THAKUR DEGREE COLLEGE OF SCIENCE & COMMERCE

KANDIVALI (EAST)

MUMBAI

A PROJECT REPORT ON

Uber Data Analysis

Designed and Developed

BY: **YOGESH SHRINATH PAL**

Submitted in partial fulfillment of

Bachelors of Science (Computer Science)

[UNIVERSITY OF MUMBAI]

Thakur Degree College of Science and Commerce
KANDIVALI (EAST)-

MUMBAI ACADEMIC YEAR 2023 - 2024

Thakur Educational Trust's (Regd.)

THAKUR COLLEGE OF SCIENCE & COMMERCE

AUTONOMOUS COLLEGE, PERMANENTLY AFFILIATED TO UNIVERSITY OF MUMBAI

NAAC Accredited Grade 'A' (3rd Cycle) & ISO 9001: 2015 (Certified)

Best College Award by University of Mumbai for the Year 2018-2019

CELEBRATING 25 YEARS OF GLORY

DATE:-

COMPUTER SCIENCE DEPARTMENT

(2023-2024)

Certificate of Approval

This is to certify that the project work entitled "Uber Data Anaysis" is prepared by Yogesh Shrinath Pal a student of "Third Year Bachelor Of Science (Computer Science)" course of University of Mumbai, which is conducted by our college.

This is the original study work and important sources used have been duly acknowledged in the report. The report is submitted in partial fulfillment of B.Sc. (Computer Science) course as per rules of University of Mumbai.

MISS: SIDDHI MA`AM — Project Guide

MR. ASHISH TRIVEDI — Head of Department

External Examiner

INDEX

ACKNOWLEDGEMENT

Achievement is finding out what you would be doing rather than what you have to do. It is not until you undertake such a project that you realize how much effort and hard work it really is, what are your capabilities and how well you can present yourself or other things. It gives me immense pleasure to present this report towards the fulfillment of my project.

It has been rightly said that we are built on the shoulder of others. For everything I have achieved, the credit goes to all those who had helped me to complete this project successfully.

I take this opportunity to express my profound gratitude to management of Thakur Degree College of Science & Commerce for giving me this opportunity to accomplish this project work.

I am very much thankful to **Dr.Mrs. C. T. Chakraborty** – Principal of Thakur College for their kind co-operation in the completion of my project.

A special vote of thanks to my faculty, Mr. Ashish Trivedi who is our HOD & also our project guide **Mr Girish Tere sir** for their most sincere, useful and encouraging contribution throughout the project span, without them we couldn't start and complete the project on time.

Finally, I would like to thank all my friends & entire Computer Science department who directly or indirectly helped me in completion of this project & to my family without whose support, motivation & encouragement this would not have been possible.

(YOGESH SHRINATH PAL)

PRELIMINARY INVESTIGATION

Organizational Overview

The Thakur College of Science and Commerce (TCSC) is a college in Kandivali in Mumbai of Maharashtra, India running by Thakur Educational Trust.

Thakur College was started in 1992 to serve the needs of students passing SSC examination from the schools around Kandivali area and Thakur Vidhya Mandir which has already established itself as one of the schools in the area. It offers courses at primarily the higher secondary and under-graduate levels. The courses at the undergraduate and postgraduate level are offered in affiliation with Mumbai University, Mumbai. An ISO 9001:2008 College with A grade as assessed by the National Assessment and Accreditation Council NAAC.

Name : Thakur College Of Science & Commerce

Founded :1997 Address : Thakur Village, Kandivali (East) Mumbai – 400101

Contacts : 022-2846 2565 / 2887 0627

Motto : Journey towards Excellence

Email : Helpdesk@tcsc.org.in

Introduction

Sometimes it's easy to give up on someone else's driving. This is less stress, more mental space and one uses that time to do other things. Yes, that's one of the ideas that grew and later became the idea behind **Uber and Lyft**.

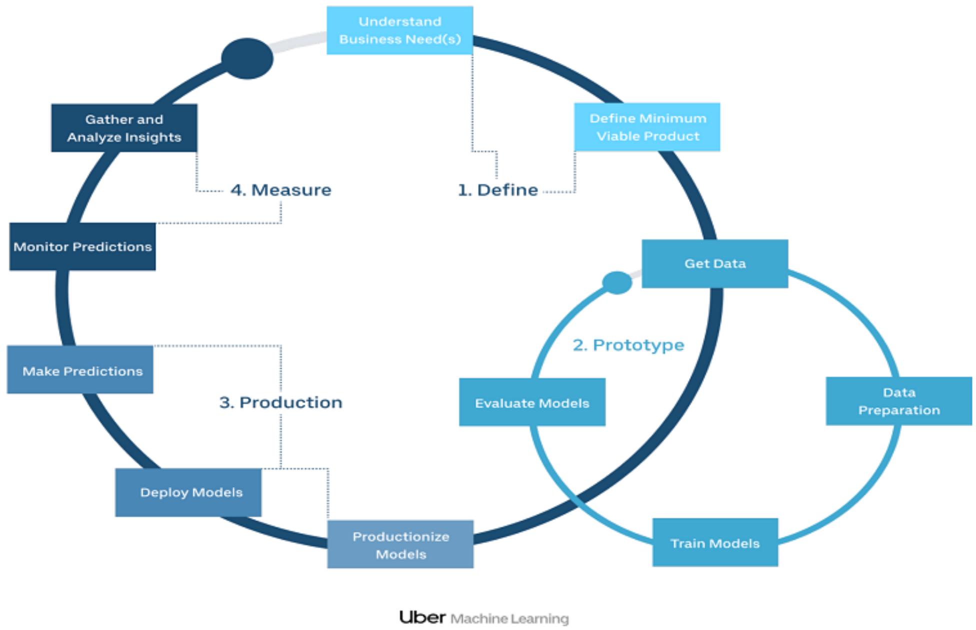
Both companies offer passenger boarding services that allow users to rent cars with drivers through websites or mobile apps. Whether traveling a short distance or traveling from one city to another, these services have helped people in many ways and have actually made their lives very difficult.

Uber is an international company located in 69 countries and around 900 cities around the world. Lyft, on the other hand, operates in approximately 644 cities in the US and 12 cities in Canada alone. However, in the US, it is the second-largest passenger company with a market share of 31%.

From booking a taxi to paying a bill, both services have similar features. But there are some exceptions when the two passenger services reach the neck. The same goes for prices, especially **Uber's** "**surge**" and "Prime Time" in Lyft. There are certain limitations that depend on where service providers are classified.

Many articles focus on algorithm/model learning, data purification, feature extraction, and fail to define the purpose of the model. Understanding the business model can help identify challenges that can be solved using analytics and scientific data. In this article, we go through the **Uber** Model, which provides a framework for end-to-end prediction analytics of **Uber** data prediction sources.

Uber's Machine Learning Model



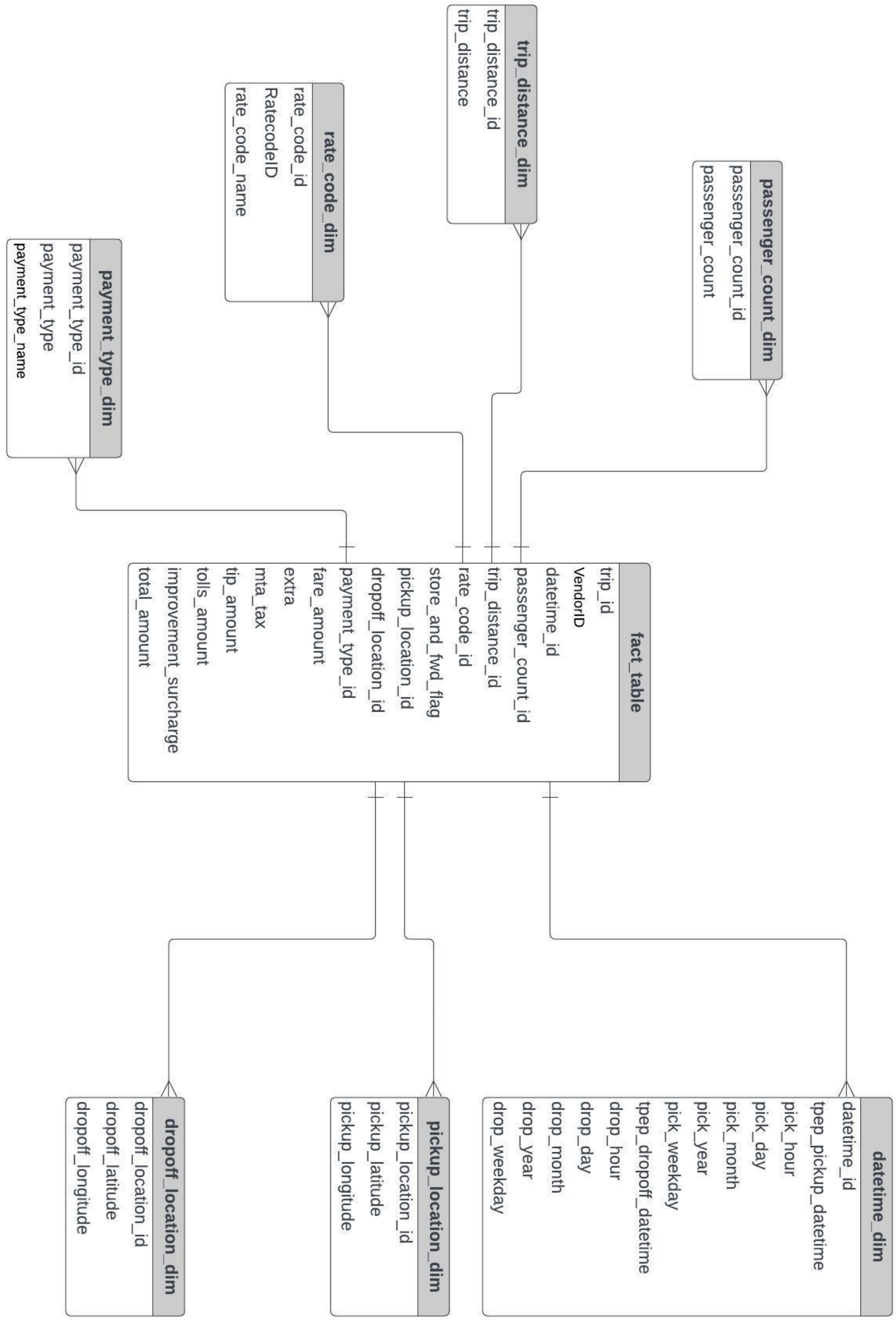
Workflow of ML learning project. Defining a problem, creating a solution, producing a solution, and measuring the impact of the solution are fundamental workflows. Barriers to workflow represent the many repetitions of the feedback collection required to create a solution and complete a project.

Michelangelo's "zero-to-one speed" or "value-to-one speed" is crucial to how ML spreads to Uber. In new applications, we focus on reducing barriers to entry by streamlining the workflow of people with different skills and having a consistent flow to achieve a basic model and work with good diversity.

A few principles have proven to be very helpful in empowering teams to develop faster:

- Solve data problems so that data scientists are not needed.
- Dealing with data access, integration, feature management, and plumbing can be time-consuming for a data expert. Michelangelo's feature shop and feature pipes are essential in solving a pile of data experts in the head.
- Change or provide powerful tools to speed up the normal flow.
- Make the delivery process faster and more magical.
- Michelangelo hides the details of deploying and monitoring models and data pipelines in production after a single click on the UI.
- Let the user use their favorite tools with small cruft – "Go to the customer".
- Michelangelo allows for the development of collaborations in Python, textbooks, CLIs, and includes production UI to manage production programs and records.
- Enable interaction and reuse.
- Also, Michelangelo's feature shop is important in enabling teams to reuse key predictive features that have already been identified and developed by other teams.
- Guide the user through organized workflows.

