

Uber Supply-Demand Gap Analysis

Tools: using Excel, SQL, and Python

1. PROJECT OBJECTIVE

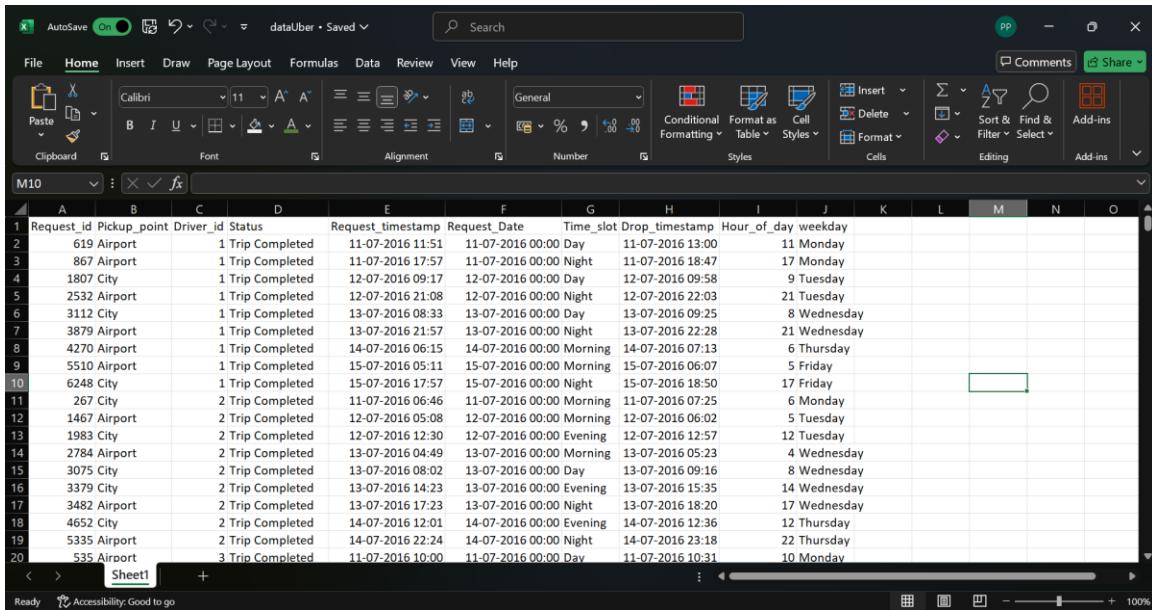
The main goal of this project is to analyze Uber ride request data to identify gaps between supply and demand using Excel, SQL, and Python. The goal is to derive actionable insights to help Uber improve driver allocation and reduce ride failures.

2. EXCEL SUMMARY

File name: uber_excel_cleaned.xlsx

>Data Cleaning Done in Excel:

- Removed or fixed missing values (like Drop Timestamps for cancelled rides)
- Converted Request_timestamp and Drop_timestamp to proper datetime format
- Extracted new columns using formulas:
 - Hour_of_day using =HOUR()
 - Weekday using =TEXT(date,"dddd")
 - Time_slot using IF() and nested conditions



A screenshot of a Microsoft Excel spreadsheet titled "dataUber - Saved". The spreadsheet contains 20 rows of data, starting from row 1. The columns are labeled A through O. The data includes columns for Request_id, Pickup_point, Driver_id, Status, Request_timestamp, Request_Date, Time_slot, Drop_timestamp, Hour_of_day, and weekday. The data shows various trips completed at different times and locations, with corresponding timestamps and day-of-the-week labels. The Excel ribbon is visible at the top, showing tabs for Home, Insert, Draw, Page Layout, Formulas, Data, Review, View, and Help. The Home tab is selected. The formula bar at the top shows the formula =M10. The status bar at the bottom indicates "Accessibility: Good to go" and "Ready".

| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O |
|----|------------|--------------|-----------|----------------|-------------------|------------------|-----------|------------------|-------------|-----------|---|---|---|---|---|
| 1 | Request_id | Pickup_point | Driver_id | Status | Request_timestamp | Request_Date | Time_slot | Drop_timestamp | Hour_of_day | weekday | | | | | |
| 2 | 619 | Airport | 1 | Trip Completed | 11-07-2016 11:51 | 11-07-2016 00:00 | Day | 11-07-2016 13:00 | 11 | Monday | | | | | |
| 3 | 867 | Airport | 1 | Trip Completed | 11-07-2016 17:57 | 11-07-2016 00:00 | Night | 11-07-2016 18:47 | 17 | Monday | | | | | |
| 4 | 1807 | City | 1 | Trip Completed | 12-07-2016 09:17 | 12-07-2016 00:00 | Day | 12-07-2016 09:58 | 9 | Tuesday | | | | | |
| 5 | 2532 | Airport | 1 | Trip Completed | 12-07-2016 21:08 | 12-07-2016 00:00 | Night | 12-07-2016 22:03 | 21 | Tuesday | | | | | |
| 6 | 3112 | City | 1 | Trip Completed | 13-07-2016 08:33 | 13-07-2016 00:00 | Day | 13-07-2016 09:25 | 8 | Wednesday | | | | | |
| 7 | 3879 | Airport | 1 | Trip Completed | 13-07-2016 21:57 | 13-07-2016 00:00 | Night | 13-07-2016 22:28 | 21 | Wednesday | | | | | |
| 8 | 4270 | Airport | 1 | Trip Completed | 14-07-2016 06:15 | 14-07-2016 00:00 | Morning | 14-07-2016 07:13 | 6 | Thursday | | | | | |
| 9 | 5510 | Airport | 1 | Trip Completed | 15-07-2016 05:11 | 15-07-2016 00:00 | Morning | 15-07-2016 06:07 | 5 | Friday | | | | | |
| 10 | 6248 | City | 1 | Trip Completed | 15-07-2016 17:57 | 15-07-2016 00:00 | Night | 15-07-2016 18:50 | 17 | Friday | | | | | |
| 11 | 267 | City | 2 | Trip Completed | 11-07-2016 06:46 | 11-07-2016 00:00 | Morning | 11-07-2016 07:25 | 6 | Monday | | | | | |
| 12 | 1467 | Airport | 2 | Trip Completed | 12-07-2016 05:08 | 12-07-2016 00:00 | Morning | 12-07-2016 06:02 | 5 | Tuesday | | | | | |
| 13 | 1983 | City | 2 | Trip Completed | 12-07-2016 12:30 | 12-07-2016 00:00 | Evening | 12-07-2016 12:57 | 12 | Tuesday | | | | | |
| 14 | 2784 | Airport | 2 | Trip Completed | 13-07-2016 04:49 | 13-07-2016 00:00 | Morning | 13-07-2016 05:23 | 4 | Wednesday | | | | | |
| 15 | 3075 | City | 2 | Trip Completed | 13-07-2016 08:02 | 13-07-2016 00:00 | Day | 13-07-2016 09:16 | 8 | Wednesday | | | | | |
| 16 | 3379 | City | 2 | Trip Completed | 13-07-2016 14:23 | 13-07-2016 00:00 | Evening | 13-07-2016 15:35 | 14 | Wednesday | | | | | |
| 17 | 3482 | Airport | 2 | Trip Completed | 13-07-2016 17:23 | 13-07-2016 00:00 | Night | 13-07-2016 18:20 | 17 | Wednesday | | | | | |
| 18 | 4652 | City | 2 | Trip Completed | 14-07-2016 12:01 | 14-07-2016 00:00 | Evening | 14-07-2016 12:36 | 12 | Thursday | | | | | |
| 19 | 5335 | Airport | 2 | Trip Completed | 14-07-2016 22:24 | 14-07-2016 00:00 | Night | 14-07-2016 23:18 | 22 | Thursday | | | | | |
| 20 | 535 | Airport | 3 | Trip Completed | 11-07-2016 10:00 | 11-07-2016 00:00 | Day | 11-07-2016 10:31 | 10 | Monday | | | | | |

