DYNAMIC PRICING

Phase 2 submission

College code: 9223

College Name: University College of Engineering, Dindigul.

Technology: AI

Total no of student in the group: 5

Student's details within the group:

-->Hema.B

-->Harini.Y.A

-->Rohithkumar.S

-->Ajay.M

-->Saranya Dheve.B

Submitted by,

Y.A.Harini

aut388617

Phase 2 Document: Data wrangling and Analysis on Dynamic pricing dataset

Introduction:

Dynamic pricing has revolutionized the way businesses approach pricing strategies by enabling real-time adjustments based on various factors. In today's fast-paced markets, static pricing models often fall short in capturing the complexities of demand fluctuations, competitor actions, and customer behavior. Dynamic pricing offers a solution by leveraging data analytics and technology to optimize prices continuously.

In this introduction, we'll explore the fundamentals of dynamic pricing and outline key considerations for implementing a dynamic pricing strategy effectively. From defining objectives to measuring performance, each step plays a crucial role in the success of a dynamic pricing project.

Objectives:

The provided task involves four main steps:

1. Data Cleansing: Identify and rectify inconsistencies, errors, and missing values in the dataset to ensure data integrity.

2. Exploratory Data Analysis (EDA): Investigate dataset characteristics through visualizations and statistical methods to understand distributions and correlations among variables.

3. Feature Engineering: Enhance model performance by creating new features or transforming existing ones based on insights gained from EDA.

4. Documentation: Document the data wrangling process comprehensively, ensuring transparency and reproducibility by recording steps taken, providing clear explanations, and promoting reliability through structured documentation.

These steps collectively contribute to preparing the dataset for accurate analysis and modeling while maintaining transparency and reproducibility in the process.

Dataset Description:

The dataset comprises historical pricing data, customer purchase history, market trends, and competitor pricing information.

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.

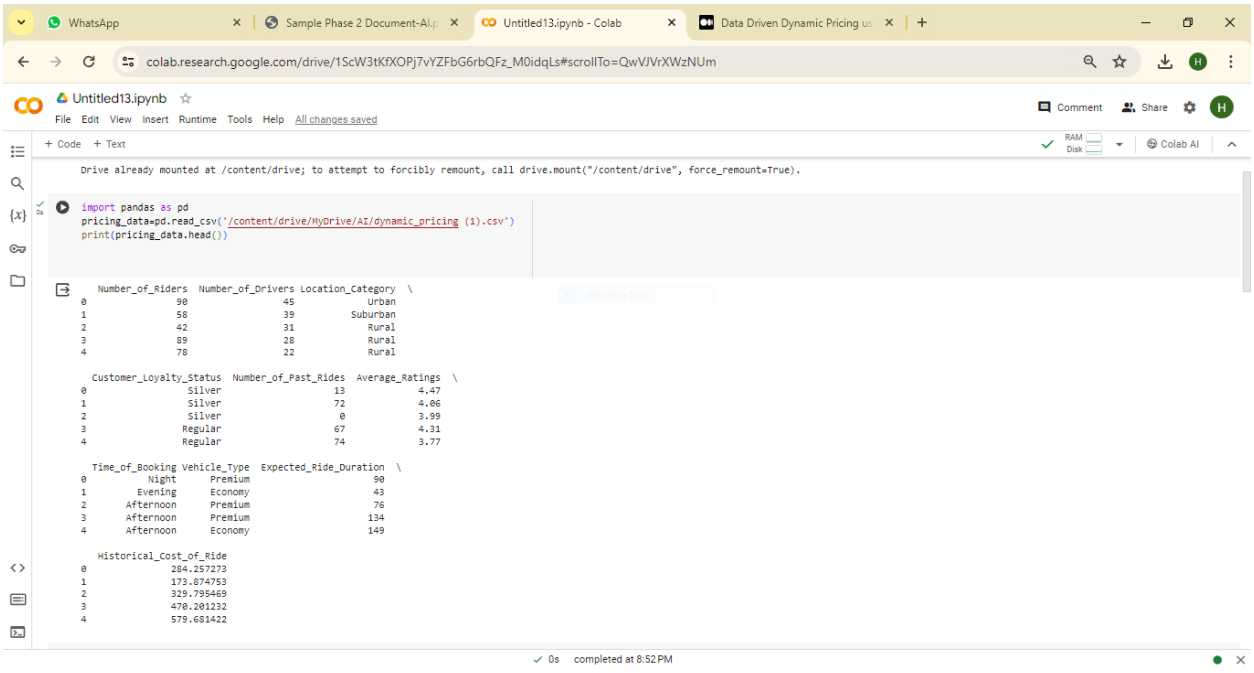
4. Describe: Generates descriptive statistics for numerical features, summarizing key statistical measures to understand data distribution and variability.

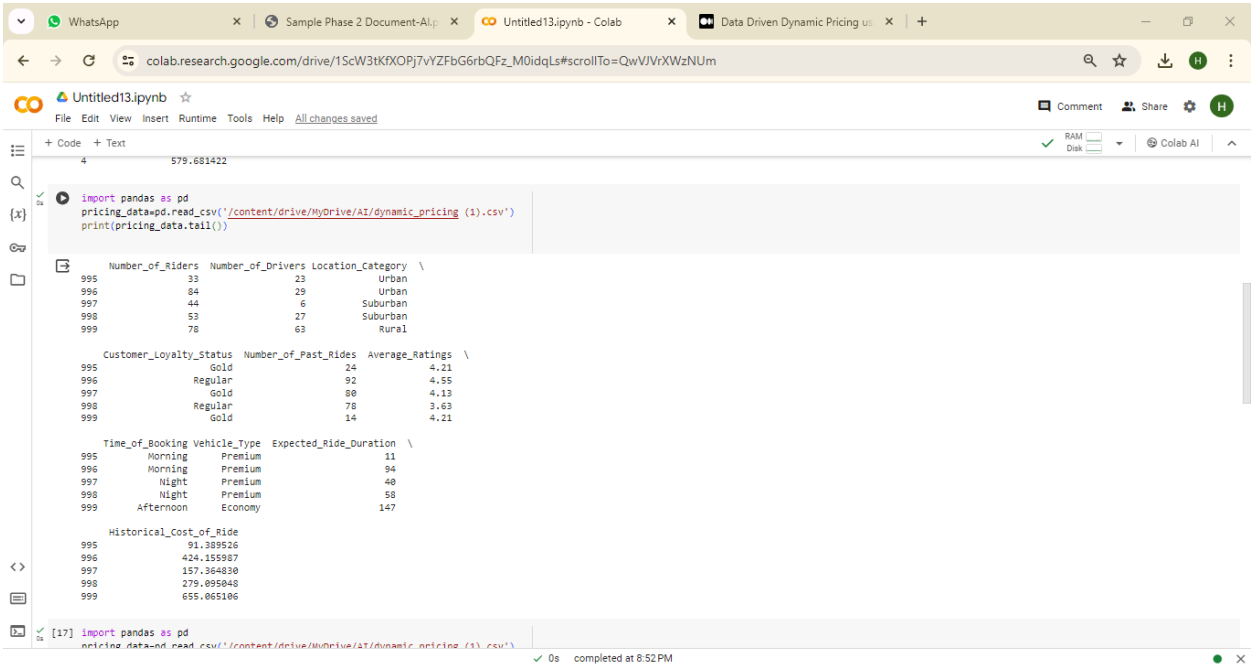
These methods collectively enable analysts to gain a thorough understanding of the dataset, facilitating further analysis and modeling.

