SELECT
2     h.city,
3     COUNT(CASE WHEN MONTH(b.date_of_booking) = 1 THEN b.booking_id END) AS jan_bookings,
4     COUNT(CASE WHEN MONTH(b.date_of_booking) = 2 THEN b.booking_id END) AS feb_bookings,
5     COUNT(CASE WHEN MONTH(b.date_of_booking) = 3 THEN b.booking_id END) AS mar_bookings,
6     COUNT(b.booking_id) AS total_bookings
7 FROM
8     bookings b
9 JOIN
10    hotels h ON b.hotel_id = h.hotel_id
11 GROUP BY
12     h.city
13 ORDER BY
14     total_bookings DESC;
```

The Results pane at the bottom displays the following data:

city	jan_bookings	feb_bookings	mar_bookings	total_bookings
1 Gurgaon	318	200	274	892
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	68	47	130
8 Jaipur	35	32	39	106
9 Chennai	41	31	26	98
10 Kolkata	7	6	9	22

The status bar at the bottom indicates 'No issues found' and 'RUTHRA-PC\SQLEXPRESS01 (16.0.1000 - RUTHRA-PC\SQLEXPRESS01) HotelBookingAnalysis 00:00:00 10 rows'.

5. New Customers in January

The screenshot shows the SQL Server Enterprise Manager interface. The Object Explorer on the left displays the database structure for 'RUTHRA-PC\SQLEXPRESS01'. The central pane shows a SQL query in the 'SQL Query7' window. The query is as follows:

```
1 WITH jan_customers AS (  
2     SELECT DISTINCT customer_id  
3     FROM bookings  
4     WHERE MONTH(date_of_booking) = 1  
5 ),  
6 prior_customers AS (  
7     SELECT DISTINCT customer_id  
8     FROM bookings  
9     WHERE date_of_booking < '2022-01-01'  
10 )  
11 SELECT  
12     COUNT(j.customer_id) AS new_customers_jan,  
13     (SELECT COUNT(DISTINCT customer_id) FROM bookings WHERE MONTH(date_of_booking) = 1) AS total_customers_jan,  
14     ROUND(COUNT(j.customer_id) * 100.0 /  
15         (SELECT COUNT(DISTINCT customer_id) FROM bookings WHERE MONTH(date_of_booking) = 1), 2) AS percentage_new  
16 FROM  
17     jan_customers j  
18 LEFT JOIN  
19     prior_customers p ON j.customer_id = p.customer_id  
20 WHERE  
21     p.customer_id IS NULL;
```

The Results pane at the bottom shows the output of the query:

new_customers_jan	total_customers_jan	percentage_new
719	719	100.000000000000

The status bar at the bottom indicates 'Query executed successfully.' and '1 rows'.

6. Net Revenue (After Cancellations)

The screenshot shows the SQL Server Enterprise Manager interface. The Object Explorer on the left displays the database structure for 'RUTHRA-PC\SQLEXPRESS01'. The central pane shows a SQL query in the 'SQL Query7' window. The query is as follows:

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

The Results pane at the bottom shows the output of the query:

net_revenue
5393361.0000000000

The status bar at the bottom indicates 'Query executed successfully.' and '1 rows'.

7. Gross Revenue (Total Booked Amount)

The screenshot shows the SQL Server Enterprise Manager interface. The Object Explorer on the left displays the database structure for 'RUTHRA-PC\SQLEXPRESS01 (SQL Server 16.0.1000 - RUTHRA-PC)'. The central query window shows the following SQL code:

```
1 SELECT
2     SUM(amount) AS gross_revenue
3 FROM
4     bookings;
```

The Results pane at the bottom shows a single row of data:

gross_revenue
11917462.0000000000

The status bar at the bottom indicates 'Query executed successfully.' and 'RUTHRA-PC\SQLEXPRESS01 (16... RUTHRA-PC\ASUS (75) HotelBookingAnalysis 00:00:00 1 rows'.

8. Cancellation Rate by City

The screenshot shows the SQL Server Enterprise Manager interface. The Object Explorer on the left displays the database structure for 'RUTHRA-PC\SQLEXPRESS01 (SQL Server 16.0.1000 - RUTHRA-PC)'. The central query window shows the following SQL code:

```
1 SELECT
2     h.city,
3     COUNT(b.booking_id) AS total_bookings,
4     SUM(CASE WHEN b.status = 'Cancelled' THEN 1 ELSE 0 END) AS cancelled_bookings,
5     ROUND(SUM(CASE WHEN b.status = 'Cancelled' THEN 1 ELSE 0 END) * 100.0 / COUNT(b.booking_id), 2) AS cancellation_rate
6 FROM
7     bookings b
8 JOIN
9     hotels h ON b.hotel_id = h.hotel_id
10 GROUP BY
11     h.city
12 ORDER BY
13     cancellation_rate DESC;
```

The Results pane at the bottom shows a table with 10 rows of data:

city	total_bookings	cancelled_bookings	cancellation_rate
Delhi	609	238	39.0800000000000000
Noida	230	87	37.8260000000000000
Hyderabad	127	48	37.8000000000000000
Mumbai	179	58	32.4000000000000000
Gurgaon	872	280	32.1100000000000000
Kolkata	22	7	31.8200000000000000
Chennai	88	29	29.5500000000000000
Japur	108	30	28.3000000000000000
Bangalore	526	148	28.1400000000000000
Pune	120	28	23.3300000000000000

The status bar at the bottom indicates 'Query executed successfully.' and 'RUTHRA-PC\SQLEXPRESS01 (16... RUTHRA-PC\ASUS (88) HotelBookingAnalysis 00:00:00 10 rows'.