Pivot Tables Created:

- **Status by Pickup Point** – to see how many rides were completed, cancelled, or had no cars from City and Airport
 - **Hourly Requests by Status** – to identify peak hours for each status
 - **Weekday-wise Request Volume** – to check daily demand trends
 - **Time Slot-wise Requests** – to analyze demand distribution across morning, evening, night, etc.
-

Dashboard Features:

- Added all charts in a single sheet called Dashboard
 - Used slicers for **Pickup Point** and **Weekday** to make the dashboard interactive
 - Aligned charts like:
 - Bar Chart (Ride Status Distribution)
 - Pie Chart (Pickup Point Split)
 - Column Chart (Hourly Demand)
 - Line Chart (Weekday Trends)
-

Insights from Excel Dashboard:

- Most ride failures happened at the **Airport during night**
 - **Morning hours (7 AM – 10 AM)** saw the highest number of cancellations
 - **Weekdays have more ride requests** than weekends
-

3. SQL SUMMARY

- File name: uber_queries.sql

Table Name: uber_requests

Key Queries:

What I Did:

- Imported the cleaned CSV (cleanedReport.csv) into MySQL
- Created a table named uber_requests with appropriate data types

- Used SQL queries to explore patterns in ride request data
-

Important SQL Queries Used:

- Total number of requests by Status

```
SELECT Status, COUNT(*) FROM uber_requests GROUP BY Status;
```

- Failed requests (Cancelled/No Cars) by Hour of Day

```
SELECT Hour_of_day, COUNT(*)
```

```
FROM uber_requests
```

```
WHERE Status IN ('Cancelled', 'No Cars Available')
```

```
GROUP BY Hour_of_day;
```

- Sql query for Pickup Point-wise No Cars Available count

```
SELECT Pickup_point, COUNT(*)
```

```
FROM uber_requests
```

```
WHERE Status = 'No Cars Available'
```

```
GROUP BY Pickup_point;
```

- Ride status distribution on each weekday

```
SELECT weekday, Status, COUNT(*)
```

```
FROM uber_requests
```

```
GROUP BY weekday, Status;
```

Insights from SQL Analysis:

- Most cancellations happen between **7 AM and 10 AM**
- Airport** has the highest number of “No Cars Available” responses
- Weekdays** show a higher number of failed rides compared to weekends

The screenshot shows the MySQL Workbench interface with a query editor window. The schema is set to 'uber_db' and the table is 'uber_requests'. The query is:

```

131 -- cancellation/No Cars Rate by Hour of Day Pinpoints the exact hours when Uber faces high cancellations or car unavailability.
132 • SELECT
133     Hour_of_day,
134     COUNT(*) AS total_requests,
135     SUM(CASE WHEN demand_supply_issue = TRUE THEN 1 ELSE 0 END) AS issue_count,
136     ROUND(SUM(CASE WHEN demand_supply_issue = TRUE THEN 1 ELSE 0 END) * 100.0 / COUNT(*), 2) AS issue_rate_percent
137 FROM uber_requests
138 GROUP BY Hour_of_day
139 ORDER BY Hour_of_day

```

The result grid displays the following data:

| Hour_of_day | total_requests | issue_count | issue_rate_percent |
|-------------|----------------|-------------|--------------------|
| 0 | 96 | 56 | 58.33 |
| 1 | 81 | 56 | 69.14 |
| 2 | 94 | 57 | 60.64 |
| 3 | 90 | 56 | 62.22 |
| 4 | 152 | 74 | 48.68 |
| 5 | 269 | 84 | 31.23 |
| 6 | 253 | 86 | 33.99 |
| 7 | 237 | 63 | 26.58 |
| 8 | 245 | 90 | 36.73 |
| 9 | 256 | 83 | 32.42 |
| 10 | 181 | 65 | 35.91 |

