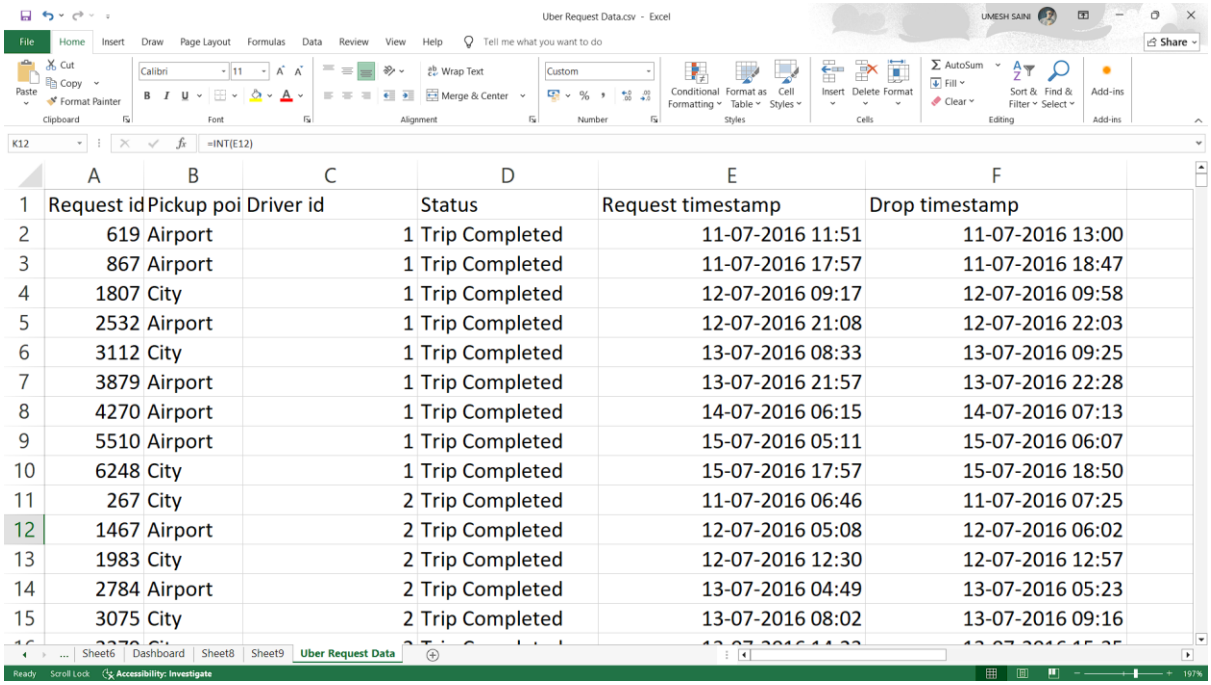
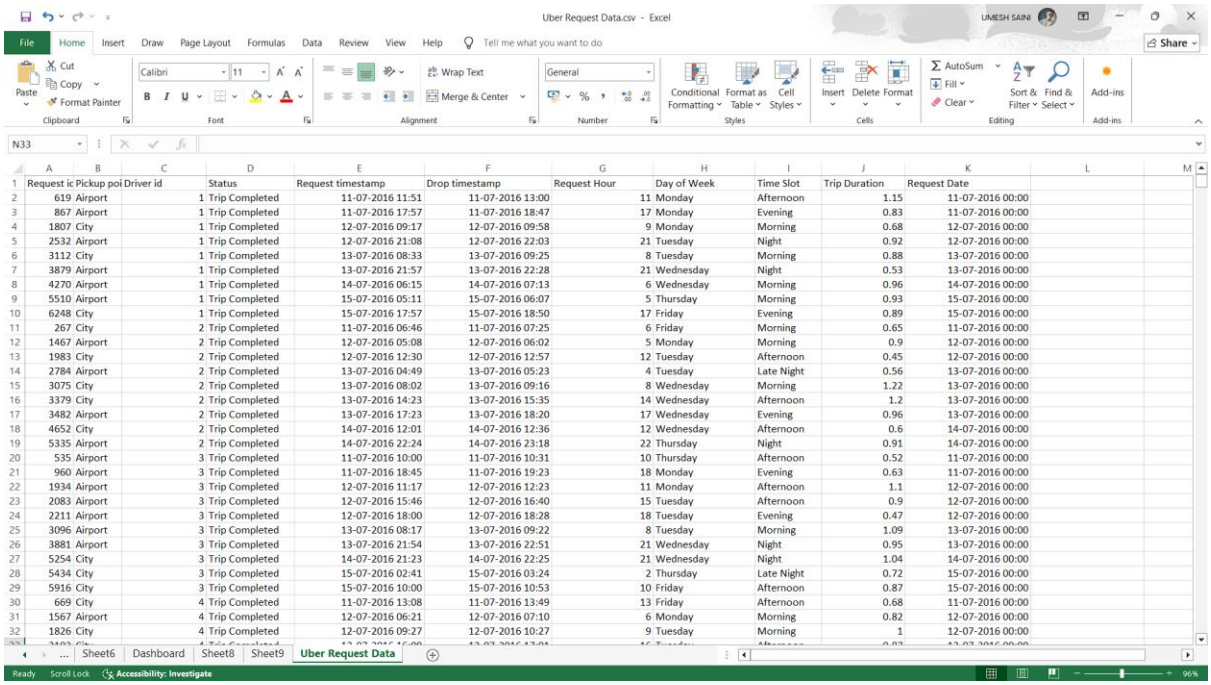
UBER SUPPLY DEMAND GAP

