**Mini Project report submitted to**

Text, logo

Description automatically generated with medium confidence

**Uber Data Analytics: Dashboard**

**Group Members:**

* Kartik Mittal (215890116)
* Ananya Gupta (215890030)

Under the supervision of

**Dr. Rashmi R Dr. Gerard Deepak**

**Assistant Professor Assistant Professor**

Department of Computer Science and Engineering

Manipal Institute of Technology

Bengaluru

**Manipal Institute of Technology**

**Bengaluru Campus-560064, Karnataka, India.**

**Abstract**

This project explores Uber's operations in New York City using the TLC cab dataset. We preprocess the data with Python, conduct ETL using MageAI, and perform Big Data analytics on Google Cloud. The insights gained are visualized through a dashboard created with Looker Studio. The findings of this project shed light on various aspects of Uber's operations in New York, including peak demand periods, popular pickup and drop-off locations, and user preferences. Moreover, they offer valuable insights for optimizing operational strategies, enhancing user experiences, and improving overall efficiency. This project underscores the transformative potential of data analytics in shaping urban mobility. In conclusion, this project demonstrates the transformative potential of data analytics in shaping the future of urban mobility. By harnessing the power of data, we can drive innovation, inform decision-making, and ultimately, create more efficient and sustainable transportation systems for the benefit of society.

|  |  |  |
| --- | --- | --- |
| **Contents** | | **Page. No.** |
| **1** | Introduction………………………………………………….. | **4** |
| **2** | Problem Statement…………………………………………... | **5** |
| **3** | Objective…………………………………………………….. | **6** |
| **4** | Methodology……………………………………………….... | **7** |
| **5** | Implementation Details…………………………………….... | **9** |
| **6** | Results……………………………………………………….. | **18** |
| **7** | Conclusion…………………………………………………... | **21** |
| References…………………………………………………………... | | **25** |

**Chapter 1**

**Introduction**

In today's rapidly evolving world, data is often hailed as the new oil, fuelling innovation, decision-making, and insights across various industries. The transportation sector, in particular, stands at the forefront of this data revolution, generating vast amounts of information ripe for analysis. Among the myriad transportation services, ride-hailing platforms like Uber have emerged as significant contributors to this data landscape, offering invaluable insights into commuting patterns, user behaviours, and operational efficiencies.

For our Big Data Analytics project, we delved into the wealth of data provided by the New York City Taxi and Limousine Commission (TLC) cab dataset, focusing on Uber's operations within the bustling metropolis. Leveraging cutting-edge tools and technologies, including Python for preprocessing, MageAI for ETL (Extract, Transform, Load) processes, Google Cloud for Big Data analytics, and Looker Studio for dashboard creation, we embarked on a journey to extract actionable insights from raw data.

The primary objective of our project is to harness the power of data analytics to uncover hidden trends, optimize operational strategies, and enhance decision-making processes within the realm of Uber's operations in New York City. By employing advanced analytical techniques and visualization tools, we aim to offer stakeholders a comprehensive understanding of various facets of Uber's business, ranging from demand-supply dynamics to user preferences and geographic patterns.

In this report, we provide a detailed account of our methodology, outlining each stage of the data analytics pipeline, from data acquisition and preprocessing to analysis and visualization. Furthermore, we present a series of insightful findings and observations derived from our analysis, shedding light on key trends, challenges, and opportunities encountered throughout the project.

Through this report, we endeavour to showcase the significance of data-driven approaches in addressing real-world challenges, fostering innovation, and driving informed decision-making across industries. Moreover, we aim to inspire future endeavours in the field of data analytics, encouraging students and professionals alike to harness the power of data to unlock actionable insights and drive positive change in the world around us.

**Chapter 2**

**Problem statement**

This project aims to use data analytics to address key challenges within Uber's operations in New York City, including demand-supply imbalance, user experience optimization, operational efficiency, and geographic insights. Through data-driven insights, we seek to empower stakeholders to make informed decisions and enhance the overall urban mobility experience.

**Chapter 3**

**Objectives**

1. **Demand-Supply Dynamics Analysis**: Explore and analyse patterns in demand and supply of Uber rides within New York City to understand fluctuations and potential areas for improvement.
2. **User Behaviour Analysis**: Investigate user behaviours and preferences to identify trends, common routes, and opportunities for enhancing the overall user experience.
3. **Operational Efficiency Assessment**: Evaluate operational processes such as driver allocation, route optimization, and fleet management to identify inefficiencies and propose optimization strategies.
4. **Geographic Insights Exploration**: Analyse geographic patterns of Uber rides to identify areas of high demand, potential underserved regions, and opportunities for market expansion.

