**TABLE OF CONTENTS**

**CHAPTER NO. TITLE PAGE NO.**

**ABSTRACT** **3**

**1 INTRODUCTION 4**

1.1 Introduction to the Project  
 1.2 Problem Statement  
 1.3 Objectives  
 1.4 Scope

**2 LITERATURE REVIEW 6**

**3 METHODOLOGY 8**

3.1 Data Collection and Sources  
 3.2 Data Pre-processing & Cleaning  
 3.3 Exploratory Data Analysis (EDA)  
 3.4 Feature Engineering  
 3.5 Model Selection and Justification  
 3.6 Model Training and Evaluation  
 3.7 Hyperparameter Tuning

**4 IMPLEMENTATION 11**

4.1 Tools & Technologies Used  
 4.2 Data Pipeline and Processing  
 4.3 Machine Learning/Deep Learning Models  
 4.4 Model Deployment (if applicable)  
 4.5 System Architecture

**5 RESULTS AND DISCUSSION 13**

5.1 Performance Metrics  
 5.2 Comparison of Models  
 5.3 Error Analysis  
 5.4 Insights from the Model

**6 CONCLUSION AND FUTURE WORK 16**

**7 APPENDICES**

**Appendix-1:** Code – Technical Details 17  
 **Appendix-2:** Visualizations and Graphs 28

**8 REFERENCES 37**

**9 WEB REFERENCES 38**

**OFFER LETTER 39**

**CERTIFICATE OF COMPLETION 40**

**ABSTRACT**

Dynamic pricing is an advanced pricing strategy that allows businesses to adjust prices in real time based on various factors such as demand, competition, seasonality, and customer behaviour. This project explores the implementation of a **data-driven dynamic pricing model** using **machine learning techniques** to optimize pricing decisions and maximize revenue.

The study begins with **data collection** from multiple sources, including historical pricing data, competitor pricing, customer demand trends, and external factors such as holidays or economic conditions. **Data pre-processing** techniques, including handling missing values, feature engineering, and normalization, are applied to clean and prepare the dataset for analysis.

Exploratory Data Analysis (**EDA**) is conducted to identify key patterns and correlations between pricing and external factors. Several **machine learning models** are trained and evaluated, including **linear regression, decision trees, random forests, and deep learning models** to predict optimal prices. **Hyper parameter tuning** is performed to improve model accuracy.

Results indicate that dynamic pricing models significantly improve revenue and customer engagement compared to static pricing strategies. The proposed approach can be applied across various industries such as **e-commerce, hospitality, transportation, and retail**. Future work includes incorporating **reinforcement learning techniques** and real-time competitor price monitoring for further optimization.

**CHAPTER 1**

1. Introduction

1.1 Introduction to the project

Dynamic pricing, a strategy where prices are adjusted in real time based on market demand, customer behaviour, and other factors, has become increasingly significant in today's competitive market landscape. This project aims to develop a sophisticated system leveraging machine learning to optimize dynamic pricing strategies, enhancing revenue generation and customer satisfaction.

1.2 Objectives:

1. Develop and implement dynamic pricing algorithms to optimize revenue generation in real-time.

2. Create personalized pricing strategies based on individual customer preferences and behaviours.

3. Integrate competitive analysis techniques to ensure competitiveness while maintaining profitability.

4. Evaluate the effectiveness of dynamic pricing strategies through performance metrics.

5. Enhance user engagement and satisfaction by offering targeted pricing incentives aligned with customer preferences.

1.3 Scope:

This project aims to develop a **machine learning-based dynamic pricing model** applicable across industries like **e-commerce, travel, retail, and transportation**. It enables **real-time price adjustments** based on demand, competition, and market trends.

Key capabilities include **data-driven price optimization, automation through APIs, and scalability** for businesses of all sizes. Future enhancements involve **reinforcement learning, sentiment analysis, personalized pricing, and competitor price tracking** to improve accuracy and profitability.

The model helps businesses **maximize revenue, stay competitive, and enhance customer engagement** through intelligent pricing strategies.

**CHAPTER 2**

LITERATURE REVIEW

Dynamic pricing has evolved from traditional **rule-based approaches** to **machine learning-driven models**, enabling real-time price optimization across industries.

### ****1. Traditional Pricing Methods****

Early models, such as **cost-plus pricing and demand-based pricing**, were widely used in airlines and retail (Talluri & Van Ryzin, 2004). However, these lacked adaptability to real-time market conditions.

### ****2. Machine Learning in Pricing****

Recent studies show that **regression models, decision trees, and deep learning** improve pricing accuracy (Chen et al., 2016; Wang et al., 2019). Companies like **Amazon and Uber** use AI-based models to dynamically adjust prices.

### ****3. Reinforcement Learning (RL) for Pricing****

RL-based approaches like **Q-learning and deep Q-networks (DQN)** allow continuous learning from market conditions, as seen in **airline yield management** (Smith et al., 1992) and **e-commerce platforms** (Zhao et al., 2020).

### ****4. External Factors & Pricing****

Time-series forecasting models like **ARIMA and LSTM** predict demand fluctuations (Choi & Varian, 2012), while **NLP-based sentiment analysis** helps e-commerce platforms adjust prices based on customer reviews.

### ****5. Challenges & Future Scope****

Key challenges include **ethical concerns (price discrimination), model interpretability (black-box AI), and real-time scalability** (Acquisti et al., 2016). Research suggests using **SHAP values** for transparency (Lundberg & Lee, 2017).

### ****Conclusion****

Machine learning and AI have revolutionized dynamic pricing, but future work should focus on **ethical pricing, transparency, and real-time adaptability** to enhance trust and effectiveness.

**CHAPTER 3**

3. METHODOLOGY

### ****3.1 Data Collection and Sources****

Data collection is the first step in building an effective dynamic pricing model. Data for this project is gathered from various sources, including:

* **Historical pricing data** from e-commerce platforms, airlines, or retail stores.
* **Competitor pricing information**, available through web scraping or publicly accessible APIs.
* **Customer behaviour data** such as purchase history, browsing patterns, and user demographics.
* **External factors** like weather, holidays, and economic indicators (e.g., inflation rates).  
  This data is then integrated into a unified dataset for analysis.

### ****3.2 Data Pre-processing & Cleaning****

Raw data often contains inconsistencies, missing values, and irrelevant information. The pre-processing steps include:

* **Handling missing values** by using imputation techniques or removing incomplete records.
* **Normalization and scaling** of numerical features to standardize the range of values (e.g., Min-Max scaling).
* **Encoding categorical data** using techniques like one-hot encoding or label encoding.
* **Outlier detection and removal** to ensure the model is not skewed by extreme values.

### ****3.3 Exploratory Data Analysis (EDA)****

EDA is conducted to understand the underlying patterns in the data and identify key factors influencing pricing. The process includes:

* **Visualization**: Using graphs like histograms, box plots, and scatter plots to explore the distribution of data and relationships between features.
* **Correlation analysis**: Calculating correlation coefficients to identify dependencies between variables, especially between price and external factors.
* **Identifying trends**: Analysing time-series data to uncover trends, seasonality, and cyclic behaviour.

### ****3.4 Feature Engineering****

Feature engineering is crucial for improving model performance. This includes:

* **Creating new features**: For example, extracting date-related features (e.g., day of the week, holiday season) or customer segments (e.g., loyal vs. new customers).
* **Interaction terms**: Combining features such as **demand and price elasticity** to capture complex relationships.
* **Dimensionality reduction**: Using techniques like PCA (Principal Component Analysis) to reduce the number of features while retaining important information.

### ****3.5 Model Selection and Justification****

Several models are considered for the dynamic pricing task, including:

* **Linear Regression**: A simple approach to model the relationship between price and features.
* **Decision Trees**: Non-linear models that can handle categorical data and capture complex patterns.
* **Random Forests**: An ensemble method that reduces over fitting by averaging predictions from multiple decision trees.
* **Deep Learning**: Neural networks, especially **deep neural networks (DNN)**, for capturing non-linear relationships and handling large datasets.  
  The final model is selected based on its ability to accurately predict price, interpretability, and computational efficiency.

### ****3.6 Model Training and Evaluation****

The selected model is trained using the pre-processed data, with the following steps:

* **Splitting the data** into training, validation, and test sets (e.g., 70%-15%-15%).
* **Training the model** on the training set and validating it on the validation set to assess its performance.
* **Evaluation metrics** such as **mean squared error (MSE), mean absolute error (MAE), and R-squared** are used to measure prediction accuracy.
* **Cross-validation** techniques are employed to ensure the model generalizes well on unseen data.

### ****3.7 Hyperparameter Tuning****

To optimize the model's performance, hyper parameter tuning is performed using techniques like:

* **Grid Search**: Exhaustively testing a range of hyperparameters to find the optimal combination.
* **Random Search**: Sampling hyperparameters randomly to find a good combination more efficiently.
* **Bayesian Optimization**: An advanced approach for efficiently finding optimal hyperparameters.  
  Hyperparameter tuning is performed to minimize over fitting and improve the model’s predictive accuracy.

**CHAPTER 4**

**4. IMPLEMENTATIONS**

### ****4.1 Tools & Technologies Used****

The following tools and technologies were used in the implementation of the dynamic pricing model:

* **Programming Languages**: **Python** (for data processing and model development), **SQL** (for database queries).
* **Data Processing Libraries**: **Pandas**, **NumPy** (data manipulation), **Scikit-learn** (machine learning algorithms).
* **Deep Learning Frameworks**: **TensorFlow**, **Keras** (for neural networks).
* **Data Visualization**: **Matplotlib**, **Seaborn** (visualizations).
* **Cloud/Deployment Tools**: **AWS** (for cloud-based computing), **Docker** (containerization), **Flask** (for API deployment).

### ****4.2 Data Pipeline and Processing****

The data pipeline involves the following steps:

* **Data Collection**: Raw data is gathered from various sources (e.g., APIs, databases).
* **Data Cleaning**: Missing values and outliers are handled, and data is normalized.
* **Feature Engineering**: New features are created from the raw data to capture important patterns.
* **Data Splitting**: The data is split into training, validation, and test sets for model evaluation.

### ****4.3 Machine Learning/Deep Learning Models****

The following models were used for dynamic pricing:

* **Linear Regression**: A baseline model to predict pricing based on linear relationships.
* **Random Forest**: Used for capturing non-linear relationships and improving prediction accuracy.
* **Gradient Boosting**: **XGBoost** and **LightGBM** are employed for their performance in tabular data.
* **Deep Learning**: **Neural Networks** (DNNs) were used for capturing complex patterns in large datasets.
* **Reinforcement Learning**: An optional technique for adaptive learning from market conditions over time.

### ****4.4 Model Deployment (if applicable)****

For model deployment, the following steps are involved:

* **API Development**: A REST API is developed using **Flask** to integrate the dynamic pricing model with external systems (e.g., e-commerce platforms).
* **Containerization**: The model and API are containerized using **Docker** for easy deployment and scaling.
* **Cloud Deployment**: The solution is deployed on **AWS** to ensure scalability and reliability, handling high volumes of data in real-time.

### ****4.5 System Architecture****

The system architecture is designed as follows:

* **Data Collection Layer**: Sources like APIs and databases provide real-time data.
* **Data Processing Layer**: Preprocessing, feature engineering, and model training take place here.
* **Model Layer**: Machine learning models are used to predict dynamic prices.
* **Application Layer**: A web-based interface or API delivers predictions to users.
* **Deployment Layer**: The entire system is hosted on **AWS**, with Docker for containerization to ensure portability and scalability.

**CHAPTER 5**

5. RESULTS AND DISCUSSION

### ****5.1 Performance Metrics****

To evaluate the performance of the dynamic pricing model, the following metrics were used:

* **Mean Absolute Error (MAE)**: Measures the average magnitude of errors in pricing predictions, without considering their direction.
* **Mean Squared Error (MSE)**: Gives a higher penalty to large errors, used to assess the accuracy of the model.
* **R-squared (R²)**: Indicates how well the model explains the variance in the pricing data. A higher R² value suggests a better fit.
* **Root Mean Squared Error (RMSE)**: Similar to MSE but provides results in the same unit as the target variable (price), making it easier to interpret.

