

Jalari Mallikarjuna

# Uber Analysis

The Uber Analysis project examines ride data to uncover trends and optimize service efficiency

[project link](#)

# Agenda

- 1 Introduction
- 2 Key Concepts
- 3 Data Collection
- 4 Data Preprocessing
- 5 EDA
- 6 Vizualizations



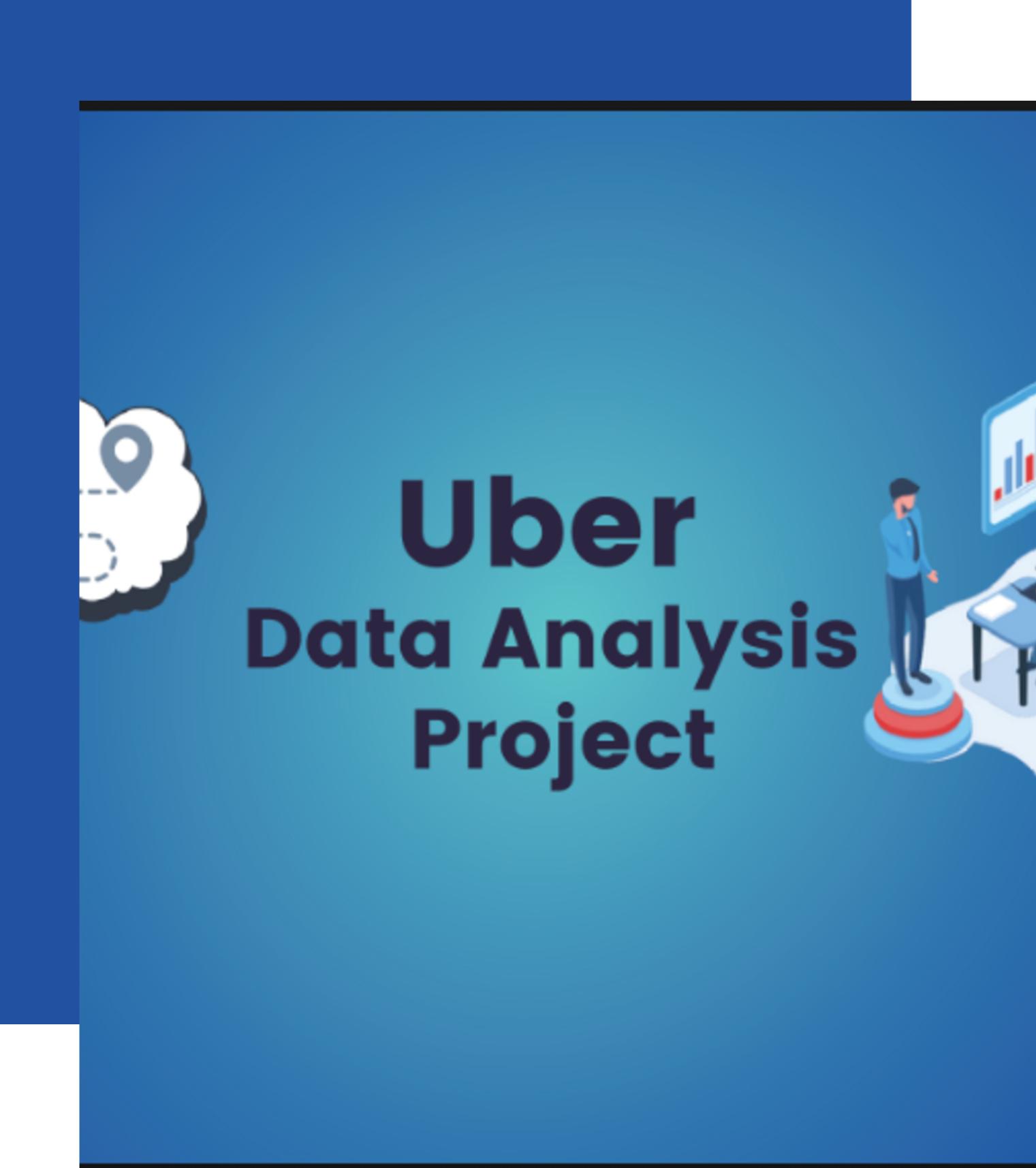
# Agenda

- 7 Insights and Findings
- 8 Conclusion
- 9 End



# Introduction

---



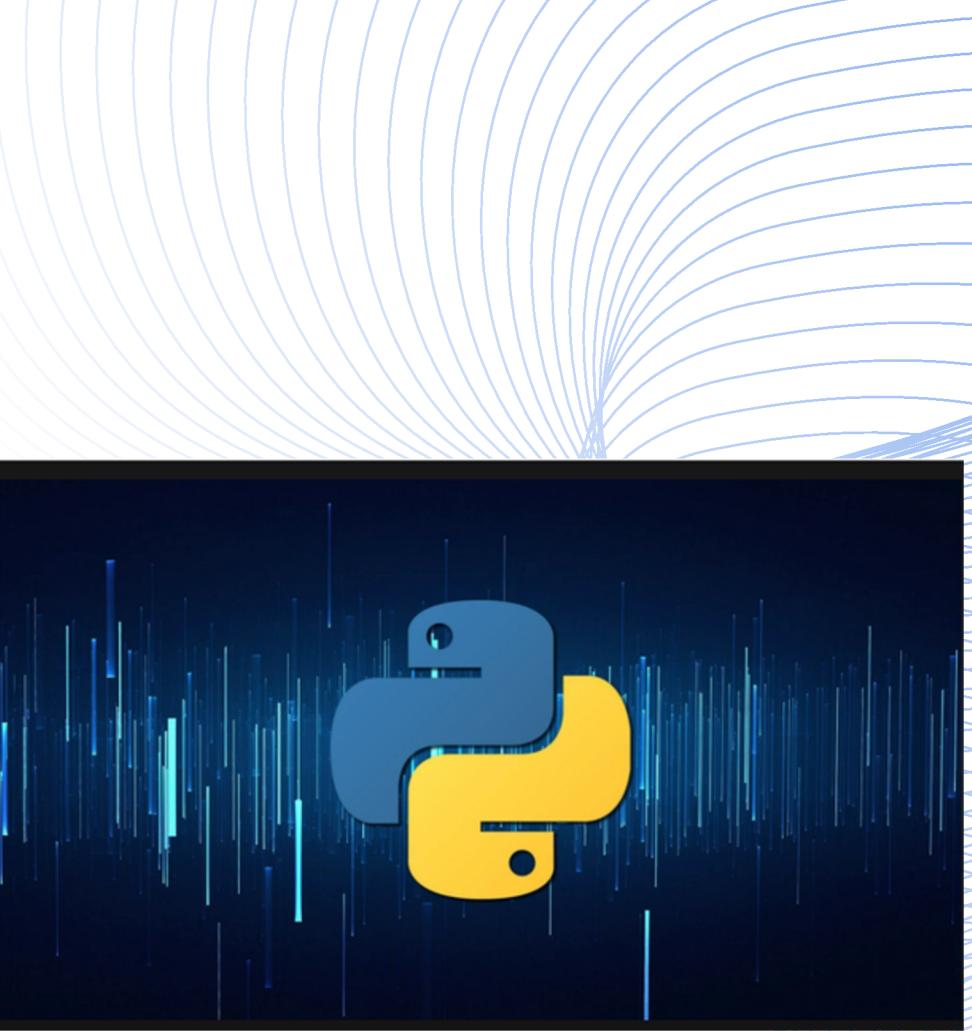
## Uber Data Analysis Project

- Objective: Analyze Uber trip data to uncover trends in ride demand, peak hours, and popular locations.
- Focus Areas: Identify trip patterns, seasonal variations, and operational efficiency opportunities.
- Key Insights: Optimize driver allocation, improve service availability, and enhance user experience.
- Data Utilized: Includes trip timings, pickup/dropoff locations, trip durations, and ride status.
- Outcome: Provide actionable recommendations to guide Uber's strategic decisions and business improvements.
- Tools: Data analysis and visualization techniques to communicate findings effectively.

# Project Needs

---

- Jupyter Notebook for interactive data analysis and visualization.
- Python as the primary language for data processing and algorithm implementation.
- Anaconda for managing packages, dependencies, and virtual environments.
- VSCode for coding and debugging with an efficient editor.
- GitHub for version control and project collaboration.



# Environmental Variables

- **Package Installation:** Installing required libraries and dependencies for smooth project execution.
- **Environment Variables:** Configuring variables to ensure compatibility with system settings.
- **Compatibility Check:** Ensuring all tools, libraries, and dependencies are compatible with the project setup.