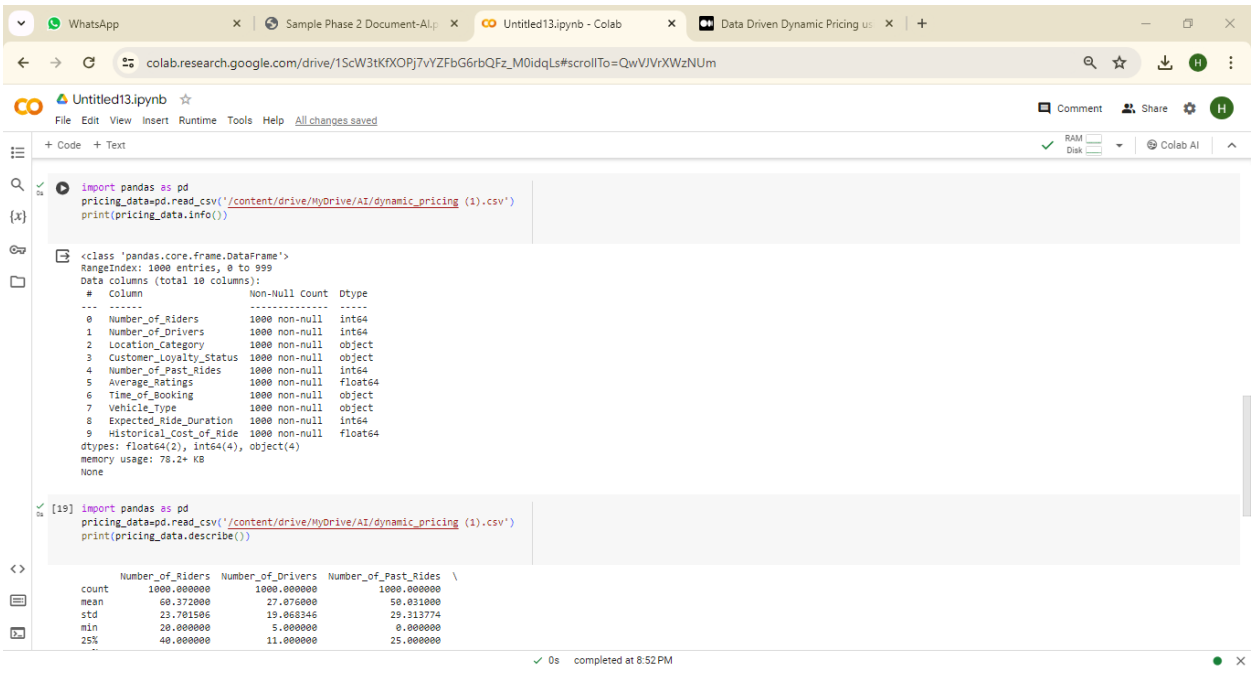
Python code:

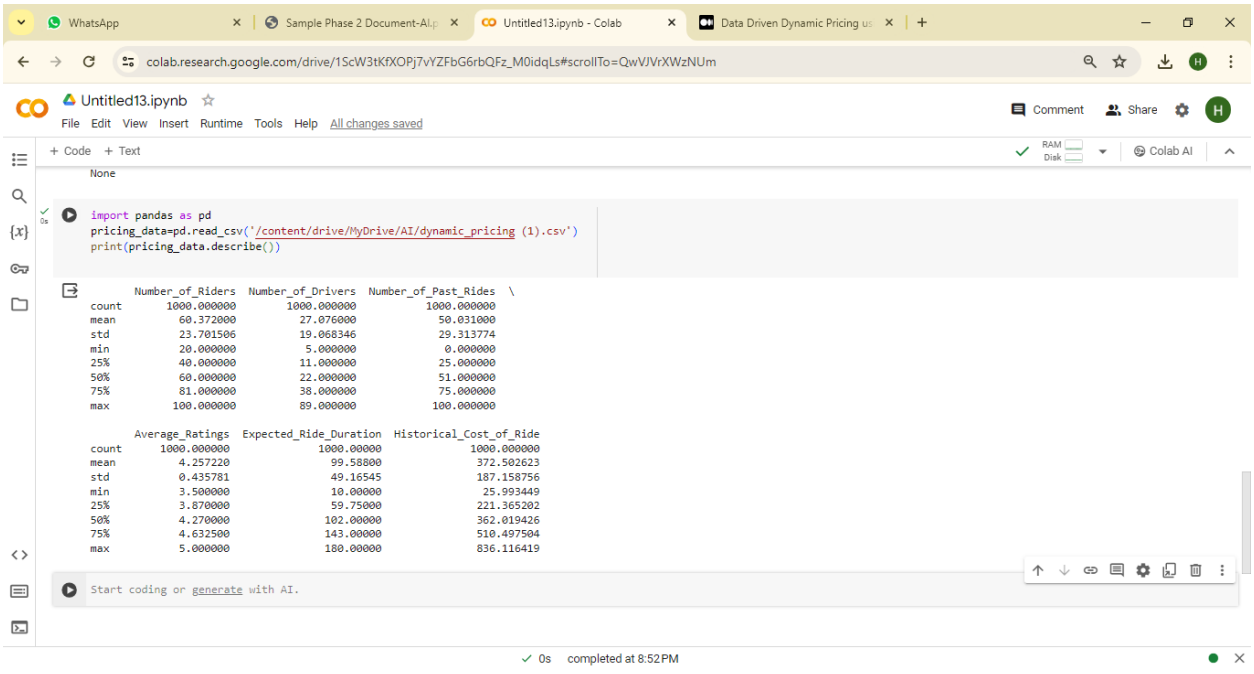
```
# Sample code for pricing data description

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())
```

Output:


WhatsApp
Sample Phase 2 Document-AI.p
Untitled13.ipynb - Colab
Data Driven Dynamic Pricing u
colab.research.google.com/drive/1ScW3tKfXOPj7vYZFbG6rbQFz_M0idqLs#scrollTo=QwVJVrXWzNUm
Untitled13.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Colab AI
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
import pandas as pd
pricing_data=pd.read_csv('/content/drive/MyDrive/AI/dynamic_pricing (1).csv')
print(pricing_data.head())
Number_of_Riders Number_of_Drivers Location_Category \
0 90 45 Urban
1 58 39 Suburban
2 42 31 Rural
3 89 28 Rural
4 78 22 Rural
Customer_Loyalty_Status Number_of_Past_Rides Average_Ratings \
0 Silver 13 4.47
1 Silver 72 4.06
2 Silver 0 3.99
3 Regular 67 4.31
4 Regular 74 3.77
Time_of_Booking Vehicle_Type Expected_Ride_Duration \
0 Night Premium 90
1 Evening Economy 43
2 Afternoon Premium 76
3 Afternoon Premium 134
4 Afternoon Economy 149
Historical_Cost_of_Ride
0 284.257273
1 173.874753
2 329.795469
3 470.201232
4 579.681422
0s completed at 8:52 PM


WhatsApp
Sample Phase 2 Document-AI.p
Untitled13.ipynb - Colab
Data Driven Dynamic Pricing u
colab.research.google.com/drive/1ScW3tKfXOPj7vYZFbG6rbQFz_M0idqLs#scrollTo=QwVJVrXWzNUm
Untitled13.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Colab AI
4 579.681422
import pandas as pd
pricing_data=pd.read_csv('/content/drive/MyDrive/AI/dynamic_pricing (1).csv')
print(pricing_data.tail())
Number_of_Riders Number_of_Drivers Location_Category \
995 33 23 Urban
996 84 29 Urban
997 44 6 Suburban
998 53 27 Suburban
999 78 63 Rural
Customer_Loyalty_Status Number_of_Past_Rides Average_Ratings \
995 Gold 24 4.21
996 Regular 92 4.55
997 Gold 80 4.13
998 Regular 78 3.63
999 Gold 14 4.21
Time_of_Booking Vehicle_Type Expected_Ride_Duration \
995 Morning Premium 11
996 Morning Premium 94
997 Night Premium 40
998 Night Premium 58
999 Afternoon Economy 147
Historical_Cost_of_Ride
995 91.389526
996 424.155987
997 157.364830
998 279.095048
999 655.065106
[17] import pandas as pd
0s completed at 8:52 PM