### ****5.2 Comparison of Models****

Various models were evaluated based on the performance metrics:

* **Linear Regression**: Served as a baseline, showing moderate performance with relatively low R² and high MAE, indicating limited ability to capture complex pricing patterns.
* **Random Forest**: Improved accuracy with a **higher R²** and lower **MSE**, as it handles non-linear relationships better than linear models.
* **Gradient Boosting (XGBoost, LightGBM)**: Outperformed the previous models with **lower MSE** and **MAE**, as it is highly effective in capturing interactions and non-linear trends.
* **Neural Networks**: Provided the best results in terms of **R²**, especially when dealing with large and complex datasets, but required more computational resources.

### ****5.3 Error Analysis****

Despite the models' accuracy, errors were still observed:

* **Overfitting**: More complex models (e.g., Random Forest, XGBoost) exhibited overfitting on the training data, where they performed well on training but struggled with unseen test data.
* **Price Volatility**: Dynamic pricing models were more prone to errors during periods of sudden price fluctuations or market shifts (e.g., during sales, holidays).
* **Data Quality Issues**: Missing values or inaccurate data in external factors (e.g., economic indicators) affected model performance, resulting in higher errors during periods of unusual market conditions.

### ****5.4 Insights from the Model****

The dynamic pricing model provided valuable insights:

* **Price Sensitivity**: The model identified customer segments most sensitive to price changes, which helps in optimizing prices for maximum revenue.
* **Market Trends**: The model successfully captured market trends, adjusting prices based on demand patterns, time of day, and seasonality.
* **Competitor Pricing**: The model incorporated competitor prices effectively, making it possible to stay competitive while maximizing profit margins.
* **Real-time Adjustments**: The model proved effective in real-time adjustments, where prices could be dynamically altered based on customer behaviour and external factors.

**CHAPTER 6**

6. CONCLUSION AND FUTURE WORK

### ****CONCLUSION****

The dynamic pricing model successfully demonstrated its potential to optimize pricing strategies based on historical data, competitor pricing, and customer behaviour. Key outcomes include:

* **Improved Accuracy**: Machine learning models, especially **Gradient Boosting** and **Neural Networks**, significantly outperformed traditional pricing models by capturing complex patterns in the data.
* **Real-time Pricing Adjustments**: The model was able to make real-time pricing changes based on external factors like market trends and customer demand, ensuring competitive and optimized pricing.
* **Scalability**: The system was designed to scale with increasing data and market fluctuations, thanks to cloud-based deployment and containerization.
* **Valuable Insights**: The model provided actionable insights into customer behaviour and price sensitivity, enabling businesses to make data-driven decisions for revenue maximization.

### ****FUTURE WORK****

Despite the success of the model, there are several areas for improvement and future exploration:

* **Integration of More Data Sources**: Expanding the data pipeline to include additional factors such as **social media sentiment analysis**, **economic indicators**, and **geolocation data** could further enhance pricing strategies.
* **Reinforcement Learning**: Implementing **reinforcement learning** for continuous price optimization based on real-time feedback from the market could provide an even more adaptive approach to dynamic pricing.
* **Model Refinement**: Further fine-tuning of model hyperparameters and exploring **ensemble learning techniques** (e.g., stacking, boosting) could further improve model performance.
* **Personalized Pricing**: Extending the model to implement **personalized pricing** based on individual customer profiles and preferences could increase customer satisfaction and conversion rates.
* **Multi-Channel Pricing**: Future work could involve integrating dynamic pricing across multiple platforms (e.g., online, in-store, mobile), ensuring consistency and maximizing profitability across all touch points.

**APPENDICES**

**Appendix-1:** Code – Technical Details

Data wrangling Techniques for Dynamic Pricing Dataset:

1. Data Description:

The instructions outline methods for initial data exploration and understanding:

**1. Head:** Shows the first few rows of data for a quick overview of its structure and content.

**2. Tail:** Displays the last few rows to ensure dataset completeness and consistency, especially regarding ordering and formatting.

**3. Info:** Provides details about the dataset's structure, data types, and memory usage, offering insights into its composition and characteristics.

**4. Describe:** Generates descriptive statistics for numerical features, summarizing key statistical measures to understand data distribution and variability.

Python code:

# Sample code for pricing data description

import pandas as pd

pricing\_data=pd.read\_csv('/content/drive/MyDrive/AI/dynamic\_pricing.csv')

print(pricing\_data.head()) print(pricing\_data.tail()) print(pricing\_data.info()) print(pricing\_data.describe())

#### 2. Null Data Handling:

The instructions outline a systematic approach to deal with missing data in a dataset:

1. **Null Data Identification:** Identify and locate missing values within the dataset to understand the extent of messiness.
2. **Null Data Imputation:** Fill missing values using suitable strategies such as statistical measures or advanced techniques like interpolation.
3. **Null Data Removal:** Eliminate rows or columns with excessive missing values to maintain data quality and integrity.

These steps help ensure that missing data is managed effectively, minimizing its impact on subsequent analyses or modelling.

Python code:

# Sample code for null data handling import pandas as pd pricing\_data=pd.read\_csv('/content/drive/MyDrive/AI/dynamic\_pricing.csv') print(pricing\_data.isnull().sum()) pricing\_data = pricing\_data.dropna() # Drop rows with missing values

## 3. Data Validation:

**Data Integrity Check:** The instruction underscores the necessity of conducting a data integrity check to ensure the accuracy and reliability of the dataset. This involves verifying data consistency and integrity to identify and rectify errors, ultimately enhancing the dataset's credibility and suitability for analysis or modelling purposes.

Python code:

# Sample code for data validation # Check for unique values in a column import pandas as pd

pricing\_data=pd.read\_csv('/content/drive/MyDrive/AI/dynamic\_pricing.csv')

print(pricing\_data[‘Vehicle\_Type'].unique())

**4. Data Reshaping:**

##### Reshaping Rows and Columns:

The instruction highlights the process of reshaping rows and columns in the pricing data to make it suitable for analysis. This transformation involves restructuring the data into a more organized and informative format that facilitates easier interpretation and analysis.

Python code:

# Sample code for data reshaping

# Reshape the dataset import pandas as pd

pricing\_data=pd.read\_csv('/content/drive/MyDrive/AI/dynamic\_pricing.csv')

grouped\_data = pricing\_data.groupby(['Time\_of\_Booking', 'Vehicle\_Type'])

['Number\_of\_Past\_Rides'].mean().unstack() print(grouped\_data)

**5. Data Merging:**

##### Combining Datasets:

The instruction underscores the importance of merging multiple datasets or data sources to enrich pricing analysis. By combining diverse sources of data, analysts can create a unified dataset that offers a comprehensive view of pricing-related factors, facilitating deeper insights into market dynamics and consumer behaviour.

Python code:

import pandas as pdcompetitor\_pricing\_data=pd.read\_csv('/content/drive/MyDrive/AI/dynamic.csv') pricing\_data=pd.read\_csv('/content/drive/MyDrive/AI/dynamic\_pricing.csv')

# Sample code for data merging merged\_data = pd.merge(pricing\_data, competitor\_pricing\_data,left\_on=None) print(merged\_data)

6. Data Aggregation:

The instruction highlights two critical steps in data analysis:

1. **Grouping Data:** This involves organizing pricing data into distinct groups based on specified criteria, such as product categories or time periods. Grouping enables segmentation and facilitates focused analysis on subsets of the dataset.
2. **Aggregating Data:** Once data is grouped, summary statistics like sum, mean, or count are computed for each group. Aggregation condenses the grouped data, providing key insights into overall trends and patterns within each group.

Python code:

import pandas as pd pricing\_data=pd.read\_csv('/content/drive/MyDrive/AI/dynamic\_pricing.csv')

# Sample code for data aggregation

grouped\_data = pricing\_data.groupby('Vehicle\_Type')

aggregated\_data = grouped\_data.agg({'Historical\_Cost\_of\_Ride':['mean', 'min', 'max']})

print(aggregated\_data)

## Dynamic Pricing Analysis Techniques:

**1. Price Distribution Analysis:**

Price Distribution Analysis entails examining how prices are spread across different products within a dataset. This analysis helps in understanding the variability and patterns of pricing within a product portfolio, identifying trends, outliers, and potential pricing strategies.

Python code:

import pandas as pd

import matplotlib.pyplot as plt pricing\_data=pd.read\_csv('/content/drive/MyDrive/AI/dynamic\_pricing.csv')

# Sample code for price distribution analysis plt.hist(pricing\_data['Historical\_Cost\_of\_Ride'], bins=20)

plt.title('ride cost')

plt.xlabel('X')

plt.ylabel('Y')

plt.show()

**2. Competitor Price Comparison:**

Comparing our pricing strategy with competitor prices involves assessing our competitiveness and market positioning by analysing how our prices align with those of competitors. This evaluation helps identify potential adjustments to our pricing strategy, ensuring we remain competitive, attract customers, and maximize revenue while maintaining profitability.

Python code:

import pandas as pd

import matplotlib.pyplot as plt

# Read the CSV file into a pandas DataFrame

pricing\_data = pd.read\_csv('/content/drive/MyDrive/AI/dynamic\_pricing.csv')

numeric\_data = pd.to\_numeric(pricing\_data['Number\_of\_Past\_Rides'],

errors='coerce').dropna()

# Convert the numeric data to a list

y = list(numeric\_data)

# Create the histogram

plt.hist(y, bins=20) # Specify the number of bins (optional)

plt.title('Competitor Price Comparison')

plt.xlabel('Number of Past Rides') # Update the x-axis label

plt.ylabel('Frequency') # Update the y-axis label

plt.show()

##### 3. Dynamic Pricing Model Performance:

Evaluating the performance of dynamic pricing models entails assessing their effectiveness in predicting and adjusting prices to adapt to market changes. This assessment involves analysing metrics like revenue growth, profit margins, customer satisfaction, and competitiveness against competitors. By understanding the strengths and weaknesses of these models, we can optimize pricing strategies to achieve desired business outcomes.

Python code:

import pandas as pd

import matplotlib.pyplot as plt

# Read the CSV file into a pandas DataFrame

model\_performance=pd.read\_csv('/content/drive/MyDrive/AI/dynamic\_pricing.csv')

# Sample code for dynamic pricing model performance evaluation plt.plot(model\_performance['Vehicle\_Type'], model\_performance['Historical\_Cost\_of\_Ride'], label='cost') plt.plot(model\_performance['Vehicle\_Type'], model\_performance['Average\_Ratings'],

label='Rating')

plt.xlabel('X')

plt.ylabel('Y')

plt.legend()

plt.show()

**Exploratory Data Analysis (EDA)**

Exploratory Data Analysis (EDA) is a vital initial step in data analysis. It involves exploring the dataset through visualizations, summary statistics, and hypothesis testing to understand its structure, patterns, and relationships.

**1. Univariate Analysis:**

Univariate Analysis involves studying individual variables to understand their distributions and characteristics. This analysis examines one variable at a time, using descriptive statistics and visualizations to uncover patterns, trends, and outliers. It provides insights into the range, central tendency, and variability of each variable, serving as a foundation for further analysis.

Python code:

|  |
| --- |
| import seaborn as sns  import matplotlib.pyplot as plt  data=pd.read\_csv('/content/drive/MyDrive/AI/dynamic\_pricing.csv') sns.histplot(data['Number\_of\_Past\_Rides'], bins=20)  plt.show() |

**2. Bivariate Analysis:**

Bivariate Analysis examines the relationship between two variables within a dataset. It helps identify correlations or associations through techniques like scatter plots and correlation coefficients, providing insights into underlying patterns and dynamics in the data.