**Chapter 4**

**Methodology**

1. **Data Acquisition:**
   * Obtain the New York TLC cab dataset from the official sources or relevant repositories.
   * It has 1 million rows and 19 columns
2. **Data Modelling:**

* Modelled the data into facts and dimension table using Lucid app

1. **Data Preprocessing:**
   * Utilize Python programming language and libraries such as Pandas and NumPy for data preprocessing tasks.
   * Perform data cleaning, including handling missing values, removing duplicates, and correcting inconsistencies.
   * Convert data types, if necessary, to ensure compatibility for subsequent analysis.
   * Creating dimension tables and fact table for visualisation.
2. **ETL Process with MageAI:**
   * Utilize MageAI for the Extract, Transform, Load (ETL) process to automate data extraction and transformation tasks.
   * Extract relevant data from the pre-processed dataset.
   * Transform the data into a suitable format for analysis.
   * Load the transformed data into the Google Cloud platform for further analysis.
3. **Big Data Analytics on Google Cloud:**
   * Leverage Google Cloud's Big Data tools such as BigQuery, Dataflow, and Dataproc for scalable and efficient data analytics.
   * Perform exploratory data analysis (EDA) to gain initial insights into the dataset.
4. **Dashboard Creation with Looker Studio:**
   * Develop an interactive dashboard using Looker Studio to visualize and communicate key insights derived from the analysis.
   * Incorporate interactive features such as filters, drill-down options, and dynamic data displays to enhance user experience.
5. **Analysis and Interpretation:**
   * Analyse the results obtained from the data analytics process, focusing on key metrics and trends related to Uber's operations in New York City.
   * Interpret the findings to derive actionable insights and recommendations for stakeholders.
   * Identify patterns, correlations, and anomalies within the data to inform decision-making processes.

**Chapter 5**

**Implementation Details**

* **Data Acquisition**:

df=pd.read\_csv("uber\_data.csv")