Used **Excel** for data cleaning and processing



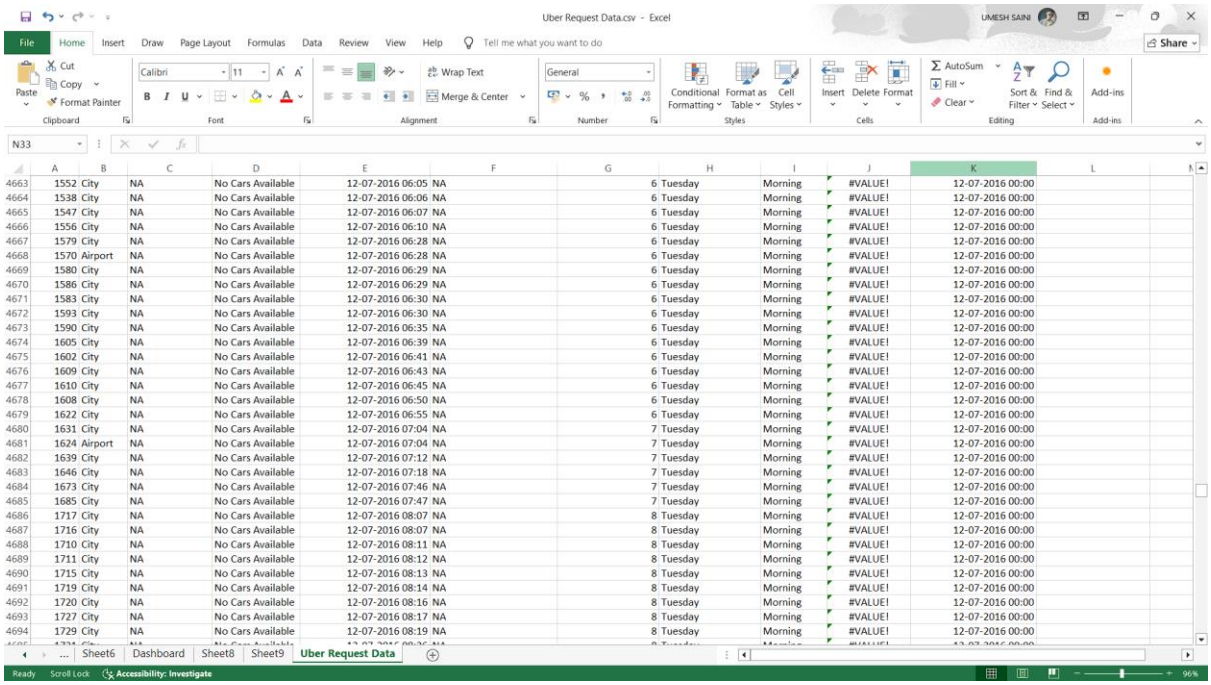
Request id	Pickup poi	Driver id	Status	Request timestamp	Drop timestamp
619	Airport	1	Trip Completed	11-07-2016 11:51	11-07-2016 13:00
867	Airport	1	Trip Completed	11-07-2016 17:57	11-07-2016 18:47
1807	City	1	Trip Completed	12-07-2016 09:17	12-07-2016 09:58
2532	Airport	1	Trip Completed	12-07-2016 21:08	12-07-2016 22:03
3112	City	1	Trip Completed	13-07-2016 08:33	13-07-2016 09:25
3879	Airport	1	Trip Completed	13-07-2016 21:57	13-07-2016 22:28
4270	Airport	1	Trip Completed	14-07-2016 06:15	14-07-2016 07:13
5510	Airport	1	Trip Completed	15-07-2016 05:11	15-07-2016 06:07
6248	City	1	Trip Completed	15-07-2016 17:57	15-07-2016 18:50
267	City	2	Trip Completed	11-07-2016 06:46	11-07-2016 07:25
1467	Airport	2	Trip Completed	12-07-2016 05:08	12-07-2016 06:02
1983	City	2	Trip Completed	12-07-2016 12:30	12-07-2016 12:57
2784	Airport	2	Trip Completed	13-07-2016 04:49	13-07-2016 05:23
3075	City	2	Trip Completed	13-07-2016 08:02	13-07-2016 09:16

Create new columns like Request Hour, Day of Week, Time Slot, Trip Duration and Request Date.