- **Efficient Execution:** Setting up the environment to facilitate seamless data analysis and algorithm execution.

```
C:\Users\jalar>conda
usage: conda-script.py [-h] [-v] [--no-plugins] [-V] COMMAND ...

conda is a tool for managing and deploying applications, environments and packages.

options:
  -h, --help            Show this help message and exit.
  -v, --verbose         Can be used multiple times. Once for detailed output, twice for INFO logging, thrice for DEBUG
                        logging, four times for TRACE logging.
  --no-plugins          Disable all plugins that are not built into conda.
  -V, --version         Show the conda version number and exit.

commands:
  The following built-in and plugins subcommands are available.

COMMAND
  activate              Activate a conda environment.
  build                 Build conda packages from a conda recipe.
  clean                 Remove unused packages and caches.
  compare               Compare packages between conda environments.
  config                Modify configuration values in .condarc.
  content-trust         Signing and verification tools for Conda
  convert               Convert pure Python packages to other platforms (a.k.a., subdirs).
  create                Create a new conda environment from a list of specified packages.
  deactivate             Deactivate the current active conda environment.
  debug                 Debug the build or test phases of conda recipes.
  develop               Install a Python package in 'development mode'. Similar to 'pip install --editable'.
```

```
C:\Program Files\Common Files\Oracle\Java\javapath
%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0
%SYSTEMROOT%\System32\OpenSSH\
%JAVA_HOME%\bin
%SPARK_HOME%\bin
%HADOOP_HOME%\bin
C:\Program Files\Git\cmd
C:\Program Files\Amazon\AWSCLIV2\
```