Data Flow(Daidram)

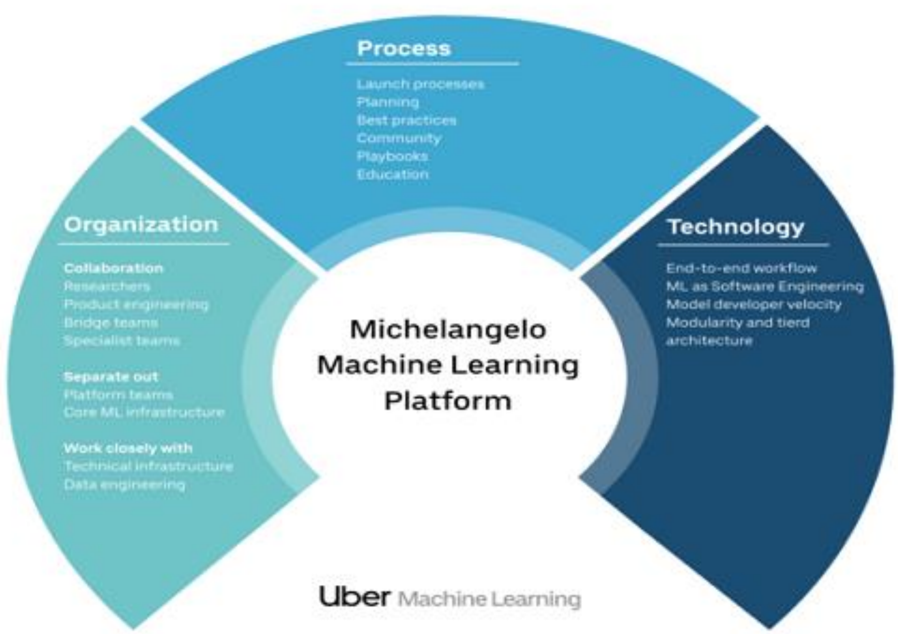


payment_type_dim
payment_type_id
payment_type
payment_type_name
rate_code_dim
rate_code_id
RatecodeID
rate_code_name
trip_distance_dim
trip_distance_id
trip_distance
passenger_count_dim
passenger_count_id
passenger_count
fact_table
trip_id
VendorID
datetime_id
passenger_count_id
trip_distance_id
rate_code_id
store_and_fwd_flag
pickup_location_id
dropoff_location_id
payment_type_id
fare_amount
extra
mta_tax
tip_amount
tolls_amount
improvement_surcharge
total_amount
dropoff_location_dim
dropoff_location_id
dropoff_latitude
dropoff_longitude
pickup_location_dim
pickup_location_id
pickup_latitude
pickup_longitude
datetime_dim
datetime_id
tpep_pickup_datetime
pick_hour
pick_day
pick_month
pick_year
pick_weekday
tpep_dropoff_datetime
drop_hour
drop_day
drop_month
drop_year
drop_weekday

Scaling Machine Learning at Uber

Data scientists, our use of tools makes it easier to create and produce on the side of building and shipping ML systems, enabling them to manage their work ultimately. For developers, Uber's ML tool simplifies data science (engineering aspect, modeling, testing, etc.) after these programs, making it easier for them to train high-quality models without the need for a data scientist. Finally, for the most experienced engineering teams forming special ML programs, we provide Michelangelo's ML infrastructure components for customization and workflow.

Successfully measuring ML at a company like Uber requires much more than just the right technology – rather than the critical considerations of process planning and processing as well. In this section, we look at critical aspects of success across all three pillars: structure, process, and technology.



Organization

The very diverse needs of ML problems and limited resources make organizational formation very important – and challenging – in machine learning. While some Uber ML projects are run by teams of many ML engineers and data scientists, others are run by teams with little technical knowledge. Similarly, some problems can be solved with novices with widely available out-of-the-box algorithms, while other problems require expert investigation of advanced techniques (and they often do not have known solutions).

Process

As Uber ML's operations mature, many processes have proven to be useful in the production and efficiency of our teams. Sharing best ML practices (e.g., data editing methods, testing, and post-management) and implementing well-structured processes (e.g., implementing reviews) are important ways to guide teams and avoid duplicating others 'mistakes. Internally focused community-building efforts and transparent planning processes involve and align ML groups under common goals.

Technology

There is a lot of detail to find the right side of the technology for any ML system. At Uber, we have identified the following high-end areas as the most important:

End-to-end workflow: ML is more than just training models; you need support for all ML workflow: manage data, train models, check models, deploy models and make predictions, and look for guesses.

ML as software engineering: We found it important to draw analogies between ML development and software development, and then use patterns from software development tools and methods to get back to our ML functionality.

Model Developer Speed: The development of a machine learning model is a very repetitive process – new methods and advanced models come from many experiments. Because of this, the speed of the model engineers is very important.

Modularity and tiered architecture: Providing end-to-end workflow is important in managing the most common causes of ML use, but to deal with rare and very special cases, it is important to have the first things that can be integrated in a directed way.

THAKUR COLLEGE OF SCIENCE AND COMMERCE

Department of Computer Science

2023-2024

Students Name: YOGESH SHRINATH PAL

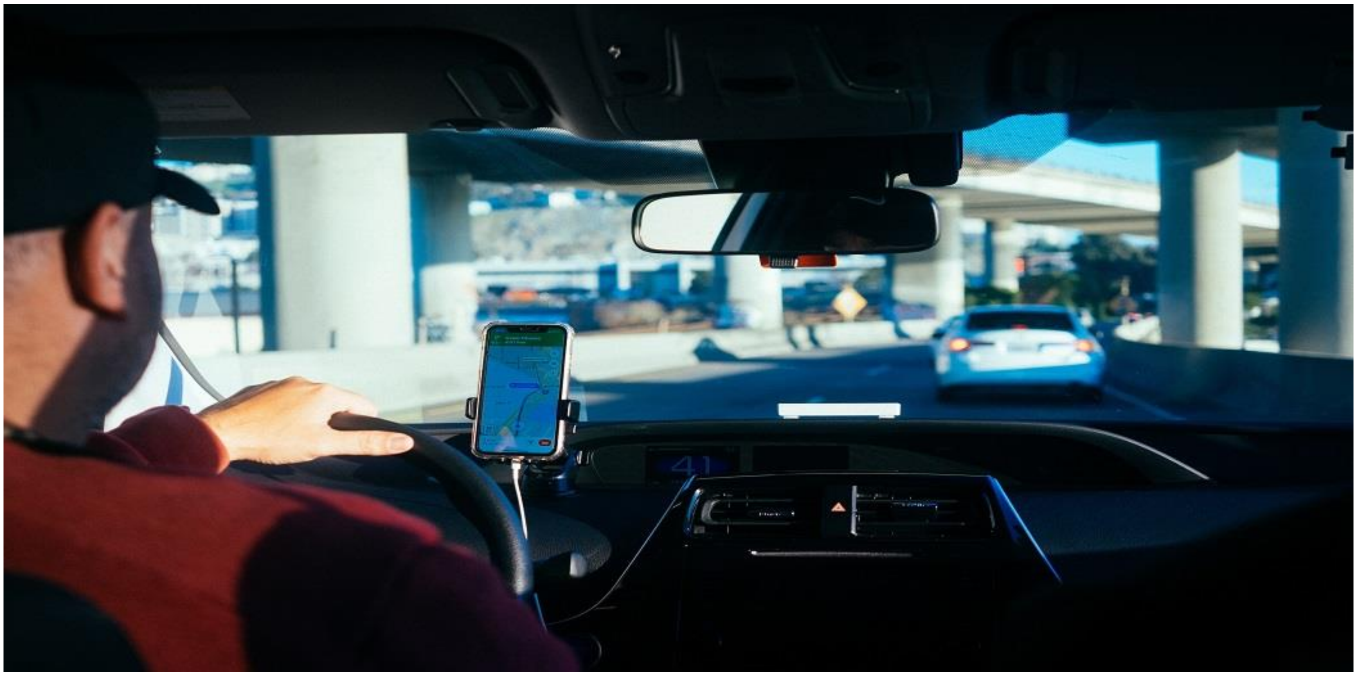
Project Name: Uber Data Analysis

College Name: Thakur College of Science and Commerce

PHASES	EXPECTED DATE OF COMPLETION	ACTUAL DATE OF COMPLETION	SIGNATURE
Preliminary Investigation			
System Analysis			
System Designing			
System Coding			
System Implementation			
Report Submission			

Uber’s Dynamic Pricing Model

Here’s a quick and easy guide to how Uber’s dynamic price model works, so you know why Uber prices are changing and what regular peak hours are the costs of Uber’s rise.



How does Uber Price Model Works?

If you request a ride on Saturday night, you may find that the price is different from the cost of the same trip a few days earlier. That’s because of our dynamic pricing algorithm, which converts prices according to several variables, such as the time and distance of your route, traffic, and the current need of the driver. In some cases, this may mean a temporary increase in price during very busy times.

Why are Uber rates changing?

As demand increases, Uber uses flexible costs to encourage more drivers to get on the road and help address a number of passenger requests. When we inform you of an increase in Uber fees, we also inform drivers. If you decide to proceed and request your ride, you will receive a warning in the app to make sure you know that ratings have changed.

Price range

When more drivers enter the road and board requests have been taken, the need will be more manageable and the fare should return to normal.

Uber Top Hours

If you're a regular passenger, you're probably already familiar with Uber's peak times, when rising demand and prices are very likely. These include:

- Friday and Saturday nights
- After-work hour fast
- Major events and celebrations
- Strong prices help us to ensure that there are always enough drivers to handle all our travel requests, so you can ride faster and easier – whether you and your friends are taking this trip or staying up to you.

Exploratory Data Analysis and Predictive Modelling on Uber Pickups

The day-to-day effect of rising prices varies depending on the location and pair of the Origin-Destination (OD pair) of the Uber trip: at accommodations/train stations, daylight hours can affect the rising price; for theaters, the hour of the important or famous play will affect the prices; finally, attractively, the price hike may be affected by certain holidays, which will increase the number of guests and perhaps even the prices; Finally, at airports, the price of escalation will be affected by the number of periodic flights and certain weather conditions, which could prevent more flights to land and land.

The weather is likely to have a significant impact on the rise in prices of Uber fares and airports as a starting point, as departure and accommodation of aircraft depending on the weather at that time. Different weather conditions will certainly affect the price increase in different ways and at different levels: we assume that weather conditions such as clouds or clearness do not have the same effect on inflation prices as weather conditions such as snow or fog. As for the day of the week, one thing that really matters is to distinguish between weekends and weekends: people often engage in different activities, go to different places, and maintain a different way of traveling during weekends and weekends. With forecasting in mind, we can now, by analyzing marine information capacity and developing graphs and formulas, investigate whether we have an impact and whether that increases their impact on Uber passenger fares in New York City.

You can download the dataset from Kaggle or you can perform it on your own Uber dataset. I have taken the dataset from Felipe Alves Santos Github.

Business Problem

Before you start managing and analyzing data, the first thing you should do is think about the PURPOSE. What it means is that you have to think about the reasons why you are going to do any analysis. If you are unsure about this, just start by asking questions about your story such as **Where? What? How? Who? Which?**

Data visualization is certainly one of the most important stages in Data Science processes. While simple, it can be a powerful tool for prioritizing data and business context, as well as determining the right treatment before creating machine learning models.