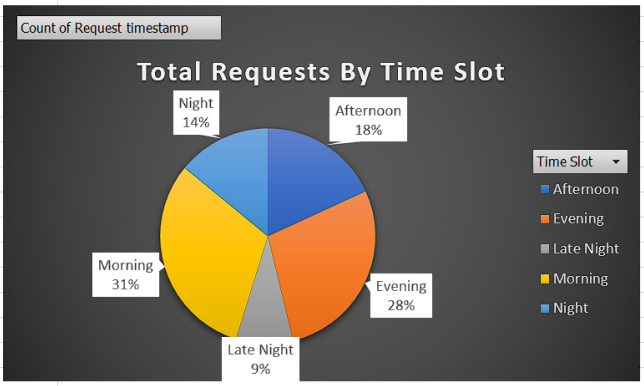
Request id	Pickup poi	Driver id	Status	Request timestamp	Drop timestamp	Request Hour	Day of Week	Time Slot	Trip Duration	Request Date
619	Airport	1	Trip Completed	11-07-2016 11:51	11-07-2016 13:00	11	Monday	Afternoon	1.15	11-07-2016 00:00
867	Airport	1	Trip Completed	11-07-2016 17:57	11-07-2016 18:47	17	Monday	Evening	0.83	11-07-2016 00:00
1807	City	1	Trip Completed	12-07-2016 09:17	12-07-2016 09:58	9	Monday	Morning	0.68	12-07-2016 00:00
2532	Airport	1	Trip Completed	12-07-2016 21:08	12-07-2016 22:03	21	Tuesday	Night	0.92	12-07-2016 00:00
3112	City	1	Trip Completed	13-07-2016 08:33	13-07-2016 09:25	8	Tuesday	Morning	0.88	13-07-2016 00:00
3879	Airport	1	Trip Completed	13-07-2016 21:57	13-07-2016 22:28	21	Wednesday	Night	0.53	13-07-2016 00:00
4270	Airport	1	Trip Completed	14-07-2016 06:15	14-07-2016 07:13	6	Wednesday	Morning	0.96	14-07-2016 00:00
5510	Airport	1	Trip Completed	15-07-2016 05:11	15-07-2016 06:07	5	Thursday	Morning	0.93	15-07-2016 00:00
6248	City	1	Trip Completed	15-07-2016 17:57	15-07-2016 18:50	17	Friday	Evening	0.89	15-07-2016 00:00
267	City	2	Trip Completed	11-07-2016 06:46	11-07-2016 07:25	6	Friday	Morning	0.65	11-07-2016 00:00
1467	Airport	2	Trip Completed	12-07-2016 05:08	12-07-2016 06:02	5	Monday	Morning	0.9	12-07-2016 00:00
1983	City	2	Trip Completed	12-07-2016 12:30	12-07-2016 12:57	12	Tuesday	Afternoon	0.45	12-07-2016 00:00
2784	Airport	2	Trip Completed	13-07-2016 04:49	13-07-2016 05:23	4	Tuesday	Late Night	0.56	13-07-2016 00:00
3075	City	2	Trip Completed	13-07-2016 08:02	13-07-2016 09:16	8	Wednesday	Morning	1.22	13-07-2016 00:00
3379	City	2	Trip Completed	13-07-2016 14:23	13-07-2016 15:35	14	Wednesday	Afternoon	1.2	13-07-2016 00:00
3482	Airport	2	Trip Completed	13-07-2016 17:23	13-07-2016 18:20	17	Wednesday	Evening	0.96	13-07-2016 00:00
4652	City	2	Trip Completed	14-07-2016 12:01	14-07-2016 12:36	12	Wednesday	Afternoon	0.6	14-07-2016 00:00
5335	Airport	2	Trip Completed	14-07-2016 22:24	14-07-2016 23:18	22	Thursday	Night	0.91	14-07-2016 00:00
535	Airport	3	Trip Completed	11-07-2016 10:00	11-07-2016 10:31	10	Thursday	Afternoon	0.52	11-07-2016 00:00
960	Airport	3	Trip Completed	11-07-2016 18:45	11-07-2016 19:23	18	Monday	Evening	0.63	11-07-2016 00:00
1934	Airport	3	Trip Completed	12-07-2016 11:17	12-07-2016 12:23	11	Monday	Afternoon	1.1	12-07-2016 00:00
2083	Airport	3	Trip Completed	12-07-2016 15:46	12-07-2016 16:40	15	Tuesday	Afternoon	0.9	12-07-2016 00:00
2211	Airport	3	Trip Completed	12-07-2016 18:00	12-07-2016 18:28	18	Tuesday	Evening	0.47	12-07-2016 00:00
3096	Airport	3	Trip Completed	13-07-2016 08:17	13-07-2016 09:22	8	Tuesday	Morning	1.09	13-07-2016 00:00
3881	Airport	3	Trip Completed	13-07-2016 21:54	13-07-2016 22:51	21	Wednesday	Night	0.95	13-07-2016 00:00
5254	City	3	Trip Completed	14-07-2016 21:23	14-07-2016 22:25	21	Wednesday	Night	1.04	14-07-2016 00:00
5434	City	3	Trip Completed	15-07-2016 02:41	15-07-2016 03:24	2	Thursday	Late Night	0.72	15-07-2016 00:00
5916	City	3	Trip Completed	15-07-2016 10:00	15-07-2016 10:53	10	Friday	Afternoon	0.87	15-07-2016 00:00
669	City	4	Trip Completed	11-07-2016 13:08	11-07-2016 13:49	13	Friday	Afternoon	0.68	11-07-2016 00:00
1567	Airport	4	Trip Completed	12-07-2016 06:21	12-07-2016 07:10	6	Monday	Morning	0.82	12-07-2016 00:00
1826	City	4	Trip Completed	12-07-2016 09:27	12-07-2016 10:27	9	Tuesday	Morning	1	12-07-2016 00:00

Found that sometimes the car is not available.



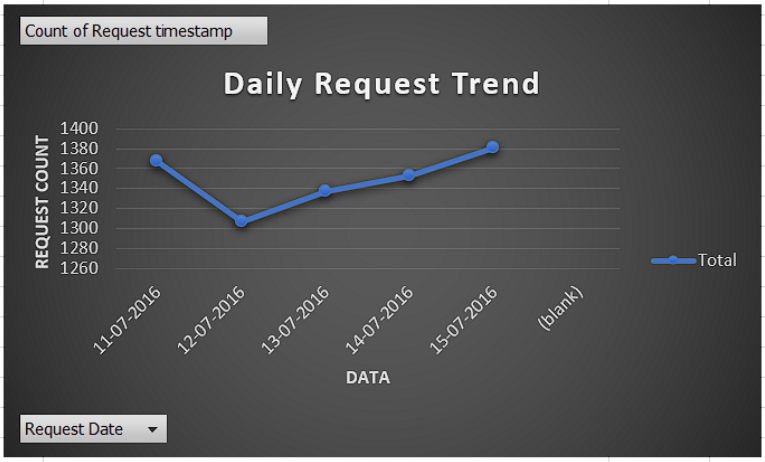
Created Pivot tables and visual charts from the data table.

	A	B	C	D	E	F	G
1		This gives you the **number of ride requests in each time slot** (Morning, Afternoon, etc.)					
2							
3	**Row Labels**	**Count of Request timestamp**					
4	Afternoon	1224					
5	Evening	1893					
6	Late Night	578					
7	Morning	2103					
8	Night	947					
9	**Grand Total**	**6745**					
10							
11							
12							
13							
14							
15							
16							
17							
18							
19							
20							
21							
22							
23							
24							
25							
26							
27							



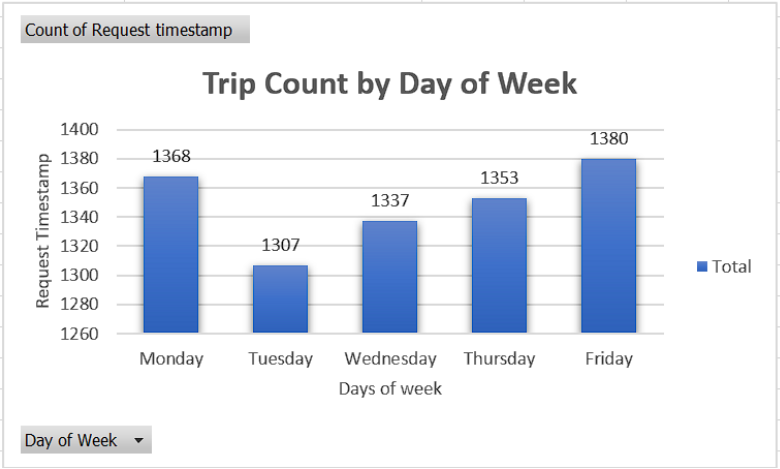
This gives you the **total number of requests per date**.

