Uber Supply-Demand Gap Analysis

By:- Tushar Garg

Date:- 21/06/2025

Project Overview

Problem Statement:-

- This project aims to analyze Uber ride request data to identify the patterns and causes of ride failures, such as driver cancellations and unavailability of cars.
- Experiences fluctuations in ride requests across different times of the day and locations. A major challenge the company faces is the supply-demand mismatch, where user demand often exceeds the number of available drivers, leading to unfulfilled ride requests, customer dissatisfaction, and lost revenue opportunities.
- By uncovering when (time slots), where (pickup points), and why (status) these issues occur, the project seeks to provide data-driven insights to help Uber

Business Objective:-

- The primary business objective of this project is to identify and understand the supply-demand gap in Uber ride requests, with the goal of improving ride fulfillment rates and enhancing customer satisfaction.

- This involves analyzing when, where, and why Uber is unable to meet user demand—whether due to unavailability of drivers, high cancellation rates, or operational inefficiencies.

Dataset Description:-

- The dataset contains detailed ride request logs from Uber, focusing on pickup and drop patterns, driver availability, and service status over a specific period.
- After loading and cleaning the dataset, I performed structural and exploratory analysis to understand its composition and key issues.

Variables Description:-

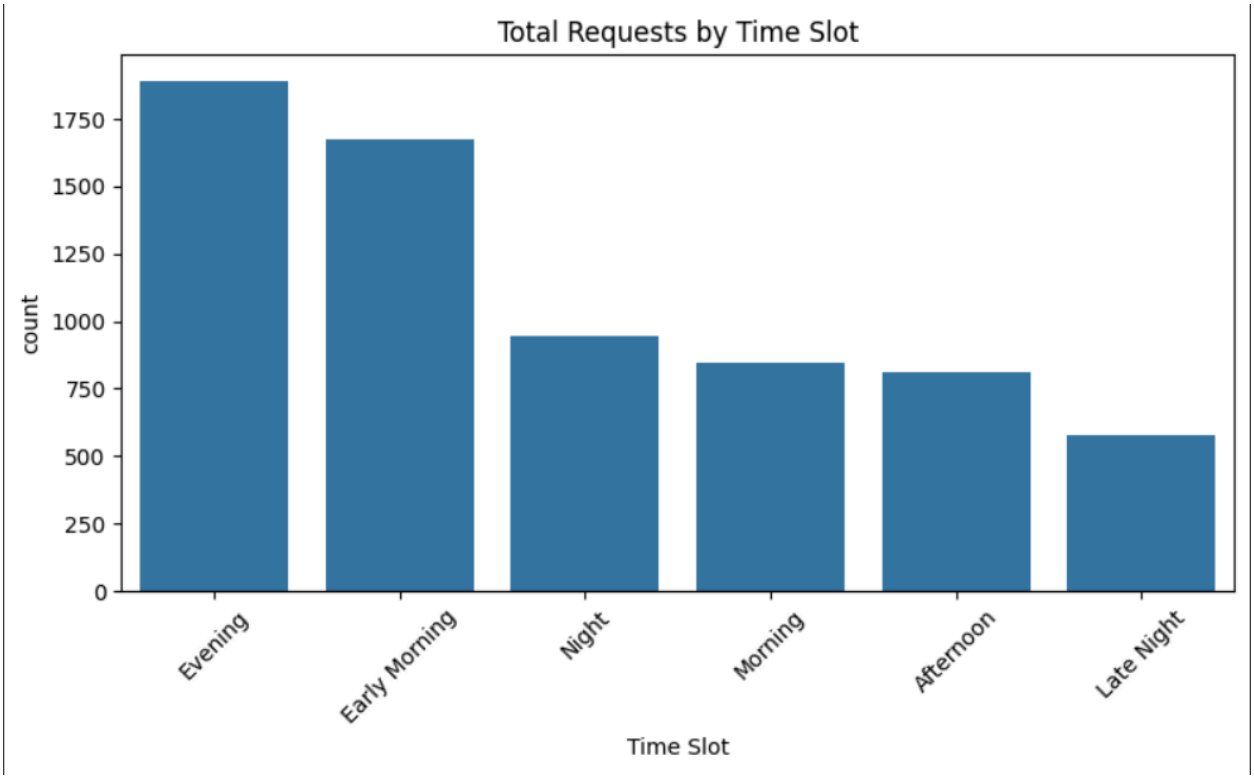
- Request timestamp-The exact date and time when the user requested a ride. Used to extract Hour, Day, and Time Slot.
- Drop timestamp - The date and time when the ride was completed. If missing, the ride was not fulfilled.
- Driver id - Unique identifier of the driver assigned to the ride. Blank if no driver was available.
- Pickup point- The location where the ride was requested: either City or Airport.
- Status- The final status of the ride

Python-Based EDA & Charts

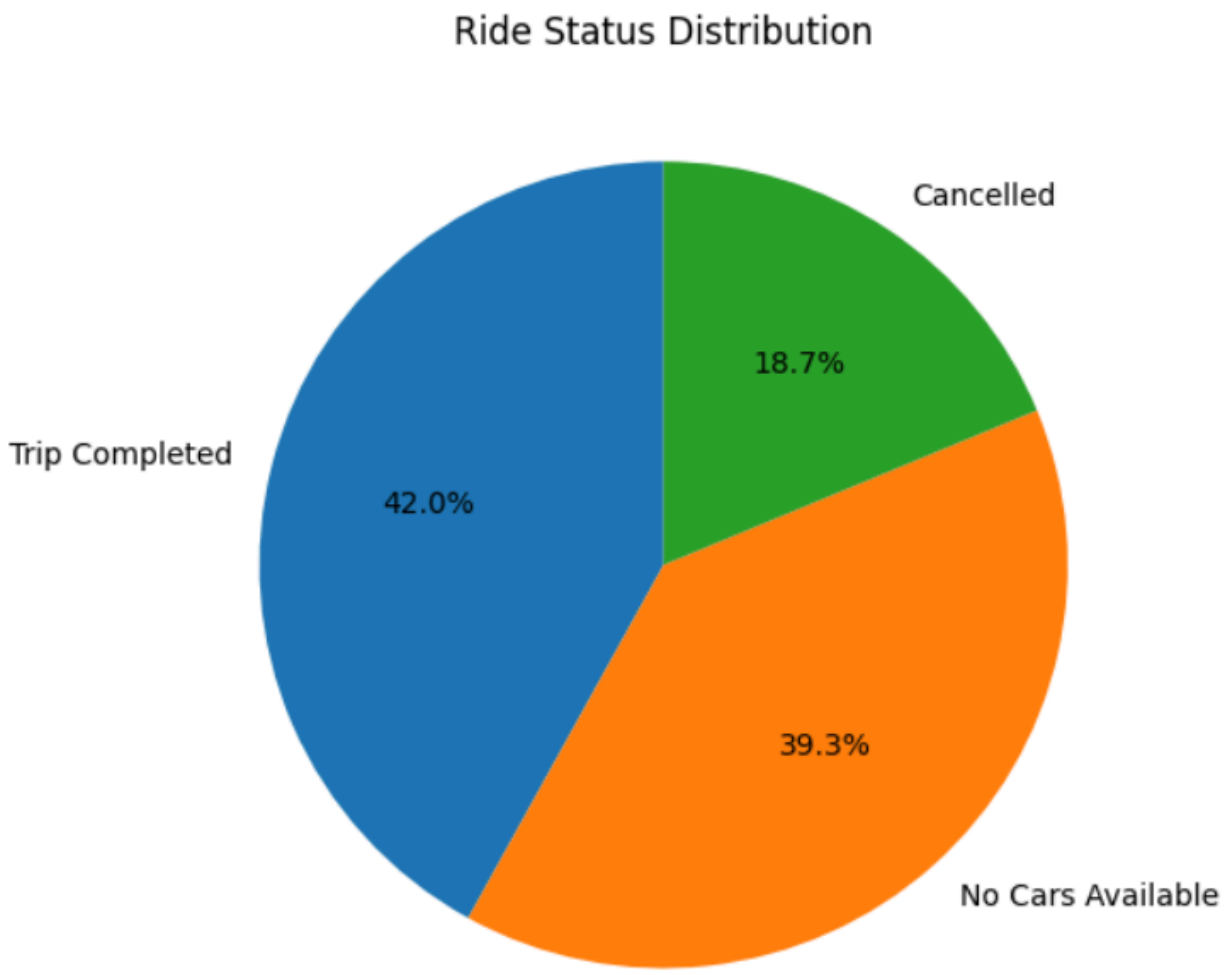
1. Univariate Charts
2. Bivariate Charts
3. Multivariate Charts