WhatsApp
Sample Phase 2 Document-AI.p
Untitled13.ipynb - Colab
Data Driven Dynamic Pricing us
colab.research.google.com/drive/1ScW3tKfXOPj7vYZFbG6rbQFz_M0idqLs#scrollTo=QwVJVrXWzNUm
Untitled13.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Colab AI
import pandas as pd
pricing_data=pd.read_csv('/content/drive/MyDrive/AI/dynamic_pricing (1).csv')
print(pricing_data.info())
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 10 columns):
Column Non-Null Count Dtype
--- ------ -------------- -----
0 Number_of_Riders 1000 non-null int64
1 Number_of_Drivers 1000 non-null int64
2 Location_Category 1000 non-null object
3 Customer_Loyalty_Status 1000 non-null object
4 Number_of_Past_Rides 1000 non-null int64
5 Average_Ratings 1000 non-null float64
6 Time_of_Booking 1000 non-null object
7 Vehicle_Type 1000 non-null object
8 Expected_Ride_Duration 1000 non-null int64
9 Historical_Cost_of_Ride 1000 non-null float64
dtypes: float64(2), int64(4), object(4)
memory usage: 78.2+ KB
None
[19] import pandas as pd
pricing_data=pd.read_csv('/content/drive/MyDrive/AI/dynamic_pricing (1).csv')
print(pricing_data.describe())
Number_of_Riders Number_of_Drivers Number_of_Past_Rides \
count 1000.000000 1000.000000 1000.000000
mean 60.372000 27.076000 50.031000
std 23.701506 19.068346 29.313774
min 20.000000 5.000000 0.000000
25% 40.000000 11.000000 25.000000
0s completed at 8:52 PM


WhatsApp
Sample Phase 2 Document-AI.p
Untitled13.ipynb - Colab
Data Driven Dynamic Pricing us
colab.research.google.com/drive/1ScW3tKfXOPj7vYZFbG6rbQFz_M0idqLs#scrollTo=QwVJVrXWzNUm
Untitled13.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Colab AI
None
import pandas as pd
pricing_data=pd.read_csv('/content/drive/MyDrive/AI/dynamic_pricing (1).csv')
print(pricing_data.describe())
Number_of_Riders Number_of_Drivers Number_of_Past_Rides \
count 1000.000000 1000.000000 1000.000000
mean 60.372000 27.076000 50.031000
std 23.701506 19.068346 29.313774
min 20.000000 5.000000 0.000000
25% 40.000000 11.000000 25.000000
50% 60.000000 22.000000 51.000000
75% 81.000000 38.000000 75.000000
max 100.000000 89.000000 100.000000
Average_Ratings Expected_Ride_Duration Historical_Cost_of_Ride
count 1000.000000 1000.00000 1000.000000
mean 4.257220 99.58800 372.502623
std 0.435781 49.16545 187.158756
min 3.500000 10.00000 25.993449
25% 3.870000 59.75000 221.365202
50% 4.270000 102.00000 362.019426
75% 4.632500 143.00000 510.497504
max 5.000000 180.00000 836.116419
Start coding or generate with AI.
0s completed at 8:52 PM

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.

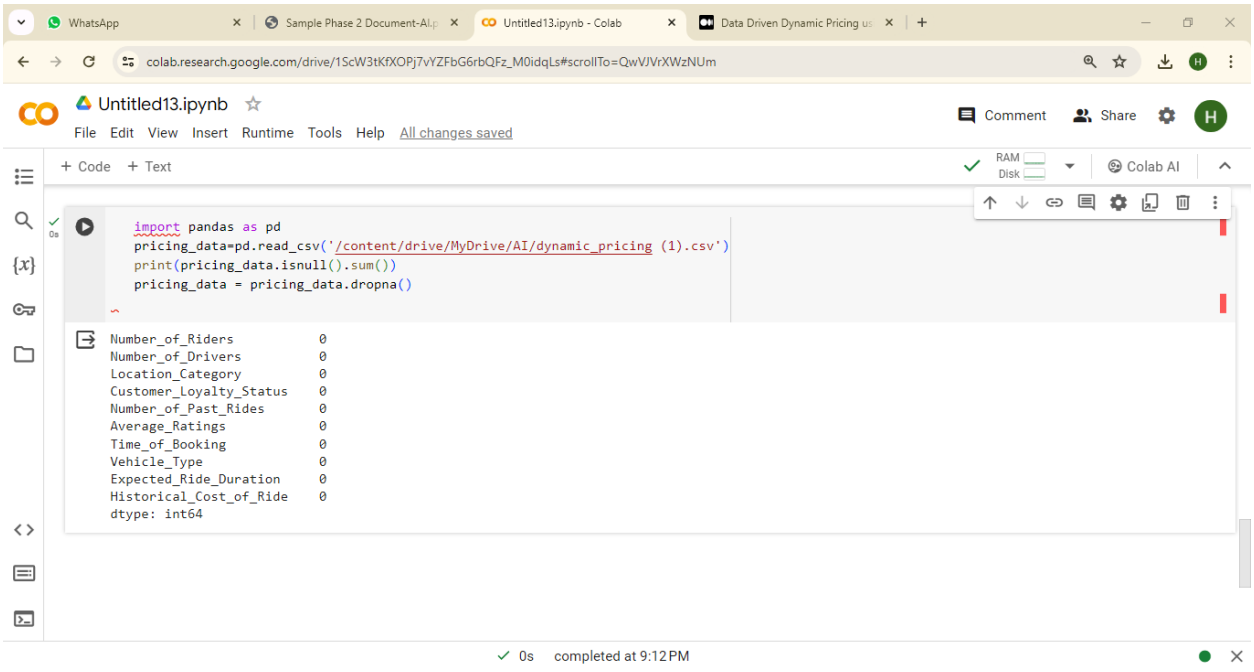
3. Null Data Removal: Eliminate rows or columns with excessive missing values to maintain data quality and integrity.

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

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
```

Output:



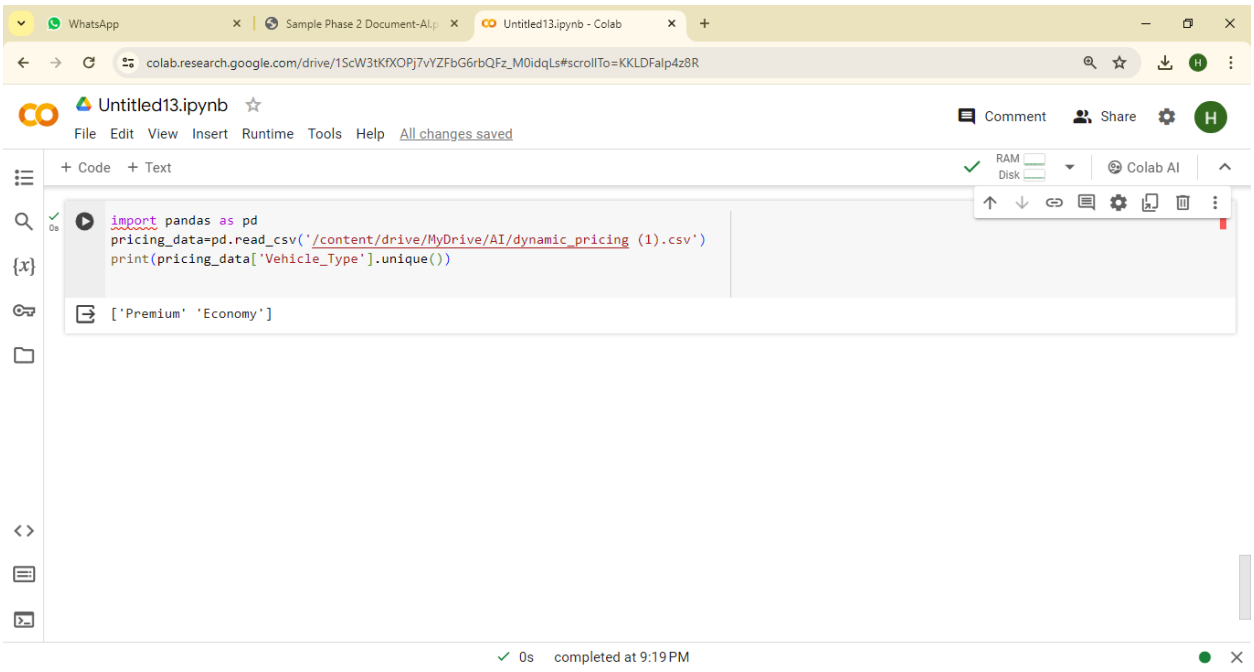
3. Data Validation:

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 modeling 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())
```