System Coding

```
from typing import Any

import streamlit as st
import pandas as pd
import preprocessor
import matplotlib.pyplot as plt
import seaborn as sns

st.sidebar.title("Uber Data Analyzer")
uploaded_file = st.sidebar.file_uploader("Choose a file")
if uploaded_file is not None:

    data = pd.read_csv(uploaded_file)

    fact_table = preprocessor.preprocessor(data)
    st.dataframe(fact_table)

    vendor_id: list | Any = fact_table['VendorID'].unique().tolist()
    vendor_id.sort()
    vendor_id.insert(0, "Overall")

    selected_vendor_id = st.sidebar.selectbox("Show Analysis", vendor_id)



    if st.sidebar.button("Start Analysis"):


        if selected_vendor_id == "Overall":

            total_amount_sum = fact_table['total_amount'].sum()
            avg_fare_amt = fact_table['fare_amount'].mean()
            avg_trip_distance = fact_table['trip_distance_id'].mean()


            st.subheader("Analysis Results (Overall)")
            st.text(f"Total of Total Amount: {total_amount_sum}")
            st.text(f"Average Fare Amount: {avg_fare_amt}")
            st.text(f"Average Trip Distance: {avg_trip_distance}")


            st.subheader("Distribution of Vendor IDs (Overall)")
            plt.figure(figsize=(10, 6))
            sns.countplot(x='VendorID', data=fact_table)
            plt.title('Distribution of Vendor IDs')
            st.pyplot(plt.gcf())


            st.subheader("Passenger Count Distribution (Overall)")
            plt.figure(figsize=(10, 6))
            sns.countplot(x='passenger_count_id', data=fact_table,
palette="viridis", order=[1, 2, 3, 4, 5])
            plt.title('Passenger Count Distribution')
            st.pyplot(plt.gcf())
```

```
st.subheader("Distribution of Rate Code IDs (Overall)")
            plt.figure(figsize=(10, 6))
            sns.countplot(x='rate_code_id', data=fact_table, palette="viridis",
order=[1, 2, 3, 4, 5])
            plt.title('Distribution of Rate Code IDs')
            st.pyplot(plt.gcf())


            st.subheader("Distribution of Payment Types (Overall)")
            plt.figure(figsize=(10, 6))
            sns.countplot(x='payment_type_id',
data=fact_table.replace({'payment_type_id': {0: 'Credit card', 1: 'Cash', 2: 'No
charge', 3: 'Dispute'}}), palette="viridis")
            plt.title('Distribution of Payment Types')
            st.pyplot(plt.gcf())

            data.rename(columns={'pickup_latitude': 'latitude', 'pickup_longitude':
'longitude'},
                        inplace=True)


            st.subheader("Pickup Locations on Map (Overall)")
            st.map(data[['latitude', 'longitude']])

        else:

            subset_data = fact_table[fact_table['VendorID'] == selected_vendor_id]

            total_amount_sum = subset_data['total_amount'].sum()
            avg_fare_amt = subset_data['fare_amount'].mean()
            avg_trip_distance = subset_data['trip_distance_id'].mean()


            st.subheader(f"Analysis Results for VendorID: {selected_vendor_id}")
            st.text(f"Total of Total Amount: {total_amount_sum}")
            st.text(f"Average Fare Amount: {avg_fare_amt}")
            st.text(f"Average Trip Distance: {avg_trip_distance}")


            st.subheader(f"Distribution of Vendor ID {selected_vendor_id}")
            plt.figure(figsize=(10, 6))
            sns.countplot(x='VendorID', data=subset_data)
            plt.title(f'Distribution of Vendor ID {selected_vendor_id}')
            st.pyplot(plt.gcf())


            st.subheader(f"Passenger Count Distribution for Vendor ID
{selected_vendor_id}")
            plt.figure(figsize=(10, 6))
            sns.countplot(x='passenger_count_id', data=subset_data,
palette="viridis", order=[1, 2, 3, 4, 5])
            plt.title(f'Passenger Count Distribution for Vendor ID
{selected_vendor_id}')
            st.pyplot(plt.gcf())


            st.subheader(f"Distribution of Rate Code IDs for Vendor ID
{selected_vendor_id}")
```

```
plt.figure(figsize=(10, 6))
            sns.countplot(x='rate_code_id', data=subset_data, palette="viridis",
order=[1, 2, 3, 4, 5])
            plt.title(f'Distribution of Rate Code IDs for Vendor ID
{selected_vendor_id}')
            st.pyplot(plt.gcf())


            st.subheader(f"Distribution of Payment Types for Vendor ID
{selected_vendor_id}")
            plt.figure(figsize=(10, 6))
            sns.countplot(x='payment_type_id',
data=subset_data.replace({'payment_type_id': {0: 'Credit card', 1: 'Cash', 2: 'No
charge', 3: 'Dispute'}}), palette="viridis")
            plt.title(f'Distribution of Payment Types for Vendor ID
{selected_vendor_id}')
            st.pyplot(plt.gcf())

            data.rename(columns={'pickup_latitude': 'latitude', 'pickup_longitude':
'longitude'},
                                       inplace=True)


            st.subheader("Pickup Locations on Map (Overall)")
            st.map(data[['latitude', 'longitude']])
```