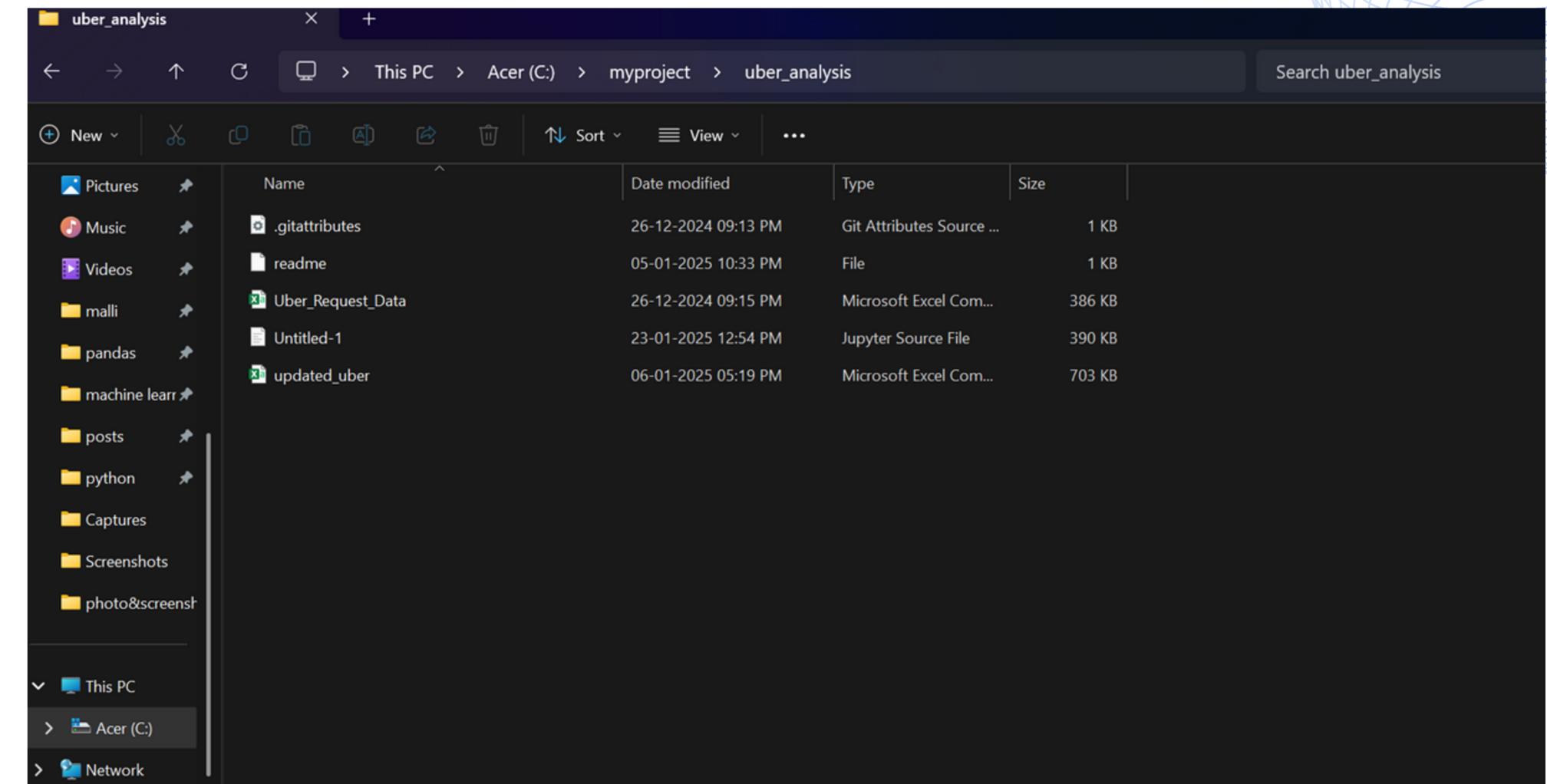
```
jalar>jupyter notebook
26 16:41:25.577 [ServerApp] Extension package jupyter_lsp took 0.3118s to import
26 16:41:25.981 [ServerApp] Extension package jupyter_server_terminals took 0.4017s to import
26 16:41:26.845 [ServerApp] jupyter_lsp | extension was successfully linked.
26 16:41:26.862 [ServerApp] jupyter_server_terminals | extension was successfully linked.
26 16:41:26.886 [ServerApp] jupyterlab | extension was successfully linked.
26 16:41:26.906 [ServerApp] notebook | extension was successfully linked.
26 16:41:27.714 [ServerApp] notebook_shim | extension was successfully linked.
26 16:41:27.776 [ServerApp] notebook_shim | extension was successfully loaded.
26 16:41:27.784 [ServerApp] jupyter_lsp | extension was successfully loaded.
26 16:41:27.784 [ServerApp] jupyter_server_terminals | extension was successfully loaded.
26 16:41:27.784 [LabApp] JupyterLab extension loaded from C:\Users\jalar\AppData\Local\jupyter\lab
26 16:41:27.792 [LabApp] JupyterLab application directory is C:\Users\jalar\AppData\Local\jupyter\lab
26 16:41:27.792 [LabApp] Extension Manager is 'pypi'.
26 16:41:28.495 [ServerApp] jupyterlab | extension was successfully loaded.
26 16:41:28.517 [ServerApp] notebook | extension was successfully loaded.
26 16:41:28.520 [ServerApp] Serving notebooks from local directory: C:\Users\jalar
26 16:41:28.520 [ServerApp] Jupyter Server 2.14.2 is running at:
26 16:41:28.520 [ServerApp] http://localhost:8888/tree?token=804c0e5bb456fbcc2d75542e
26 16:41:28.520 [ServerApp] http://127.0.0.1:8888/tree?token=804c0e5bb456fbcc2d75542e
26 16:41:28.520 [ServerApp] Use Control-C to stop this server and shut down all kernels.
```