1. Univariate Charts

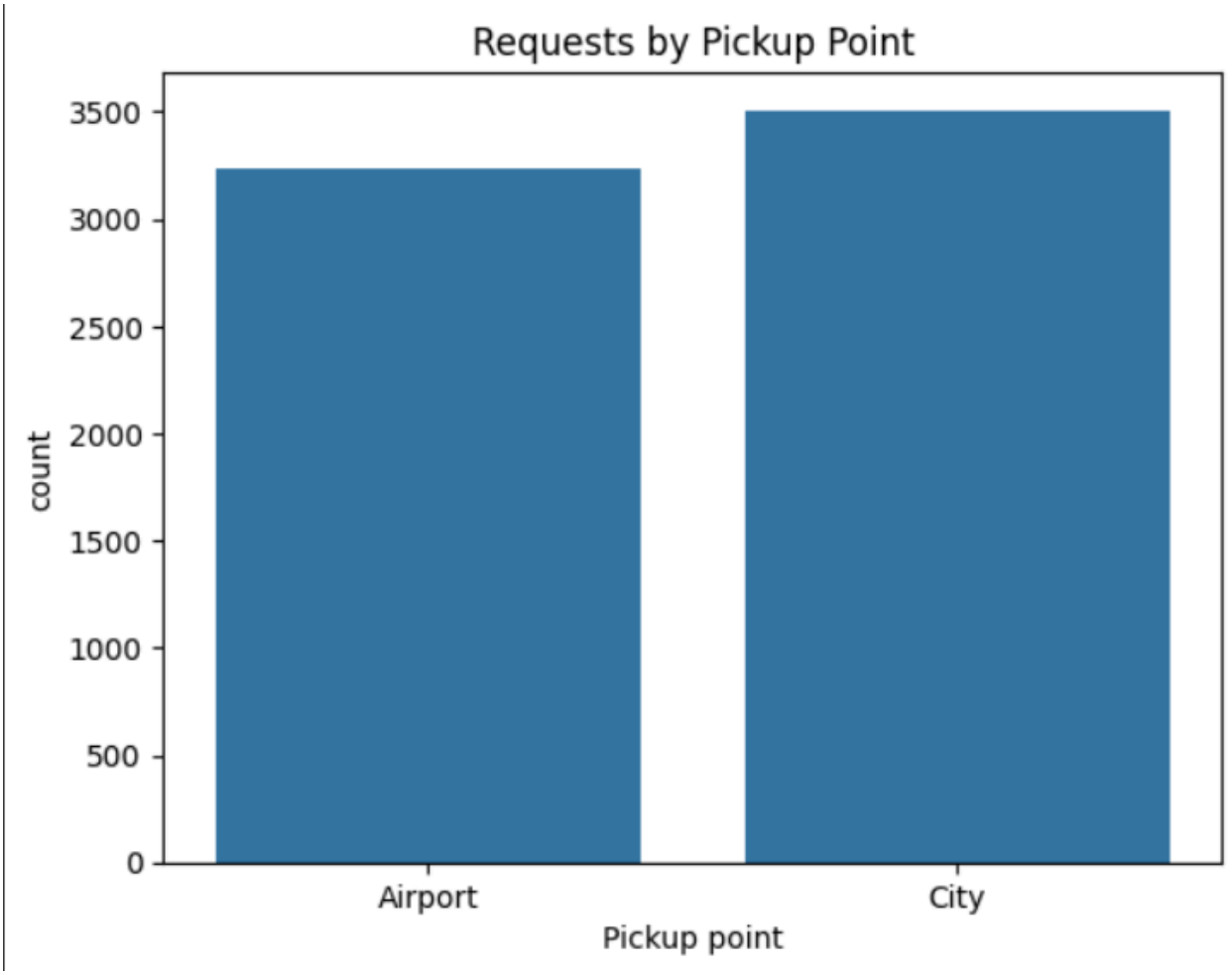
```python
# Chart 1: Requests by Time Slot
# Why: Identifies peak periods of demand throughout the day.
# Insight: High demand during Morning and Evening.
# Business Impact: Positive – helps allocate drivers efficiently.
# Risk: Missed demand = customer churn.
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(8,5))
sns.countplot(data=df, x='Time Slot', order=df['Time Slot'].value_counts().index)
plt.title("Total Requests by Time Slot")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

```



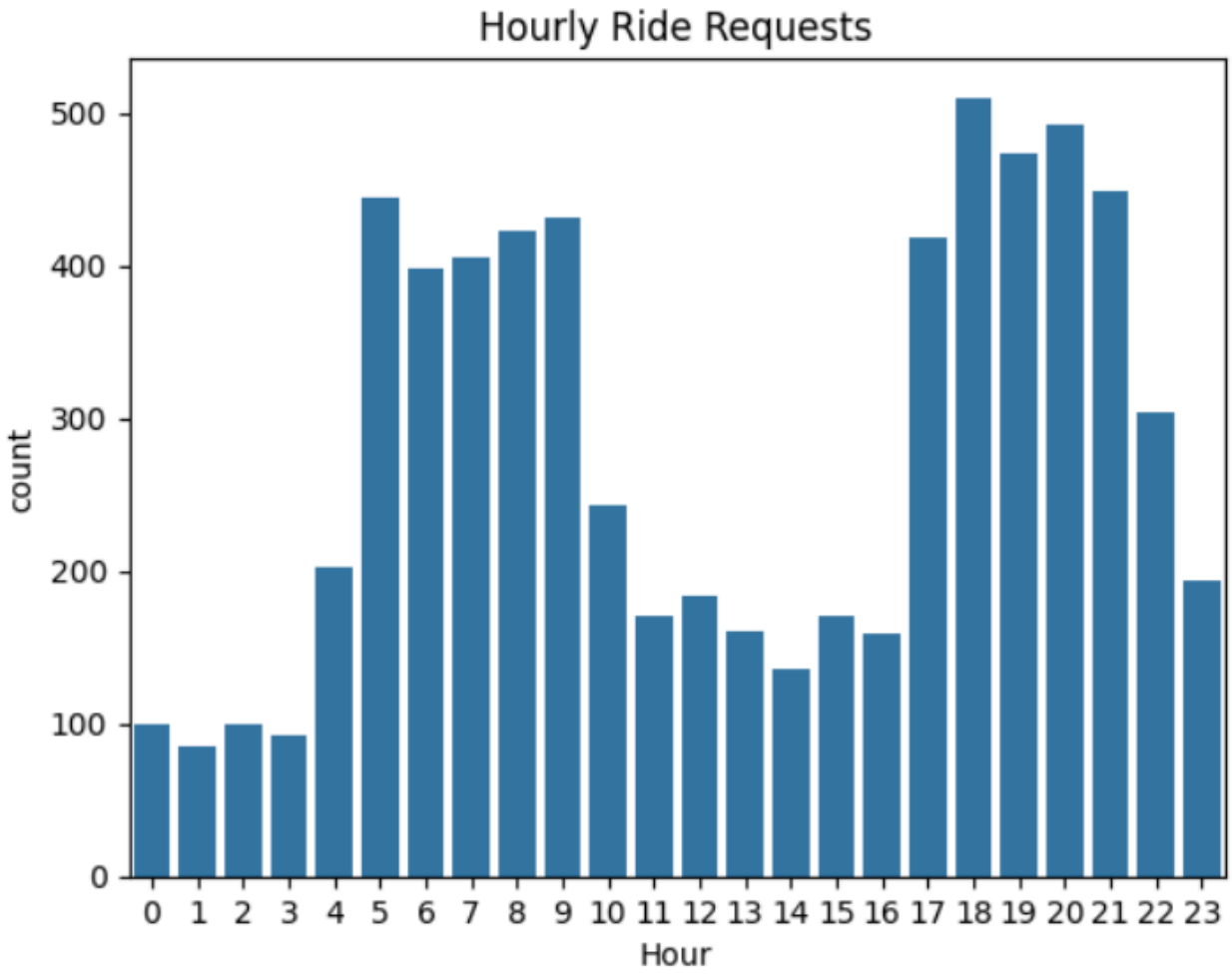
```
1
2 # Chart 2: Status Distribution
3 # Why: Understand how rides are fulfilled or fail.
4 # Insight: A large share of rides are either cancelled or unfulfilled.
5 # Business Impact: Negative – points to reliability issues.
6 plt.figure(figsize=(6,6))
7 df['Status'].value_counts().plot.pie(autopct='%1.1f%%', startangle=90)
8 plt.title("Ride Status Distribution")
9 plt.ylabel('')
10 plt.tight_layout()
11 plt.show()
12
```



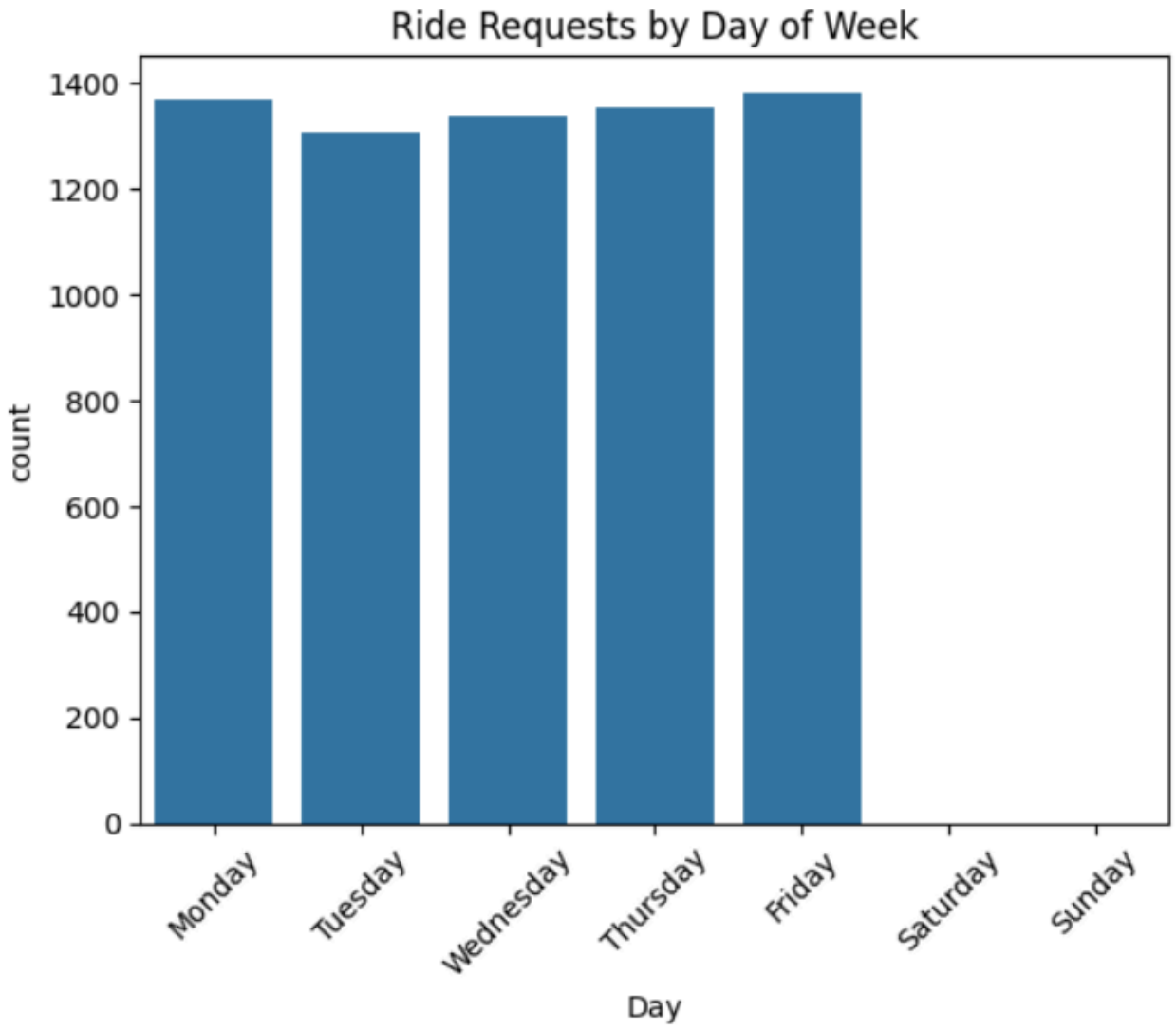
```
# Chart 3: Requests by Pickup Point
# Why: Shows which location (City/Airport) has more demand.
# Insight: Airport has more frequent requests.
# Business Impact: Positive – guide for targeted driver allocation.
sns.countplot(data=df, x='Pickup point')
plt.title("Requests by Pickup Point")
plt.show()