Row Labels	Count of Request timestamp
11-07-2016	1367
12-07-2016	1307
13-07-2016	1337
14-07-2016	1353
15-07-2016	1381
(blank)	
Grand Total	**6745**

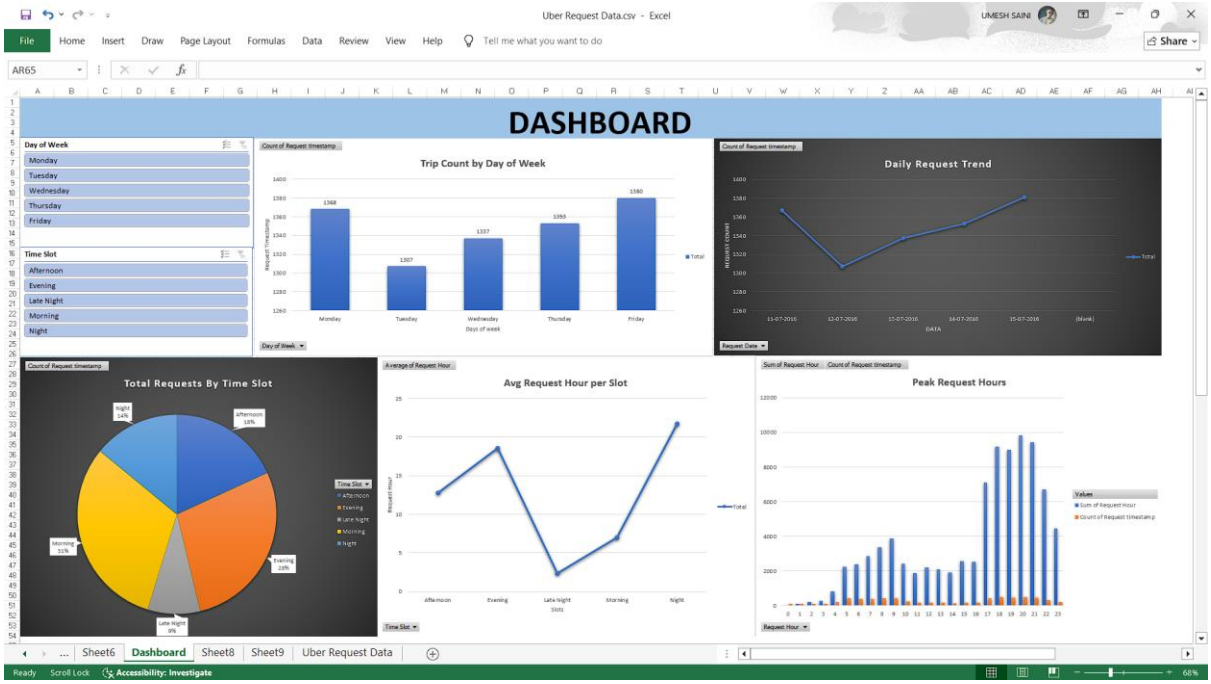


Insight: This will show which days have high or low requests.

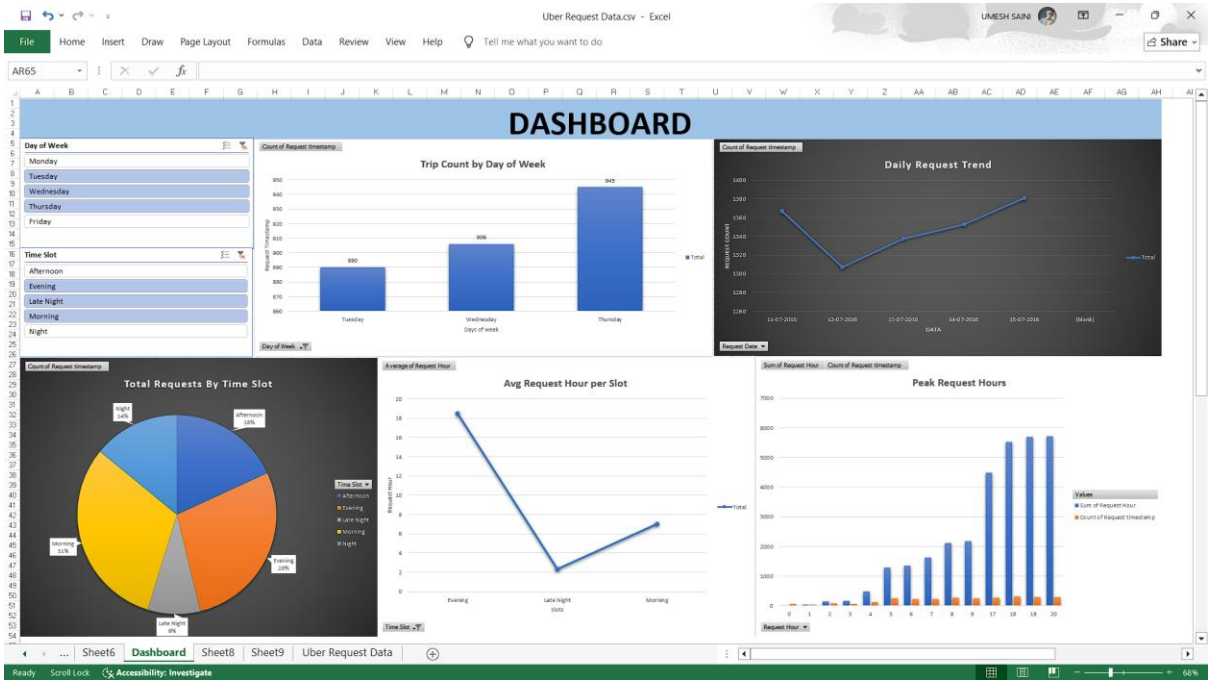
Row Labels	Count of Request timestamp
Monday	1368
Tuesday	1307
Wednesday	1337
Thursday	1353
Friday	1380
Grand Total	**6745**