# Project Setup

## Create Environment

- **Local Storage:** Jupyter Notebook files are stored on the local desktop for easy access and management.
- **Organized Directory:** A dedicated directory is created to structure and organize notebooks systematically.
- **Global Python Script:** A global script (environment.py) is used to execute the analysis, containing essential imports, configurations, and reusable functions.
- **Efficiency:** This setup ensures that all project files are easily accessible and that Python code execution is streamlined.

C:\myproject\uber\_analysis



# Project Setup

---

## Clone GitHub Repository to VSCode

- 1.Create a GitHub repository.
- 2.Clone the repository to VSCode.
- 3.Set up Git for version control.

## Project Requirements and Libraries

- 1.Required libraries include Matplotlib, Pandas, and NumPy etc .
- 2.Install libraries using pip or conda.
- 3.Verify library versions.

# Data Analytics Process

---

## Data Analytics Process



DATA  
COLLECTION



DATA  
PREPARATION



DATA  
ANALYSIS



DATA  
VISUALIZATION

# Data Collection

---

- **Libraries Imported:** Used Pandas for data manipulation, NumPy for numerical operations, and Matplotlib for data visualization.
- **Data Source:** Dataset provided by Instrovate Technologies, containing detailed Uber ride request information.
- **Key Attributes:** Includes request ID, pickup/dropoff points, driver IDs, status, and timestamps.
- **Data Processing:** Utilized Pandas for loading, cleaning, and analyzing the data to generate actionable insights.



# Data collection

```
[93] import matplotlib.pyplot as plt  
import seaborn as s  
import warnings  
warnings.filterwarnings("ignore")
```

```
[94] uber=pd.read_csv("Uber_Request_Data.csv")
```

```
[95] uber.head()
```

...	Request id	Pickup point	Driver id	Status	Request timestamp	Drop timestamp
0	619	Airport	1.0	Trip Completed	11/7/2016 11:51	11/7/2016 13:00
1	867	Airport	1.0	Trip Completed	11/7/2016 17:57	11/7/2016 18:47
2	1807	City	1.0	Trip Completed	12/7/2016 9:17	12/7/2016 9:58
3	2532	Airport	1.0	Trip Completed	12/7/2016 21:08	12/7/2016 22:03
4	3112	City	1.0	Trip Completed	13-07-2016 08:33:16	13-07-2016 09:25:47

# Data Preprocessing

---

- **Data Loading:** Imported the dataset into a Pandas DataFrame using `pd.read_csv()` for easy manipulation.
- **Handling Missing Values:** Dropped rows with missing data using `dropna()` to maintain data integrity.
- **Datetime Conversion:** Converted `request_timestamp` and `drop_timestamp` columns to datetime format using `pd.to_datetime()` for easier time-based analysis.
- Extracted Hour of the Day to analyze hourly trends.
- Derived Day of the Week to examine weekday vs. weekend patterns.
- Created a Month feature to observe seasonal trends.
- **Data Type Conversion:** Optimized data by converting columns like `request_id` and `driver_id` to string types.

# Exploratory Data Analysis

---

## Summary Statistics:

- Computed basic statistics to overview numeric features and identify outliers.

## Request Status Distribution:

- Analyzed the status of ride requests (completed, canceled, no cars available) and visualized with a bar plot.