```



```
# Chart 4: Hourly Demand
# Why: Reveal demand distribution across hours.
# Insight: Peaks around early morning and evening.
# Business Impact: Positive – helps manage shift planning.
sns.countplot(data=df, x='Hour')
plt.title("Hourly Ride Requests")
plt.show()


```



```python
# Chart 5: Weekly Demand by Day
# Why: Compare day-of-week demand.
# Insight: Demand is slightly higher on weekdays.
# Business Impact: Moderate – helps plan weekly schedules.
sns.countplot(data=df, x='Day', order=["Monday","Tuesday","Wednesday","Thursday","Friday","Saturday","Sunday"])
plt.title("Ride Requests by Day of Week")
plt.xticks(rotation=45)
plt.show()

```



```
[ ]
# Chart 6: Missing Driver IDs
# Why: Show how often rides had no assigned driver.
# Insight: High count of NULLs = supply shortage.
# Business Impact: Negative – shows missed opportunities.
missing_drivers = df['Driver id'].isna().sum()
print(f"Number of missing driver IDs: {missing_drivers}")
```

```
Number of missing driver IDs: 2650
```

```
[ ]
# Chart 7: Missing Drop Timestamps
# Why: Capture incomplete trips.
# Insight: High = high failure/cancellation rate.
# Business Impact: Negative – damages customer trust.
missing_drop = df['Drop timestamp'].isna().sum()
print(f"Number of missing drop timestamps: {missing_drop}")

```

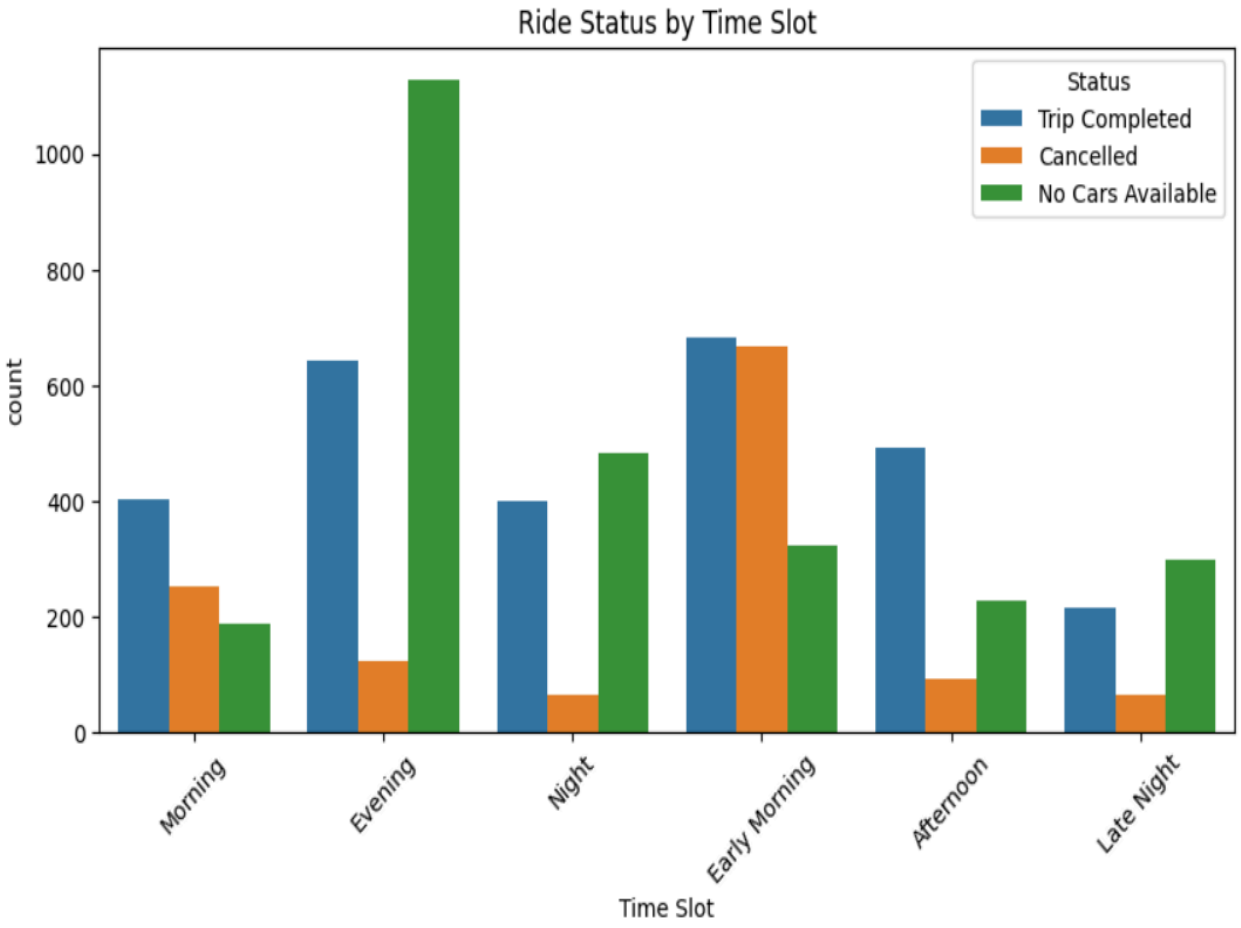
```
Number of missing drop timestamps: 3914
```

```
[ ]
# Chart 8: Total Requests KPI
# Why: Baseline metric.
# Insight: Total ride demand.
# Business Impact: Shows market size and platform use.
print(f"Total ride requests: {df.shape[0]}")

```

```
Total ride requests: 6745
```

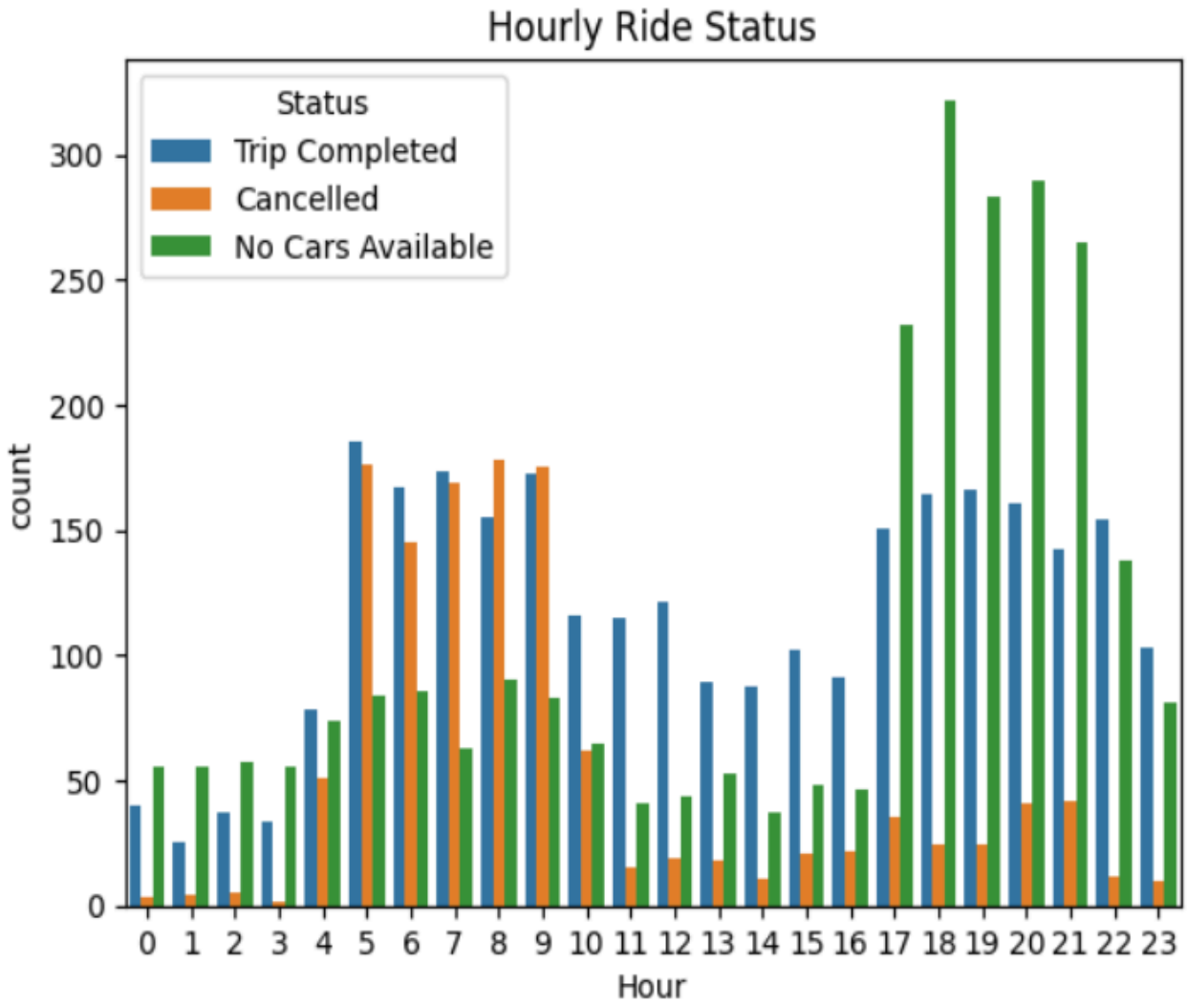
2. Bivariate Charts

```
# Chart 9: Time Slot vs Status
# Why: Understand performance over time.
# Insight: Failures peak at night and early morning.
# Business Impact: Target incentives during weak supply hours.
plt.figure(figsize=(10,5))
sns.countplot(data=df, x='Time Slot', hue='Status')
plt.title("Ride Status by Time Slot")
plt.xticks(rotation=45)
plt.show()
```