Output:



4. Data Reshaping:

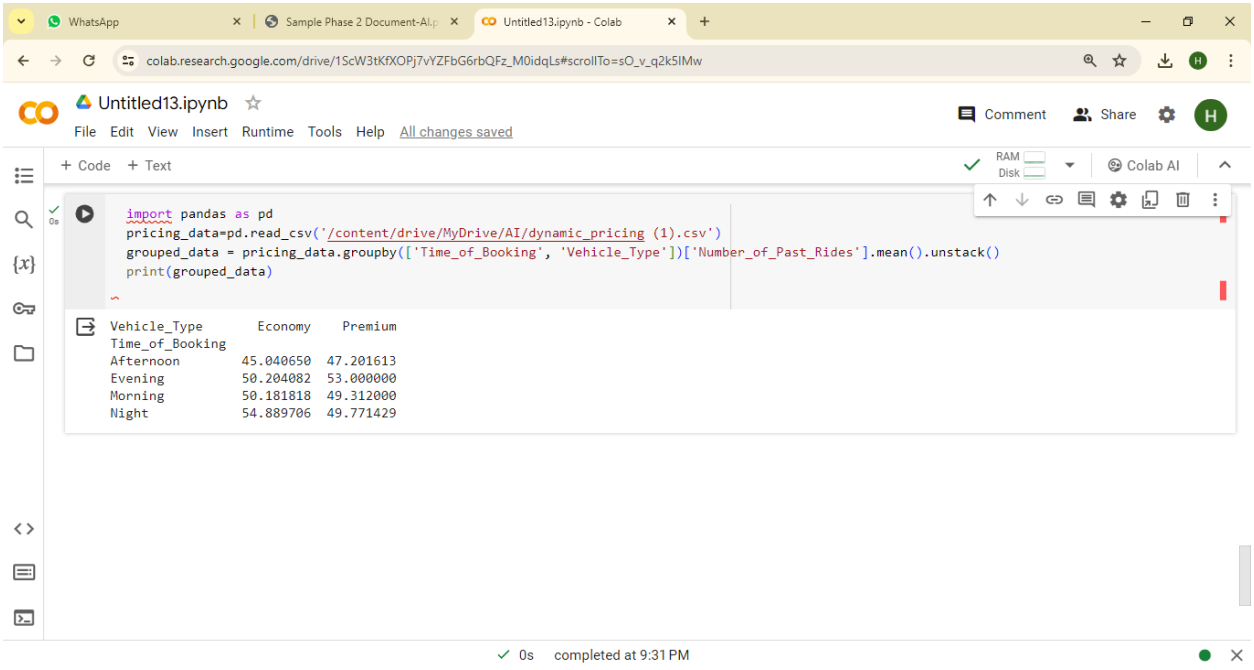
Reshaping Rows and Columns:

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. By reshaping rows and columns, analysts can effectively manipulate the data to extract valuable insights and patterns, ultimately aiding in decision-making processes and strategy formulation.

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)
```

Output:



5. Data Merging:

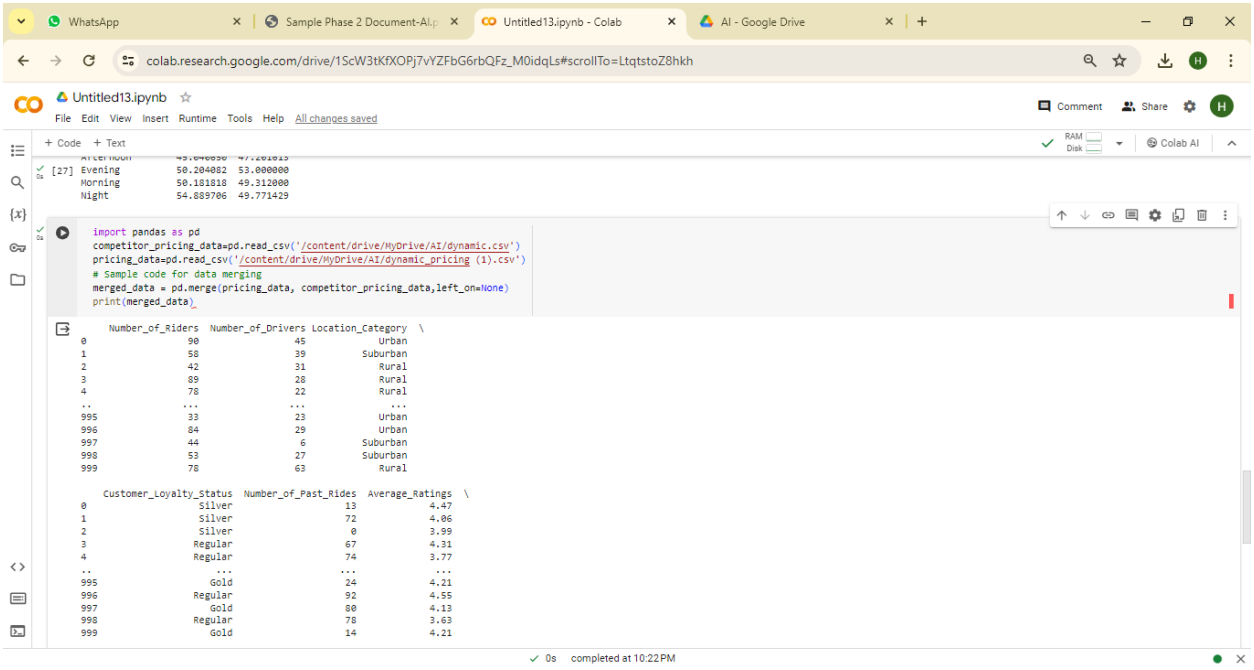
Combining Datasets:

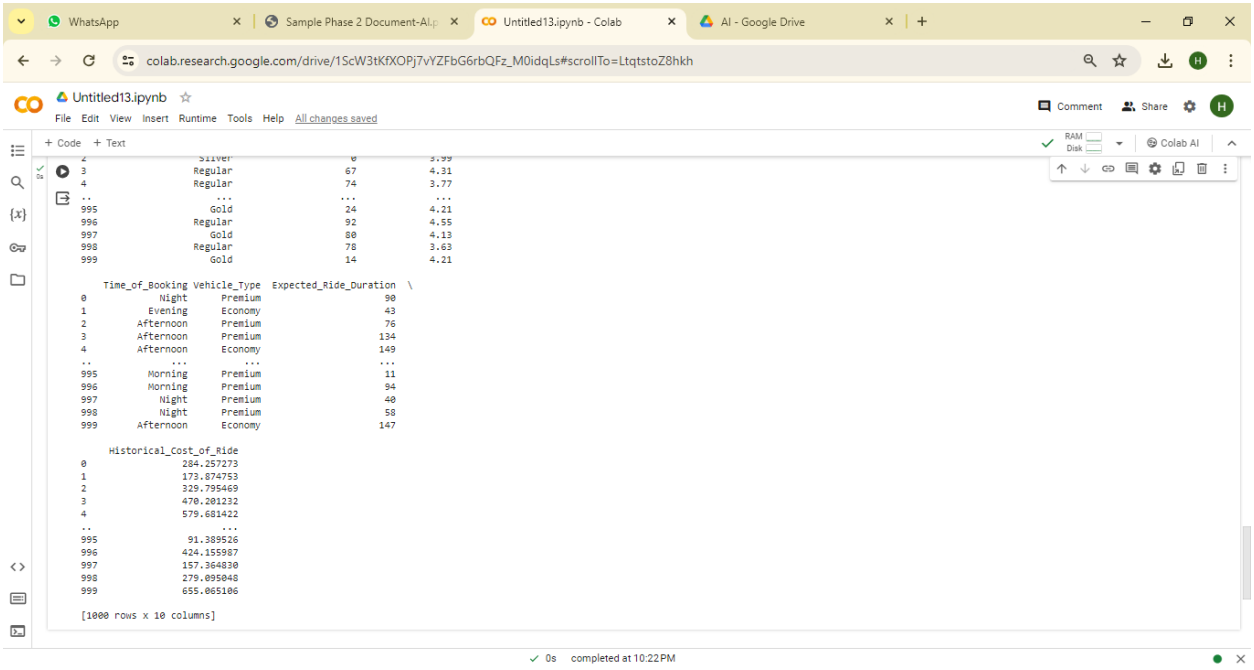
The instruction underscores the importance of merging multiple datasets or data sources to enrich pricing analysis. By combining diverse sources of data, analysts can create a unified dataset that offers a comprehensive view of pricing-related factors, facilitating deeper insights into market dynamics and consumer behavior. This integrated approach enhances decision-making and strategy development in pricing initiatives, ultimately leading to more informed and effective outcomes.

Python code:

```
import pandas as pd

competitor_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)
```

Output:


WhatsApp
Sample Phase 2 Document-AI.p
Untitled13.ipynb - Colab
AI - Google Drive
colab.research.google.com/drive/1ScW3tKfXOPj7vYZFbG6rbQFz_M0idqLs#scrollTo=LtqtstoZ8hkh
Untitled13.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Colab AI
[27] Evening 50.204082 53.000000
Morning 50.181818 49.312000
Night 54.889706 49.771429
import pandas as pd
competitor_pricing_data=pd.read_csv('/content/drive/MyDrive/AI/dynamic.csv')
pricing_data=pd.read_csv('/content/drive/MyDrive/AI/dynamic_pricing (1).csv')
# Sample code for data merging
merged_data = pd.merge(pricing_data, competitor_pricing_data,left_on=None)
print(merged_data)
Number_of_Riders Number_of_Drivers Location_Category \
0 90 45 Urban
1 58 39 Suburban
2 42 31 Rural
3 89 28 Rural
4 78 22 Rural
.. ..
995 33 23 Urban
996 84 29 Urban
997 44 6 Suburban
998 53 27 Suburban
999 78 63 Rural
Customer_Loyalty_Status Number_of_Past_Rides Average_Ratings \
0 Silver 13 4.47
1 Silver 72 4.06
2 Silver 0 3.99
3 Regular 67 4.31
4 Regular 74 3.77
.. ..
995 Gold 24 4.21
996 Regular 92 4.55
997 Gold 80 4.13
998 Regular 78 3.63
999 Gold 14 4.21
0s completed at 10:22PM


WhatsApp
Sample Phase 2 Document-AI.p
Untitled13.ipynb - Colab
AI - Google Drive
colab.research.google.com/drive/1ScW3tKfXOPj7vYZFbG6rbQFz_M0idqLs#scrollTo=LtqtstoZ8hkh
Untitled13.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Colab AI
3 Regular 67 4.31
4 Regular 74 3.77
.. ..
995 Gold 24 4.21
996 Regular 92 4.55
997 Gold 80 4.13
998 Regular 78 3.63
999 Gold 14 4.21
Time_of_Booking Vehicle_Type Expected_Ride_Duration \
0 Night Premium 90
1 Evening Economy 43
2 Afternoon Premium 76
3 Afternoon Premium 134
4 Afternoon Economy 149
.. ..
995 Morning Premium 11
996 Morning Premium 94
997 Night Premium 40
998 Night Premium 58
999 Afternoon Economy 147
Historical_Cost_of_Ride
0 284.257273
1 173.874753
2 329.795469
3 470.201232
4 579.681422
.. ...
995 91.389526
996 424.155987
997 157.364830
998 279.095048
999 655.065106
[1000 rows x 10 columns]
0s completed at 10:22PM

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.

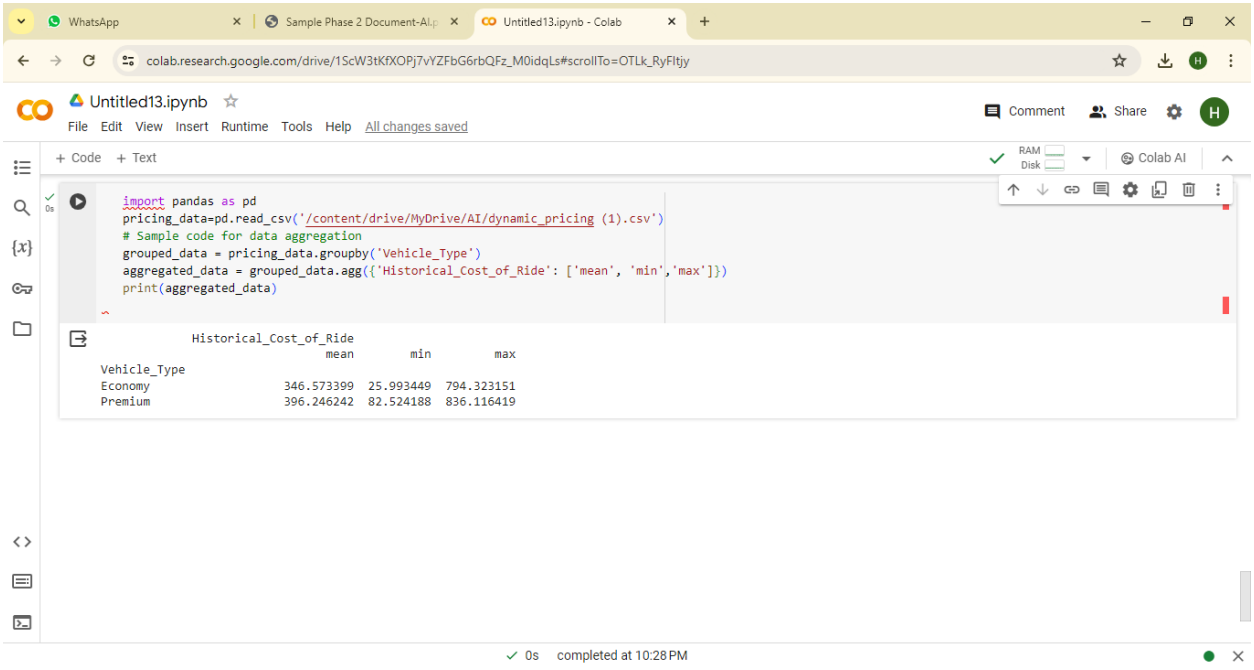
Together, grouping and aggregating data allow analysts to derive meaningful insights from the pricing dataset, aiding in decision-making processes and strategy formulation.

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)
```