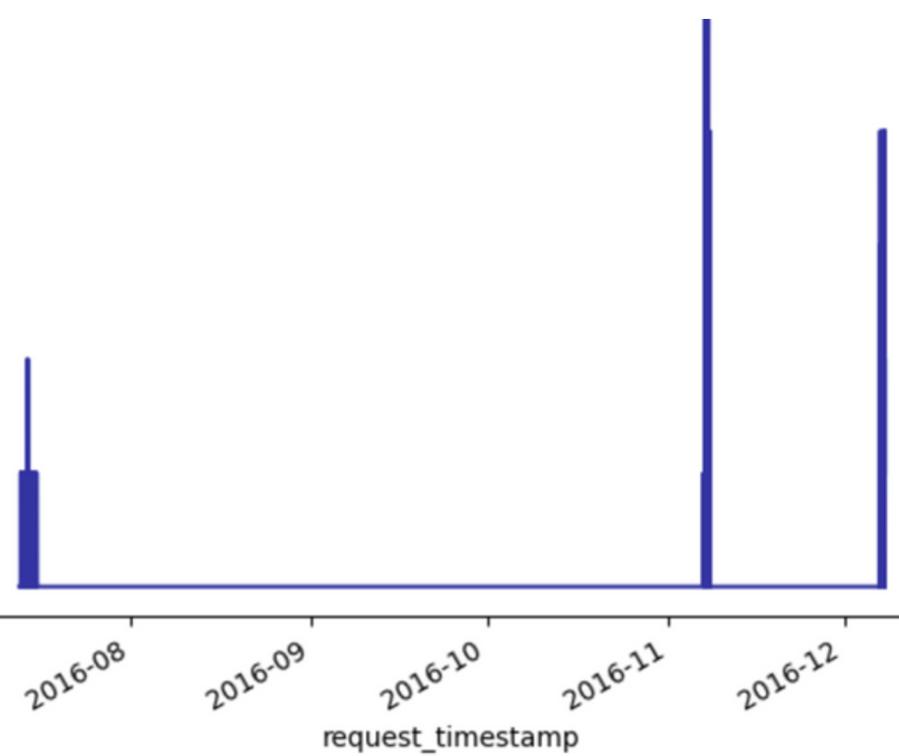
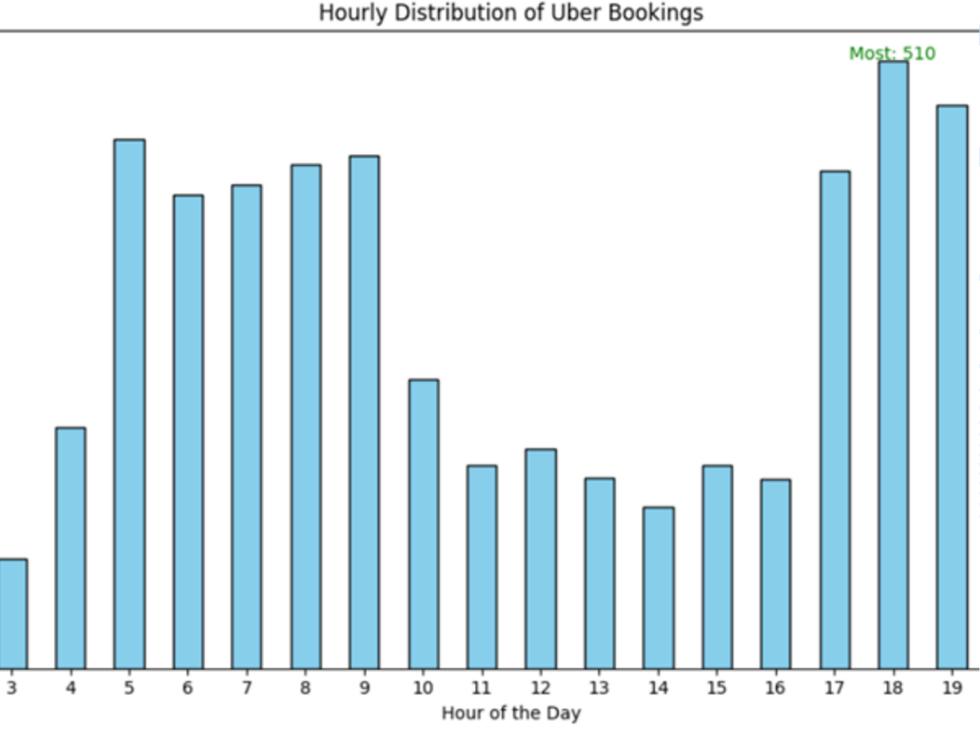
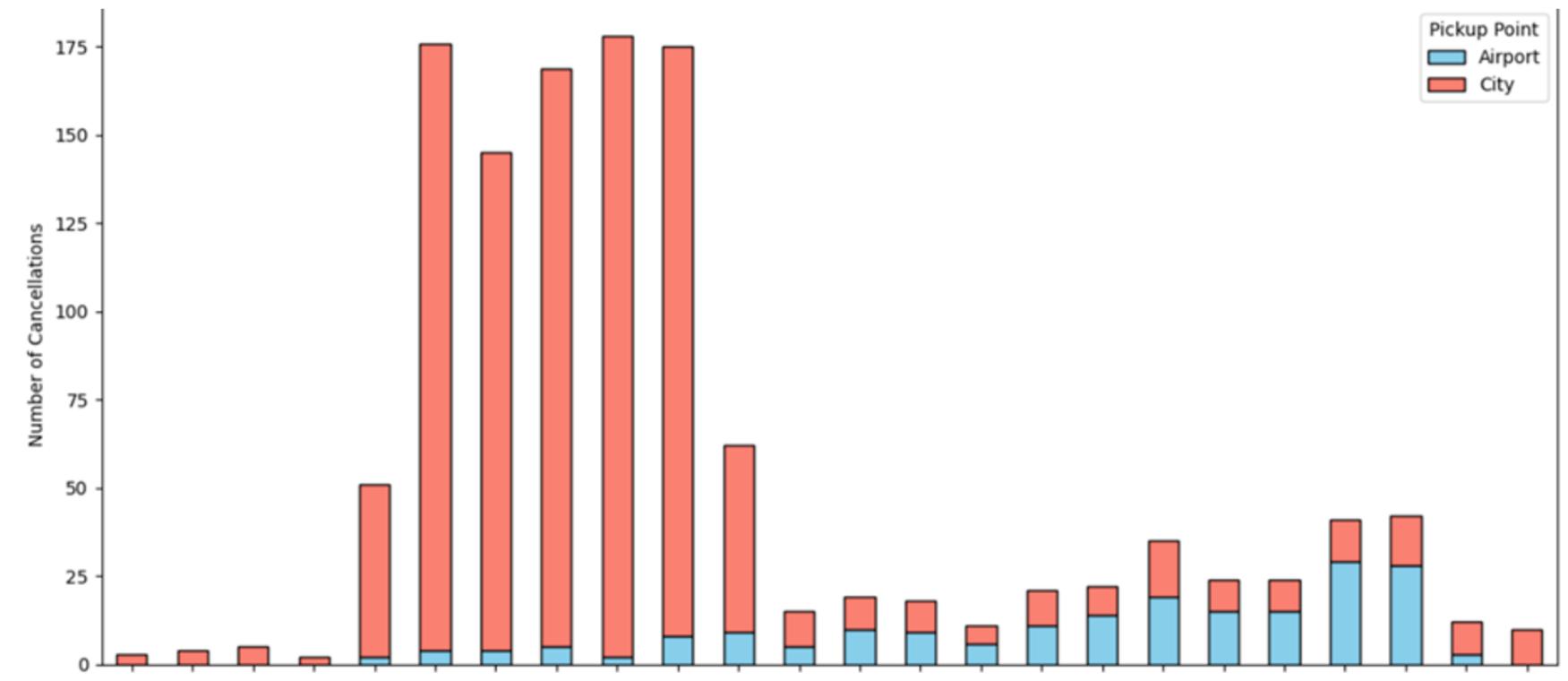
## Request Timing Analysis:

- Studied distribution of requests by time of day, weekdays, and months using histograms.

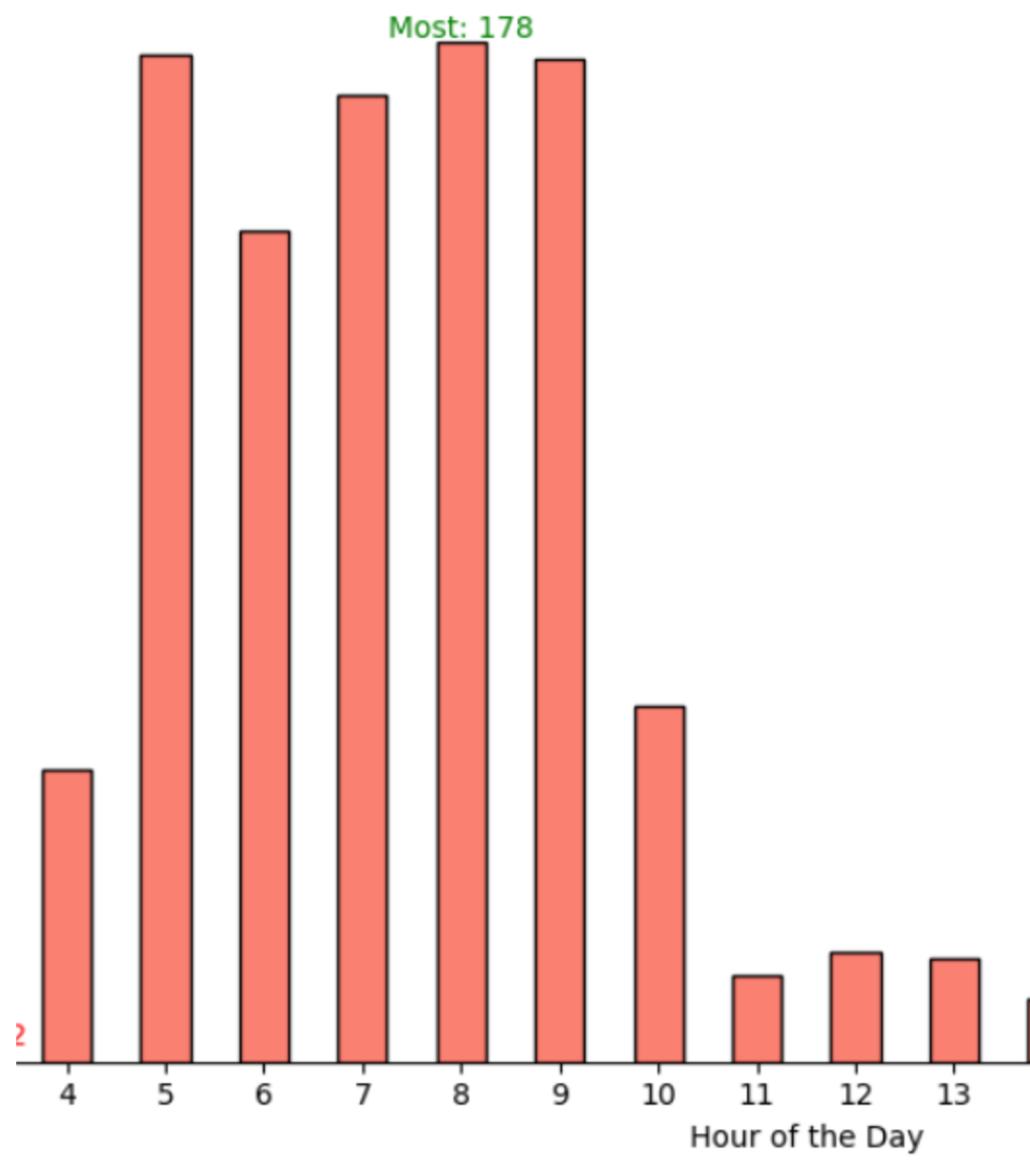
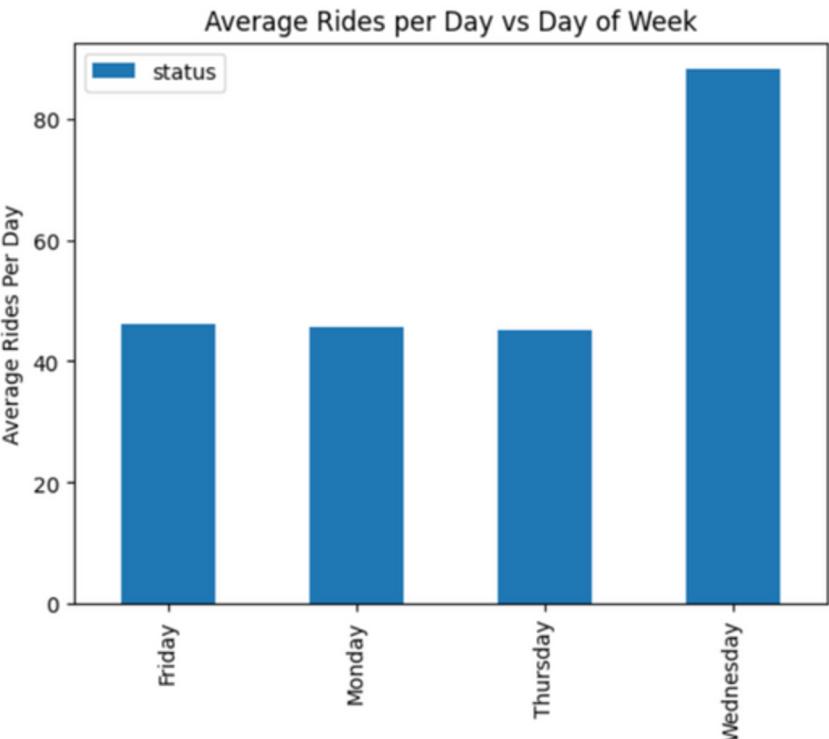