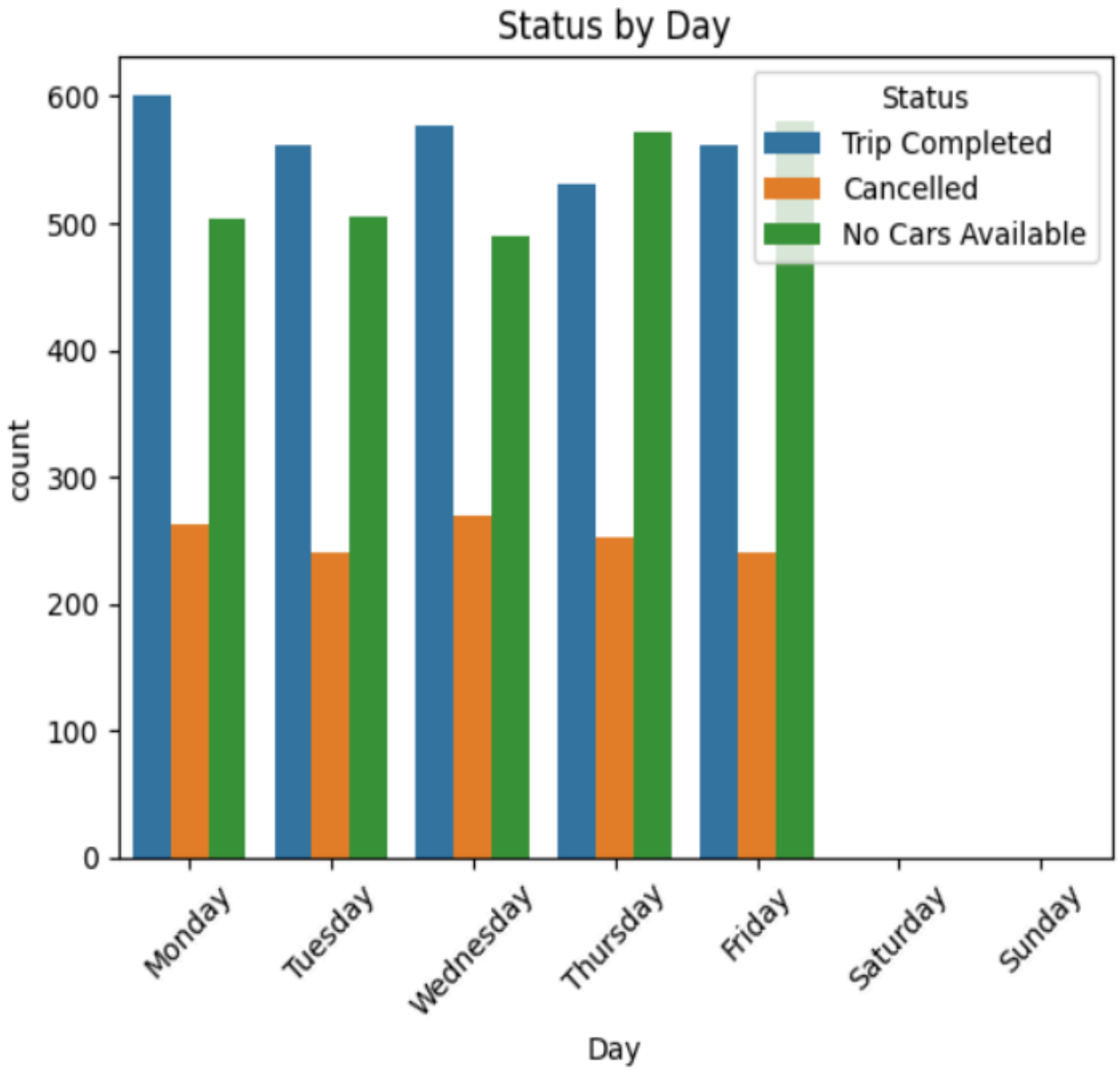


```python
# Chart 10: Pickup Point vs Status
# Why: Compare success/failure by location.
# Insight: Airport sees more no-car availability issues.
# Business Impact: Adds urgency for better airport supply.
sns.countplot(data=df, x='Pickup point', hue='Status')
plt.title("Ride Status by Pickup Location")
plt.show()

```

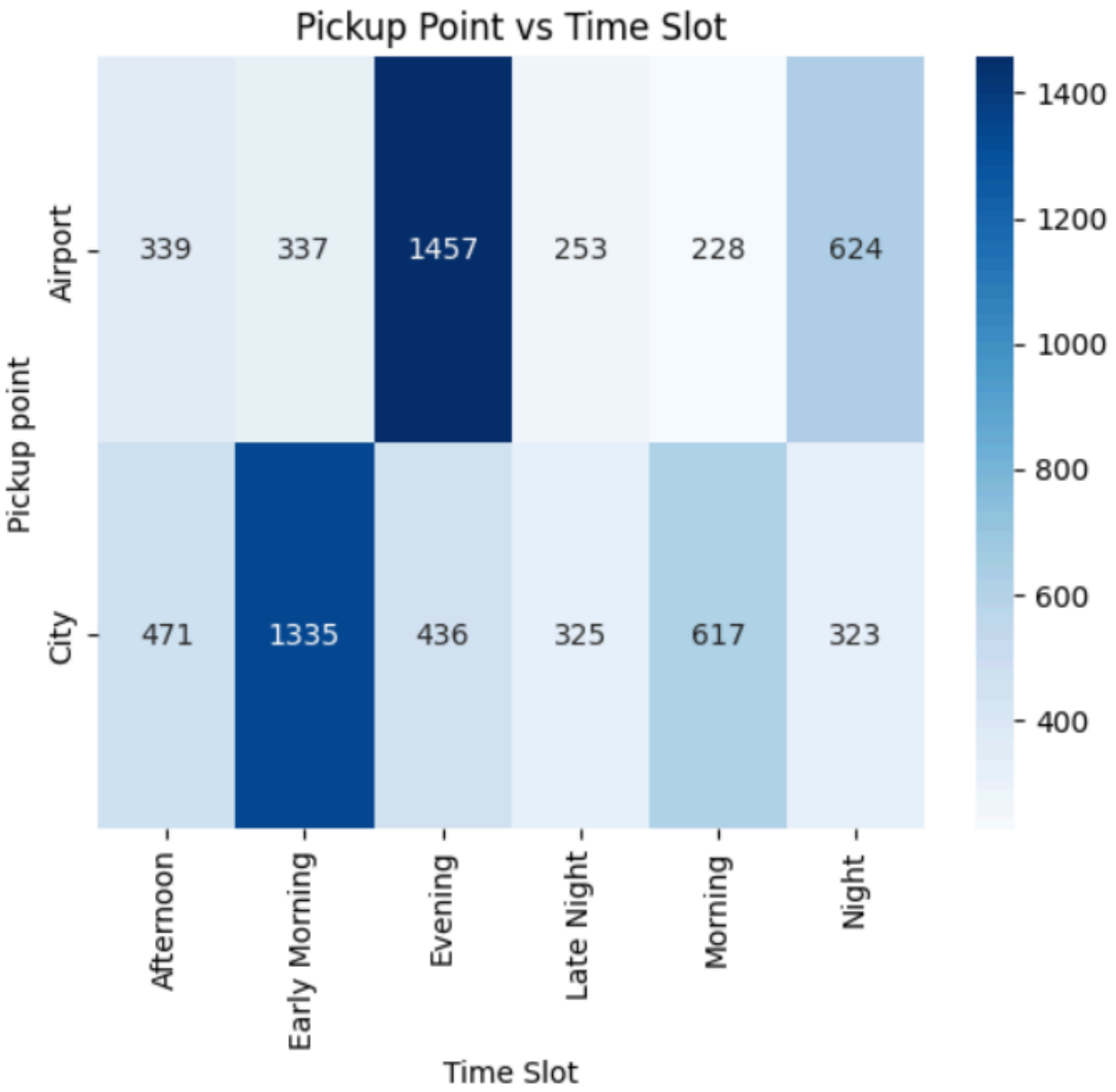
```python
# Chart 11: Hour vs Status
# Why: Breakdown hourly demand/success.
# Insight: Early morning has max cancellations.
# Business Impact: Optimize early shifts.
sns.countplot(data=df, x='Hour', hue='Status')
plt.title("Hourly Ride Status")
plt.show()
```



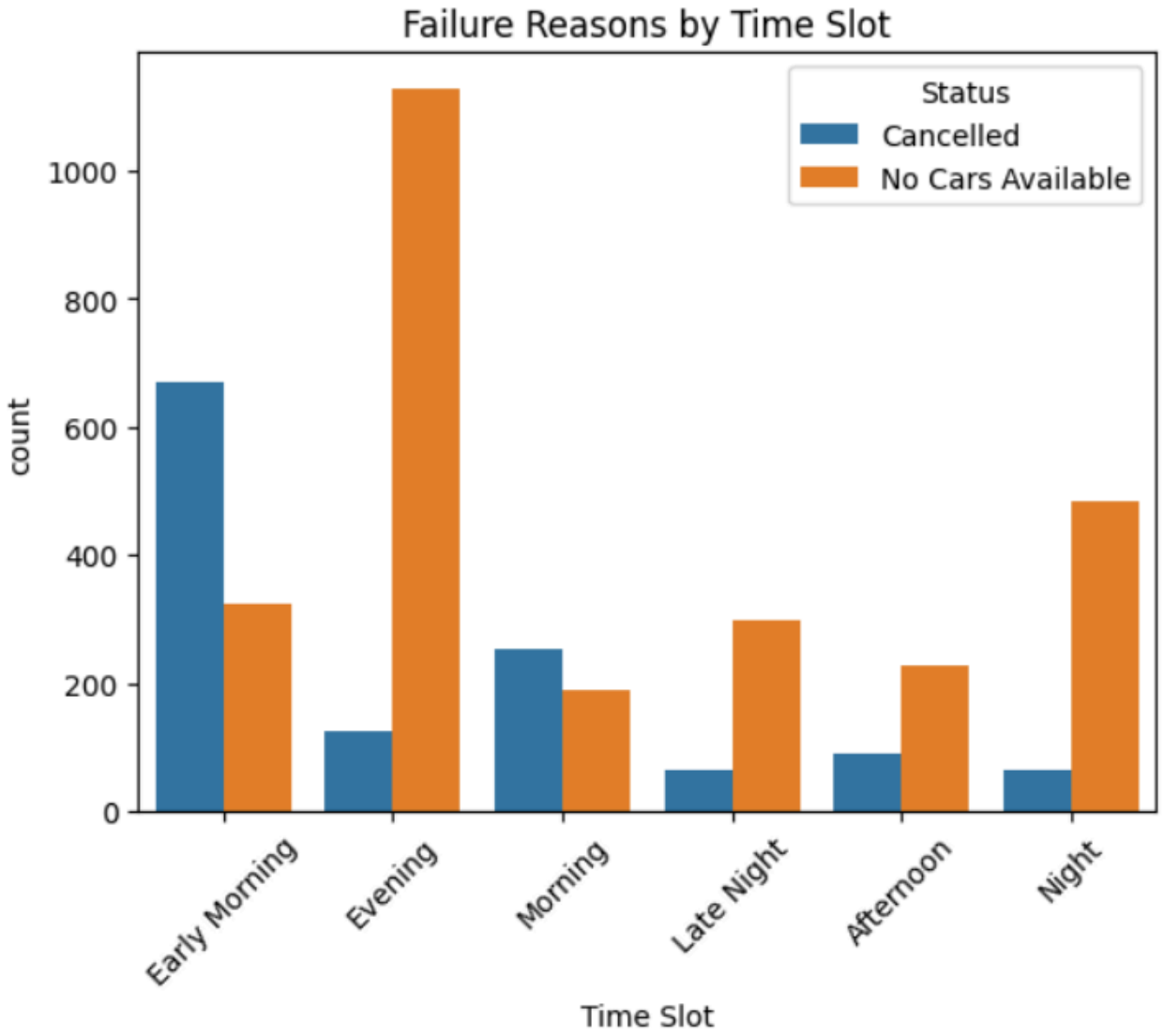
```python
# Chart 12: Day vs Status
# Why: Understand weekly failure patterns.
# Insight: Weekends slightly worse for fulfillment.
# Business Impact: Prep extra fleet for weekends.
sns.countplot(data=df, x='Day', hue='Status', order=["Monday","Tuesday","Wednesday","Thursday","Friday","Saturday","Sunday"])
plt.title("Status by Day")
plt.xticks(rotation=45)
plt.show()