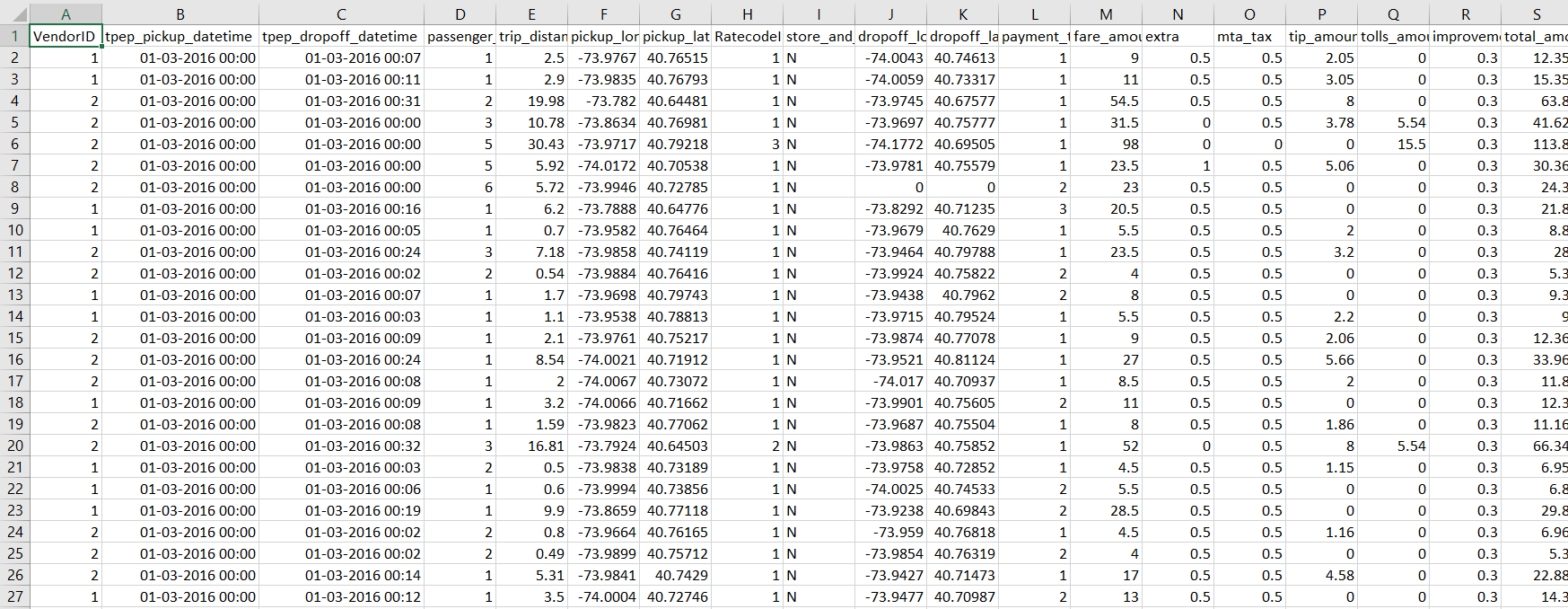


Figure 1 Dataset

df.info()

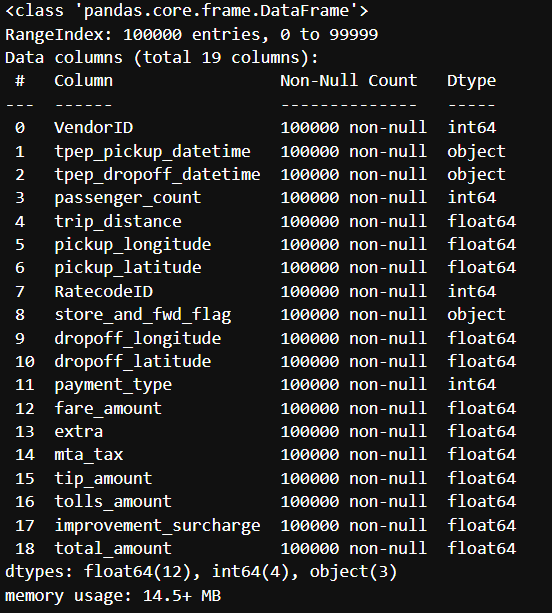


Figure 2 Data Description

* **Data Modelling:**

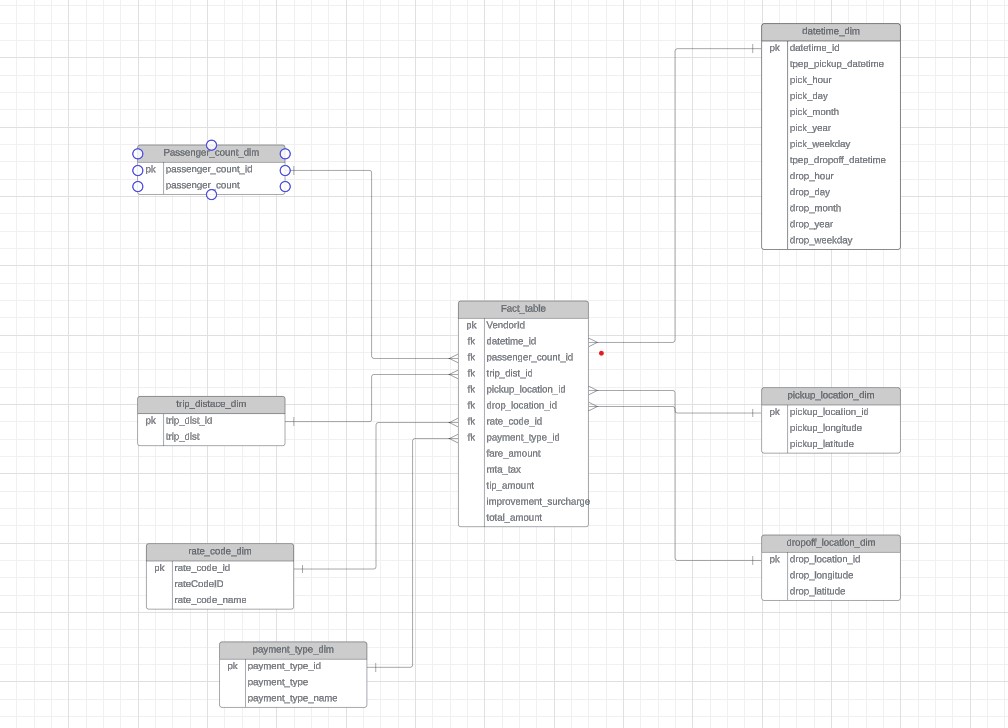


Figure 3 Data Model

* **Data Preprocessing:**

df['tpep\_pickup\_datetime']=pd.to\_datetime(df["tpep\_pickup\_datetime"])

df['tpep\_dropoff\_datetime']=pd.to\_datetime(df["tpep\_dropoff\_datetime"])

df=df.drop\_duplicates().reset\_index(drop=True)

df['trip\_id']=df.index

**DateTime Dimension Table:**

datetime\_dim=df[['tpep\_pickup\_datetime','tpep\_dropoff\_datetime']].reset\_index(drop=True)

datetime\_dim['tpep\_pickup\_datetime']=datetime\_dim['tpep\_pickup\_datetime']