Python code:

|  |
| --- |
| import seaborn as sns import matplotlib.pyplot as plt  data=pd.read\_csv('/content/drive/MyDrive/AI/dynamic\_pricing.csv') sns.scatterplot(data['Number\_of\_Drivers'])  plt.show() |

### 3. Multivariate Analysis:

Multivariate Analysis examines the relationships between multiple variables simultaneously, offering a comprehensive understanding of complex patterns within the dataset. It uncovers interactions and dependencies through techniques like regression, principal component analysis, and clustering, enabling deeper insights for decision-making.

Python code:

|  |
| --- |
| import seaborn as sns import matplotlib.pyplot as plt  data=pd.read\_csv('/content/drive/MyDrive/AI/dynamic\_pricing.csv') sns.pairplot(data)  plt.show() |

**Dynamic Pricing Model for Ride Duration Prediction Using Random Forest**

Python code:

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split, GridSearchCV

from sklearn.ensemble import RandomForestRegressor

from sklearn.impute import SimpleImputer

from sklearn.preprocessing import OneHotEncoder

from sklearn.compose import ColumnTransformer

from sklearn.pipeline import Pipeline

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error

# Load the dataset

data = pd.read\_csv('dynamic\_pricing (1).csv')

# Preprocess the dataset

data = data.dropna()

# Separate features and target variable

X = data.drop(columns=['Expected\_Ride\_Duration'])

y = data['Expected\_Ride\_Duration']

# Identify categorical and numerical columns

categorical\_features = X.select\_dtypes(include=['object']).columns numerical\_features = X.select\_dtypes(include=[np.number]).columns

# Create preprocessing pipelines for both numerical and categorical data numeric\_transformer = Pipeline(steps=[

('imputer', SimpleImputer(strategy='median'))])

categorical\_transformer = Pipeline(steps=[

('imputer', SimpleImputer(strategy='constant', fill\_value='missing')),

('onehot', OneHotEncoder(handle\_unknown='ignore'))])

# Combine preprocessing steps

preprocessor = ColumnTransformer( transformers=[

('num', numeric\_transformer, numerical\_features),

('cat', categorical\_transformer, categorical\_features)])

# Create the model pipeline

model = Pipeline(steps=[ ('preprocessor', preprocessor), ('regressor', RandomForestRegressor())])

# Define hyperparameters for tuning param\_grid = {

'regressor\_\_n\_estimators': [100, 200, 300],

'regressor\_\_max\_depth': [None, 10, 20, 30],

'regressor\_\_min\_samples\_split': [2, 5, 10]

}

# Split the dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Perform grid search

grid\_search = GridSearchCV(estimator=model, param\_grid=param\_grid, cv=5, scoring='neg\_mean\_squared\_error', n\_jobs=-1)

grid\_search.fit(X\_train, y\_train)

# Best model from grid search

best\_model = grid\_search.best\_estimator\_

# Predict on test data

y\_pred = best\_model.predict(X\_test)

# Evaluate the model

mae = mean\_absolute\_error(y\_test, y\_pred)

rmse = np.sqrt(mean\_squared\_error(y\_test, y\_pred))

total\_revenue = np.sum(y\_pred) # Simplified revenue calculation

revenue\_uplift = total\_revenue - np.sum(y\_test) # Simplified uplift calculation

# Print evaluation results

print(f"Mean Absolute Error (MAE): {mae}")

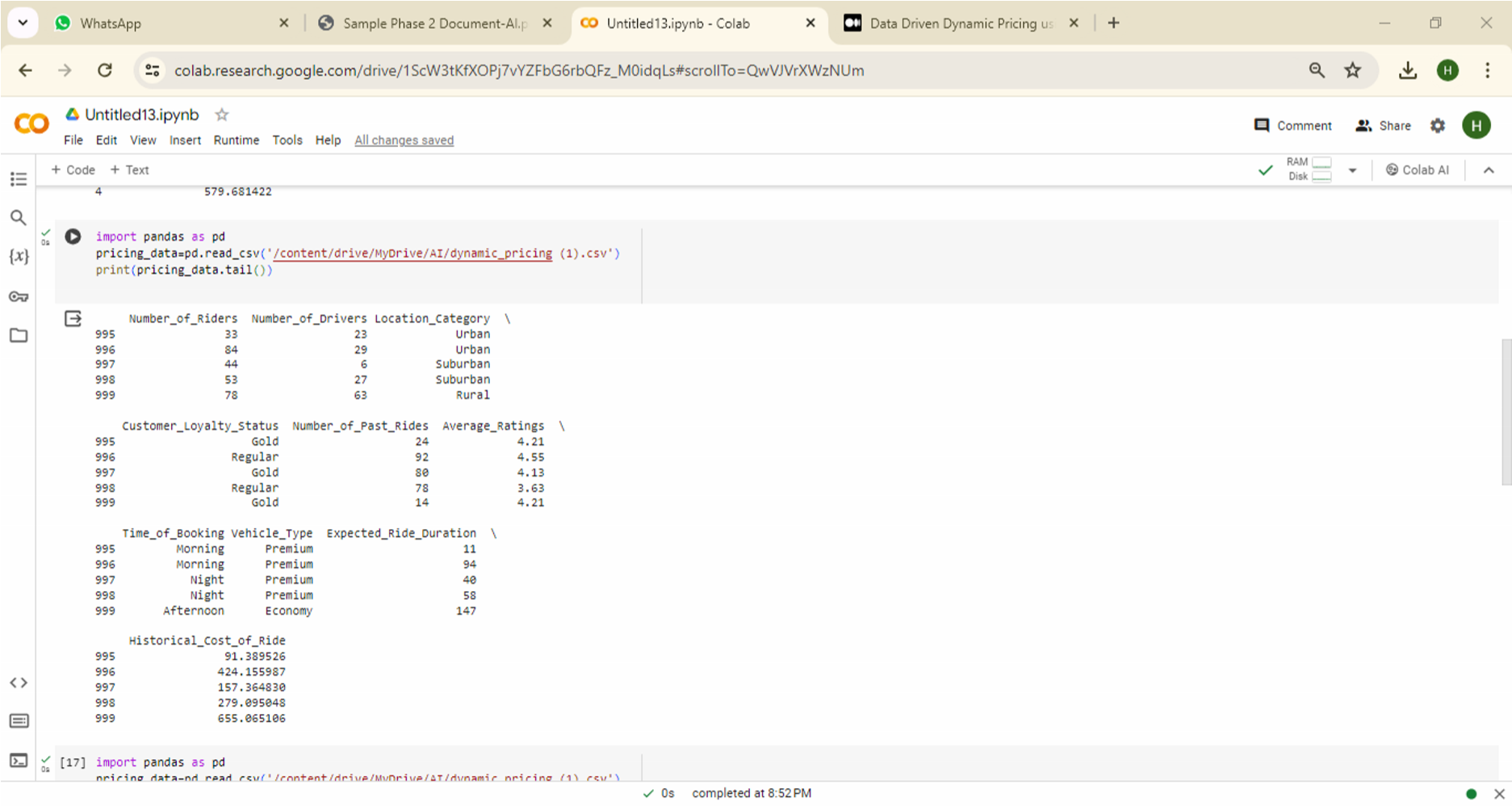
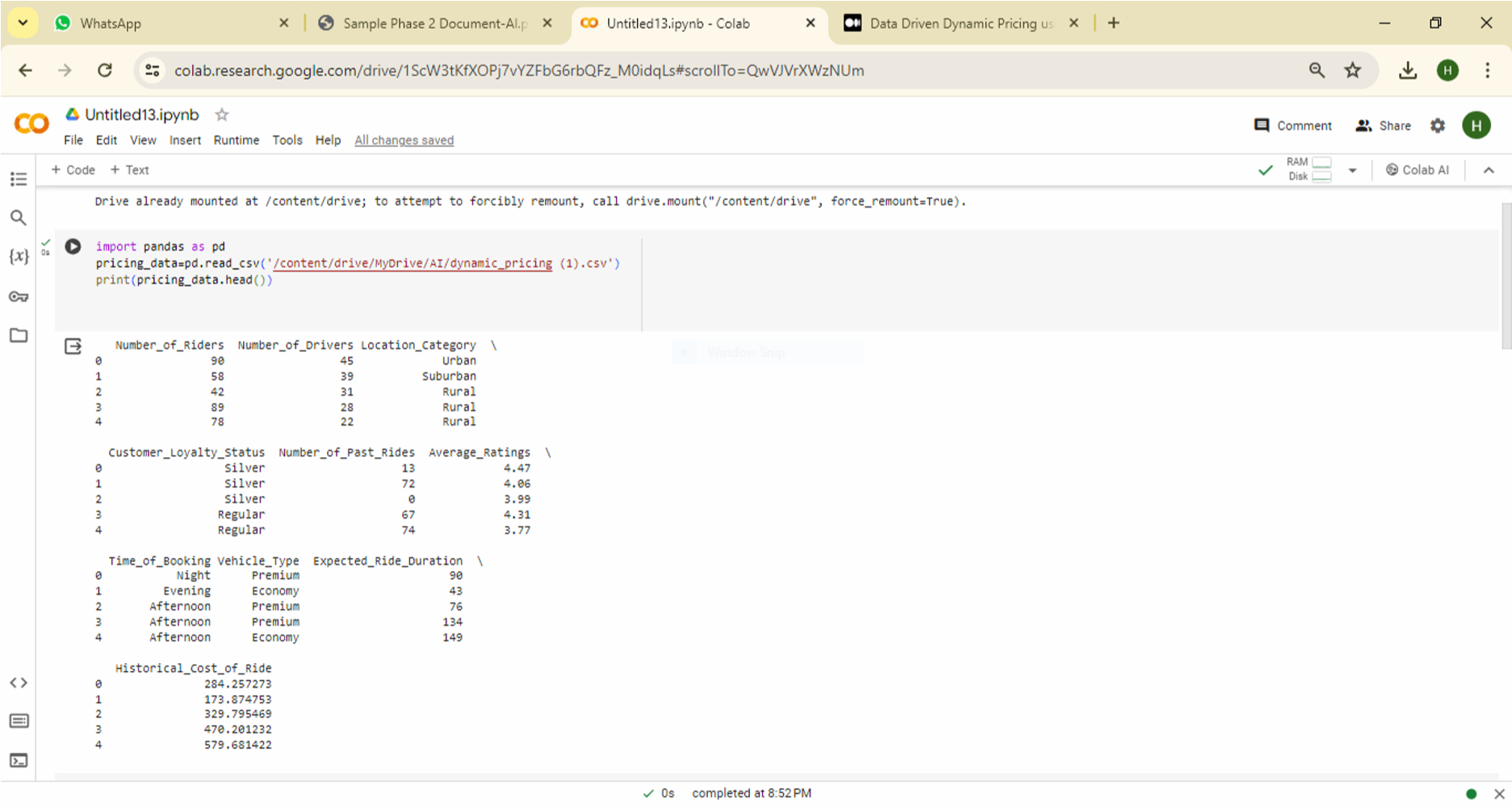
print(f"Root Mean Squared Error (RMSE): {rmse}")