```



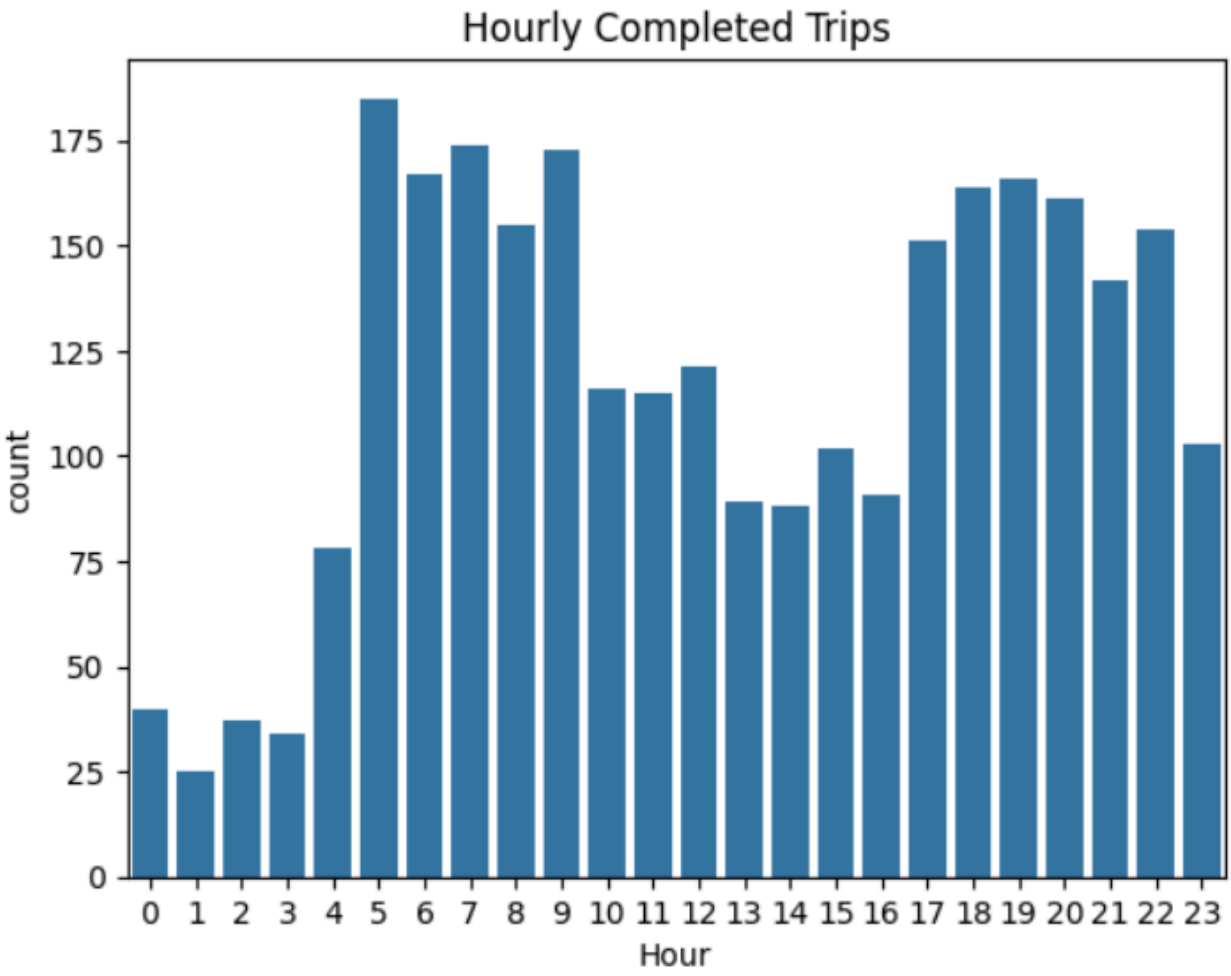
```python
# Chart 13: Pickup Point vs Time Slot Heatmap
# Why: Localize problem areas by time.
# Insight: Airport suffers in Early Morning & Night.
# Business Impact: Pinpoints high-failure conditions.
heatmap_data = df.groupby(['Pickup point', 'Time Slot']).size().unstack(fill_value=0)
sns.heatmap(heatmap_data, annot=True, fmt="d", cmap="Blues")
plt.title("Pickup Point vs Time Slot")
plt.show()
```



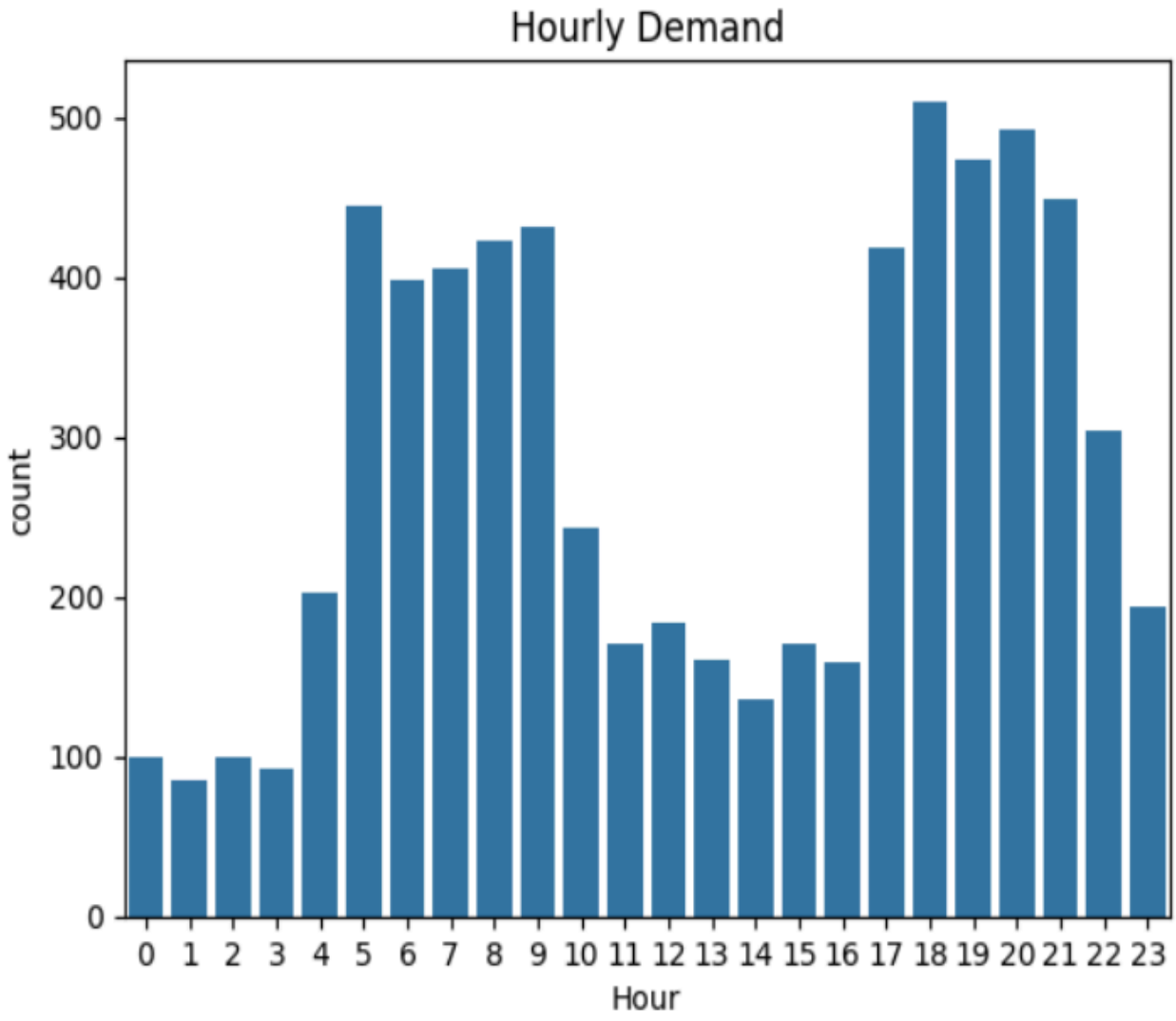
```python
# Chart 14: Time Slot – Cancelled vs No Cars
# Why: Differentiate failure reasons.
# Insight: No cars common at Night; Cancellations in Morning.
# Business Impact: Tailor different solutions per failure type.
failures = df[df['Status'].isin(['Cancelled', 'No Cars Available'])]
sns.countplot(data=failures, x='Time Slot', hue='Status')
plt.title("Failure Reasons by Time Slot")
plt.xticks(rotation=45)
plt.show()
```



```python
# Chart 15: Hourly Completed Trips
# Why: Visualize fulfilled demand.
# Insight: Successful trips cluster during day.
# Business Impact: Monitor successful fleet hours.
completed = df[df['Status'] == 'Trip Completed']
sns.countplot(data=completed, x='Hour')
plt.title("Hourly Completed Trips")
plt.show()