Output:



Dynamic Pricing Analysis Techniques:

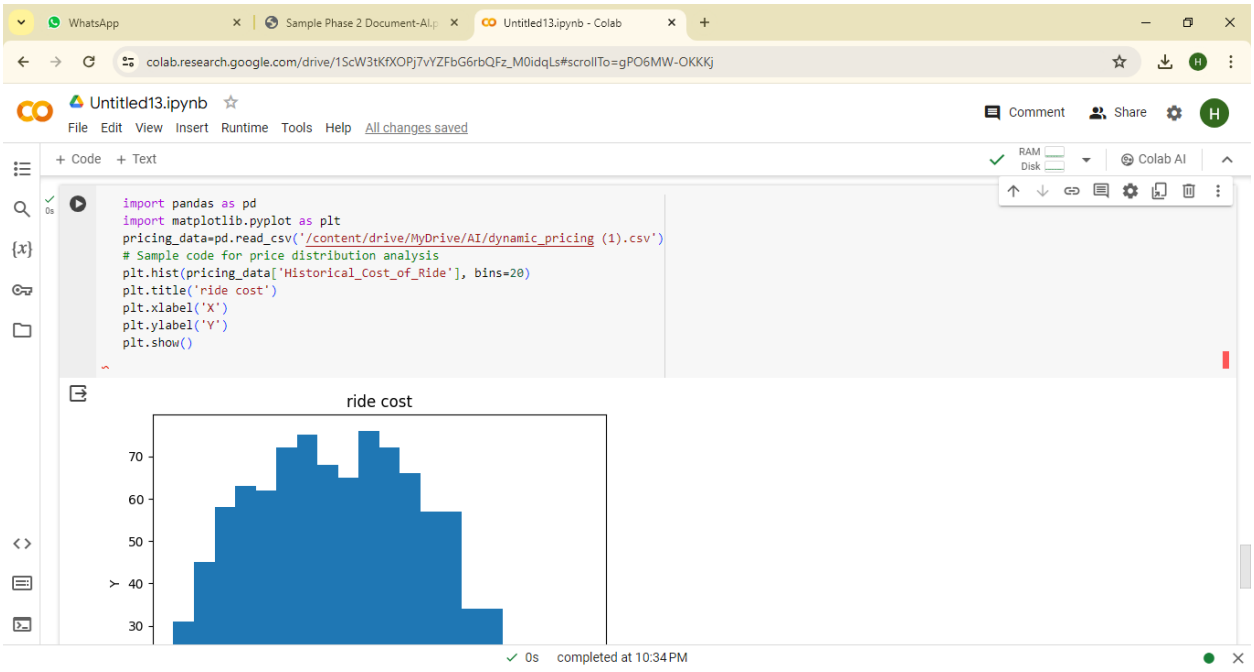
1. Price Distribution Analysis:

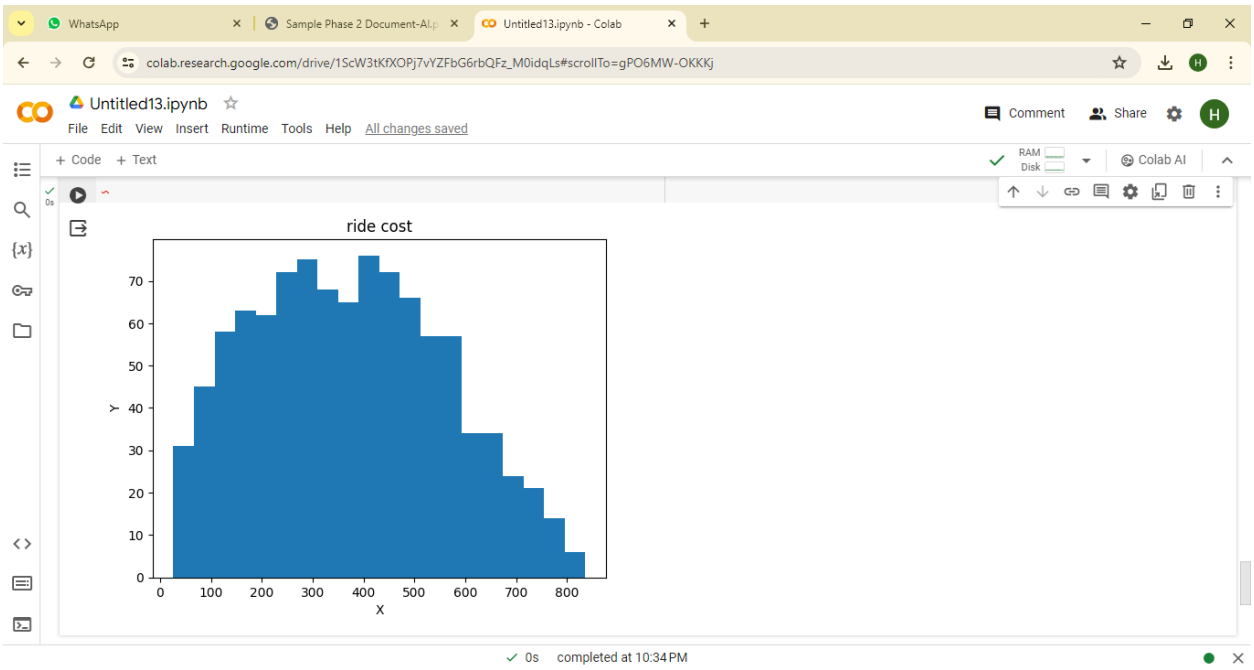
Price Distribution Analysis entails examining how prices are spread across different products within a dataset. This analysis helps in understanding the variability and patterns of pricing within a product portfolio, identifying trends, outliers, and potential pricing strategies. Ultimately, it assists in making informed decisions regarding pricing optimization, market positioning, and revenue management.