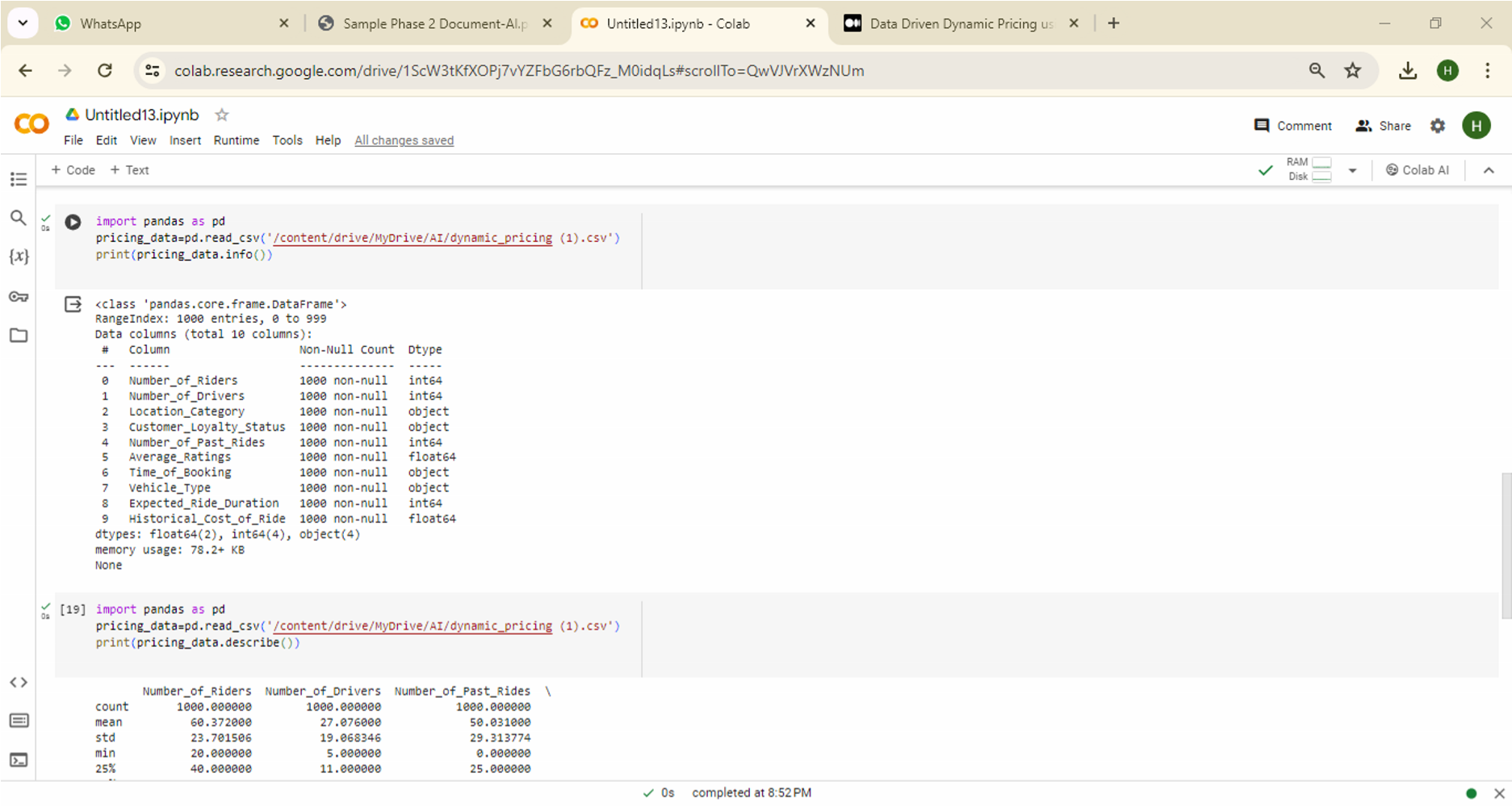
print(f"Total Revenue: {total\_revenue}")

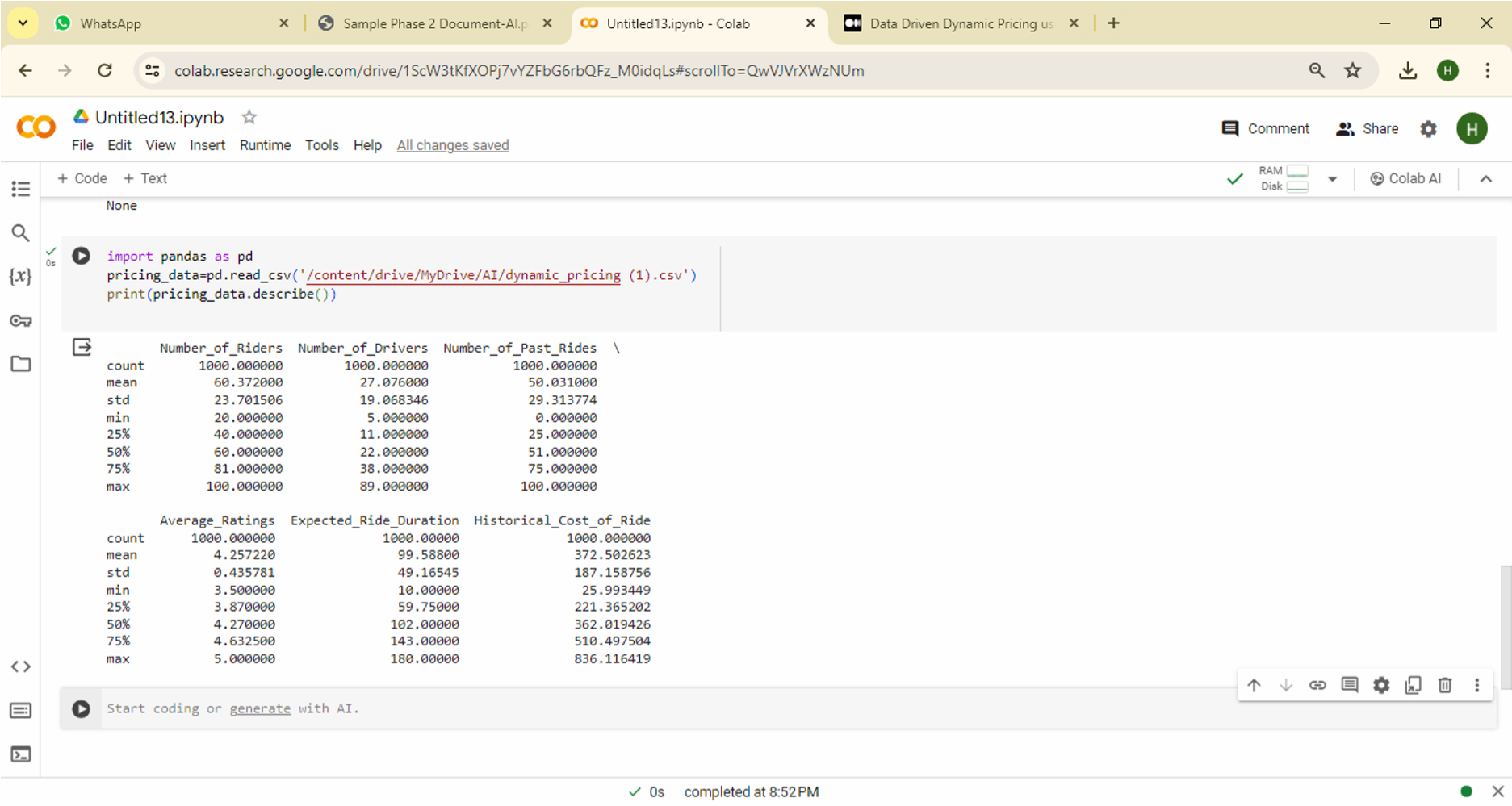
print(f"Revenue Uplift: {revenue\_uplift}")

**Appendix-2:** Visualizations and Graphs

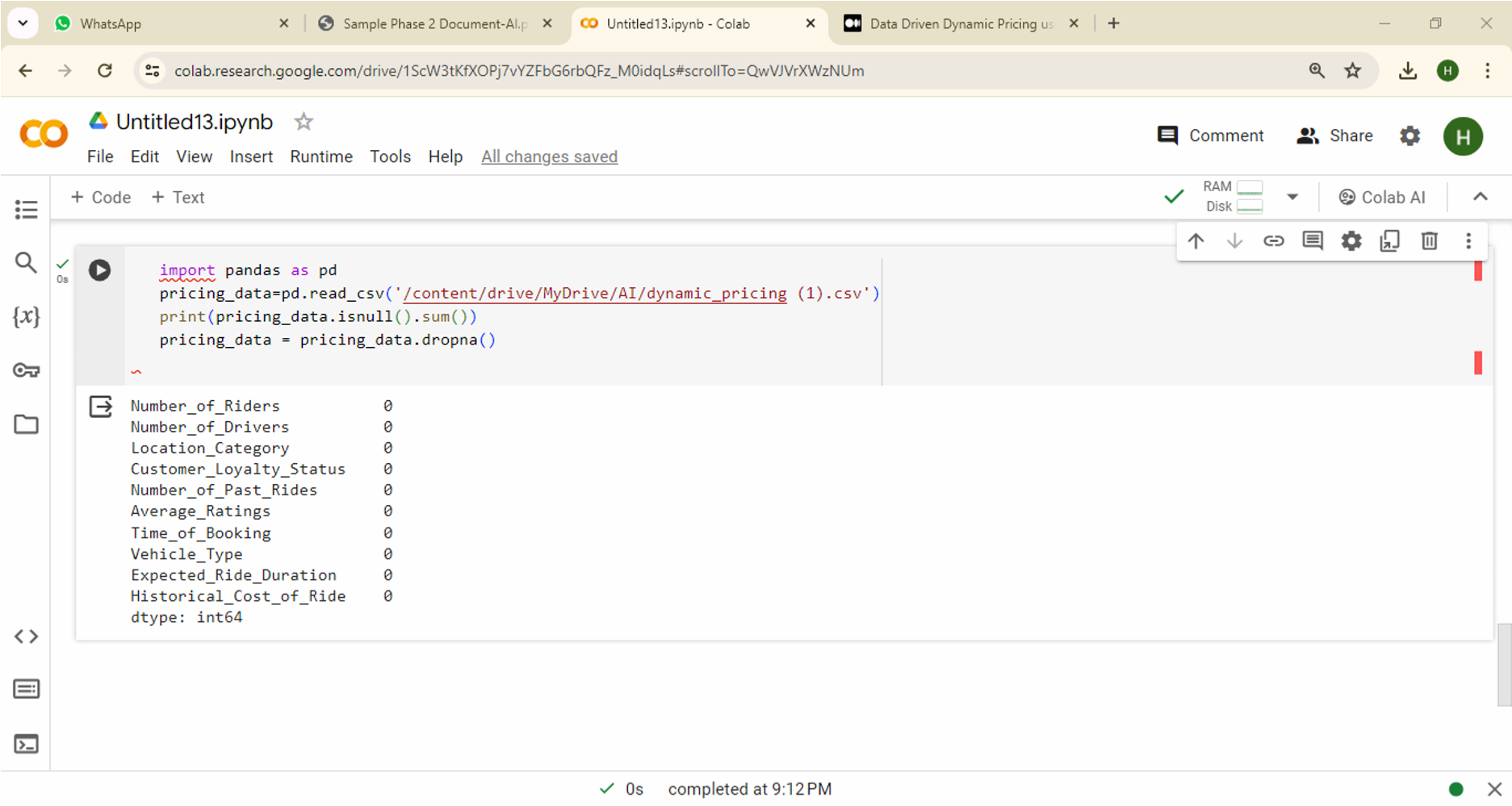
1. Output for Data Description:



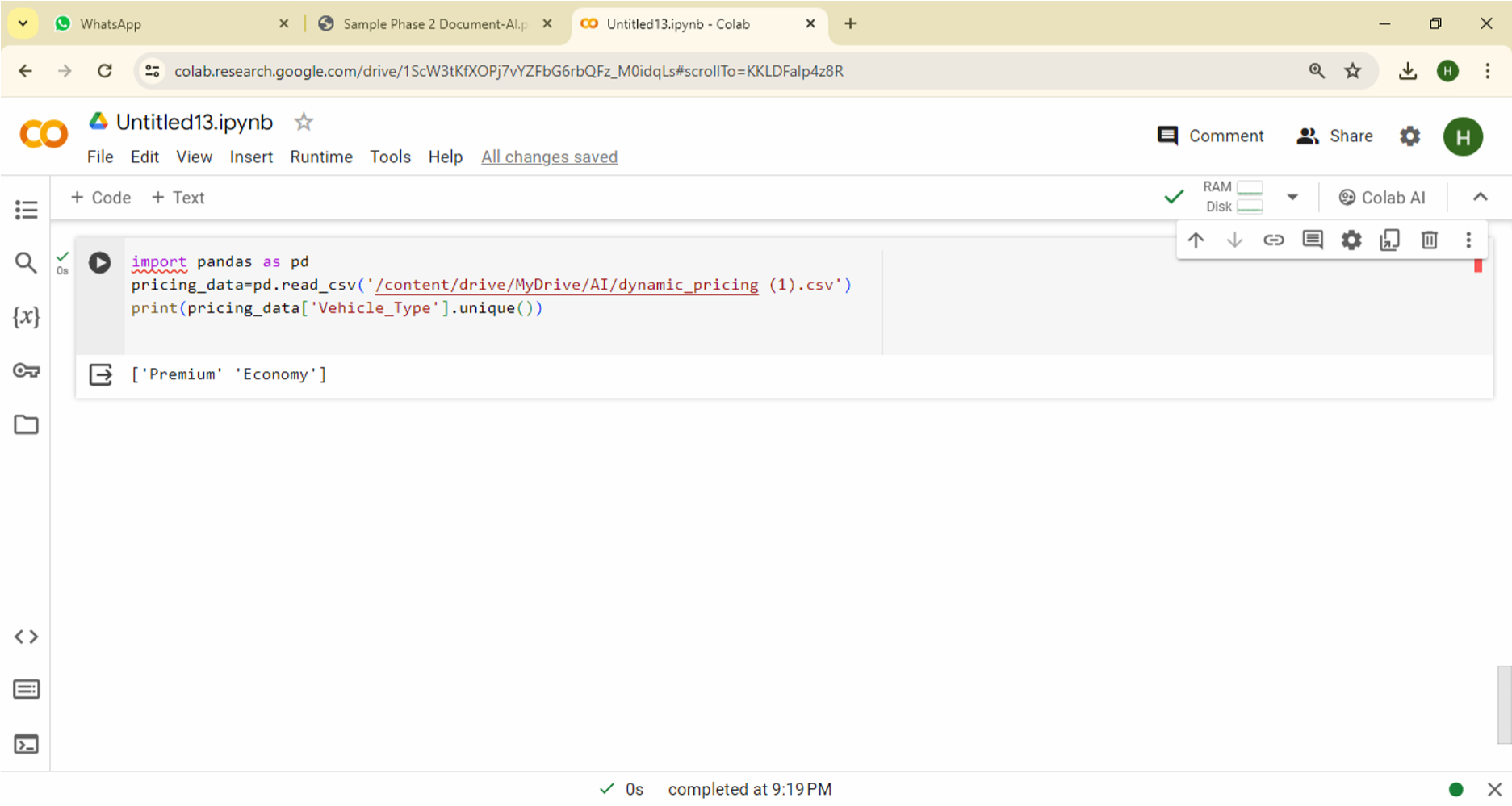




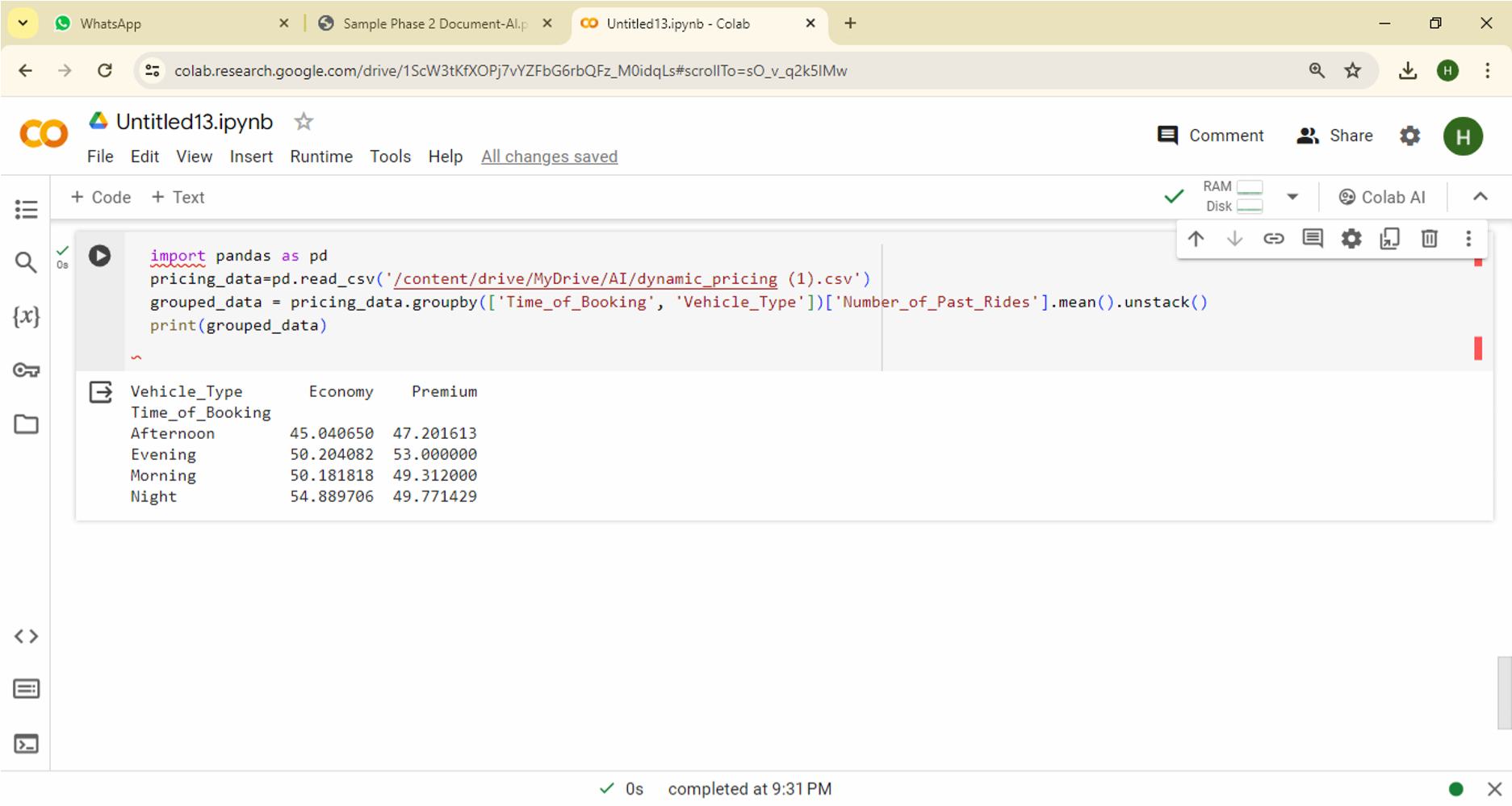
#### 2. Output for Null Data Handling:



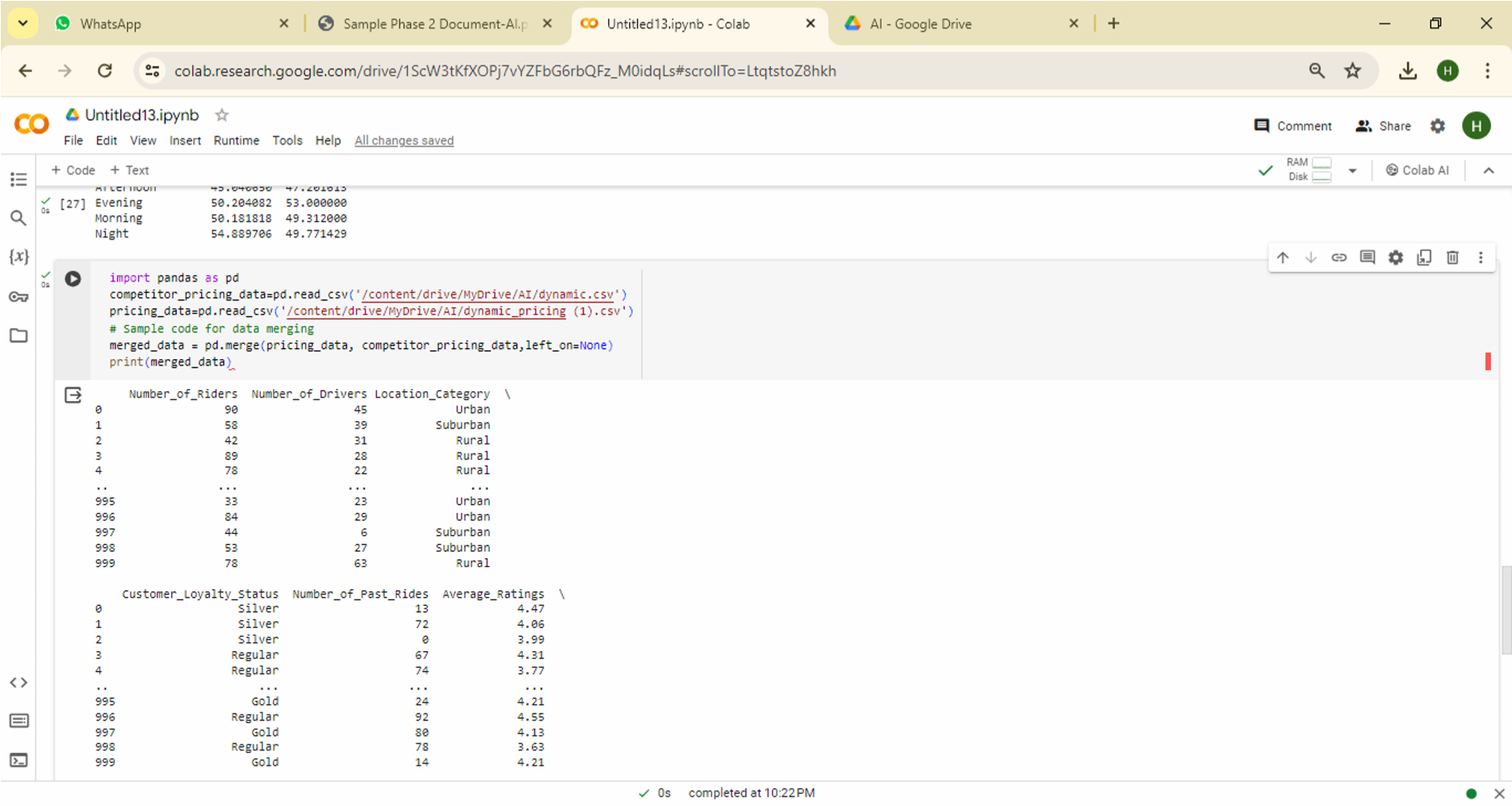
## 3. Output for Data Validation:

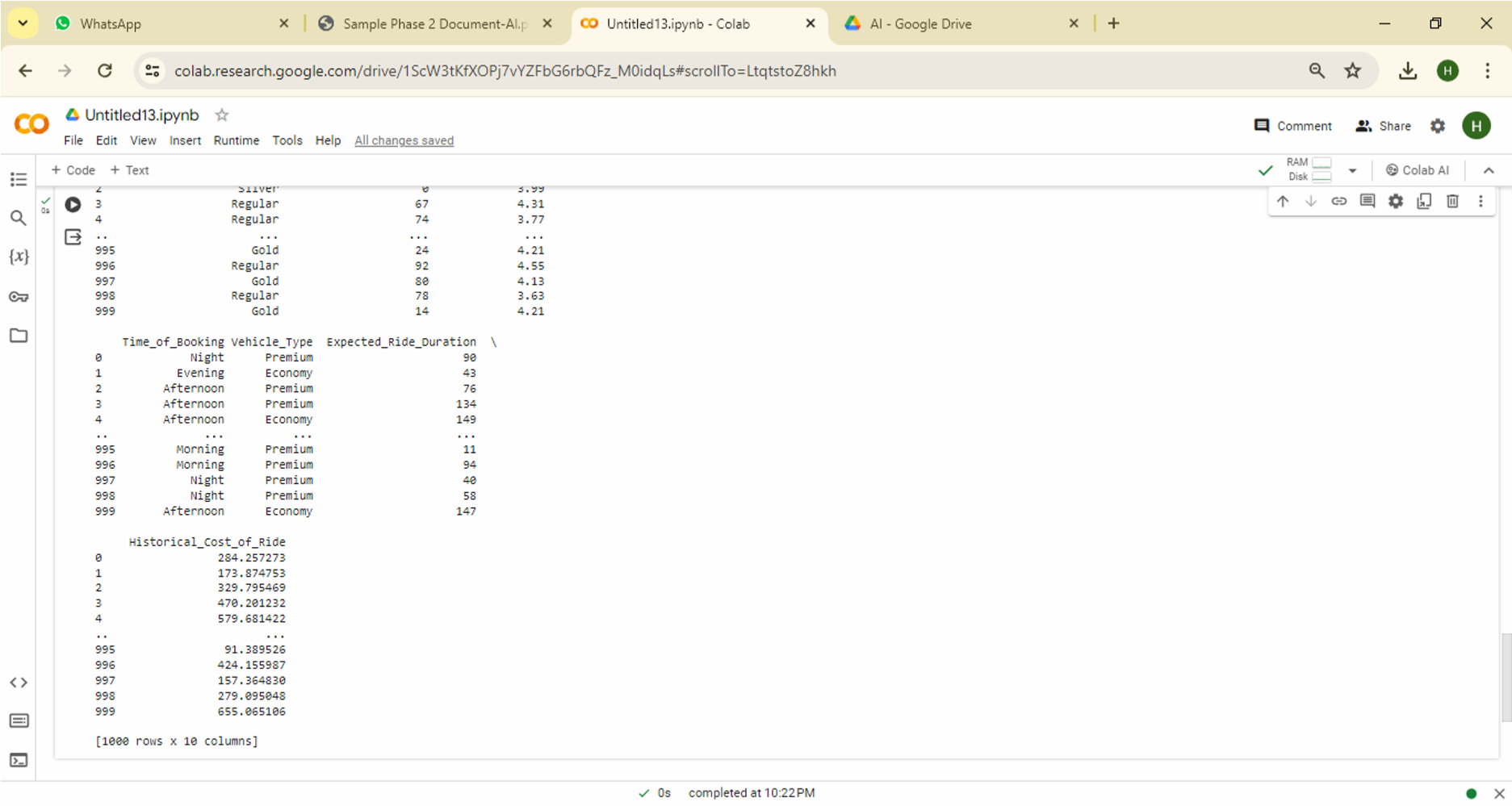


**4.** Output for **Data Reshaping:**

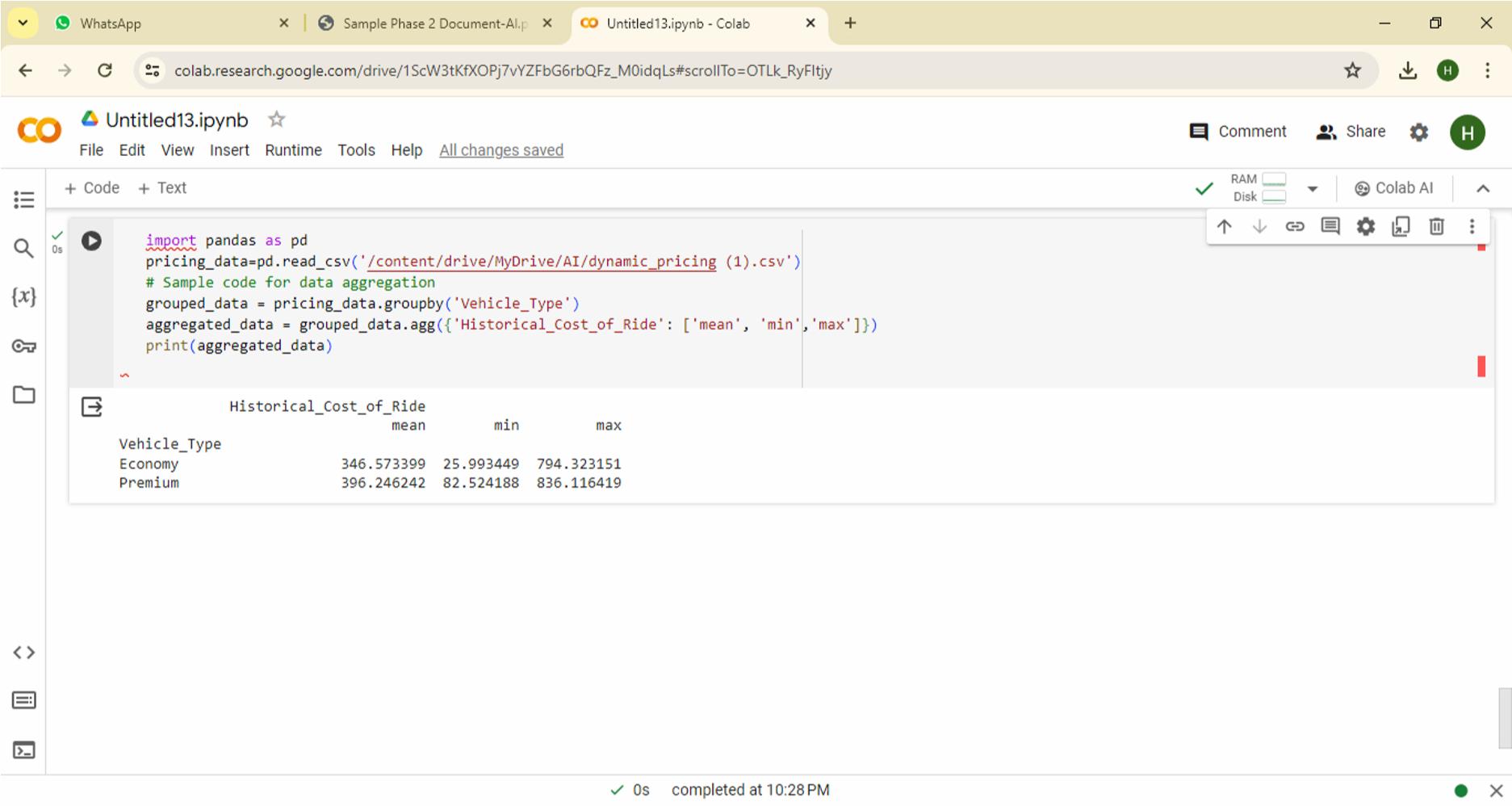


**5.** Output for **Data Merging:**



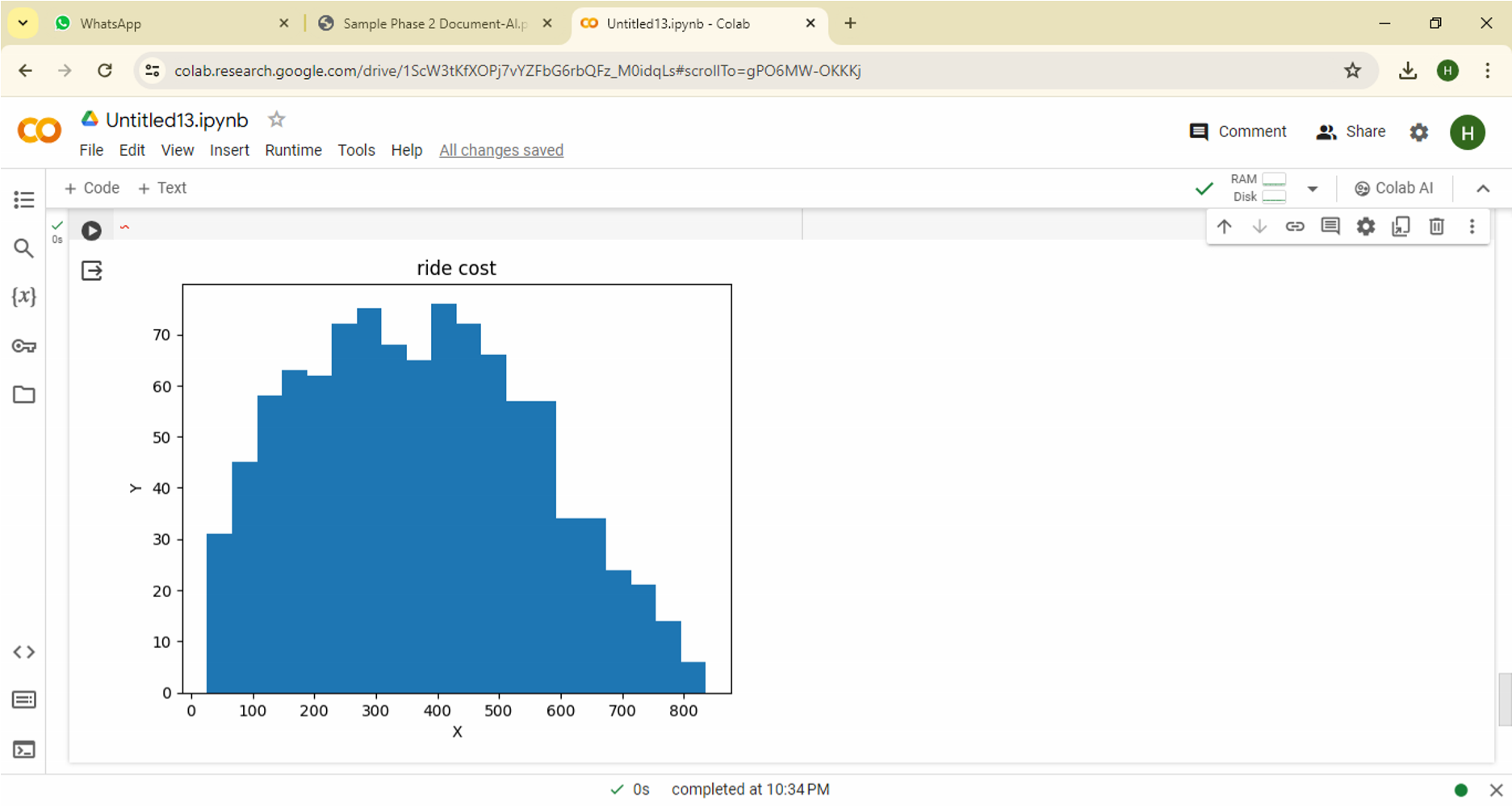