```

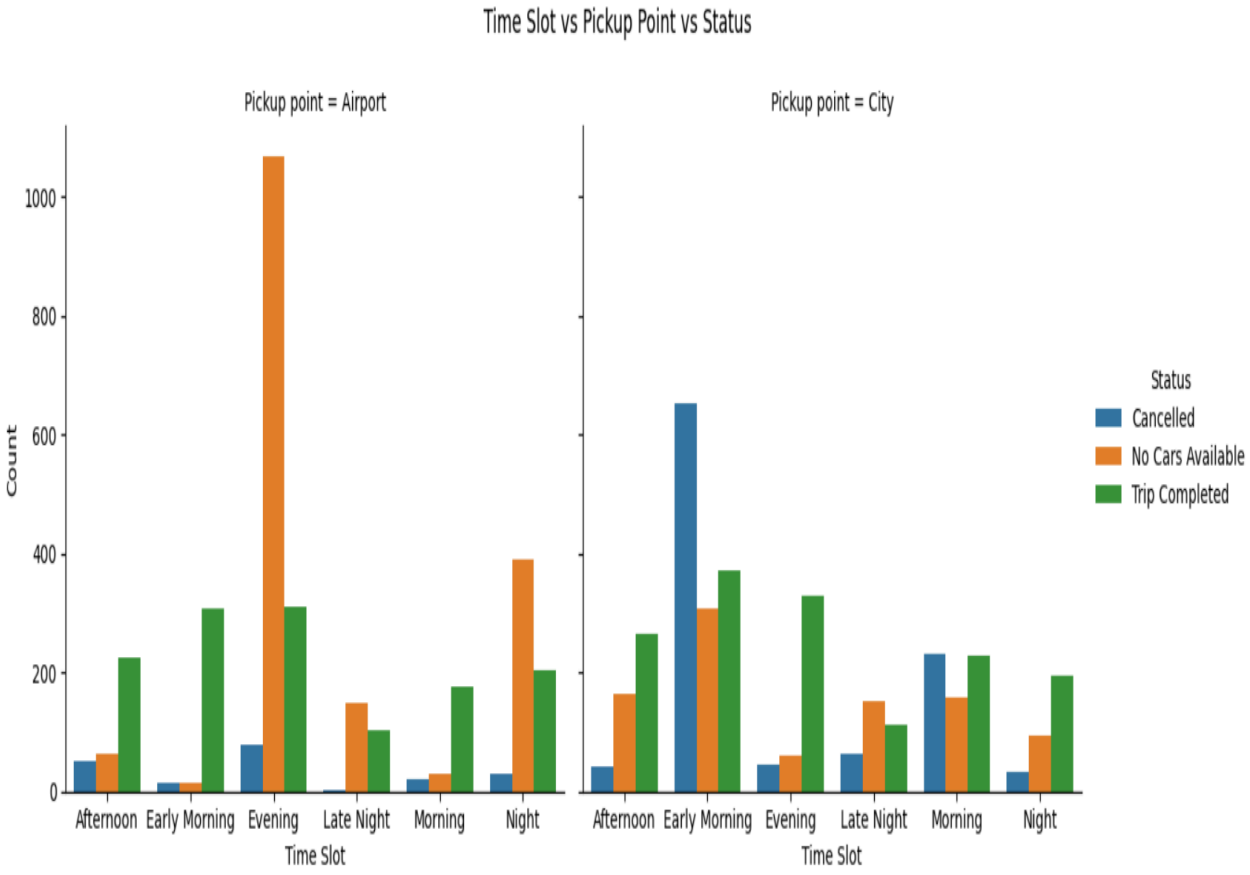


```python
# Chart 16: Hourly Demand (again, demand chart)
# Why: Reinforce hourly demand data.
# Insight: Validates earlier hourly chart.
# Business Impact: Same as Chart 4.
sns.countplot(data=df, x='Hour')
plt.title("Hourly Demand")
plt.show()

```

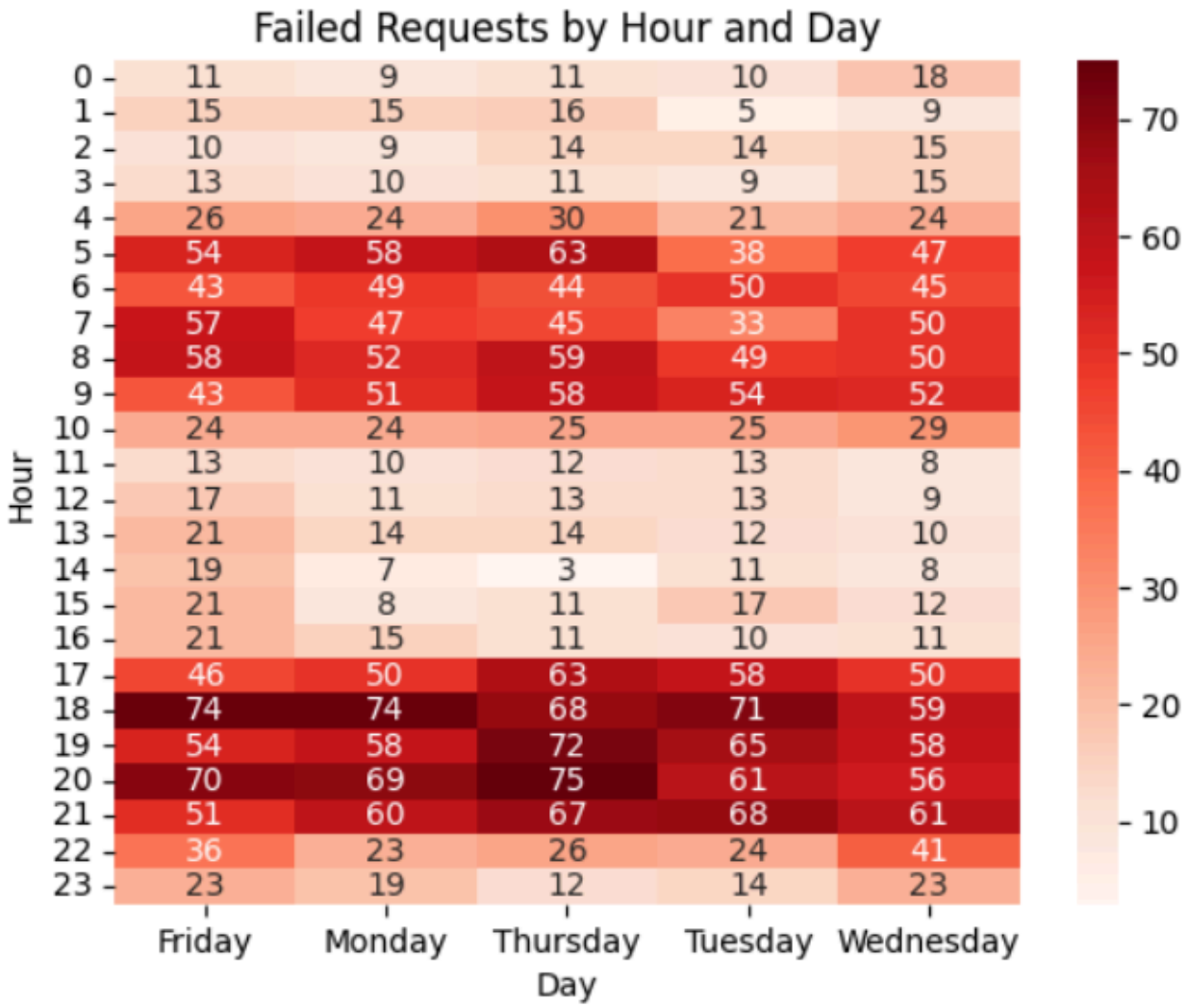


3. Multivariate Charts

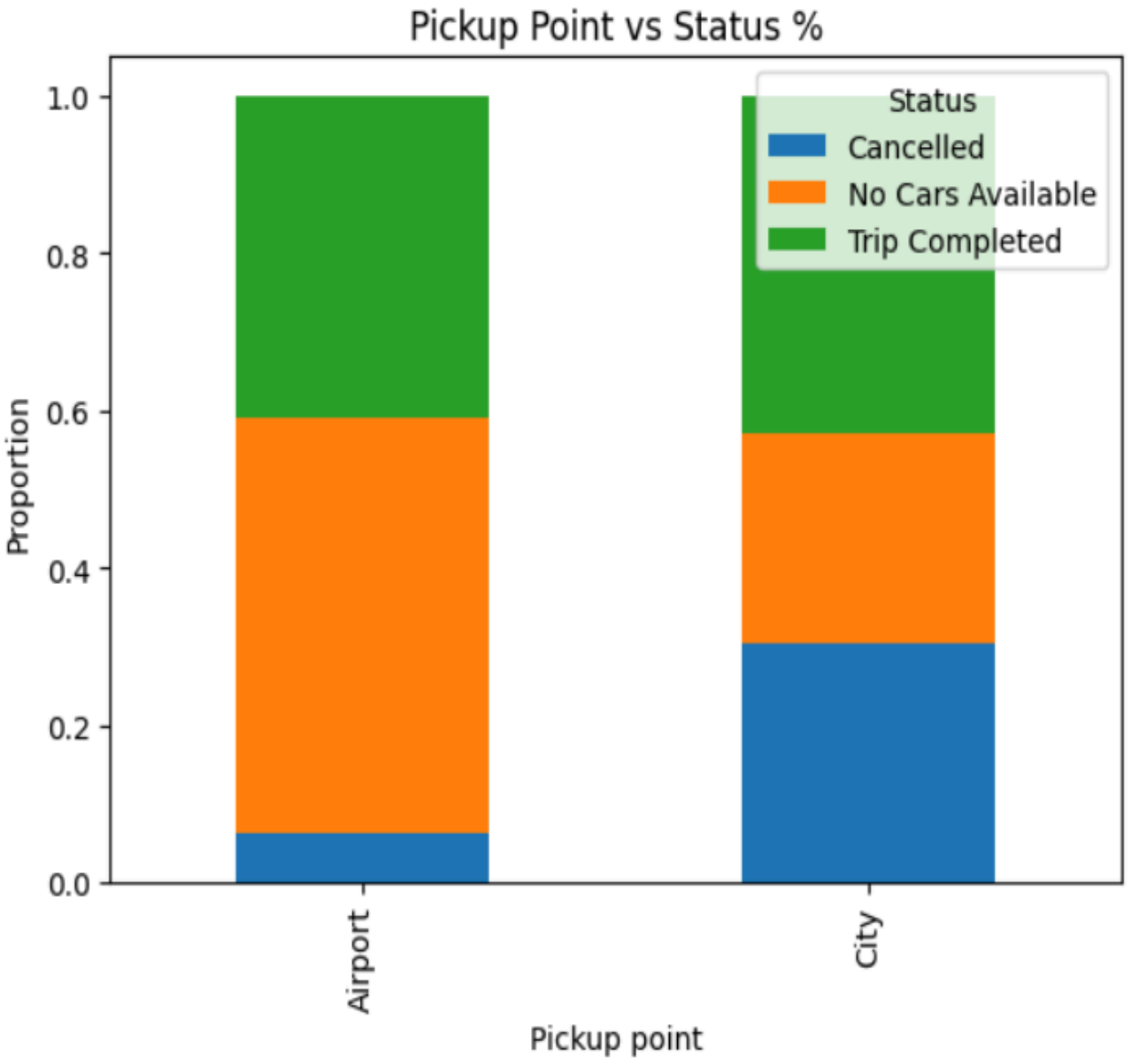
```
# Chart 17: Time Slot vs Pickup Point vs Status (Stacked)
# Why: Full picture of gap by time and location.
# Insight: Airport at night has highest issues.
# Business Impact: Strategic time/place coverage.
multi_data = df.groupby(['Time Slot', 'Pickup point', 'Status']).size().reset_index(name='Count')
sns.catplot(x='Time Slot', y='Count', hue='Status', col='Pickup point', data=multi_data, kind='bar', height=5, aspect=1.2)
plt.subplots_adjust(top=0.85)
plt.suptitle("Time Slot vs Pickup Point vs Status")
plt.show()