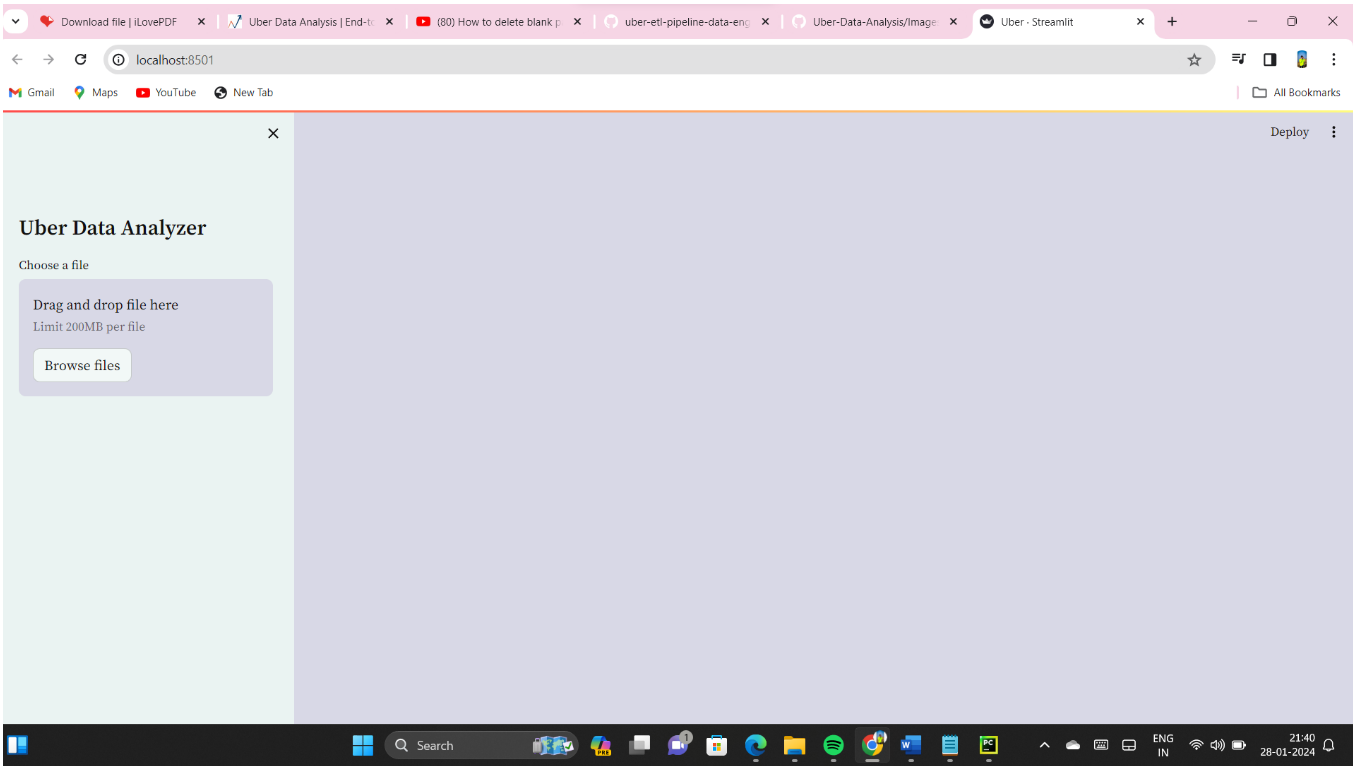
Data Analysis code:-

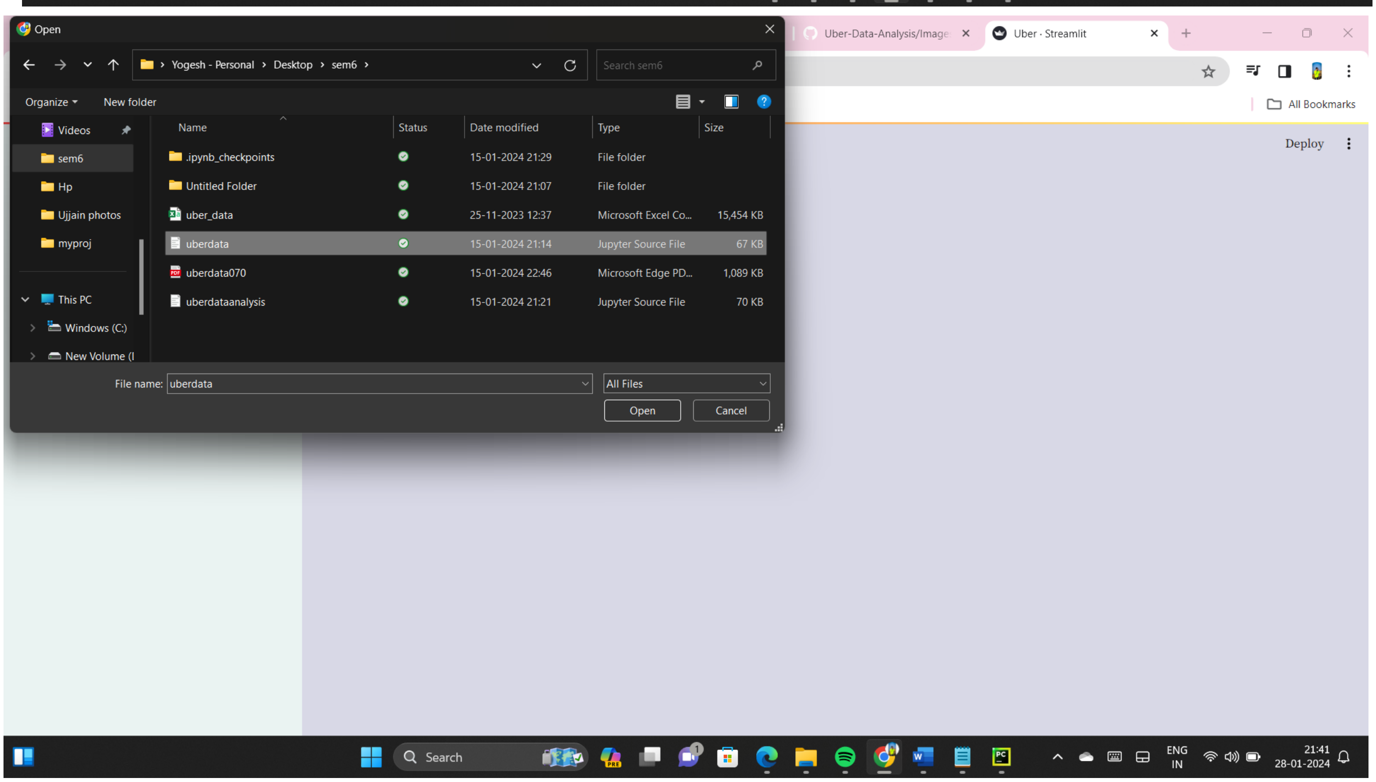
```
import pandas as pd
def preprocessor(data):
    df = pd.DataFrame(data)


    df['tpep_pickup_datetime'] = pd.to_datetime(df['tpep_pickup_datetime'])
    df['tpep_dropoff_datetime'] = pd.to_datetime(df['tpep_dropoff_datetime'])

    datetime_dim = df[['tpep_pickup_datetime',
'tpep_dropoff_datetime']].drop_duplicates().reset_index(drop=True)
    datetime_dim['pick_hour'] = datetime_dim['tpep_pickup_datetime'].dt.hour
    datetime_dim['pick_day'] = datetime_dim['tpep_pickup_datetime'].dt.day
    datetime_dim['pick_month'] = datetime_dim['tpep_pickup_datetime'].dt.month
    datetime_dim['pick_year'] = datetime_dim['tpep_pickup_datetime'].dt.year
    datetime_dim['pick_weekday'] = datetime_dim['tpep_pickup_datetime'].dt.weekday

    datetime_dim['drop_hour'] = datetime_dim['tpep_dropoff_datetime'].dt.hour
    datetime_dim['drop_day'] = datetime_dim['tpep_dropoff_datetime'].dt.day
    datetime_dim['drop_month'] = datetime_dim['tpep_dropoff_datetime'].dt.month
    datetime_dim['drop_year'] = datetime_dim['tpep_dropoff_datetime'].dt.year
    datetime_dim['drop_weekday'] = datetime_dim['tpep_dropoff_datetime'].dt.weekday
```

```
datetime_dim['datetime_id'] = datetime_dim.index

    datetime_dim = datetime_dim[
        ['datetime_id', 'tpep_pickup_datetime', 'pick_hour', 'pick_day',
'pick_month', 'pick_year', 'pick_weekday',
         'tpep_dropoff_datetime', 'drop_hour', 'drop_day', 'drop_month',
'drop_year', 'drop_weekday']]

    passenger_count_dim =
df[['passenger_count']].drop_duplicates().reset_index(drop=True)
    passenger_count_dim['passenger_count_id'] = passenger_count_dim.index
    passenger_count_dim = passenger_count_dim[['passenger_count_id',
'passenger_count']]

    trip_distance_dim =
df[['trip_distance']].drop_duplicates().reset_index(drop=True)
    trip_distance_dim['trip_distance_id'] = trip_distance_dim.index
    trip_distance_dim = trip_distance_dim[['trip_distance_id', 'trip_distance']]

    rate_code_type = {1: "Standard rate", 2: "JFK", 3: "Newark", 4: "Nassau or
Westchester", 5: "Negotiated fare",
                      6: "Group ride"}
    rate_code_dim = df[['RatecodeID']].drop_duplicates().reset_index(drop=True)
    rate_code_dim['rate_code_id'] = rate_code_dim.index
    rate_code_dim['rate_code_name'] =
rate_code_dim['RatecodeID'].map(rate_code_type)
    rate_code_dim = rate_code_dim[['rate_code_id', 'RatecodeID', 'rate_code_name']]

    payment_type_name = {1: "Credit card", 2: "Cash", 3: "No charge", 4: "Dispute",
5: "Unknown", 6: "Voided trip"}
    payment_type_dim = df[['payment_type']].drop_duplicates().reset_index(drop=True)
    payment_type_dim['payment_type_id'] = payment_type_dim.index
    payment_type_dim['payment_type_name'] =
payment_type_dim['payment_type'].map(payment_type_name)
    payment_type_dim = payment_type_dim[['payment_type_id', 'payment_type',
'payment_type_name']]


    pickup_location_dim = df[['pickup_longitude',
'pickup_latitude']].drop_duplicates().reset_index(drop=True)
    pickup_location_dim['pickup_location_id'] = pickup_location_dim.index
    pickup_location_dim = pickup_location_dim[['pickup_location_id',
'pickup_latitude', 'pickup_longitude']]

    dropoff_location_dim = df[['dropoff_longitude',
'dropoff_latitude']].drop_duplicates().reset_index(drop=True)
    dropoff_location_dim['dropoff_location_id'] = dropoff_location_dim.index
    dropoff_location_dim = dropoff_location_dim[['dropoff_location_id',
'dropoff_latitude', 'dropoff_longitude']]
```

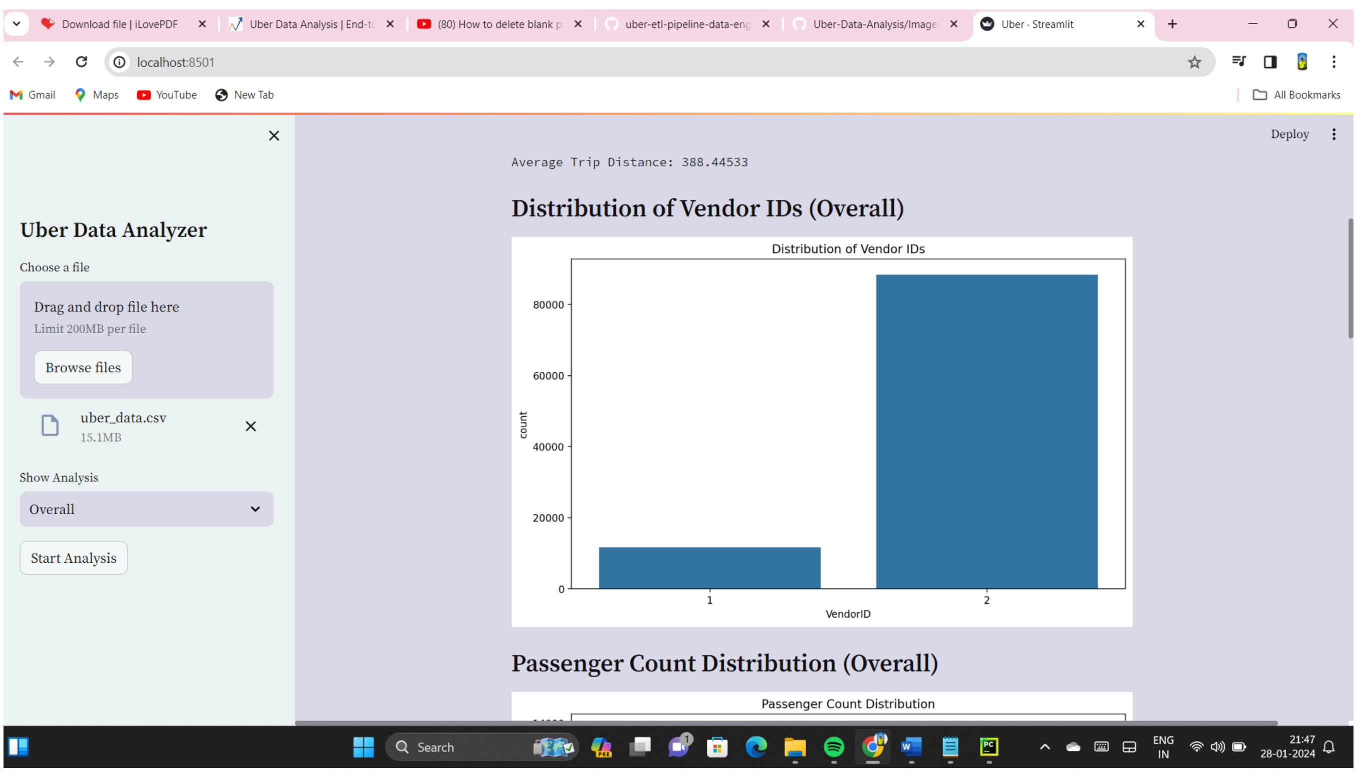
```
fact_table = df.merge(passenger_count_dim, on='passenger_count') \
        .merge(trip_distance_dim, on='trip_distance') \
        .merge(rate_code_dim, on='RatecodeID') \
        .merge(pickup_location_dim, on=['pickup_longitude', 'pickup_latitude']) \
        .merge(dropoff_location_dim, on=['dropoff_longitude', 'dropoff_latitude']) \
        .merge(datetime_dim, on=['tpep_pickup_datetime', 'tpep_dropoff_datetime']) \
        .merge(payment_type_dim, on='payment_type') \
        [['VendorID', 'datetime_id', 'passenger_count_id',
          'trip_distance_id', 'rate_code_id', 'store_and_fwd_flag',
'pickup_location_id', 'dropoff_location_id',
          'payment_type_id', 'fare_amount', 'extra', 'mta_tax', 'tip_amount',
'tolls_amount',
          'improvement_surcharge', 'total_amount']]

    return fact_table
```