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['tpep\_dropoff\_datetime']=datetime\_dim['tpep\_dropoff\_datetime']

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.head()

datetime\_dim['datetime\_id']=datetime\_dim.index

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 Dimension Table:**

passenger\_count\_dim=df[['passenger\_count']].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']].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 Dimension Table:**

rate\_code\_type={

1:'Standard rate',

2:'JFK',

3:'NewYark',

4:'Nassau or Westchester',

5:'Negotiated fare',

6:'Group ride'

}

rate\_code\_dim=df[['RatecodeID']].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']]

rate\_code\_dim.head(5)

**Location Dimenstion Table:**

pickup\_location\_dim = df[['pickup\_longitude', 'pickup\_latitude']].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']].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']]

**Payment Dimension Table:**

payment\_type\_name = {

1:"Credit card",

2:"Cash",

3:"No charge",

4:"Dispute",

5:"Unknown",

6:"Voided trip"

}

payment\_type\_dim = df[['payment\_type']].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']]

**Fact Table:**

fact\_table = df.merge(passenger\_count\_dim, left\_on='trip\_id', right\_on='passenger\_count\_id') \

.merge(trip\_distance\_dim, left\_on='trip\_id', right\_on='trip\_distance\_id') \

.merge(rate\_code\_dim, left\_on='trip\_id', right\_on='rate\_code\_id') \

.merge(pickup\_location\_dim, left\_on='trip\_id', right\_on='pickup\_location\_id') \

.merge(dropoff\_location\_dim, left\_on='trip\_id', right\_on='dropoff\_location\_id')\

.merge(datetime\_dim, left\_on='trip\_id', right\_on='datetime\_id') \

.merge(payment\_type\_dim, left\_on='trip\_id', right\_on='payment\_type\_id') \

[['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']]

* **ETL Process:**

**Export:**

from mage\_ai.settings.repo import get\_repo\_path

from mage\_ai.io.bigquery import BigQuery

from mage\_ai.io.config import ConfigFileLoader

from pandas import DataFrame

from os import path

if 'data\_exporter' not in globals():

from mage\_ai.data\_preparation.decorators import data\_exporter

@data\_exporter

def export\_data\_to\_big\_query(data: DataFrame, \*\*kwargs) -> None:

"""

Template for exporting data to a BigQuery warehouse.

Specify your configuration settings in 'io\_config.yaml'.

Docs: https://docs.mage.ai/design/data-loading#bigquery

"""

table\_id = 'uberdataprojectkartik.uber\_data\_project.fact\_table'

config\_path = path.join(get\_repo\_path(), 'io\_config.yaml')

config\_profile = 'default'

BigQuery.with\_config(ConfigFileLoader(config\_path, config\_profile)).export(

DataFrame(data['fact\_table']),

table\_id,

if\_exists='replace', )

**Transform:**

@transformer

def transform(df, \*args, \*\*kwargs):

"""

Template code for a transformer block.

Add more parameters to this function if this block has multiple parent blocks.

There should be one parameter for each output variable from each parent block.

Args:

data: The output from the upstream parent block

args: The output from any additional upstream blocks (if applicable)

Returns:

Anything (e.g. data frame, dictionary, array, int, str, etc.)

"""

df['tpep\_pickup\_datetime']=pd.to\_datetime(df["tpep\_pickup\_datetime"])

df['tpep\_dropoff\_datetime']=pd.to\_datetime(df["tpep\_dropoff\_datetime"])

df=df.drop\_duplicates().reset\_index(drop=True)

df['trip\_id']=df.index

datetime\_dim=df[['tpep\_pickup\_datetime','tpep\_dropoff\_datetime']].reset\_index(drop=True)

datetime\_dim['tpep\_pickup\_datetime']=datetime\_dim['tpep\_pickup\_datetime']

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['tpep\_dropoff\_datetime']=datetime\_dim['tpep\_dropoff\_datetime']