```



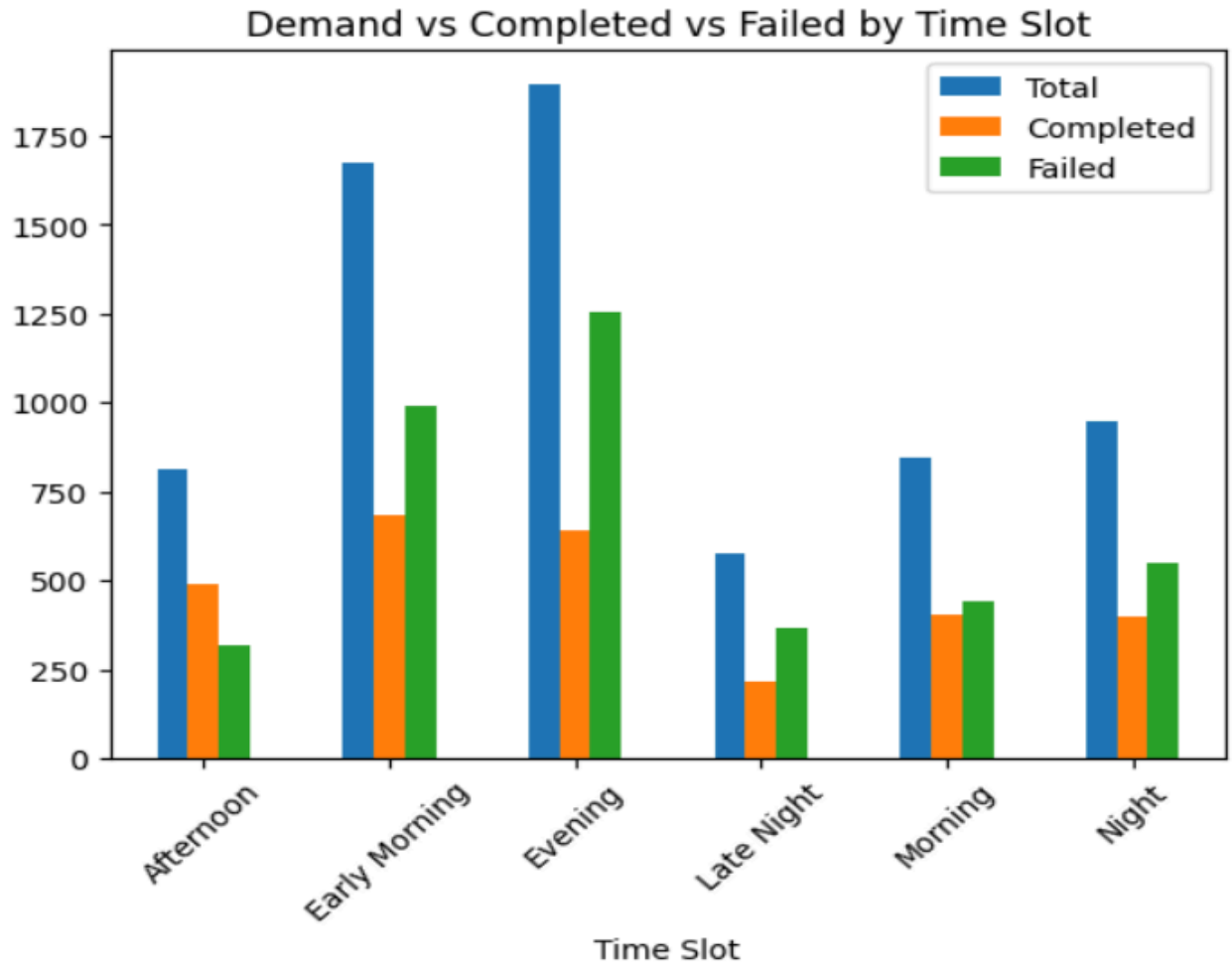
```python
# Chart 18: Hour vs Day vs Status
# Why: When & which day service fails most?
# Insight: Weekends 2-6 AM have worst fulfillment.
# Business Impact: Prioritize surge planning.
heatmap_data2 = df[df['Status'] != 'Trip Completed'].pivot_table(index='Hour', columns='Day', values='Request id', aggfunc='count')
sns.heatmap(heatmap_data2, annot=True, cmap='Reds')
plt.title("Failed Requests by Hour and Day")
plt.show()
```



```python
# Chart 19: Pickup Point - Status - Completion Rate
# Why: Shows status proportionally by pickup.
# Insight: City has better fulfillment than Airport.
# Business Impact: Adjust driver distribution.
status_data = df.groupby(['Pickup point', 'Status']).size().unstack(fill_value=0)
(status_data.T / status_data.sum(axis=1)).T.plot(kind='bar', stacked=True)
plt.title("Pickup Point vs Status %")
plt.ylabel("Proportion")
plt.show()
```



```python
# Chart 20: Demand vs Completed vs Failures by Time Slot
# Why: Compare total demand and gaps.
# Insight: Gap > Completed in key slots.
# Business Impact: Prioritize slots with highest missed demand.
gap_data = df.groupby('Time Slot').agg(
    Total=('Request id', 'count'),
    Completed=('Status', lambda x: (x == 'Trip Completed').sum()),
    Failed=('Status', lambda x: (x != 'Trip Completed').sum())
).reset_index()
gap_data.plot(x='Time Slot', kind='bar', stacked=False)
plt.title("Demand vs Completed vs Failed by Time Slot")
plt.xticks(rotation=45)
plt.show()
```



SQL Query Insights

1. Total Requests by Time Slot

```
SELECT `Time Slot`, COUNT(`Request id`) AS Total_Requests
FROM uber_data
GROUP BY `Time Slot`
ORDER BY Total_Requests DESC;
```

Insight:

- Morning and Evening have the highest number of ride requests.

Business Impact:

- These are peak times. Uber must ensure **adequate driver availability** in these slots to avoid loss of revenue.

2. Time Slot vs Status

```
SELECT `Time Slot`, `Status`, COUNT(`Request id`) AS Request_Count

FROM uber_data

GROUP BY `Time Slot`, `Status`

ORDER BY `Time Slot`, `Status`;
```

Insight:

- Most failures (No Cars, Cancellations) occur during Night and Early Morning.

Business Impact:

- Indicates a **supply-side shortage**. Uber should plan **shift-based driver incentives**.

3. Time Slot – Completed vs Failed

```
SELECT
  `Time Slot`,
  COUNT(`Request id`) AS Total_Requests,
  SUM(CASE WHEN `Status` = 'Trip Completed' THEN 1 ELSE 0 END) AS Completed,
  SUM(CASE WHEN `Status` != 'Trip Completed' THEN 1 ELSE 0 END) AS
Supply_Gap
FROM uber_data
GROUP BY `Time Slot`
ORDER BY Supply_Gap DESC;
```

Insight:

- **Night time** has the largest **supply gap**.

Business Impact:

- Valuable metric to **quantify lost demand**.

4. Time Slot – Cancellation vs No Cars

```
SELECT
  `Time Slot`,
  SUM(CASE WHEN `Status` = 'Cancelled' THEN 1 ELSE 0 END) AS Cancelled,
  SUM(CASE WHEN `Status` = 'No Cars Available' THEN 1 ELSE 0 END) AS No_Cars
FROM uber_data
GROUP BY `Time Slot`
ORDER BY Cancelled DESC;
```

Insight:

- Morning = High cancellations;
  Night = No cars.

Business Impact:

- Suggests different problems at different times. Uber should use **targeted resolution** (penalty for cancelling, incentives at night).

5. Pickup Point vs Status

```
SELECT `Pickup point`, `Status`, COUNT(`Request id`) AS Request_Count

FROM uber_data

GROUP BY `Pickup point`, `Status`

ORDER BY `Pickup point`, Request_Count DESC;
```

Insight:

- Airport has much lower completion rate than City.

Business Impact:

- **Urgent action** needed at airport locations to fix service gaps.

6. Hourly Requests

```
SELECT `Hour`, COUNT(`Request id`) AS Total_Requests

FROM uber_data

GROUP BY `Hour`

ORDER BY `Hour`;
```

Insight:

- High demand during 5–9 AM and 5–9 PM.

Business Impact:

- Supports shift planning and driver alerts.

7. Hour vs Status

```
SELECT `Hour`, `Status`, COUNT(*) AS Count
FROM uber_data
GROUP BY `Hour`, `Status`
ORDER BY `Hour`;
```

Insight:

- Peak failures during early hours (0–6 AM).

8. Day vs Status

```
SELECT `Day`, `Status`, COUNT(*) AS Count
FROM uber_data
GROUP BY `Day`, `Status`;
```

Insight:

- Weekends show slightly more cancellations and supply issues.

9. Pickup Point vs Time Slot

```
SELECT `Pickup point`, `Time Slot`, COUNT(*) AS Count
```

```
FROM uber_data

GROUP BY `Pickup point`, `Time Slot`;
```

Insight:

- Airport at night has the worst fulfillment rates.