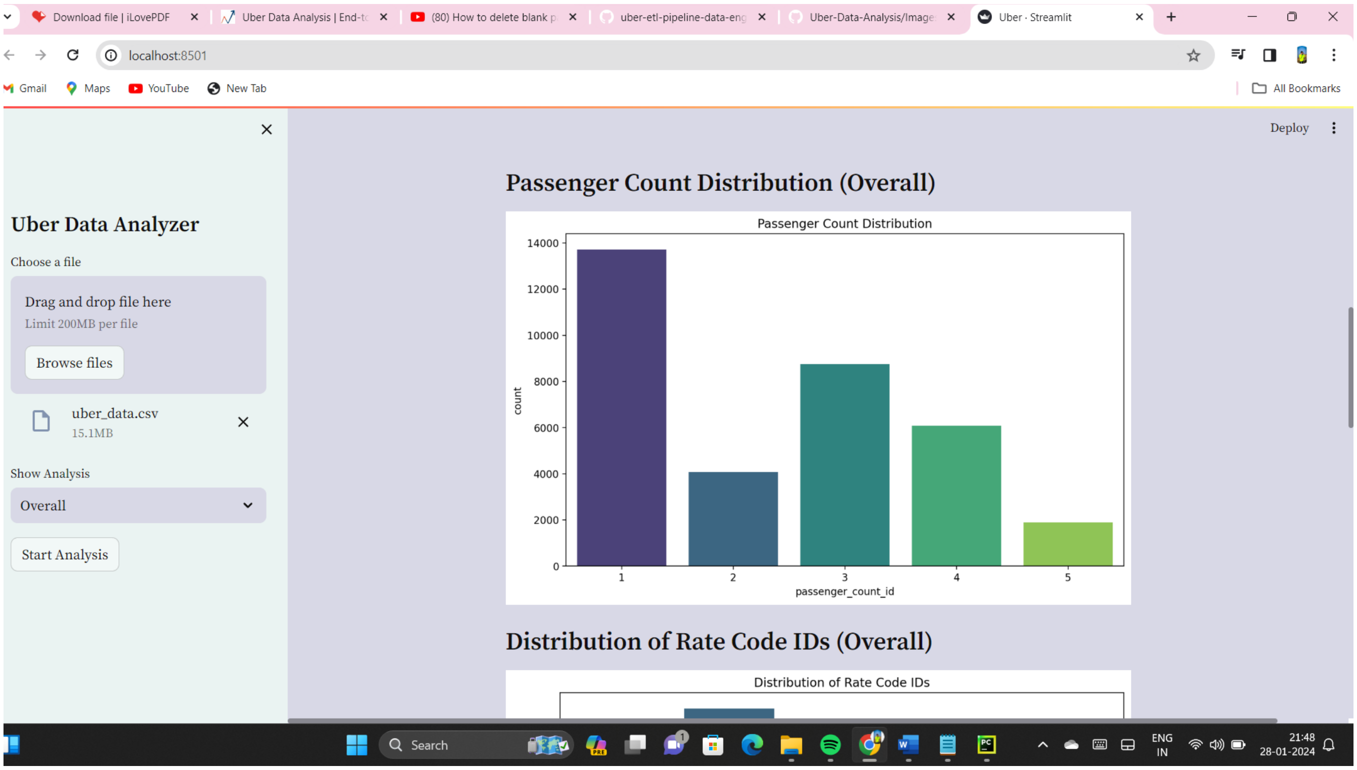
Outputs:-

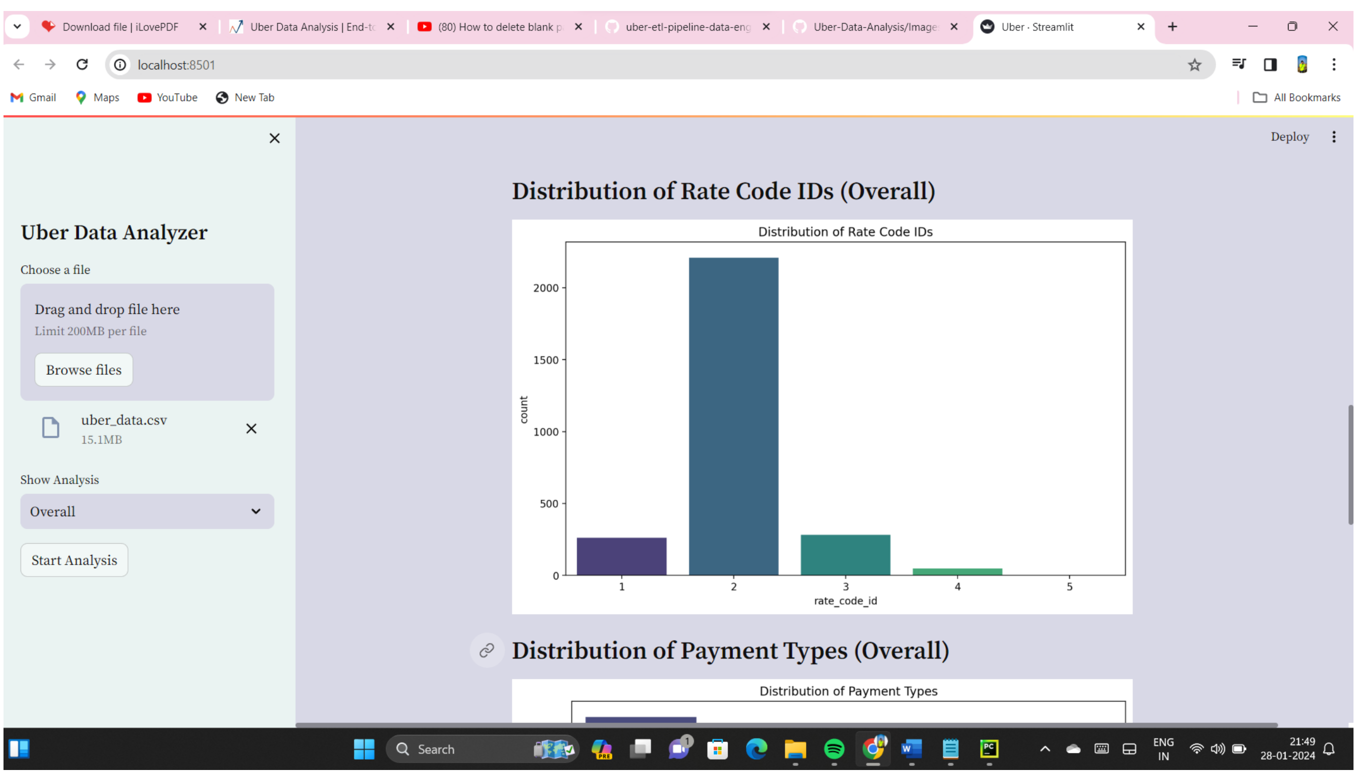

localhost:8501
Deploy
Uber Data Analyzer
Choose a file
Drag and drop file here
Limit 200MB per file
Browse files

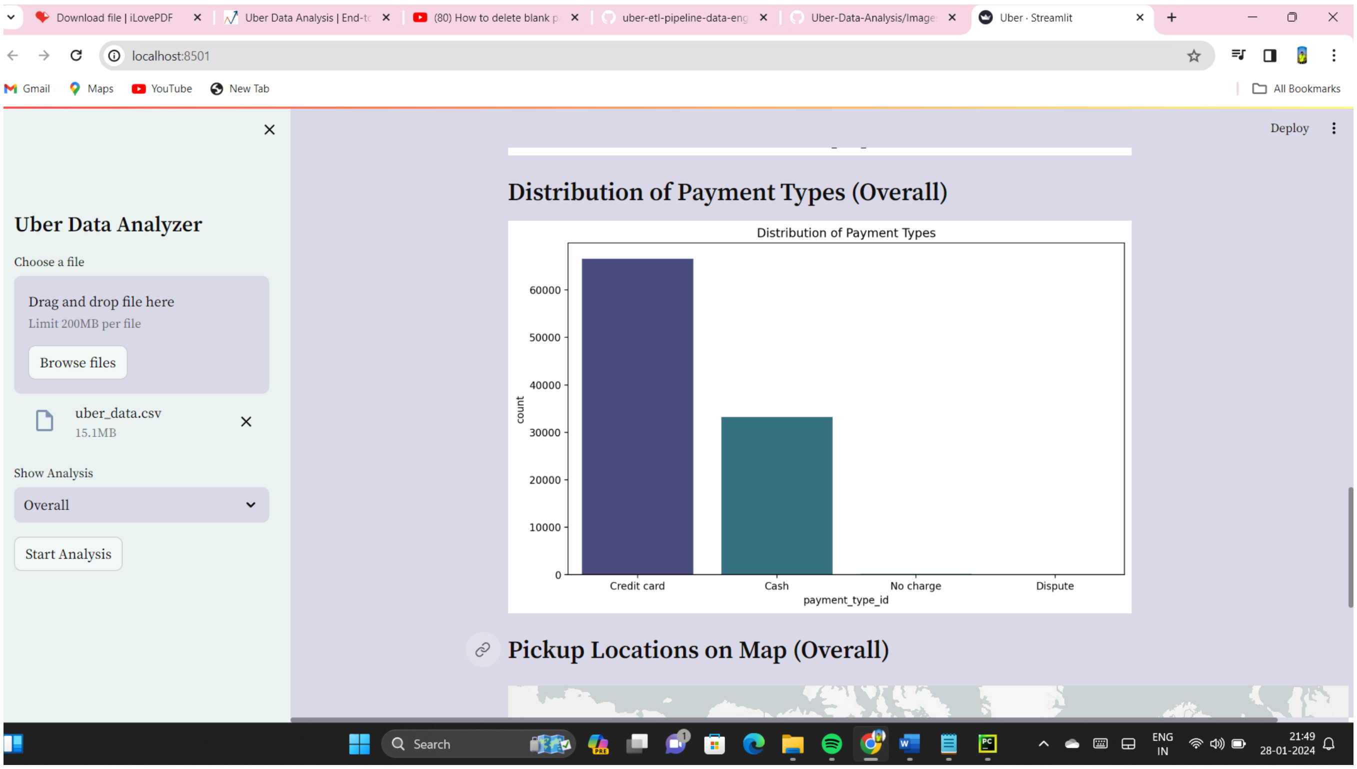

Open
Yogesh - Personal › Desktop › sem6
Organize
New folder
Name
Status
Date modified
Type
Size
.ipynb_checkpoints 15-01-2024 21:29 File folder
Untitled Folder 15-01-2024 21:07 File folder
uber_data 25-11-2023 12:37 Microsoft Excel Co... 15,454 KB
uberdata 15-01-2024 21:14 Jupyter Source File 67 KB
uberdata070 15-01-2024 22:46 Microsoft Edge PD... 1,089 KB
uberdataanalysis 15-01-2024 21:21 Jupyter Source File 70 KB
File name: uberdata
All Files
Open
Cancel
Deploy

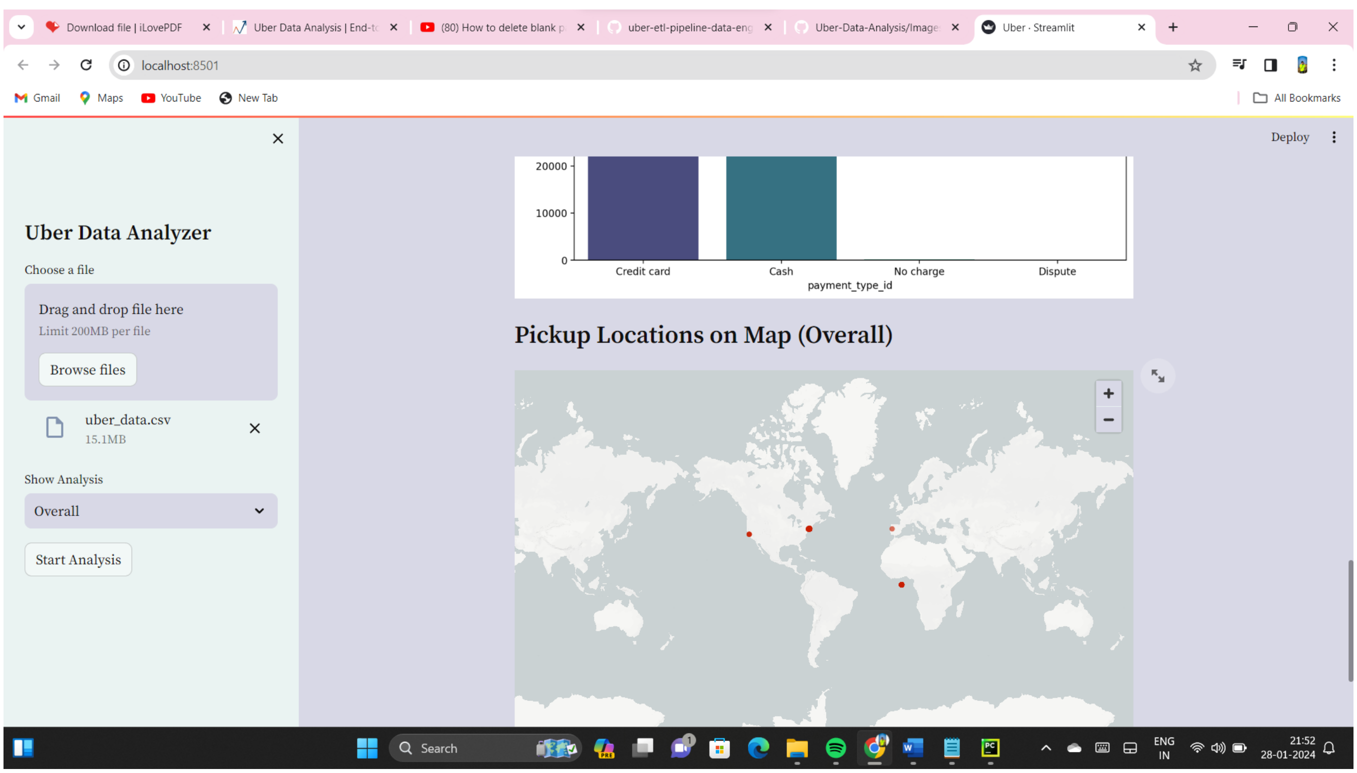

localhost:8501
Deploy
Uber Data Analyzer
Choose a file
Drag and drop file here
Limit 200MB per file
Browse files
uber_data.csv
15.1MB
Show Analysis
Overall
Start Analysis
Analysis Results (Overall)
Total of Total Amount: 1639072.09
Average Fare Amount: 13.252608400000003
Average Trip Distance: 388.44533
Distribution of Vendor IDs (Overall)