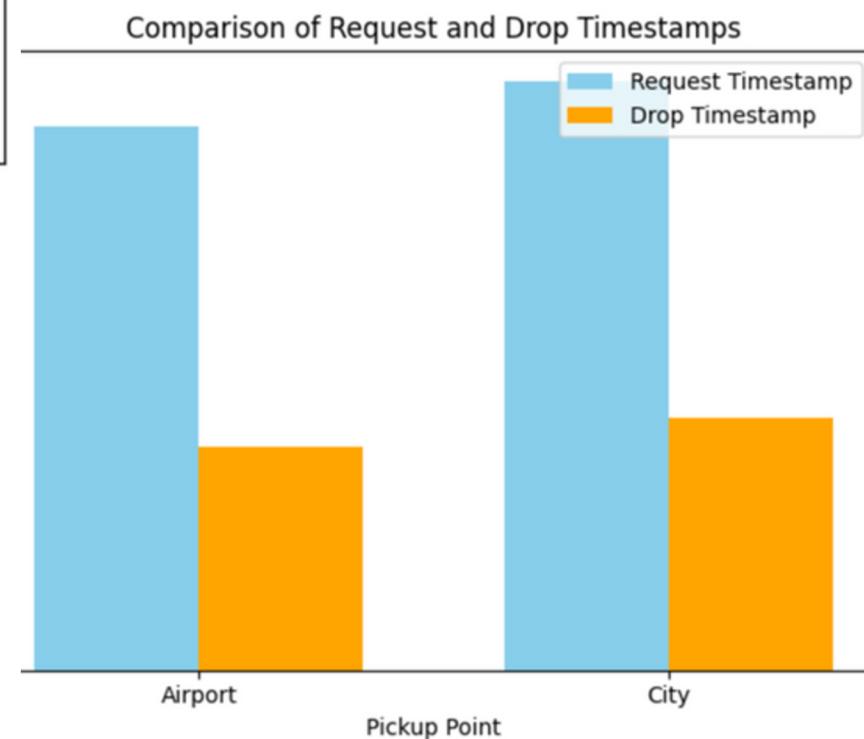
## Driver Activity Analysis:

- Analyzed the frequency of requests per driver to identify any imbalances in driver workload.