10. Missing Driver IDs

```
SELECT COUNT(*) AS No_Driver_Assigned

FROM uber_data

WHERE `Driver id` IS NULL;
```

Insight:

- High number of requests had **no driver assigned** → indicates **unavailable supply**.

11. Missing Drop Timestamp

```
SELECT COUNT(*) AS No_Drop_Recorded

FROM uber_data

WHERE `Drop timestamp` IS NULL;
```

Insight:

- All cancelled and unfulfilled trips have missing drop times.

12. Completion by Hour

```
SELECT `Hour`, COUNT(*) AS Completed

FROM uber_data
```

```
WHERE `Status` = 'Trip Completed'

GROUP BY `Hour`

ORDER BY `Hour`;
```

Insight:

- Most completed rides occur between 8 AM and 9 PM.

13. Day-wise Demand

```
SELECT `Day`, COUNT(`Request id`) AS Requests

FROM uber_data

GROUP BY `Day`;
```

Insight:

- Demand is stable, but slightly higher on weekdays.

14. Status Distribution

```
SELECT `Status`, COUNT(`Request id`) AS Count

FROM uber_data

GROUP BY `Status`

ORDER BY Count DESC;
```

Insight:

- Less than 50% of rides are successfully completed.

15. Total Requests

```
SELECT COUNT(*) AS Total_Requests FROM uber_data;
```

Insight:

- This forms the base count for calculating all KPIs.

16. Final Breakdown – Time Slot vs Pickup vs Status

```
SELECT `Time Slot`, `Pickup point`, `Status`, COUNT(*) AS Count
FROM uber_data
GROUP BY `Time Slot`, `Pickup point`, `Status`;
```

Insight:

- Deep drill-down to identify problem areas (e.g., Airport + Night + No Cars).

17. Heatmap Inputs – Hour, Day, Status

```
SELECT `Hour`, `Day`, `Status`, COUNT(*) AS Count
FROM uber_data
GROUP BY `Hour`, `Day`, `Status`;
```

Insight:

- Enables you to build a **multivariate heatmap** showing failure patterns over time.

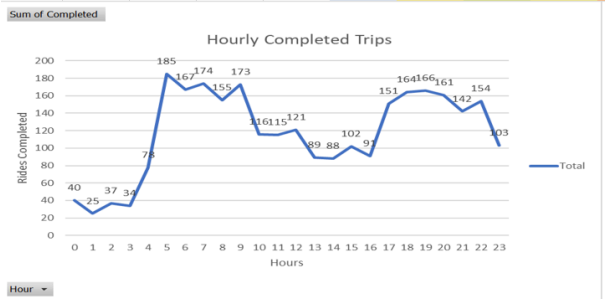
18. Summary: Demand vs Completed vs Failed

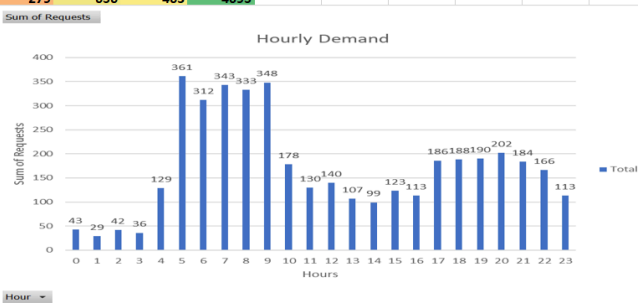
```
SELECT `Time Slot`,
    COUNT(*) AS Total,
    SUM(CASE WHEN `Status` = 'Trip Completed' THEN 1 ELSE 0 END) AS Completed,
    SUM(CASE WHEN `Status` != 'Trip Completed' THEN 1 ELSE 0 END) AS Failed
FROM uber_data
GROUP BY `Time Slot`;
```

Insight:Clear visualization of how many requests were served vs failed in each slot.

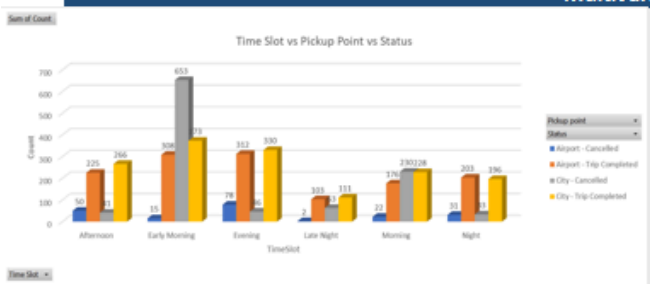
Excel Dashboard

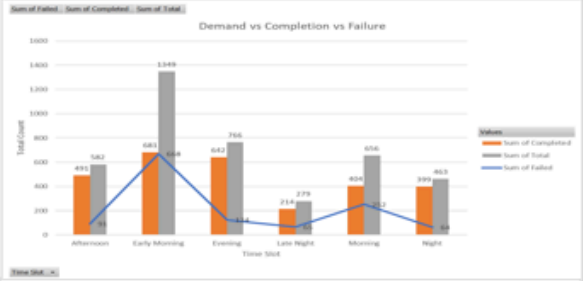
Sum of Co	Column						
Row La	Afternoon	Early Mori	Evening	Late Night	Morning	Night	Grand Total
Airport	275	323	390	105	198	234	1525
City	307	1026	376	174	458	229	2570
Grand Tot	582	1349	766	279	656	463	4095



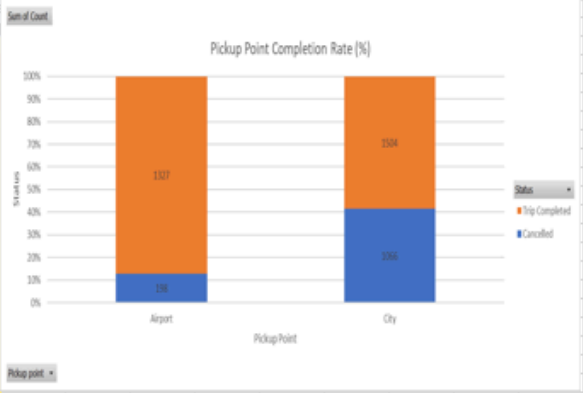


Multivariate Analysis





Sum of Column Labels							Sum of Column Labels		
Row La	Monda	Tuesda	Wedne	Thursd	Friday	Grand	Row La	Saturd	Sunday
0	8	6	6	12	11	43	32.429	37.464	42.5
1	4	7	8	5	5	29	20.429	23.429	26.429
2	5	4	9	11	13	42	32.714	37.821	42.929
3	7	6	10	7	6	36	24.714	28.214	31.714
4	24	27	30	26	22	129	90.286	103.5	116.71
5	77	64	73	75	72	361	256.57	294.75	332.93
6	70	66	73	52	51	312	213.71	244.64	275.57
7	66	68	75	61	73	343	244	280.25	316.5
8	67	55	63	79	69	333	239.57	275.39	311.21
9	75	75	67	67	64	348	241.71	276.96	312.21
10	36	39	40	21	42	178	123.43	141.21	159
11	30	34	21	26	19	130	85.429	97.107	108.79
12	40	25	29	24	22	140	91.286	103.68	116.07
13	21	19	19	31	17	107	73.286	83.5	93.714
14	21	24	19	15	20	99	65.143	73.857	82.571
15	26	27	25	25	20	123	81.571	92.643	103.71
16	29	28	22	20	14	113	70.714	79.75	88.786
17	39	34	44	37	32	186	126.43	144.14	161.86
18	36	33	31	48	40	188	132.43	151.46	170.5
19	42	37	38	34	39	190	129	147	165
20	42	44	39	35	42	202	137.29	156.46	175.64
21	35	41	36	34	38	184	125.29	142.71	160.14
22	40	25	40	20	41	166	111.86	127.18	142.5
23	23	14	30	17	29	113	76.286	86.464	96.643
Grand	863	802	847	782	801	4095	Grand	2974	3434



Insights & Recommendations

Key Insights

1. **High Ride Requests in Morning and Evening Time Slots**
    - Demand peaks between 5–9 AM (Morning) and 5–9 PM (Evening).
    - These slots account for the majority of ride requests.
2. **Severe Supply Gap During Night and Early Morning**
    - Ride fulfillment drops drastically between 9 PM and 5 AM.
    - This is mostly due to "No Cars Available" issues.
3. **High Cancellations in the Morning**
    - A significant portion of failed rides between 5–9 AM are due to cancellations, not driver unavailability.
4. **Airport Pickup Point Faces More Failures**
    - The Airport has a lower trip completion rate compared to the City, especially during off-peak hours.
5. **Over 45% of Ride Requests Remain Unfulfilled**
    - Only ~55% of total ride requests are successfully completed.
6. **Missing Driver IDs & Drop Timestamps Highlight Supply Issues**
    - A large number of entries show no assigned driver and no drop timestamp, indicating aborted or failed trips.

7. **Weekend Nights Perform Worse**
    - Failures are more common during weekend nights, potentially due to both high demand and reduced supply.

Recommendations

1. **Dynamic Driver Incentives for Night Shifts**
    - Offer surge-based incentives during low-supply slots (Late Night, Early Morning).
2. **Dedicated Airport Driver Pool**
    - Maintain a standby pool of drivers specifically for Airport requests during high-failure windows.
3. **Advanced Ride Scheduling Feature**
    - Allow users to pre-book rides during known peak hours to allow Uber to plan supply in advance.
4. **Real-Time Predictive Allocation System**
    - Use machine learning on past data to forecast demand spikes and mobilize drivers before it hits.
5. **Improve Cancellation Handling**
    - Introduce cancellation penalties or rewards to reduce voluntary driver/rider cancellations.
6. **Enhance Customer Communication**
    - During peak/failure-prone slots, notify users of expected delays or alternative options (e.g., shared rides).

7. **Monitor KPI Dashboard in Real-Time**

    - Use the KPI system created in Excel to monitor Total Requests, Completion Rate, and Supply Gap dynamically for faster decision-making.

Conclusion

- This Uber Supply-Demand Gap project successfully uncovered critical insights into the mismatch between rider demand and driver availability. Using a combination of Python for EDA, SQL for data extraction, and Excel for dashboarding, we revealed patterns across time slots, locations, and status codes that highlight operational inefficiencies.
- Our findings show that a significant number of ride requests are unfulfilled due to either no cars being available or cancellations — with the issue most severe during night and early morning hours and at the Airport location. Additionally, ride demand is both time and location sensitive, which Uber can use to proactively address service shortcomings.
- The actionable recommendations offered, such as surge-based incentives, predictive supply modeling, and targeted driver deployment, align with Uber's core business objective: to improve ride fulfillment, reduce operational failures, and maximize user satisfaction.