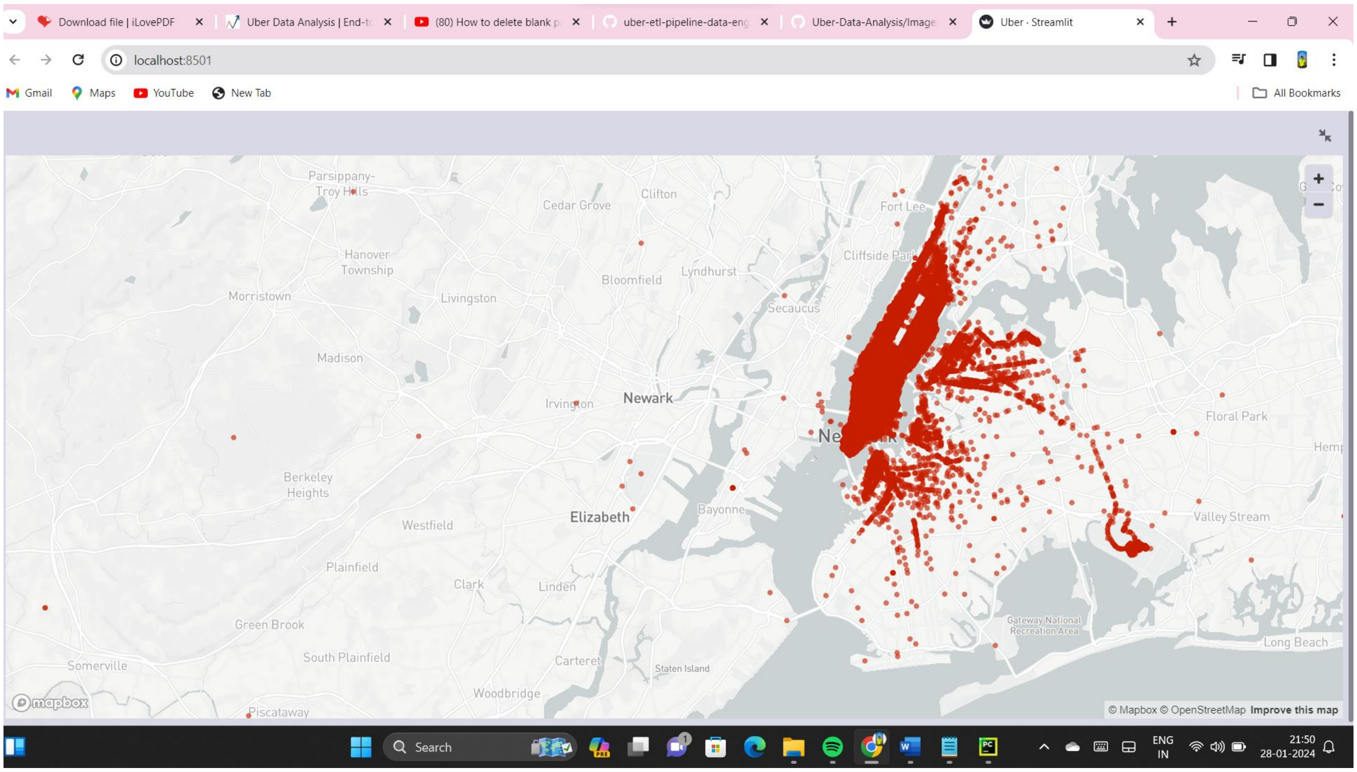

localhost:8501
Deploy
Uber Data Analyzer
Choose a file
Drag and drop file here
Limit 200MB per file
Browse files
uber_data.csv
15.1MB
Show Analysis
Overall
Start Analysis
Average Trip Distance: 388.44533
Distribution of Vendor IDs (Overall)
Distribution of Vendor IDs
count
VendorID
Passenger Count Distribution (Overall)
Passenger Count Distribution


localhost:8501
Deploy
Uber Data Analyzer
Choose a file
Drag and drop file here
Limit 200MB per file
Browse files
uber_data.csv
15.1MB
Show Analysis
Overall
Start Analysis
Passenger Count Distribution (Overall)
Passenger Count Distribution
count
passenger_count_id
Distribution of Rate Code IDs (Overall)
Distribution of Rate Code IDs

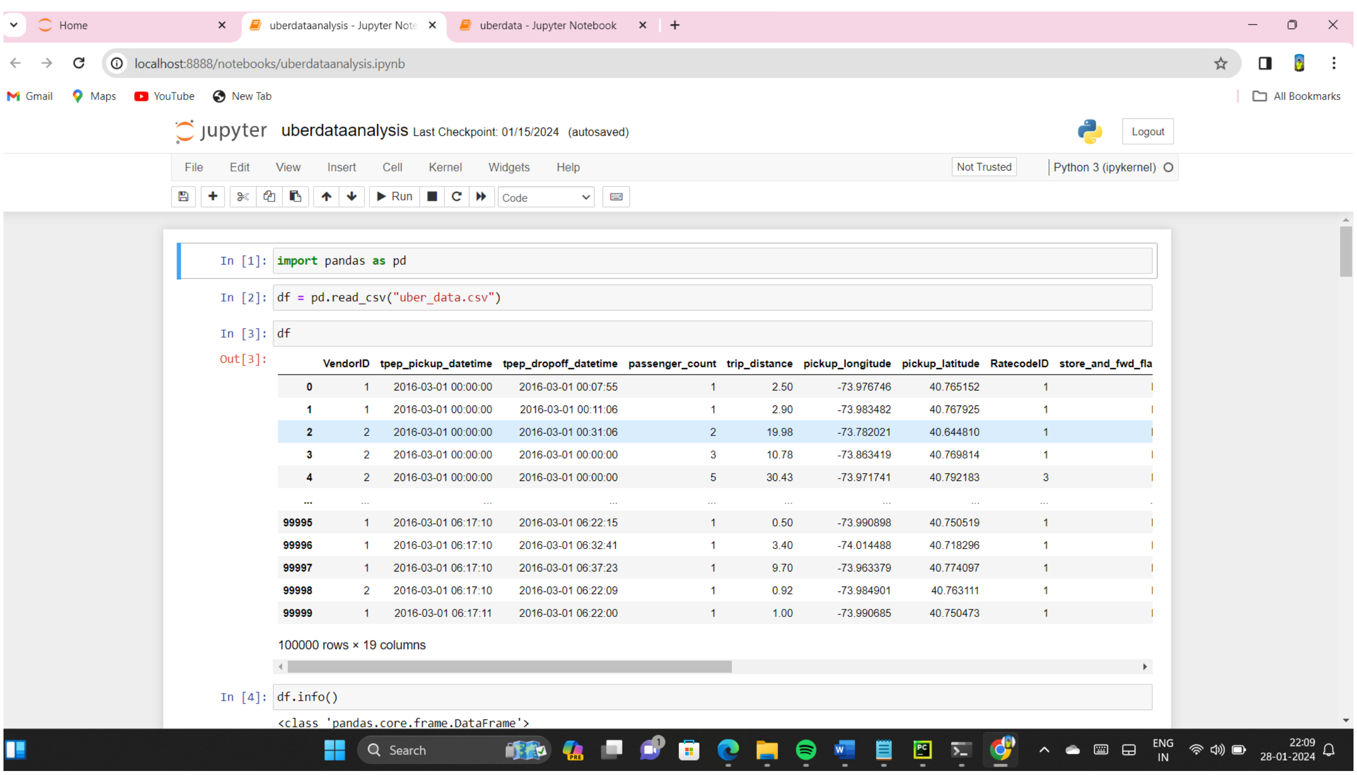

localhost:8501
Deploy
Uber Data Analyzer
Choose a file
Drag and drop file here
Limit 200MB per file
Browse files
uber_data.csv
15.1MB
Show Analysis
Overall
Start Analysis
Distribution of Rate Code IDs (Overall)
Distribution of Rate Code IDs
count
rate_code_id
Distribution of Payment Types (Overall)
Distribution of Payment Types


Uber Data Analyzer
Choose a file
Drag and drop file here
Limit 200MB per file
Browse files
uber_data.csv
15.1MB
Show Analysis
Overall
Start Analysis
Deploy
Distribution of Payment Types (Overall)
Distribution of Payment Types
count
payment_type_id
Credit card
Cash
No charge
Dispute
Pickup Locations on Map (Overall)


Uber Data Analyzer
Choose a file
Drag and drop file here
Limit 200MB per file
Browse files
uber_data.csv
15.1MB
Show Analysis
Overall
Start Analysis
Deploy
Credit card
Cash
No charge
Dispute
payment_type_id
Pickup Locations on Map (Overall)



Data Analysis in jupyter notebook



	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	pickup_longitude	pickup_latitude	RatecodeID	store_and_fwd_fla
0	1	2016-03-01 00:00:00	2016-03-01 00:07:55	1	2.50	-73.976746	40.765152	1	
1	1	2016-03-01 00:00:00	2016-03-01 00:11:06	1	2.90	-73.983482	40.767925	1	
2	2	2016-03-01 00:00:00	2016-03-01 00:31:06	2	19.98	-73.782021	40.644810	1	
3	2	2016-03-01 00:00:00	2016-03-01 00:00:00	3	10.78	-73.863419	40.769814	1	
4	2	2016-03-01 00:00:00	2016-03-01 00:00:00	5	30.43	-73.971741	40.792183	3	
...	...	...	...	...	...	...	...	...	...
99995	1	2016-03-01 06:17:10	2016-03-01 06:22:15	1	0.50	-73.990898	40.750519	1	
99996	1	2016-03-01 06:17:10	2016-03-01 06:32:41	1	3.40	-74.014488	40.718296	1	
99997	1	2016-03-01 06:17:10	2016-03-01 06:37:23	1	9.70	-73.963379	40.774097	1	
99998	2	2016-03-01 06:17:10	2016-03-01 06:22:09	1	0.92	-73.984901	40.763111	1	
99999	1	2016-03-01 06:17:11	2016-03-01 06:22:00	1	1.00	-73.990685	40.750473	1	

Home | uberdataanalysis - Jupyter Note | uberdata - Jupyter Notebook

localhost:8888/notebooks/uberdataanalysis.ipynb

Gmail | Maps | YouTube | New Tab | All Bookmarks

jupyter uberdataanalysis Last Checkpoint: 01/15/2024 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

```
datetime_dim['pick_hour'] = datetime_dim['tpep_pickup_datetime'].dt.hour
datetime_dim['pick_day'] = datetime_dim['tpep_pickup_datetime'].dt.day
datetime_dim['pick_month'] = datetime_dim['tpep_pickup_datetime'].dt.month
datetime_dim['pick_year'] = datetime_dim['tpep_pickup_datetime'].dt.year
datetime_dim['pick_weekday'] = datetime_dim['tpep_pickup_datetime'].dt.weekday

datetime_dim['drop_hour'] = datetime_dim['tpep_dropoff_datetime'].dt.hour
datetime_dim['drop_day'] = datetime_dim['tpep_dropoff_datetime'].dt.day
datetime_dim['drop_month'] = datetime_dim['tpep_dropoff_datetime'].dt.month
datetime_dim['drop_year'] = datetime_dim['tpep_dropoff_datetime'].dt.year
datetime_dim['drop_weekday'] = datetime_dim['tpep_dropoff_datetime'].dt.weekday
```

In [8]:
```
datetime_dim
```

Out[8]:

	tpep_pickup_datetime	tpep_dropoff_datetime	pick_hour	pick_day	pick_month	pick_year	pick_weekday	drop_hour	drop_day	drop_month	drop_year
0	2016-03-01 00:00:00	2016-03-01 00:07:55	0	1	3	2016	1	0	1	3	2016
1	2016-03-01 00:00:00	2016-03-01 00:11:06	0	1	3	2016	1	0	1	3	2016
2	2016-03-01 00:00:00	2016-03-01 00:31:06	0	1	3	2016	1	0	1	3	2016
3	2016-03-01 00:00:00	2016-03-01 00:00:00	0	1	3	2016	1	0	1	3	2016
4	2016-03-01 00:00:01	2016-03-01 00:16:04	0	1	3	2016	1	0	1	3	2016
...	...	...	...	...	...	...	...	...	...	...	...
99848	2016-03-01 06:17:10	2016-03-01 06:22:15	6	1	3	2016	1	6	1	3	2016
99849	2016-03-01 06:17:10	2016-03-01 06:32:41	6	1	3	2016	1	6	1	3	2016
99850	2016-03-01 06:17:10	2016-03-01 06:37:23	6	1	3	2016	1	6	1	3	2016
99851	2016-03-01 06:17:10	2016-03-01 06:22:09	6	1	3	2016	1	6	1	3	2016
99852	2016-03-01 06:17:11	2016-03-01 06:22:00	6	1	3	2016	1	6	1	3	2016