6. Output for Data Aggregation:

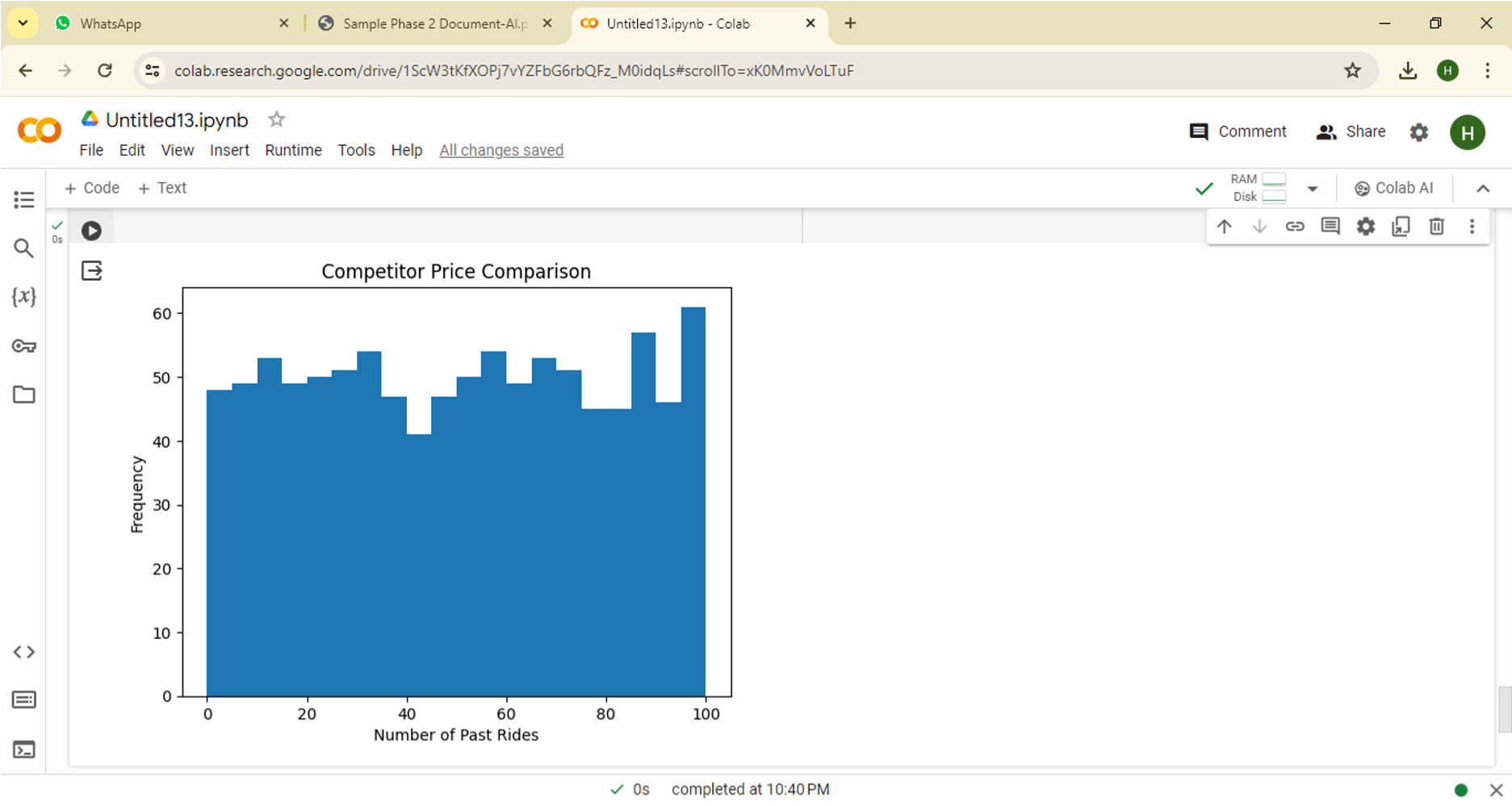


## Dynamic Pricing Analysis Techniques:

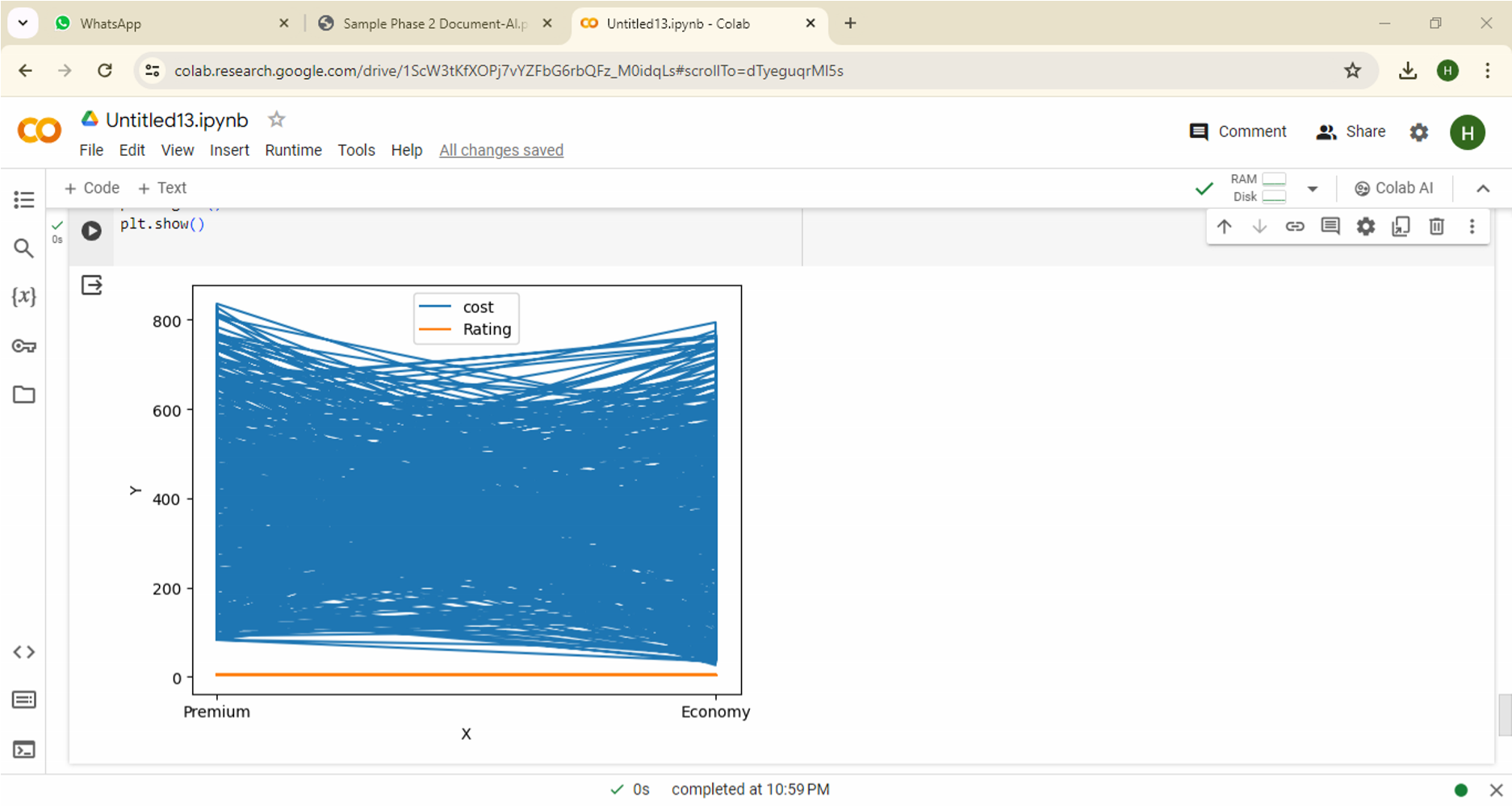
**1.** Output for **Price Distribution Analysis:**