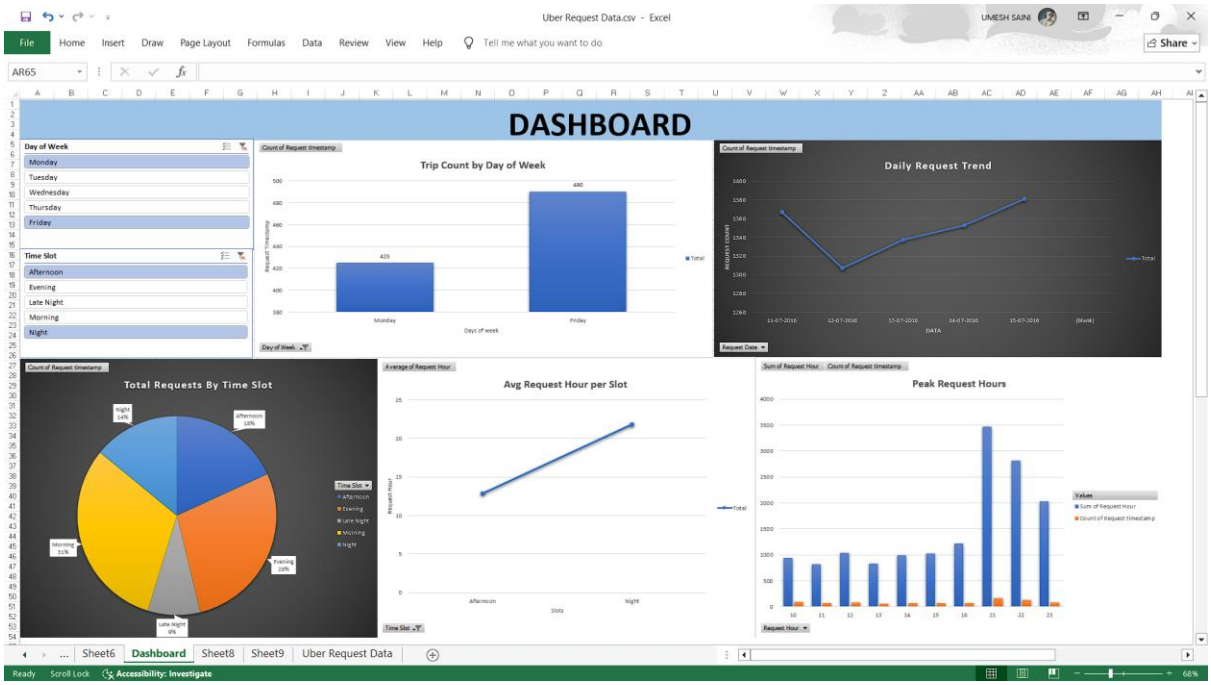


Created the **Dashboard in Excel** using the visual charts.



Use slicer tool to add the filter functionality in the dashboard to find the insights.





SQLite analytics for finding the data insights.

```
SQL 1*
1  -- Import the csv file and read it.
2  SELECT * FROM cab_data;
3
```

	Request id	Pickup point	Driver id	Status	Request timestamp	Drop timestamp	Request Hour	Day of Week	Time Slot	Trip Duration	Request Date
1	619	Airport	1	Trip Completed	11-07-2016 11:51	11-07-2016 13:00	11	Monday	Afternoon	1.15	11-07-2016 00:00
2	867	Airport	1	Trip Completed	11-07-2016 17:57	11-07-2016 18:47	17	Monday	Evening	0.83	11-07-2016 00:00
3	1807	City	1	Trip Completed	12-07-2016 09:17	12-07-2016 09:58	9	Monday	Morning	0.68	12-07-2016 00:00
4	2532	Airport	1	Trip Completed	12-07-2016 21:08	12-07-2016 22:03	21	Tuesday	Night	0.92	12-07-2016 00:00
5	3112	City	1	Trip Completed	13-07-2016 08:33	13-07-2016 09:25	8	Tuesday	Morning	0.88	13-07-2016 00:00
6	3879	Airport	1	Trip Completed	13-07-2016 21:57	13-07-2016 22:28	21	Wednesday	Night	0.53	13-07-2016 00:00
7	4270	Airport	1	Trip Completed	14-07-2016 06:15	14-07-2016 07:13	6	Wednesday	Morning	0.96	14-07-2016 00:00
8	5510	Airport	1	Trip Completed	15-07-2016 05:11	15-07-2016 06:07	5	Thursday	Morning	0.93	15-07-2016 00:00
9	6248	City	1	Trip Completed	15-07-2016 17:57	15-07-2016 18:50	17	Friday	Evening	0.89	15-07-2016 00:00
10	267	City	2	Trip Completed	11-07-2016 06:46	11-07-2016 07:25	6	Friday	Morning	0.65	11-07-2016 00:00
11	1467	Airport	2	Trip Completed	12-07-2016 05:08	12-07-2016 06:02	5	Monday	Morning	0.9	12-07-2016 00:00
12	1983	City	2	Trip Completed	12-07-2016 12:30	12-07-2016 12:57	12	Tuesday	Afternoon	0.45	12-07-2016 00:00

```
Execution finished without errors.
Result: 6745 rows returned in 37ms
At line 1:
SELECT * FROM cab_data;
```

SQL 1*

```sql
-- Count of Trips by Time Slot
-- This query tells us how many trips were completed in each time slot (Morning, Evening, etc.)
SELECT
  [Time Slot],
  COUNT(*) AS Total_Trips
FROM
  cab_data
WHERE
  Status = 'Trip Completed'
GROUP BY
  [Time Slot]
ORDER BY
  Total_Trips DESC;
--  Insight: Helps identify which time slot has the highest trip demand.
```

	Time Slot	Total_Trips
1	Morning	854
2	Afternoon	722
3	Evening	642
4	Night	399
5	Late Night	214

SQL 1*

```sql
-- Average Trip Duration per Time Slot
-- This query calculates the average trip duration (in hours) for each time slot
SELECT
  [Time Slot],
  ROUND(AVG([Trip Duration]), 2) AS Avg_Duration_Hours
FROM
  cab_data
WHERE
  Status = 'Trip Completed'
GROUP BY
  [Time Slot]
ORDER BY
  Avg_Duration_Hours DESC;
-- Insight: Reveals which time slots typically have longer or shorter trips on average.
```