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\_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']].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']].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:'NewYark',

4:'Nassau or Westchester',

5:'Negotiated fare',

6:'Group ride'

}

rate\_code\_dim=df[['RatecodeID']].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']]

pickup\_location\_dim = df[['pickup\_longitude', 'pickup\_latitude']].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']].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']]

payment\_type\_name = {

1:"Credit card",

2:"Cash",

3:"No charge",

4:"Dispute",

5:"Unknown",

6:"Voided trip"

}

payment\_type\_dim = df[['payment\_type']].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']]

fact\_table = df.merge(passenger\_count\_dim, left\_on='trip\_id', right\_on='passenger\_count\_id') \

.merge(trip\_distance\_dim, left\_on='trip\_id', right\_on='trip\_distance\_id') \

.merge(rate\_code\_dim, left\_on='trip\_id', right\_on='rate\_code\_id') \

.merge(pickup\_location\_dim, left\_on='trip\_id', right\_on='pickup\_location\_id') \

.merge(dropoff\_location\_dim, left\_on='trip\_id', right\_on='dropoff\_location\_id')\

.merge(datetime\_dim, left\_on='trip\_id', right\_on='datetime\_id') \

.merge(payment\_type\_dim, left\_on='trip\_id', right\_on='payment\_type\_id') \

[['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']]

return {"datetime\_dim":datetime\_dim.to\_dict(orient='dict'),

"passenger\_count\_dim":passenger\_count\_dim.to\_dict(orient='dict'),

"trip\_distance\_dim":trip\_distance\_dim.to\_dict(orient='dict'),

"rate\_code\_dim":rate\_code\_dim.to\_dict(orient='dict'),

"pickup\_location\_dim":pickup\_location\_dim.to\_dict(orient='dict'),

"dropoff\_location\_dim":dropoff\_location\_dim.to\_dict(orient='dict'),

"payment\_type\_dim":payment\_type\_dim.to\_dict(orient='dict'),

"fact\_table":fact\_table.to\_dict(orient='dict')}

@test

def test\_output(output, \*args) -> None:

"""

Template code for testing the output of the block.

"""

assert output is not None, 'The output is undefined'

**Load:**

@data\_loader

def load\_data\_from\_api(\*args, \*\*kwargs):

"""

Template for loading data from API

"""

url = 'https://storage.googleapis.com/uber-data-project-kartik/uber\_data.csv'

response = requests.get(url)

return pd.read\_csv(io.StringIO(response.text), sep=',')

@test

def test\_output(output, \*args) -> None:

"""

Template code for testing the output of the block.

"""

assert output is not None, 'The output is undefined'

* **Big Data Analytics on Google Cloud:**

select VendorID, AVG(fare\_amount) from thinking-bonsai-412117.uber\_data\_project.analyticstbl group by VendorID

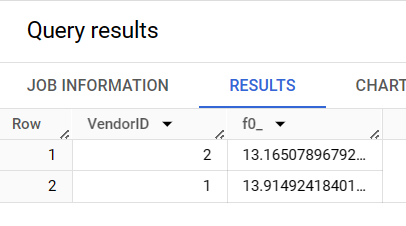


Figure 4 Query Result 1

select b.payment\_type\_name,sum(a.tip\_amount) from thinking-bonsai-412117.uber\_data\_project.fact\_table a

join thinking-bonsai-412117.uber\_data\_project.payment\_type\_dim b

on a.payment\_type\_id=b.payment\_type\_id

group by b.payment\_type\_name;