99853 rows × 12 columns

Search ENG IN 22:11 28-01-2024

Home | uberdataanalysis - Jupyter Note | uberdata - Jupyter Notebook

localhost:8888/notebooks/uberdataanalysis.ipynb

Gmail | Maps | YouTube | New Tab | All Bookmarks

jupyter uberdataanalysis Last Checkpoint: 01/15/2024 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

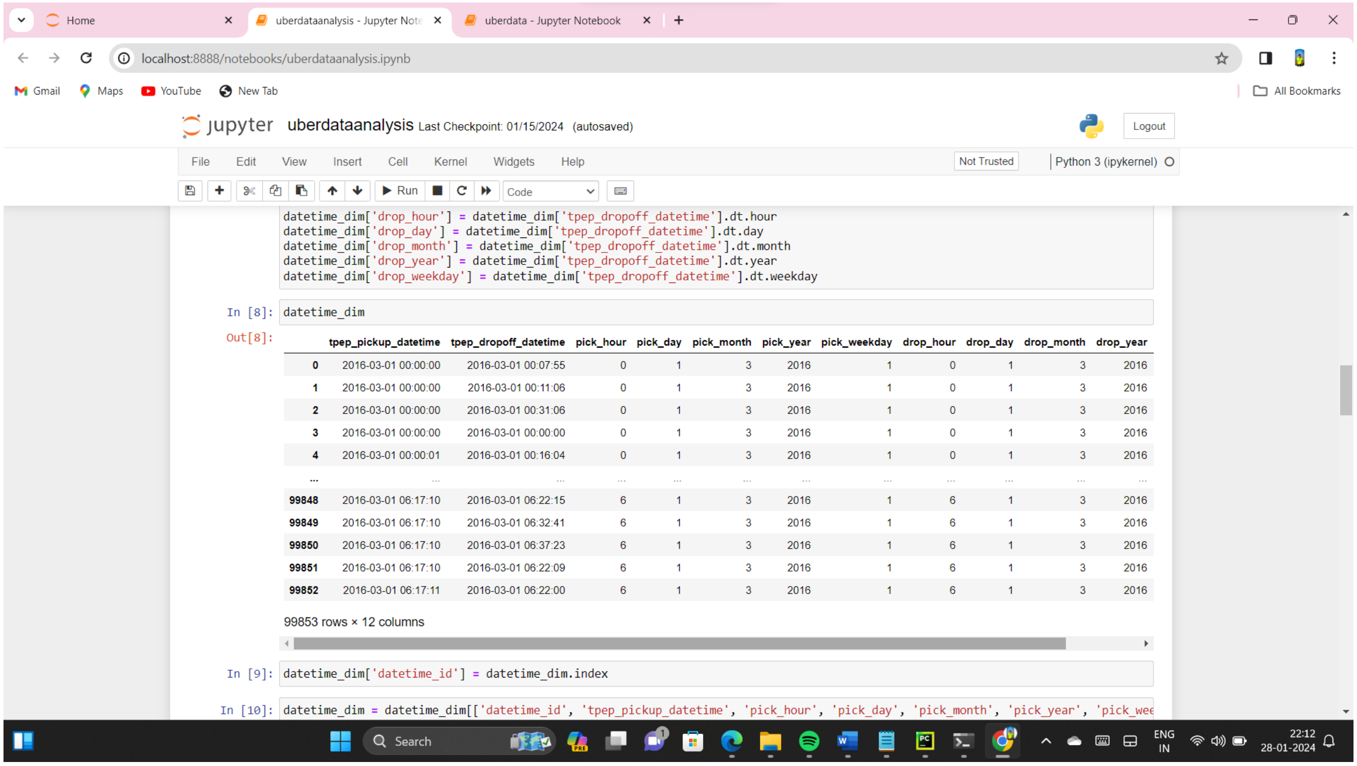
In [5]:
```
df['tpep_pickup_datetime'] = pd.to_datetime(df['tpep_pickup_datetime'])
df['tpep_dropoff_datetime'] = pd.to_datetime(df['tpep_dropoff_datetime'])
```

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 19 columns):
 #   Column                 Non-Null Count   Dtype
---  ------                 --------------   -----
 0   VendorID               100000 non-null  int64
 1   tpep_pickup_datetime   100000 non-null  datetime64[ns]
 2   tpep_dropoff_datetime  100000 non-null  datetime64[ns]
 3   passenger_count        100000 non-null  int64
 4   trip_distance          100000 non-null  float64
 5   pickup_longitude       100000 non-null  float64
 6   pickup_latitude        100000 non-null  float64
 7   RatecodeID             100000 non-null  int64
 8   store_and_fwd_flag     100000 non-null  object
 9   dropoff_longitude      100000 non-null  float64
 10  dropoff_latitude       100000 non-null  float64
 11  payment_type           100000 non-null  int64
 12  fare_amount            100000 non-null  float64
 13  extra                  100000 non-null  float64
 14  mta_tax                100000 non-null  float64
 15  tip_amount             100000 non-null  float64
 16  tolls_amount           100000 non-null  float64
 17  improvement_surcharge  100000 non-null  float64
 18  total_amount           100000 non-null  float64
dtypes: datetime64[ns](2), float64(12), int64(4), object(1)
memory usage: 14.5+ MB
```

In [7]:
```
datetime_dim = df[['tpep_pickup_datetime','tpep_dropoff_datetime']].drop_duplicates().reset_index(drop=True)
datetime_dim['pick_hour'] = datetime_dim['tpep_pickup_datetime'].dt.hour
```

Search ENG IN 22:12 28-01-2024



```
datetime_dim['drop_hour'] = datetime_dim['tpep_dropoff_datetime'].dt.hour
datetime_dim['drop_day'] = datetime_dim['tpep_dropoff_datetime'].dt.day
datetime_dim['drop_month'] = datetime_dim['tpep_dropoff_datetime'].dt.month
datetime_dim['drop_year'] = datetime_dim['tpep_dropoff_datetime'].dt.year
datetime_dim['drop_weekday'] = datetime_dim['tpep_dropoff_datetime'].dt.weekday
```

In [8]:
```
datetime_dim
```

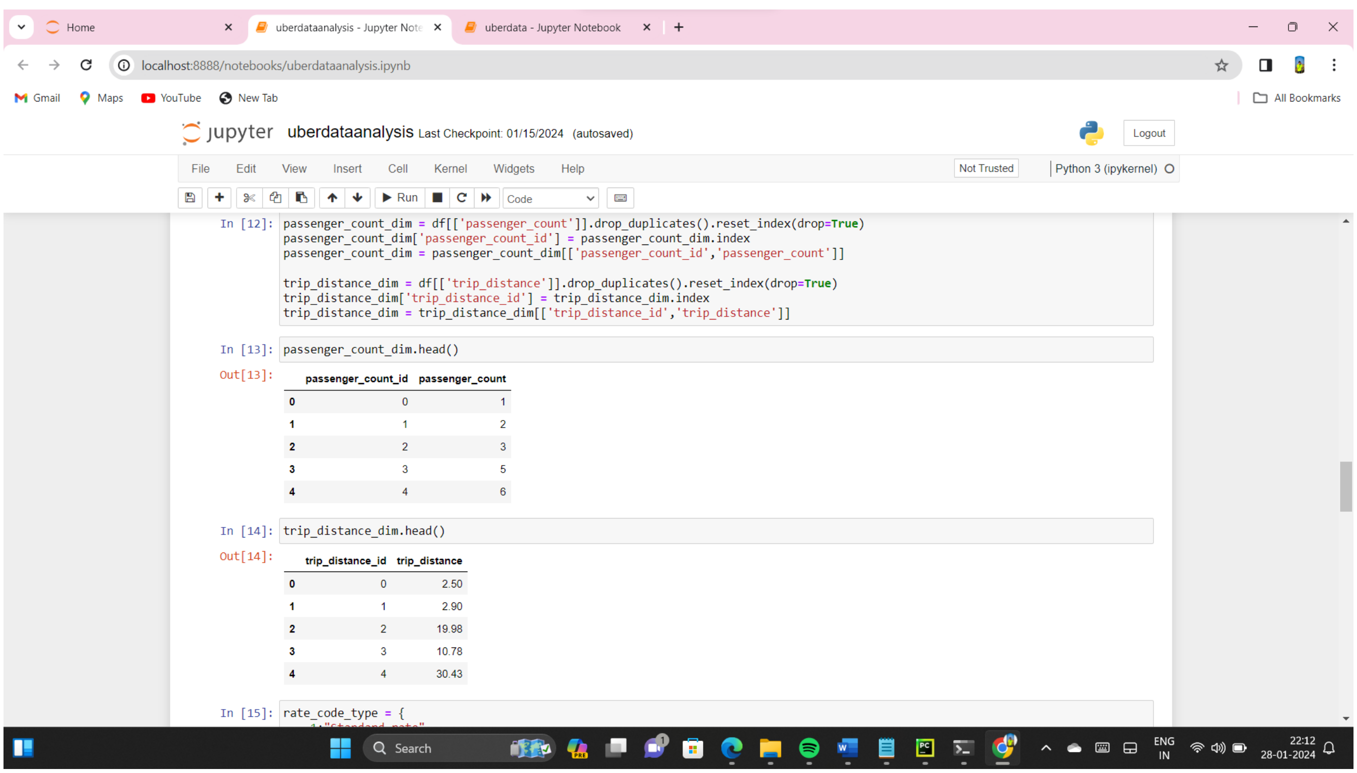
Out[8]:

	tpep_pickup_datetime	tpep_dropoff_datetime	pick_hour	pick_day	pick_month	pick_year	pick_weekday	drop_hour	drop_day	drop_month	drop_year
0	2016-03-01 00:00:00	2016-03-01 00:07:55	0	1	3	2016	1	0	1	3	2016
1	2016-03-01 00:00:00	2016-03-01 00:11:06	0	1	3	2016	1	0	1	3	2016
2	2016-03-01 00:00:00	2016-03-01 00:31:06	0	1	3	2016	1	0	1	3	2016
3	2016-03-01 00:00:00	2016-03-01 00:00:00	0	1	3	2016	1	0	1	3	2016
4	2016-03-01 00:00:01	2016-03-01 00:16:04	0	1	3	2016	1	0	1	3	2016
...	...	...	...	...	...	...	...	...	...	...	...
99848	2016-03-01 06:17:10	2016-03-01 06:22:15	6	1	3	2016	1	6	1	3	2016
99849	2016-03-01 06:17:10	2016-03-01 06:32:41	6	1	3	2016	1	6	1	3	2016
99850	2016-03-01 06:17:10	2016-03-01 06:37:23	6	1	3	2016	1	6	1	3	2016
99851	2016-03-01 06:17:10	2016-03-01 06:22:09	6	1	3	2016	1	6	1	3	2016
99852	2016-03-01 06:17:11	2016-03-01 06:22:00	6	1	3	2016	1	6	1	3	2016

99853 rows × 12 columns

In [9]:
```
datetime_dim['datetime_id'] = datetime_dim.index
```

In [10]:
```
datetime_dim = datetime_dim[['datetime_id', 'tpep_pickup_datetime', 'pick_hour', 'pick_day', 'pick_month', 'pick_year', 'pick_wee
```



In [12]:
```
passenger_count_dim = df[['passenger_count']].drop_duplicates().reset_index(drop=True)
passenger_count_dim['passenger_count_id'] = passenger_count_dim.index
passenger_count_dim = passenger_count_dim[['passenger_count_id','passenger_count']]