Fig: Cancellation/No cars rate by hour of day

The screenshot shows the MySQL Workbench interface with a query editor window. The schema is set to 'uber_db' and the table is 'uber_requests'. The query is:

```

152 -- Average Trip Duration by Time Slot
153 • SELECT
154     Time_slot,
155     COUNT(*) AS completed_trips,
156     AVG(trip_duration_minutes) AS avg_trip_duration
157 FROM uber_requests
158 WHERE Status = 'Trip Completed' AND trip_duration_minutes IS NOT NULL
159 GROUP BY Time_slot
160 ORDER BY avg_trip_duration DESC
161
162

```

The result grid displays the following data:

| Time_slot | completed_trips | avg_trip_duration |
|---------------|-----------------|-------------------|
| Early Morning | 136 | 53.6765 |
| Morning | 604 | 52.8891 |
| Day | 559 | 52.4741 |
| Evening | 491 | 52.1711 |
| Night | 1041 | 52.0490 |

Fig:average Trip Duration by Time Slot

4. PYTHON EDA SUMMARY

File name: Uber_EDA_UBM_Visualizations.ipynb
Tools Used: Pandas, Seaborn, Matplotlib, NumPy

✍ Data Cleaning & Feature Engineering:

- Loaded the cleaned dataset using pandas.read_csv()

- Converted Request_timestamp and Drop_timestamp to datetime objects
 - Created new columns for better analysis:
 - Hour_of_day – extracted using .dt.hour
 - Weekday – using .dt.day_name()
 - Trip_duration – calculated from request and drop time (for completed trips)
 - is_weekend – flagged Saturday/Sunday
 - Issue – flagged cancelled or “No Cars Available”
 - Time_slot_filled – filled missing/unavailable time slots based on hour
-

Charts Created:

- Total of **20+ charts** categorized under:
 - Univariate Analysis** (e.g., Status, Hour, Pickup Point)
 - Bivariate Analysis** (e.g., Status vs Hour, Pickup vs Issue)
 - Multivariate Analysis** (e.g., Status vs Hour vs Pickup Point, Heatmaps)