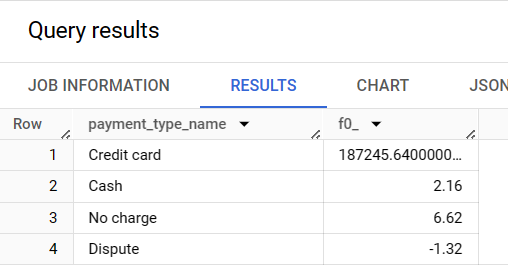


Figure 5 Query Result 2

* **Dashboard Creation:**

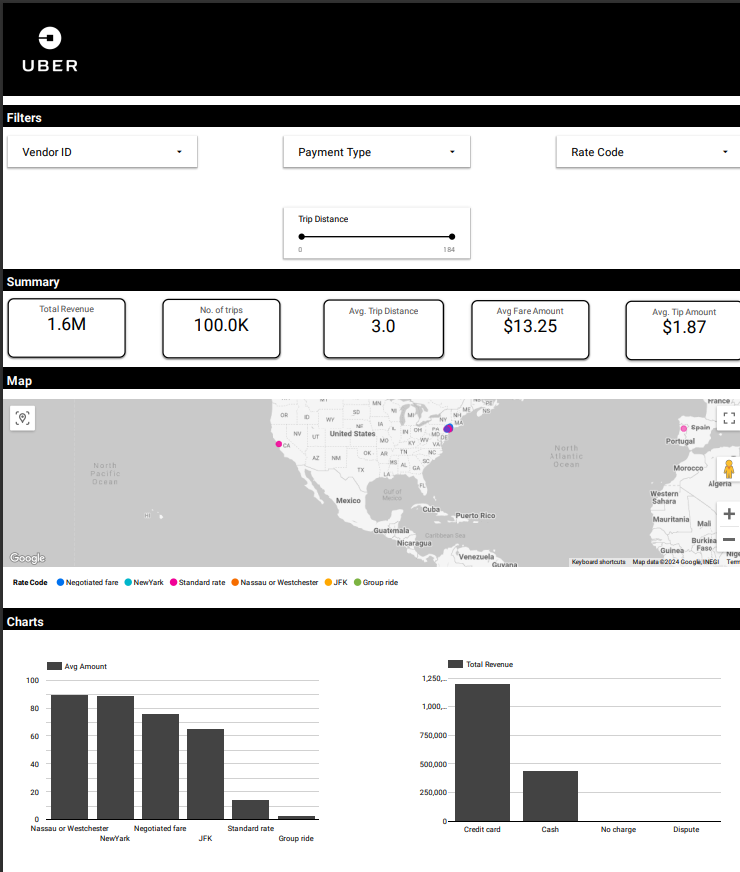
****

Figure 6 Dashboard

**Chapter 6**

**Results**

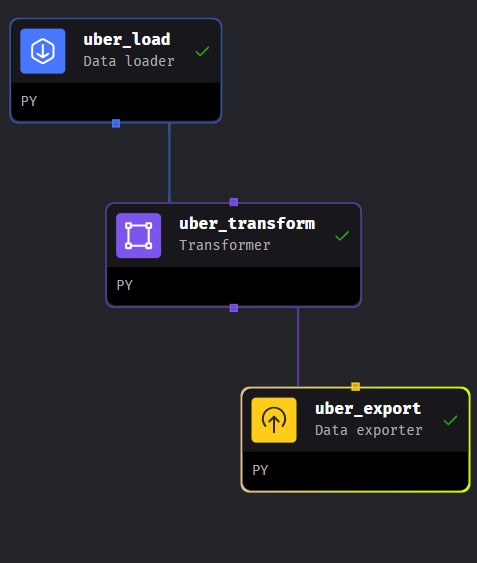


Figure 7 Pipeline

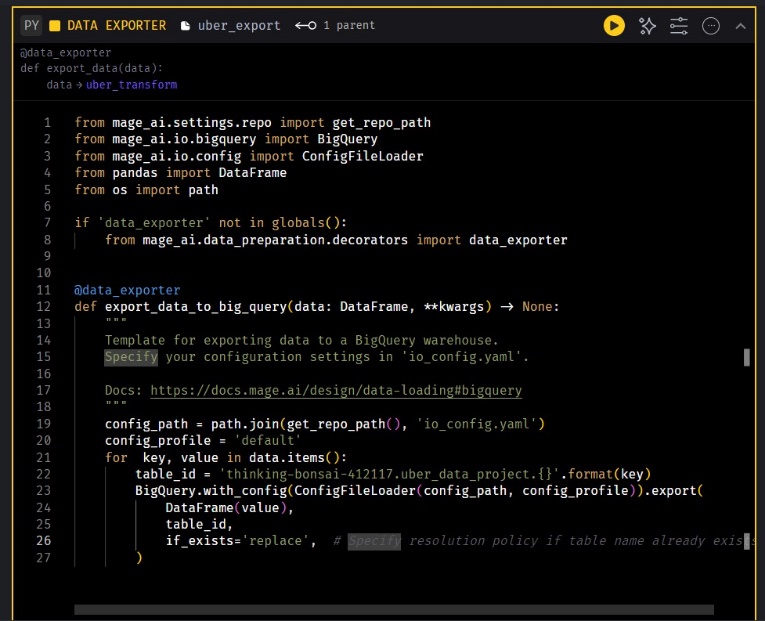


Figure 8 Exporter

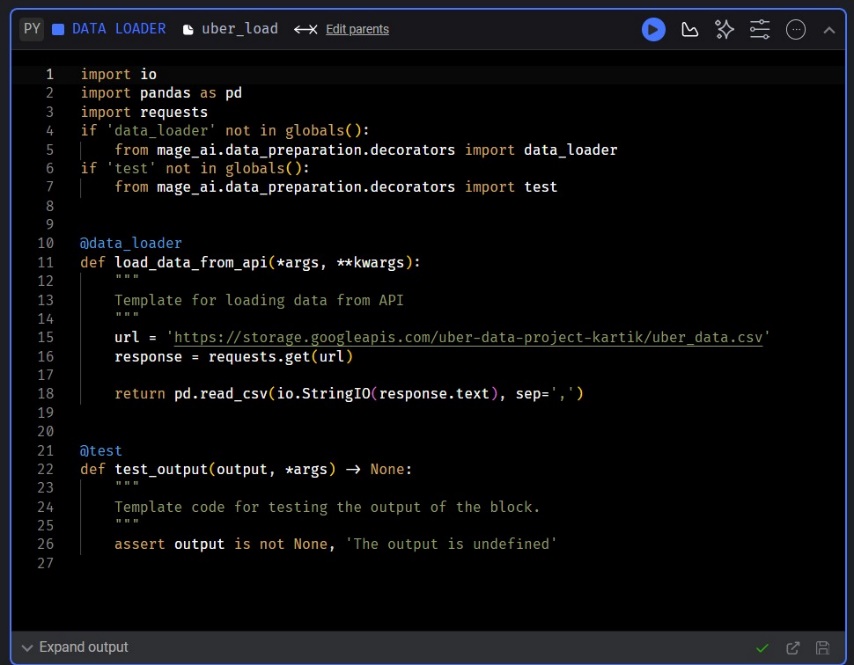


Figure 9 Loader

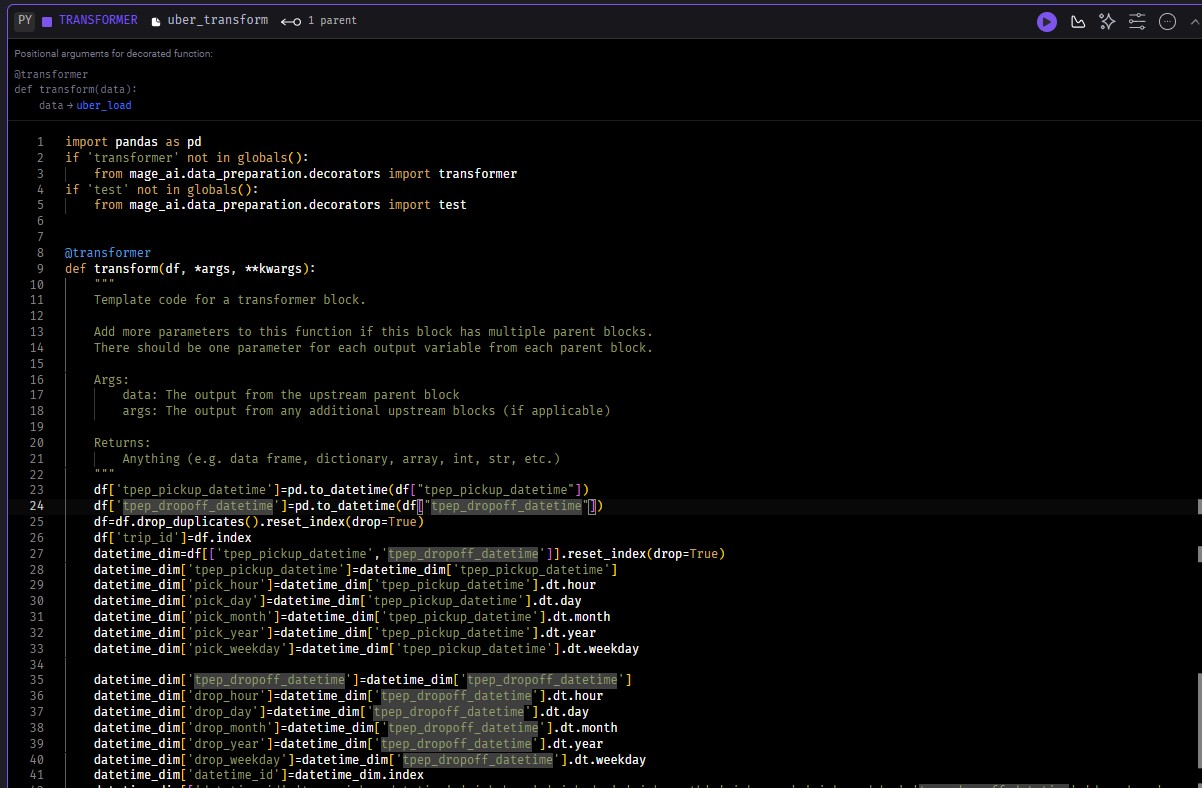


Figure 10 Transformer

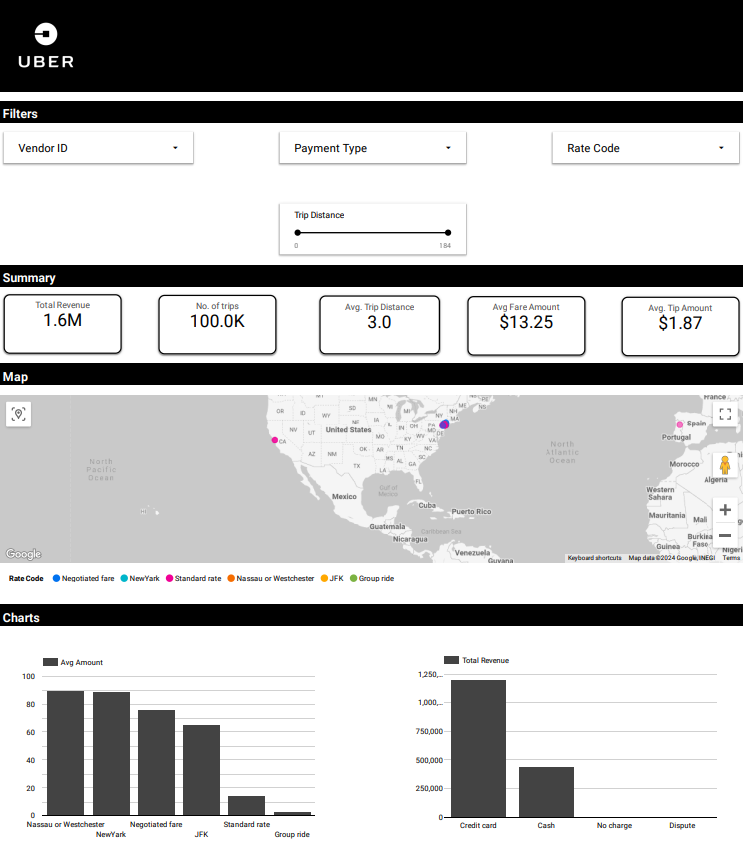


Figure 11 Dashboard

**Chapter 7**

**Conclusion**

In conclusion, this Uber data analytics project has provided valuable insights into the operational dynamics and user behaviours within the New York City transportation ecosystem. Leveraging the New York TLC cab dataset, we