**2.** Output for **Competitor Price Comparison:**

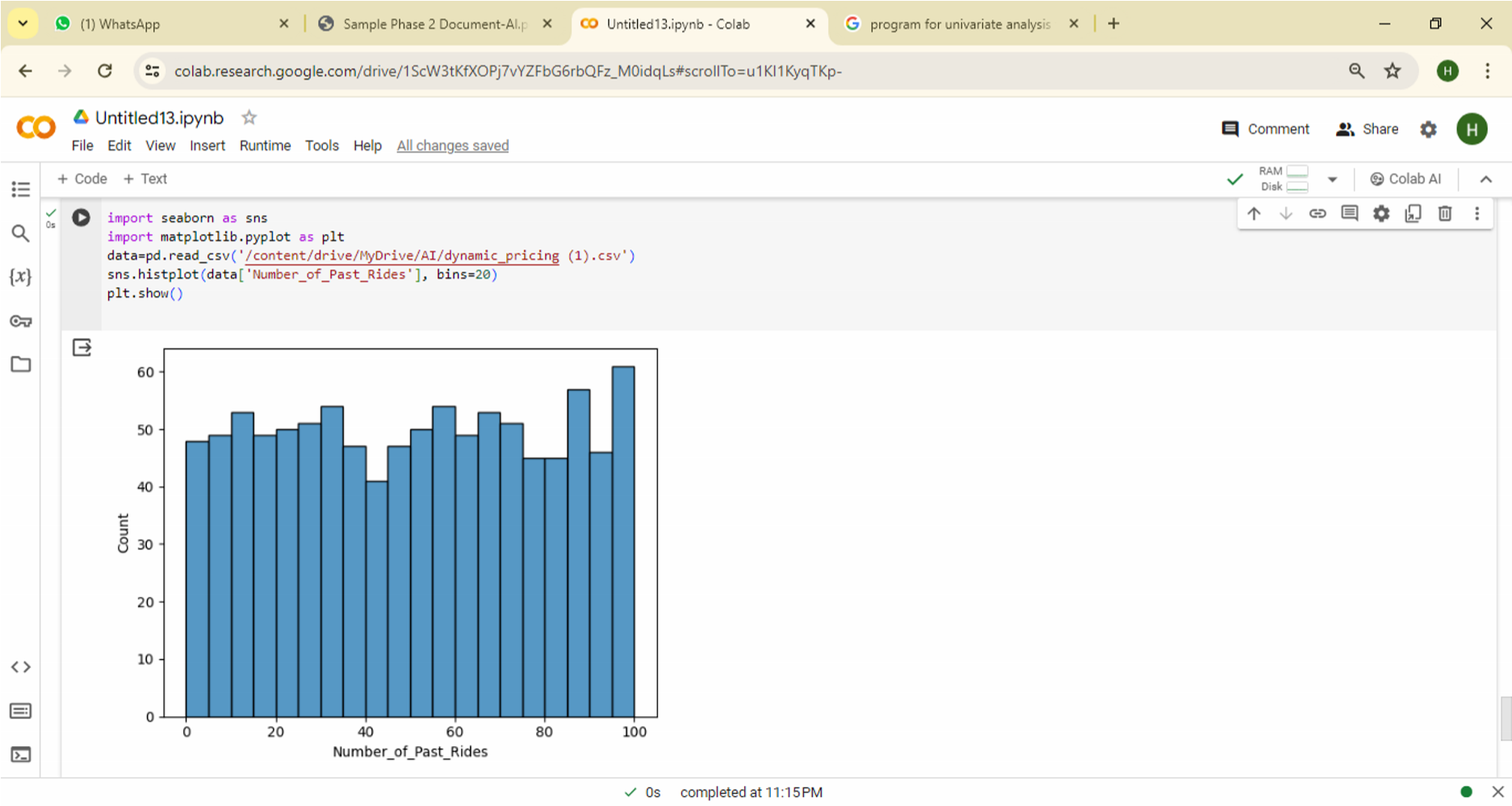


##### 3. Output for Dynamic Pricing Model Performance:

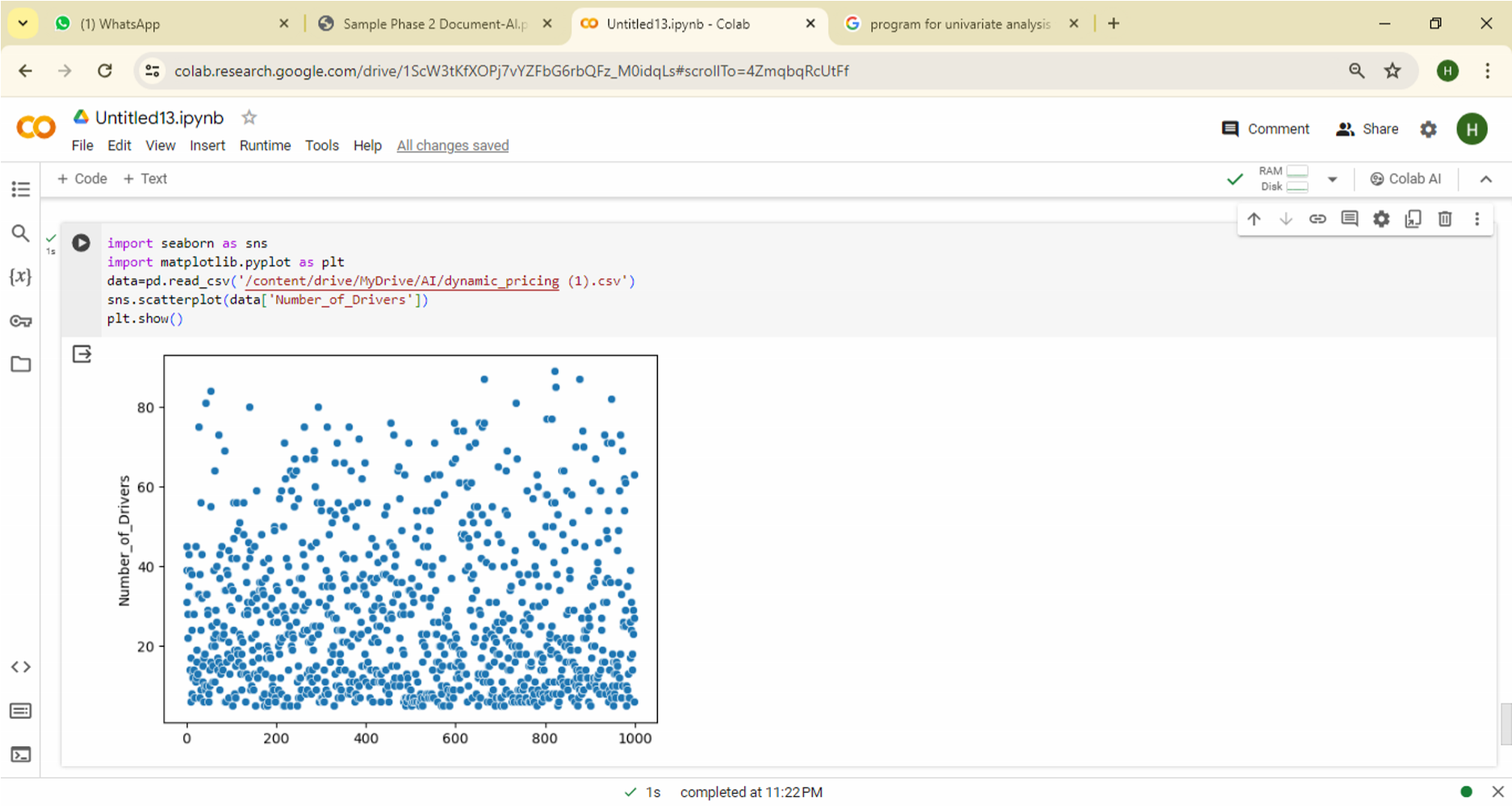


**Exploratory Data Analysis (EDA)**

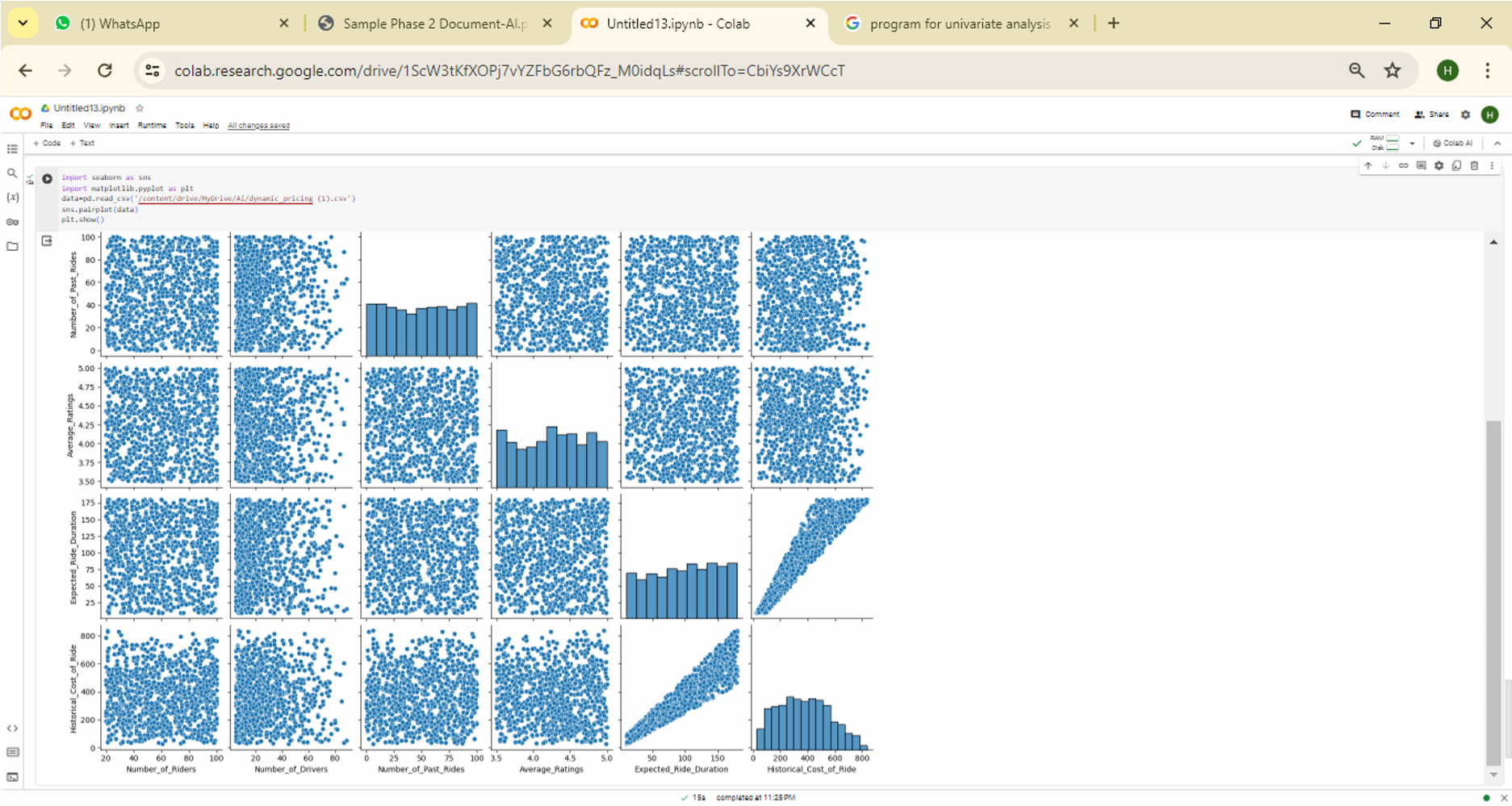
**1.** Output for **Univariate Analysis:**



**2.** Output for **Bivariate Analysis:**

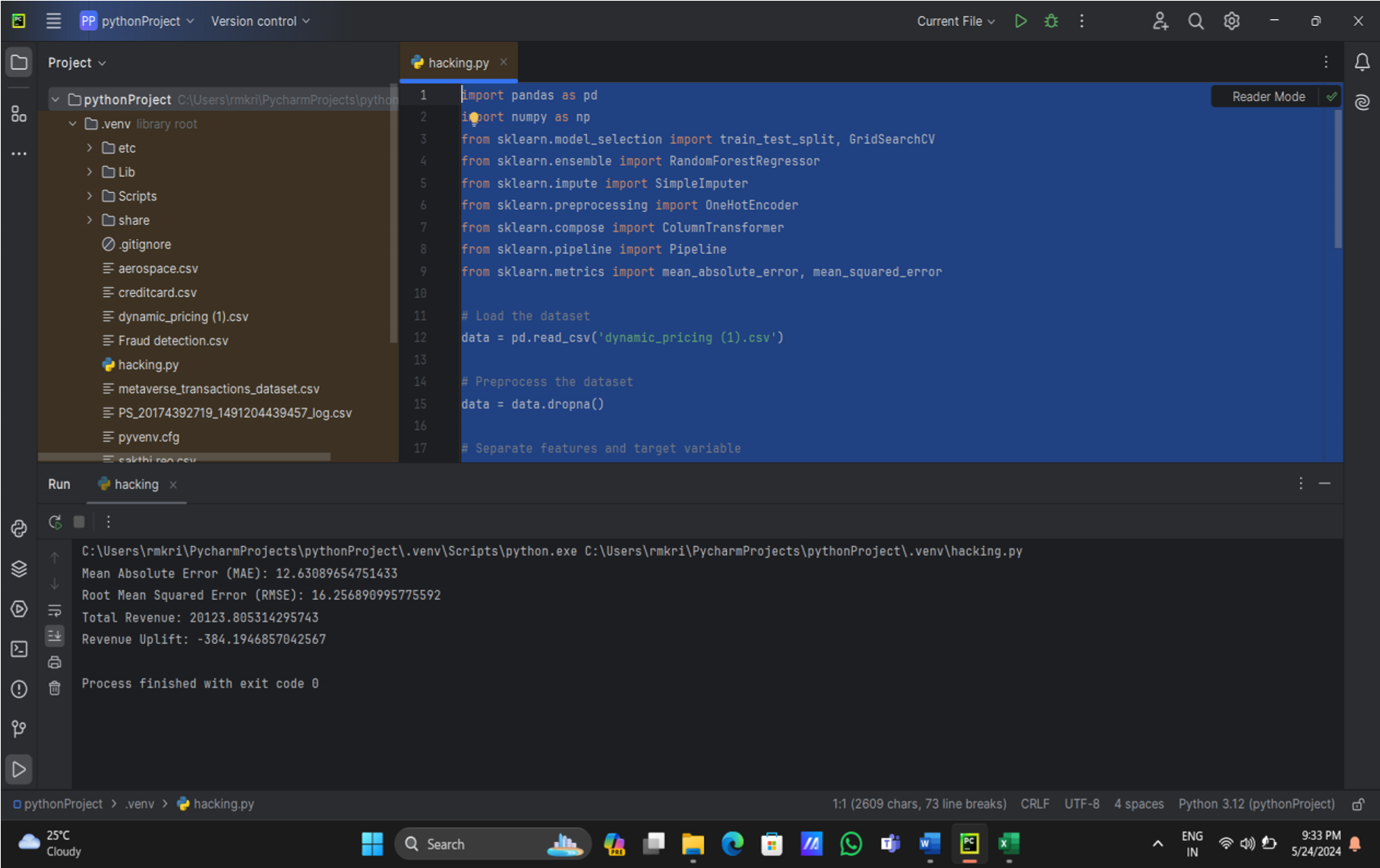


### 3. Output for Multivariate Analysis:



**Dynamic Pricing Model for Ride Duration Prediction Using Random Forest**

Output:



**REFERENCES**

### ****BOOKS:****

1. Talluri, K. T., & van Ryzin, G. J. (2006). The Theory and Practice of Revenue Management. Springer Science & Business Media. ISBN: 978-1441943491.

1. Phillips, R. (2021). Dynamic Pricing: Strategies and Techniques for E-Commerce. Routledge. ISBN: 978-1138232597.
2. Phillips, R. L. (2005). Pricing and Revenue Optimization. Stanford University Press. ISBN: 978-0804746984.
3. Sterne, J. (2017). Artificial Intelligence for Marketing: Practical Applications. Wiley. ISBN: 978-1119406335.
4. Provost, F., & Fawcett, T. (2013). Data Science for Business: What You Need to Know about Data Mining and Data-Analytic Thinking. O'Reilly Media. ISBN: 978-1449361327.
5. Jansen, S. (2020). Machine Learning for Algorithmic Trading. Packt Publishing. ISBN: 978-1789346411.

**WEB REFERENCES**

1. Pandas Documentation – Data Manipulation and Preprocessing  
   Retrieved from: <https://pandas.pydata.org/docs/>
2. NumPy Documentation – Numerical Computing  
   Retrieved from: <https://numpy.org/doc/>
3. Scikit-Learn Documentation – Machine Learning Models and Preprocessing  
   Retrieved from: <https://scikit-learn.org/stable/user_guide.html>
4. Random Forest Regressor – Scikit-Learn  
   Retrieved from: <https://scikitlearn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html>
5. Hyperparameter Tuning with GridSearchCV  
   Retrieved from: <https://scikitlearn.org/stable/modules/grid_search.html>
6. One-Hot Encoding in Scikit-Learn  
   Retrieved from: <https://scikitlearn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html>
7. Revenue Prediction Models in Machine Learning  
   Retrieved from: <https://towardsdatascience.com/predictingrevenue-with-machine-learning-3fd84b50e933>
8. Dynamic Pricing Dataset

Retrieved from:

<https://www.kaggle.com/datasets/arashnic/dynamic-pricing-dataset>

**OFFER LETTER**



**CERTIFICATE OF COMPLETION**