trip_distance_dim = df[['trip_distance']].drop_duplicates().reset_index(drop=True)
trip_distance_dim['trip_distance_id'] = trip_distance_dim.index
trip_distance_dim = trip_distance_dim[['trip_distance_id','trip_distance']]
```

In [13]:
```
passenger_count_dim.head()
```

Out[13]:

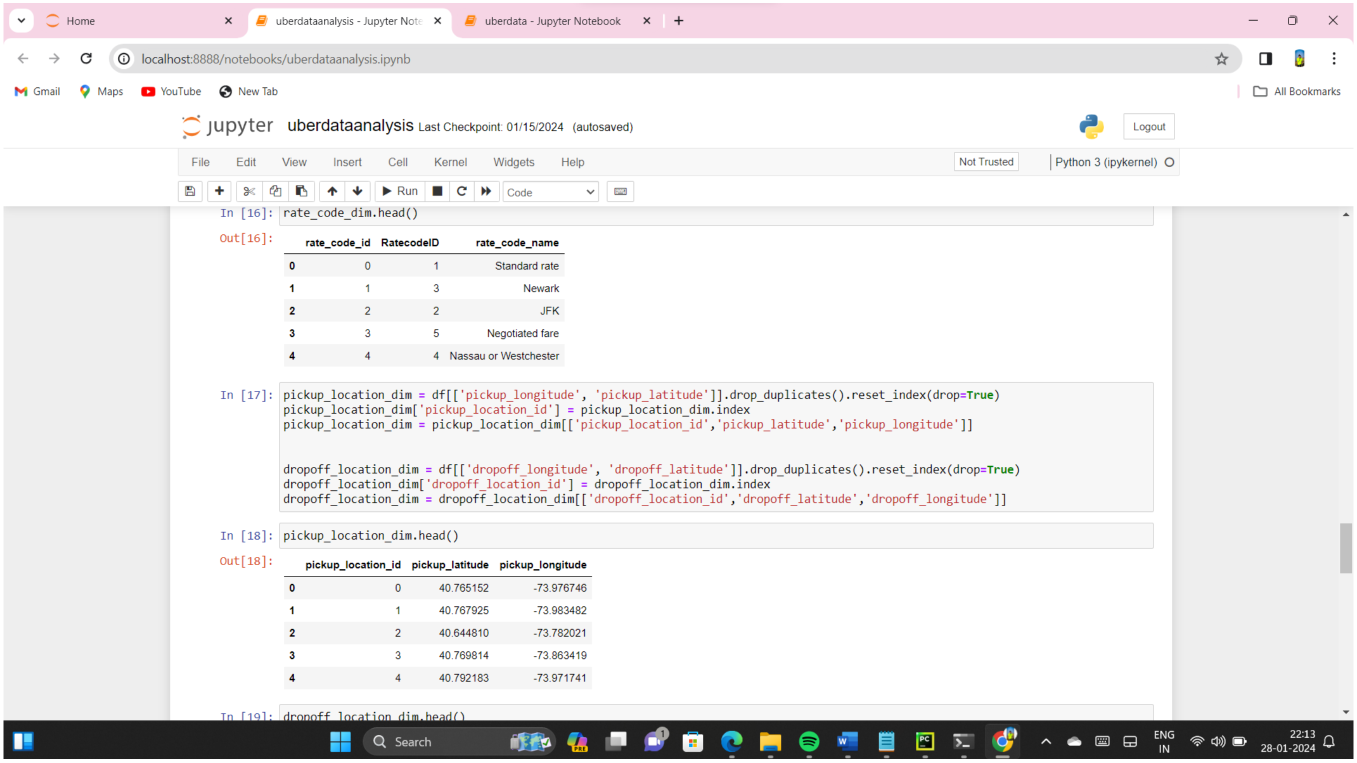
	passenger_count_id	passenger_count
0	0	1
1	1	2
2	2	3
3	3	5
4	4	6

In [14]:
```
trip_distance_dim.head()
```

Out[14]:

	trip_distance_id	trip_distance
0	0	2.50
1	1	2.90
2	2	19.98
3	3	10.78
4	4	30.43

In [15]:
```
rate_code_type = {
```


In [16]:
```
rate_code_dim.head()
```
Out[16]:

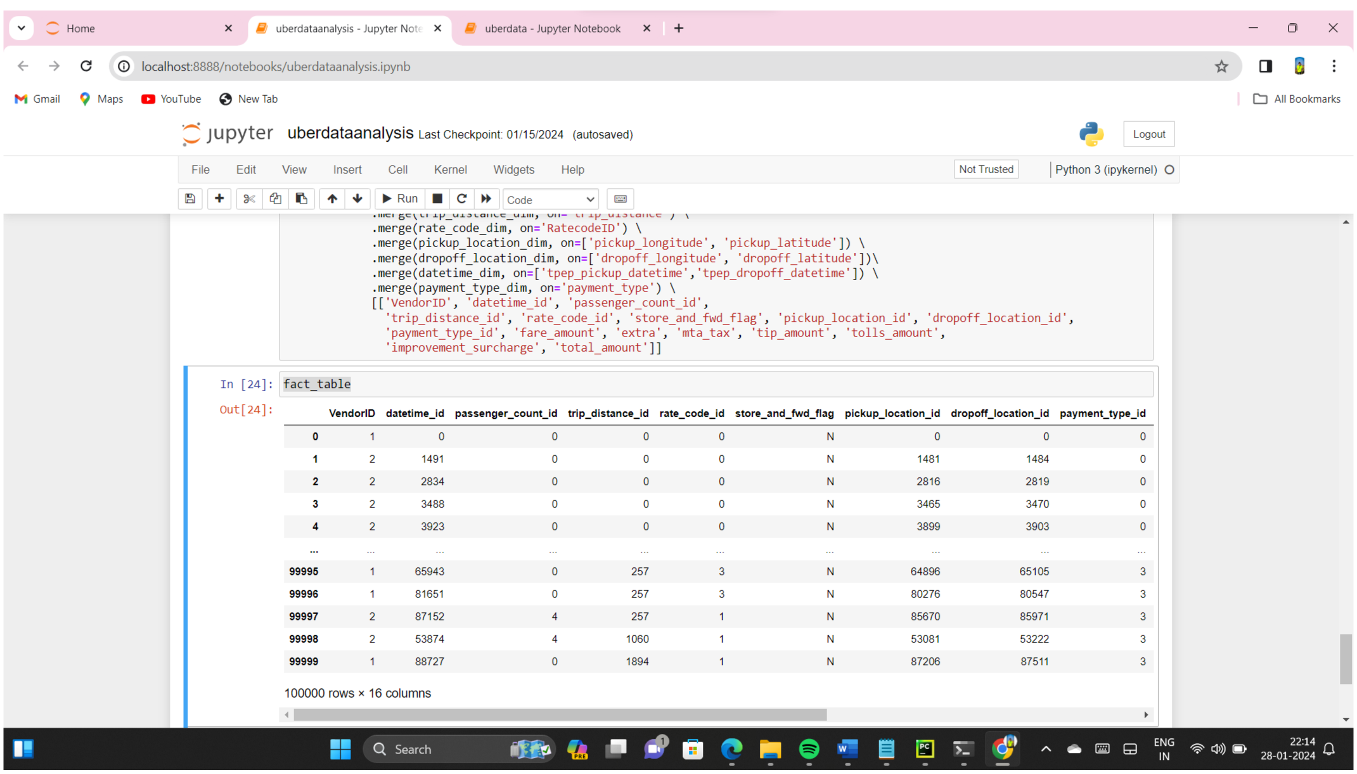
	rate_code_id	RatecodeID	rate_code_name
0	0	1	Standard rate
1	1	3	Newark
2	2	2	JFK
3	3	5	Negotiated fare
4	4	4	Nassau or Westchester

In [17]:
```
pickup_location_dim = df[['pickup_longitude', 'pickup_latitude']].drop_duplicates().reset_index(drop=True)
pickup_location_dim['pickup_location_id'] = pickup_location_dim.index
pickup_location_dim = pickup_location_dim[['pickup_location_id','pickup_latitude','pickup_longitude']]


dropoff_location_dim = df[['dropoff_longitude', 'dropoff_latitude']].drop_duplicates().reset_index(drop=True)
dropoff_location_dim['dropoff_location_id'] = dropoff_location_dim.index
dropoff_location_dim = dropoff_location_dim[['dropoff_location_id','dropoff_latitude','dropoff_longitude']]
```

In [18]:
```
pickup_location_dim.head()
```
Out[18]:

	pickup_location_id	pickup_latitude	pickup_longitude
0	0	40.765152	-73.976746
1	1	40.767925	-73.983482
2	2	40.644810	-73.782021
3	3	40.769814	-73.863419
4	4	40.792183	-73.971741

In [19]:
```
dropoff_location_dim.head()
```


```
         .merge(trip_distance_dim, on='trip_distance') \
         .merge(rate_code_dim, on='RatecodeID') \
         .merge(pickup_location_dim, on=['pickup_longitude', 'pickup_latitude']) \
         .merge(dropoff_location_dim, on=['dropoff_longitude', 'dropoff_latitude'])\
         .merge(datetime_dim, on=['tpep_pickup_datetime','tpep_dropoff_datetime']) \
         .merge(payment_type_dim, on='payment_type') \
         [['VendorID', 'datetime_id', 'passenger_count_id',
           'trip_distance_id', 'rate_code_id', 'store_and_fwd_flag', 'pickup_location_id', 'dropoff_location_id',
           'payment_type_id', 'fare_amount', 'extra', 'mta_tax', 'tip_amount', 'tolls_amount',
           'improvement_surcharge', 'total_amount']]
```

In [24]:
```
fact_table
```
Out[24]:

	VendorID	datetime_id	passenger_count_id	trip_distance_id	rate_code_id	store_and_fwd_flag	pickup_location_id	dropoff_location_id	payment_type_id
0	1	0	0	0	0	N	0	0	0
1	2	1491	0	0	0	N	1481	1484	0
2	2	2834	0	0	0	N	2816	2819	0
3	2	3488	0	0	0	N	3465	3470	0
4	2	3923	0	0	0	N	3899	3903	0
...	...	...	...	...	...	...	...	...	...
99995	1	65943	0	257	3	N	64896	65105	3
99996	1	81651	0	257	3	N	80276	80547	3
99997	2	87152	4	257	1	N	85670	85971	3
99998	2	53874	4	1060	1	N	53081	53222	3
99999	1	88727	0	1894	1	N	87206	87511	3

100000 rows × 16 columns

Reference

1. Kaggle – Explanatory Data Analysis
2. Research Gate – Explanatory Data Analysis
3. Medium (towardsdatascience) Felipe Alves Santos – Explanatory Data Analysis and Business Problem
4. Uber Blog – Dynamic pricing model
5. Unsplash – Images
6. Uber Auth – Dataset
7. Kaggle – Dataset for New York Pickups
8. Uber Blog – Scaling Machine Learning at Uber & Uber's Machine Learning Model

Conclusion

Explanatory Data Analysis is no small feat! It takes a lot of work and patience, but it is certainly a powerful tool if used properly in the context of your business.

After analyzing the various parameters, here are a few guidelines that we can conclude. If you were a Business analyst or data scientist working for Uber or Lyft, you could come to the following conclusions:

- Uber is very economical; however, Lyft also offers fair competition.
- People prefer to have a shared ride in the middle of the night.
- People avoid riding when it rains.
- When traveling long distances, the price does not increase by line. However, based on time and demand, increases can affect costs.
- Uber could be the first choice for long distances.

This guide briefly outlines some of the tips and tricks to simplify analysis and undoubtedly highlighted the critical importance of a well-defined business problem, which directs all coding efforts to a particular purpose and reveals key details. This business case also attempted to demonstrate the basic use of python in everyday business activities, showing how fun, important, and fun it can be.

THANK YOU