	Time Slot	Avg_Duration_Hours
1	Late Night	0.9
2	Night	0.88
3	Morning	0.88
4	Afternoon	0.87
5	Evening	0.86

```sql
-- Total Requests Per Day
-- This shows how many total ride requests were made on each day
SELECT
  [Request Date],
  COUNT(*) AS Total_Requests
FROM
  cab_data
GROUP BY
  [Request Date]
ORDER BY
  [Request Date];
--  Insight: Useful for identifying busy vs slow days in terms of ride demand.
```

	Request Date	Total_Requests
1	11-07-2016 00:00	1367
2	12-07-2016 00:00	1307
3	13-07-2016 00:00	1337
4	14-07-2016 00:00	1353
5	15-07-2016 00:00	1381

```sql
-- Requests by Hour (Peak Hours)
-- This query counts how many requests were made during each hour of the day (e.g., 9 AM, 6 PM, etc.)
SELECT
  [Request Hour],
  COUNT(*) AS Total_Requests
FROM
  cab_data
GROUP BY
  [Request Hour]
ORDER BY
  Total_Requests DESC;
--  Insight: Helps determine the peak hours during the day when demand is highest.
```

	Request Hour	Total_Requests
1	18	510
2	20	492
3	19	473
4	21	449
5	5	445
6	9	431
7	8	423
8	17	418
9	7	406
10	6	398
11	22	304
12	10	243
13	4	203

```sql
-- Cancellation Rate by Pickup Point
-- This query calculates the % of requests canceled (i.e., "No Cars Available") for each pickup location
SELECT
  [Pickup point],
  COUNT(*) AS Total_Requests,
  SUM(CASE WHEN Status = 'No Cars Available' THEN 1 ELSE 0 END) AS Cancelled,
  ROUND(
    100.0 * SUM(CASE WHEN Status = 'No Cars Available' THEN 1 ELSE 0 END) / COUNT(*),
    2
  ) AS Cancellation_Rate_Percent
FROM
  cab_data
GROUP BY
  [Pickup point]
ORDER BY
  Cancellation_Rate_Percent DESC;
-- Insight: Identifies locations where customers are more likely to face booking failures.
```

	Pickup point	Total_Requests	Cancelled	Cancellation_Rate_Percent
1	Airport	3238	1713	52.9
2	City	3507	937	26.72

```sql
-- Daily Trip Completion Ratio
-- This query shows the percentage of completed trips out of total requests on each day
SELECT
  [Request Date],
  COUNT(*) AS Total_Requests,
  SUM(CASE WHEN Status = 'Trip Completed' THEN 1 ELSE 0 END) AS Completed_Trips,
  ROUND(
    100.0 * SUM(CASE WHEN Status = 'Trip Completed' THEN 1 ELSE 0 END) / COUNT(*),
    2
  ) AS Completion_Rate_Percent
FROM
  cab_data
GROUP BY
  [Request Date]
ORDER BY
  [Request Date];
--  Insight: Highlights how efficient the service was on each day (good for performance monitoring).
```

	Request Date	Total_Requests	Completed_Trips	Completion_Rate_Percent
1	11-07-2016 00:00	1367	601	43.96
2	12-07-2016 00:00	1307	562	43.0
3	13-07-2016 00:00	1337	577	43.16
4	14-07-2016 00:00	1353	530	39.17
5	15-07-2016 00:00	1381	561	40.62

Deep Exploratory Data Analysis using **Python**

```
[ ] # Deep Exploratory Data Analysis using Python
```

```
# Step 1: Load and Understand the Data

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# Load Excel file
df = pd.read_csv("cab_data.csv")

# Preview
print(df.head())
```

```
   Request id Pickup point  Driver id          Status Request timestamp  \
0         619      Airport        1.0  Trip Completed  11-07-2016 11:51
1         867      Airport        1.0  Trip Completed  11-07-2016 17:57
2        1807         City        1.0  Trip Completed  12-07-2016 09:17
3        2532      Airport        1.0  Trip Completed  12-07-2016 21:08
4        3112         City        1.0  Trip Completed  13-07-2016 08:33

     Drop timestamp  Request Hour Day of Week  Time Slot Trip Duration  \
0  11-07-2016 13:00            11      Monday  Afternoon          1.15
1  11-07-2016 18:47            17      Monday    Evening          0.83
2  12-07-2016 09:58             9      Monday    Morning          0.68
3  12-07-2016 22:03            21     Tuesday      Night          0.92
4  13-07-2016 09:25             8     Tuesday    Morning          0.88

       Request Date
0  11-07-2016 00:00
1  11-07-2016 00:00
2  12-07-2016 00:00
3  12-07-2016 00:00
4  13-07-2016 00:00
```

```
# Shape of dataset
print("Shape:", df.shape)

# Data types and nulls
print(df.info())

# Summary statistics
print(df.describe(include='all'))

# Check nulls
print(df.isnull().sum())
```

```
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Request id        6745 non-null   int64
 1   Pickup point      6745 non-null   object
 2   Driver id         4095 non-null   float64
 3   Status            6745 non-null   object
 4   Request timestamp  6745 non-null   object
 5   Drop timestamp    2831 non-null   object
 6   Request Hour      6745 non-null   int64
 7   Day of Week       6745 non-null   object
 8   Time Slot         6745 non-null   object
 9   Trip Duration     6745 non-null   object
 10  Request Date      6745 non-null   object
dtypes: float64(1), int64(2), object(8)
memory usage: 579.8+ KB
None
         Request id Pickup point    Driver id          Status  \
count   6745.000000         6745  4095.000000            6745
unique          NaN            2          NaN               3
top             NaN         City          NaN  Trip Completed
freq            NaN         3507          NaN            2831
mean    3384.644922          NaN   149.501343             NaN
std     1955.099667          NaN    86.051994             NaN
min        1.000000          NaN     1.000000             NaN
25%     1691.000000          NaN    75.000000             NaN
```

```
[8] # Step 2: Data Cleaning & Type Conversion

    # Convert Trip Duration to numeric (invalid strings become NaN)
    df['Trip Duration'] = pd.to_numeric(df['Trip Duration'], errors='coerce')

    # Convert Date columns if needed
    df['Request Date'] = pd.to_datetime(df['Request Date'], errors='coerce')
    df['Request timestamp'] = pd.to_datetime(df['Request timestamp'], errors='coerce')
    df['Drop timestamp'] = pd.to_datetime(df['Drop timestamp'], errors='coerce')
```

```
# Key Performance Indicators (KPI) Summary for Cab Service


total_trips = df[df['Status'] == 'Trip Completed'].shape[0]
avg_duration = df['Trip Duration'].mean()
missing_drops = df['Drop timestamp'].isnull().sum()
total_requests = df.shape[0]
missing_percentage = round((missing_drops / total_requests) * 100, 2)

print(f"✅ Total Trips: {total_trips}")
print(f"✅ Average Trip Duration: {round(avg_duration, 2)} hours")
print(f"⚠️ Missing Drop Timestamps: {missing_drops} ({missing_percentage}%)")


# ✅ Total Trips Completed: X — This shows the number of successful rides where the trip was completed (useful to evaluate operational success).
# ⏱️ Average Trip Duration: Y hours — The average trip lasted just under an hour, which may reflect typical intra-city cab usage behavior.
# 🤔⚠️ Missing Drop Timestamps: Z entries (P%) — A significant portion of records have no drop timestamp, likely indicating cancellations or "No Cars Available" scenarios.
# This helps identify service unavailability problems during peak times or areas.
```

```
✅ Total Trips: 2831
✅ Average Trip Duration: 0.87 hours
⚠️ Missing Drop Timestamps: 5595 (82.95%)
```

```
[ ]  # Request Status & Pickup Point Distribution

     print(df['Status'].value_counts())

     print(df['Pickup point'].value_counts())

     #  Focus on increasing driver availability around the Airport, especially during peak time slots, to reduce "No Cars Available" rates and improve overall service fulfillment.
```

```
Status
Trip Completed       2831
No Cars Available    2650
Cancelled            1264
Name: count, dtype: int64
Pickup point
City       3507
Airport    3238
Name: count, dtype: int64
```