# Visualization



# Insights and Findings

---

01

## Hourly Trends

Ride requests peaked during morning rush hours (7 AM–9 AM) and evening rush hours (5 PM–8 PM), while late-night demand (12 AM–5 AM) was significantly lower.

02

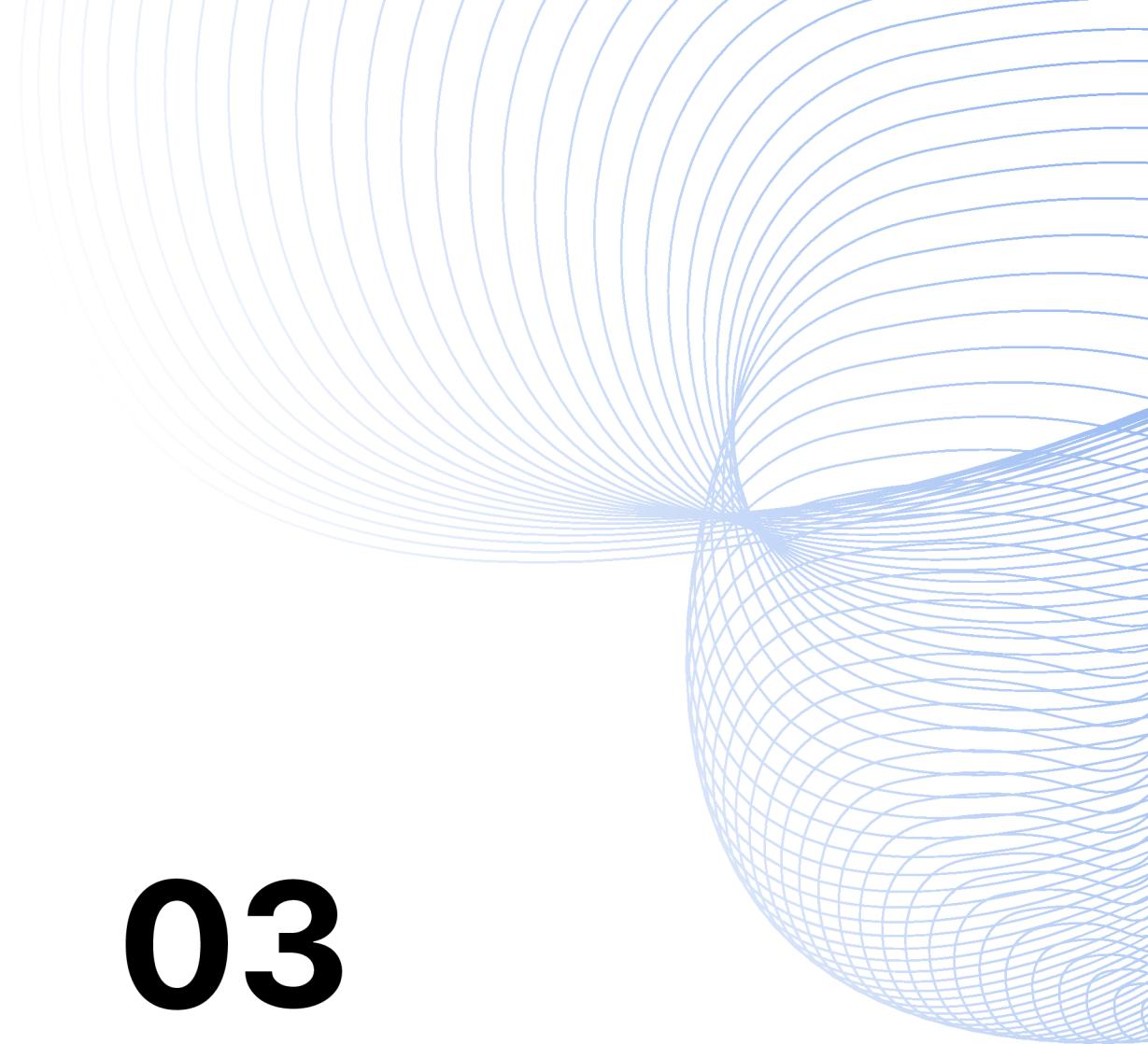
## Request Status

A significant percentage of requests were canceled or marked as "No Cars Available," particularly during high-demand periods

03

## Geospatial Trends

High-demand locations included city centers and airports, with dense clusters of pickups and drop-offs in these areas.



## Conclusion

This project uncovered critical patterns in Uber's ride request data, including peak demand during morning and evening rush hours and higher daytime demand compared to nighttime. Wednesdays recorded the highest number of successful rides. Geospatial analysis identified city centers and airports as high-demand zones, vital for operational planning. Additionally, an imbalance in driver workload was observed, highlighting opportunities for optimizing resource allocation.



thank you!