successfully pre-processed the data using Python, ensuring its quality and readiness for analysis. The adoption of MageAI facilitated an efficient Extract, Transform, Load (ETL) process, while Google Cloud's robust infrastructure enabled scalable and powerful Big Data analytics.

Key findings from our analysis include insights into peak demand periods, popular pickup and drop-off locations, and user travel preferences. These findings have direct implications for optimizing operational strategies, enhancing user experiences, and improving overall efficiency within Uber's operations in New York City.

Looking ahead, opportunities for further exploration include real-time data integration, advanced machine learning applications, and additional data source incorporation. Ultimately, our project demonstrates the power of data-driven approaches in enhancing transportation efficiency and shaping the future of urban mobility.

**References**

* <https://www.nyc.gov/site/tlc/about/data.page> (Data - TLC (nyc.gov))
* <https://cloud.google.com/looker/docs/creating-user-defined-dashboards>
* <https://cloud.google.com/bigquery/docs/query-overview> (Overview of BigQuery analytics  |  Google Cloud)
* [VM instance pricing  |  Compute Engine: Virtual Machines (VMs)  |  Google Cloud](https://cloud.google.com/compute/vm-instance-pricing?hl=en&text=vm%20instance)
* [🧙‍♀️ Welcome to Mage - Mage](https://docs.mage.ai/introduction/overview)
* [ETL pipeline tutorial - Mage](https://docs.mage.ai/guides/load-api-data)