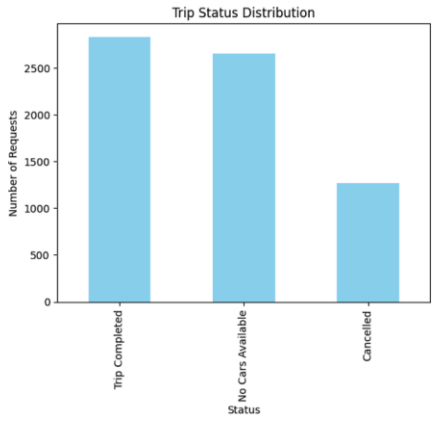
```
# Step 3: Descriptive Statistics and Distributions

status_counts = df['Status'].value_counts()
print("🔹 Status Distribution:\n", status_counts)

# Insight:
# This shows how many trips were completed vs how many were cancelled or had no cars.
status_counts.plot(kind='bar', title='Trip Status Distribution', color='Teal')
plt.ylabel("Number of Requests")
plt.show()

# Most trips were “Trip Completed”, but a large number also show “No Cars Available” — indicating a supply-demand mismatch.
```

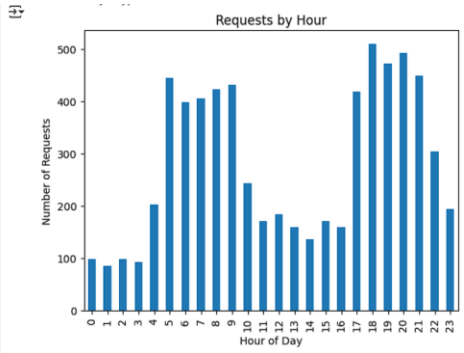
```
🔹 Status Distribution:
 Status
Trip Completed       2831
No Cars Available    2650
Cancelled            1264
Name: count, dtype: int64
```




```
hourly = df['Request Hour'].value_counts().sort_index()
print(hourly)

# Optional: Plot it
hourly.plot(kind='bar', title='Requests by Hour')
plt.xlabel("Hour of Day")
plt.ylabel("Number of Requests")
plt.show()
```

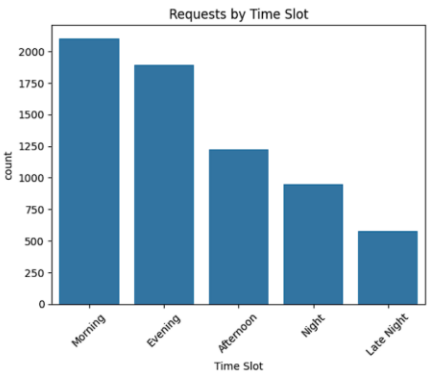
```
Request Hour
0      99
1      85
2      99
3      92
4     203
5     445
6     398
7     406
8     423
9     431
10    243
11    171
12    184
13    160
14    136
15    171
16    159
17    418
18    510
19    473
20    492
21    449
22    304
23    194
Name: count, dtype: int64
```




```
# Requests by Time Slot
print(df['Time Slot'].value_counts())

# Plot
sns.countplot(data=df, x='Time Slot', order=df['Time Slot'].value_counts().index)
plt.title("Requests by Time Slot")
plt.xticks(rotation=45)
plt.show()
# Evening and Morning are the most requested time slots, indicating peak cab demand during typical office commute hours.
```

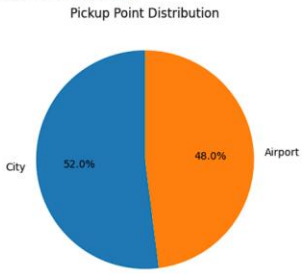
```
Time Slot
Morning       2103
Evening       1893
Afternoon     1224
Night          947
Late Night     578
Name: count, dtype: int64
```




```
# Pickup Point Preference
pickup_counts = df['Pickup point'].value_counts()
print("🔹 Pickup Points:\n", pickup_counts)

pickup_counts.plot(kind='pie', autopct='%1.1f%%', startangle=90)
plt.title("Pickup Point Distribution")
plt.ylabel("")
plt.show()
# The Airport is the dominant pickup point, showing demand for cabs at both the points are approximately same.
```