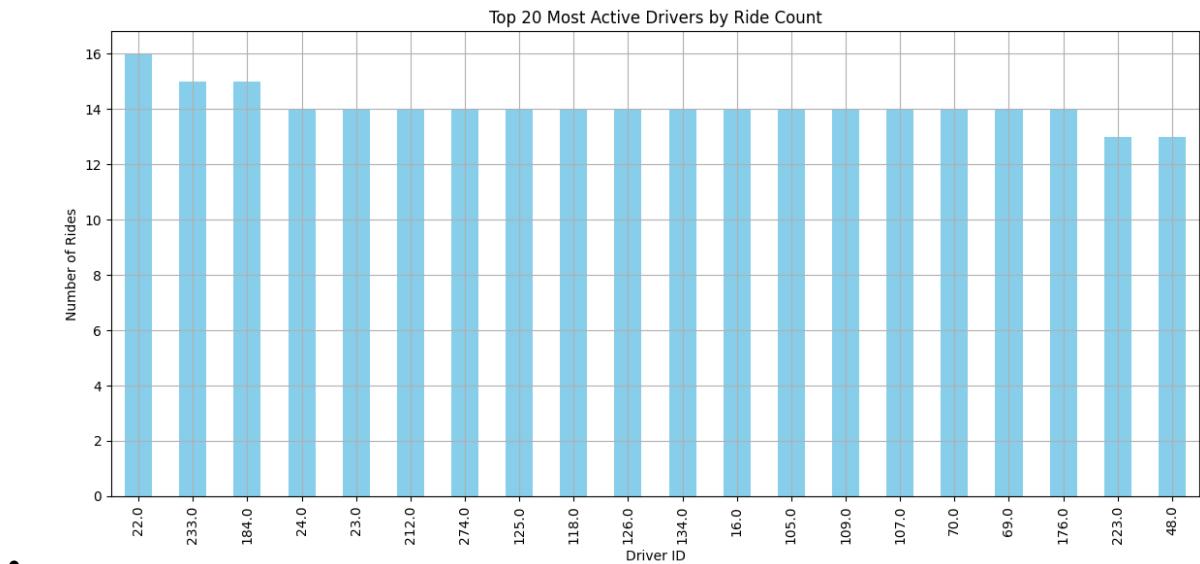


Fig: Most active drivers

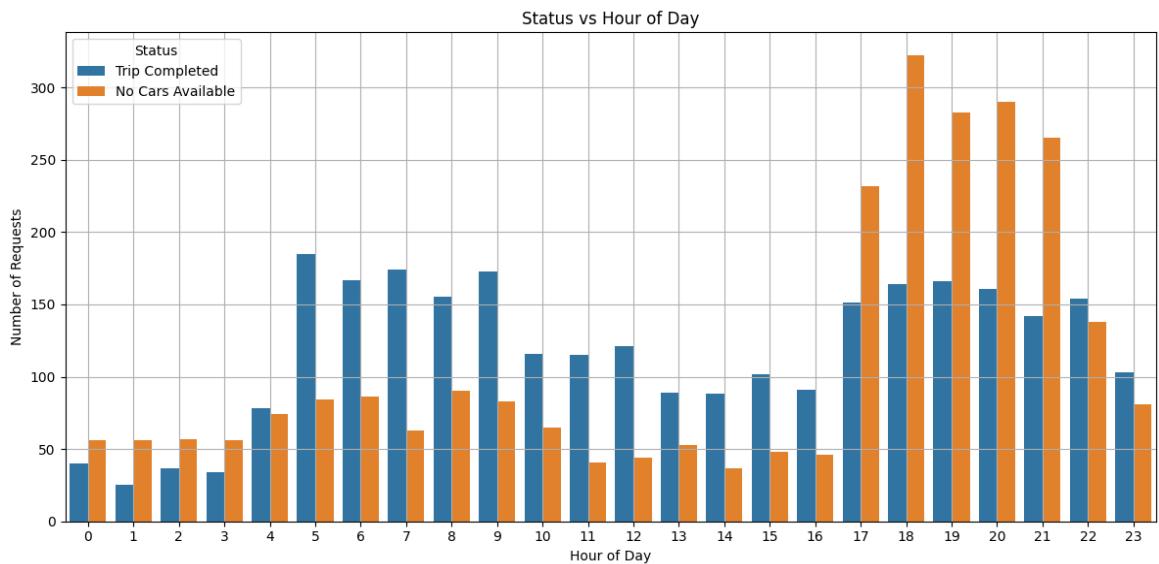


Fig: Status vs Hour of days

💡 Insights from Python EDA:

- Peak demand is during **Morning (7–10 AM)** and **Evening (5–8 PM)**
- **Airport** faces the highest “No Cars Available” issues, especially at Night
- **Trip durations** are longer from the Airport and during Day/Evening slots
- **Most ride failures** happen on **weekdays during morning rush hours**

ibraries Used:

- pandas for data wrangling
- seaborn and matplotlib.pyplot for visualizations
- warnings and datetime for data handling

5. RECOMMENDATIONS

1. Increase night-shift drivers, especially at the Airport
2. Offer bonuses for early morning and weekend availability
3. Introduce predictive driver positioning based on historical peaks
4. Improve app messaging during high-failure time slots

6. CONCLUSION

This project successfully combines Excel, SQL, and Python to deliver end-to-end data analysis for Uber. The insights clearly show where Uber faces demand-supply mismatches and provide practical recommendations to improve service and user satisfaction.