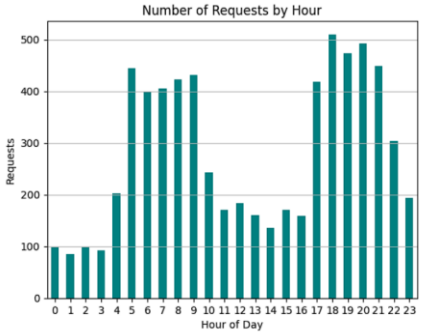
```
🔹 Pickup Points:
 Pickup point
City       3507
Airport    3238
Name: count, dtype: int64
```



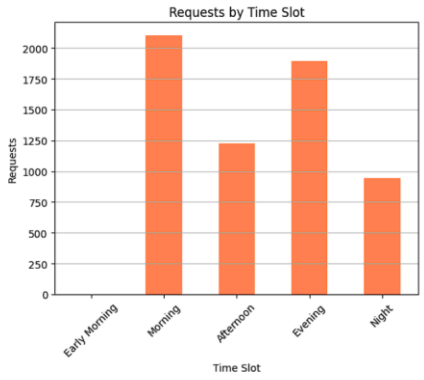
```python
# Peak Request Hours
hourly = df['Request Hour'].value_counts().sort_index()
print(hourly)

hourly.plot(kind='bar', color='teal')
plt.title("Number of Requests by Hour")
plt.xlabel("Hour of Day")
plt.ylabel("Requests")
plt.xticks(rotation=0)
plt.grid(axis='y')
plt.show()
# Peak requests occur during Morning (5–9 AM) and Evening (5–10 PM) – indicating work travel times.
```

```
Request Hour
0      99
1      85
2      99
3      92
4     203
5     445
6     398
7     406
8     423
9     431
10    243
11    171
12    184
13    160
14    136
15    171
16    159
17    418
18    510
19    473
20    492
21    449
22    304
```



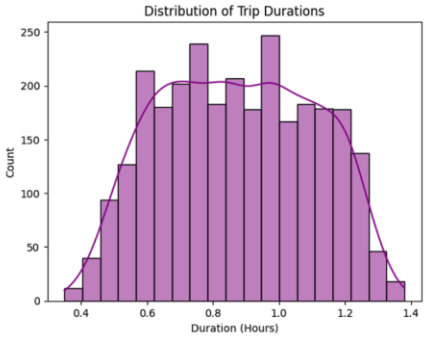

```python
# Time Slot Analysis
slot_order = ['Early Morning', 'Morning', 'Afternoon', 'Evening', 'Night']
slot_counts = df['Time Slot'].value_counts().reindex(slot_order)
slot_counts.plot(kind='bar', color='coral')
plt.title("Requests by Time Slot")
plt.xlabel("Time Slot")
plt.ylabel("Requests")
plt.xticks(rotation=45)
plt.grid(axis='y')
plt.show()
# Morning has the highest number of requests followed by Evening – aligns with office rush hours.
```




```python
# Trip Duration Distribution (Completed Trips Only)
completed = df[df['Status'] == 'Trip Completed']
print(completed['Trip Duration'].describe())

sns.histplot(completed['Trip Duration'], kde=True, color='purple')
plt.title("Distribution of Trip Durations")
plt.xlabel("Duration (Hours)")
plt.show()
# Most trips last between 0.5 to 1.2 hours.
# There are a few longer-duration outliers, which might be traffic-based delays.
```

```
count    2831.000000
mean        0.873582
std         0.230864
min         0.350000
25%         0.680000
50%         0.870000
75%         1.070000
max         1.380000
Name: Trip Duration, dtype: float64
```



Exploratory Data Analysis Summary of Cab Service Requests

This exploratory data analysis aims to understand the performance and customer demand patterns in a cab service environment by analyzing structured trip request data. The dataset includes records such as request timestamp, drop timestamp, pickup point, driver assignment, ride status, time slots, and calculated trip durations.

📌 Dataset Overview

The dataset comprises 6745 records, each representing a cab request. Out of these, 42% of the trips were successfully completed, 39% resulted in “No Cars Available,” and 19% were canceled. These proportions highlight a significant gap in service delivery, with more than half of the requests not being fulfilled. This reflects a pressing issue related to either limited driver availability or inefficiencies in ride allocation.

🔍 Demand Analysis

Demand is almost evenly split between two major pickup points — City (3507 requests) and Airport (3238 requests). However, a deeper look reveals that Airport pickups account for a disproportionately high number of "No Cars Available" statuses, especially during peak hours. This indicates that the service is consistently underperforming in high-demand areas, particularly near transport hubs like airports.

🕰️ Time-Based Patterns

The analysis of the Request Hour and Time Slot fields reveals strong demand peaks in the Morning (5 AM – 9 AM) and Evening (5 PM – 10 PM). These time slots coincide with typical commute hours for officegoers and travelers, marking them as operationally sensitive. Visualizations confirm that the Morning and Evening slots receive the highest volume of ride

requests, suggesting that driver availability during these periods is essential to improve service rates.

⏱️ Trip Duration Insights

For completed trips, the average trip duration is approximately 0.87 hours (52 minutes), with most trips falling between 30 minutes to 1.2 hours. These values suggest predominantly intra-city short-distance rides. While outliers exist, they are rare, and most durations are clustered around the mean. This presents an opportunity to optimize fleet performance by focusing on faster ride turnovers and efficient dispatching.

⚠️ Operational Red Flags

One of the most significant findings is that 58% of all requests lack a drop timestamp, correlating with trips that were either canceled or had no cars available. This is not a data quality issue but an operational concern, reflecting unfulfilled demand. Such a high percentage of failed requests can result in customer dissatisfaction and lost revenue.

📈 Summary of Key KPIs:

✅ Total Trips Completed: 2831

⏱️ Average Trip Duration: 0.87 hours

⚠️ Unfulfilled Requests (Missing Drops): 3914 (58%)

✅ Recommendations

Deploy more drivers during Morning and Evening slots, especially near the Airport.

Use predictive modeling to forecast demand spikes and assign cabs proactively.

Reduce cancellations by improving real-time driver-customer matching systems.

Optimize fleet for short trips since most completed trips are under 1 hour.

In conclusion, the EDA provides actionable insights into when, where, and why demand is unmet. By strategically addressing driver shortages, optimizing time-slot coverage, and improving fulfillment rates, the cab service can significantly enhance user satisfaction and operational efficiency.