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()
```

Output:





2. Competitor Price Comparison:

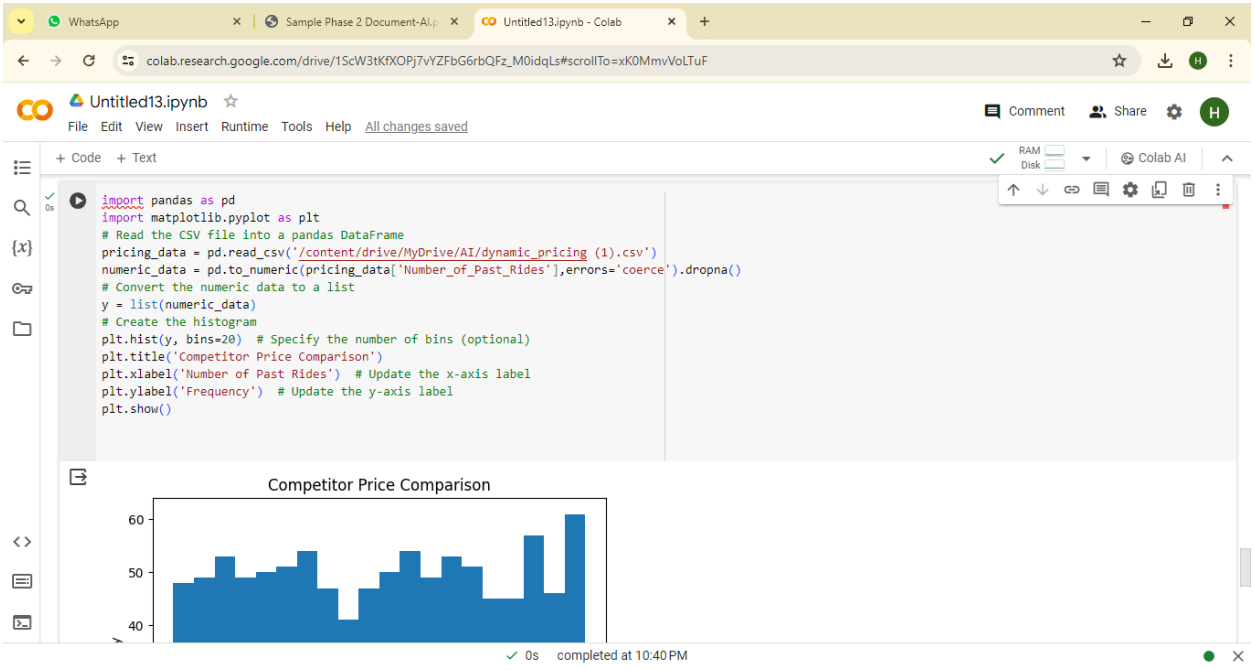
Comparing our pricing strategy with competitor prices involves assessing our competitiveness and market positioning by analyzing how our prices align with those of competitors. This evaluation helps identify potential adjustments to our pricing strategy, ensuring we remain competitive, attract customers,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

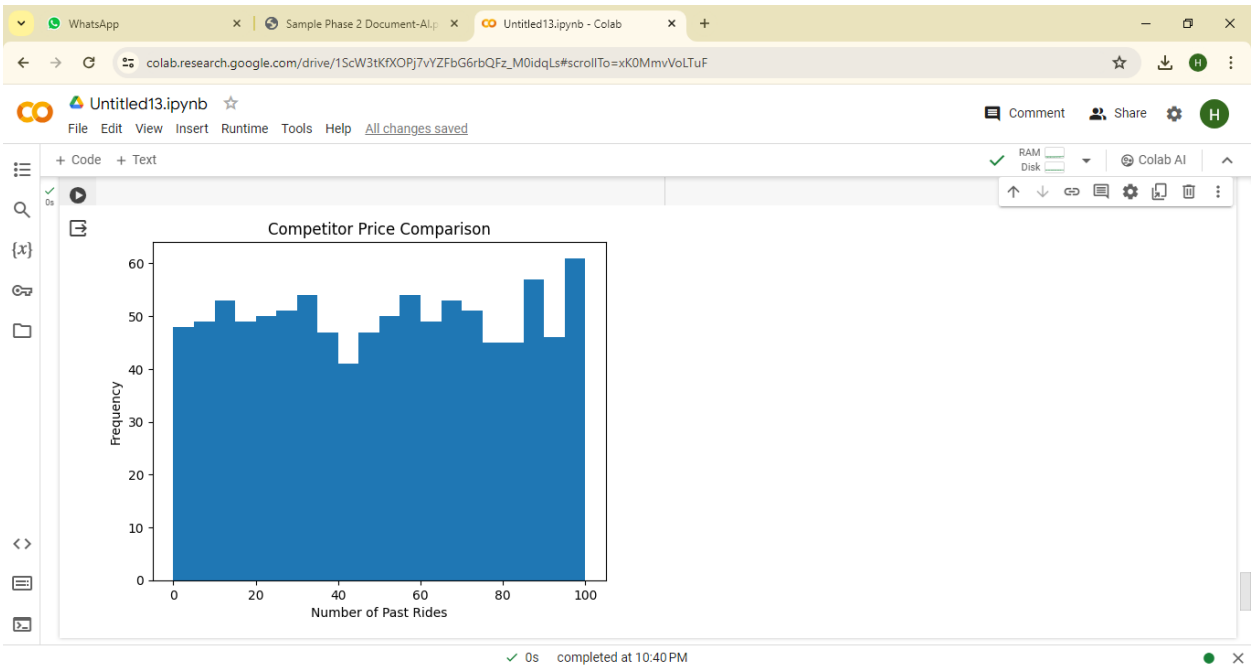
Python code:

```
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()
```

Output:





3. Dynamic Pricing Model Performance:

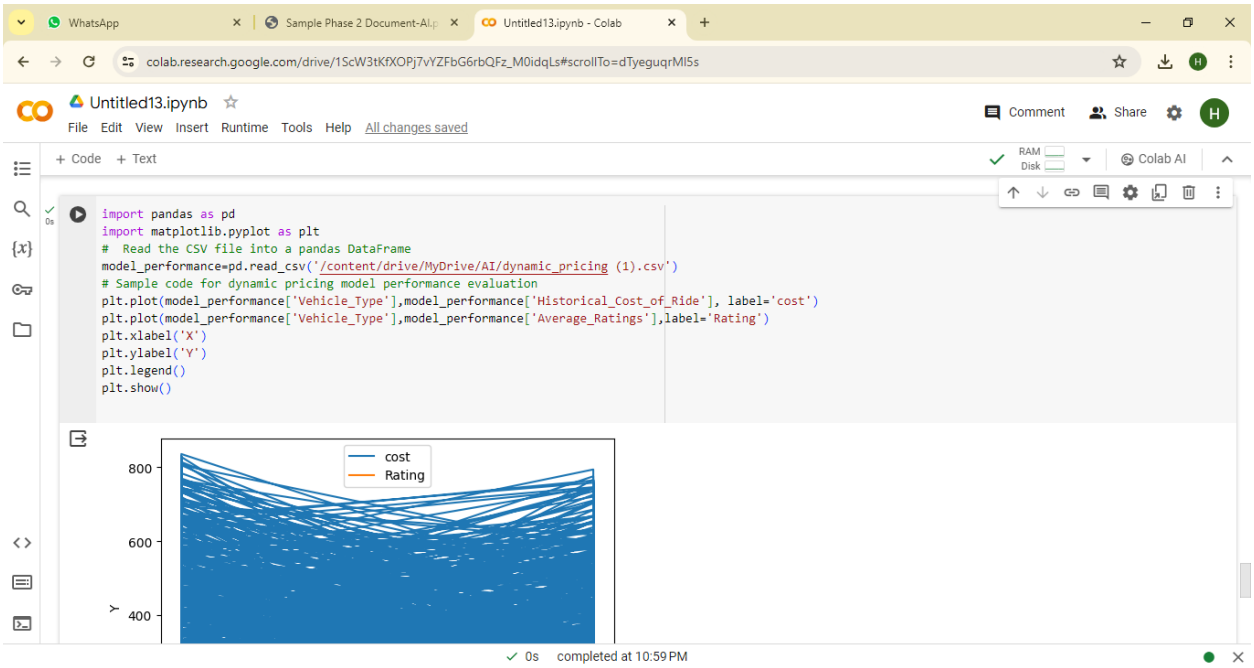
Evaluating the performance of dynamic pricing models entails assessing their effectiveness in predicting and adjusting prices to adapt to market changes. This assessment involves analyzing metrics like revenue growth, profit margins, customer satisfaction, and competitiveness against competitors. By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

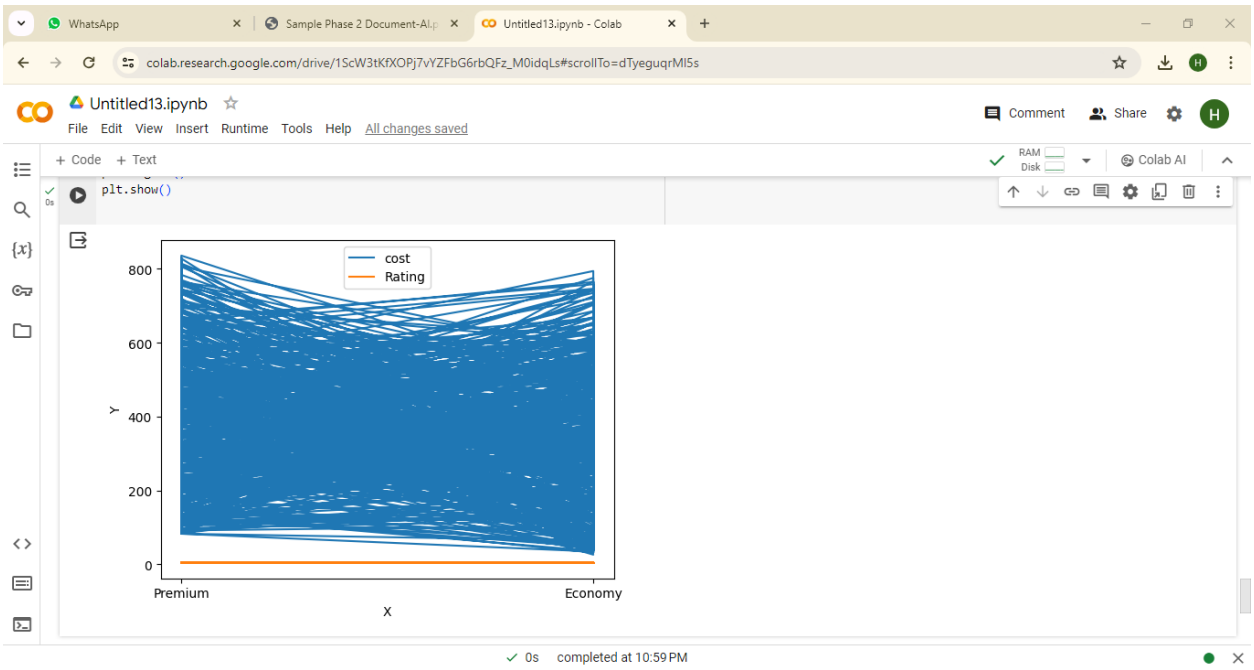
Python code:

```
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()
```

Output:





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()
```

Output:

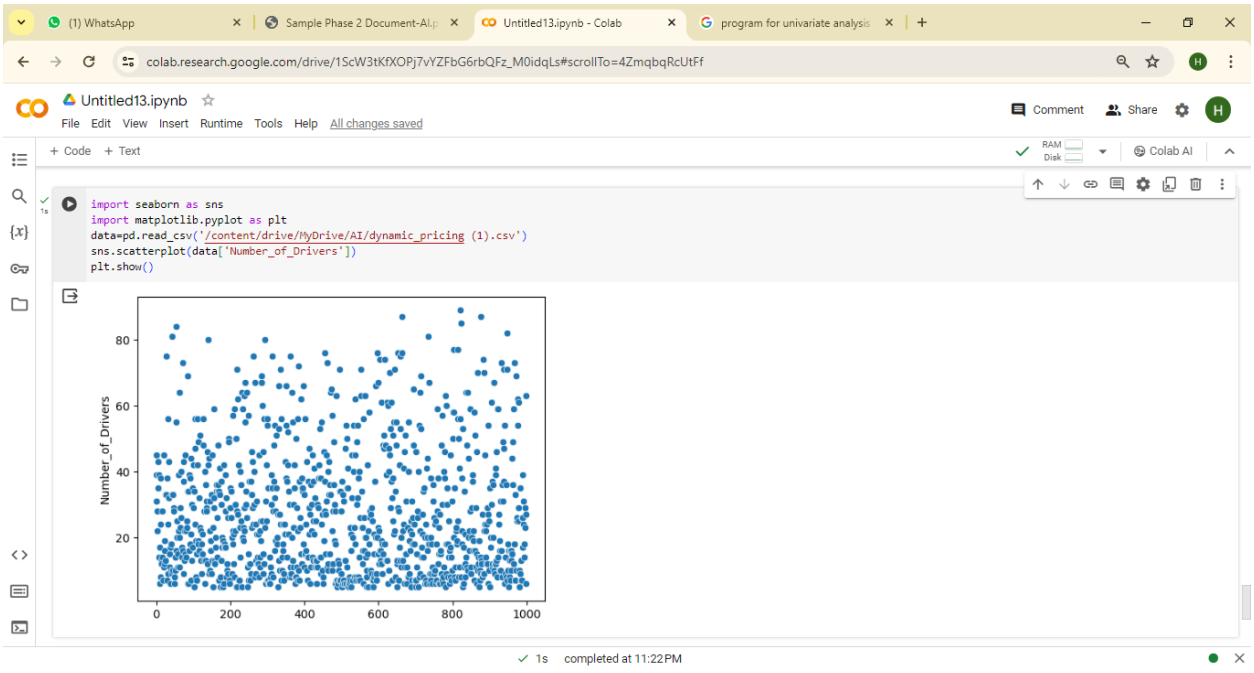
2. Bivariate Analysis:

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()
```

Output:



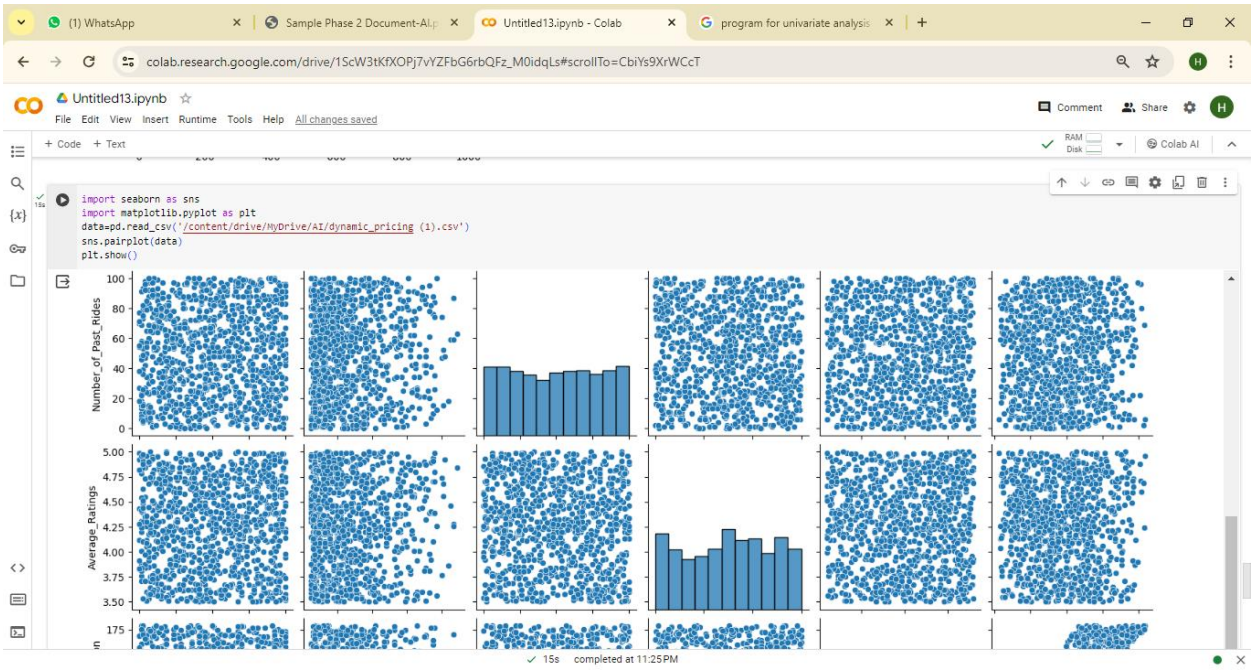
3. Multivariate Analysis:

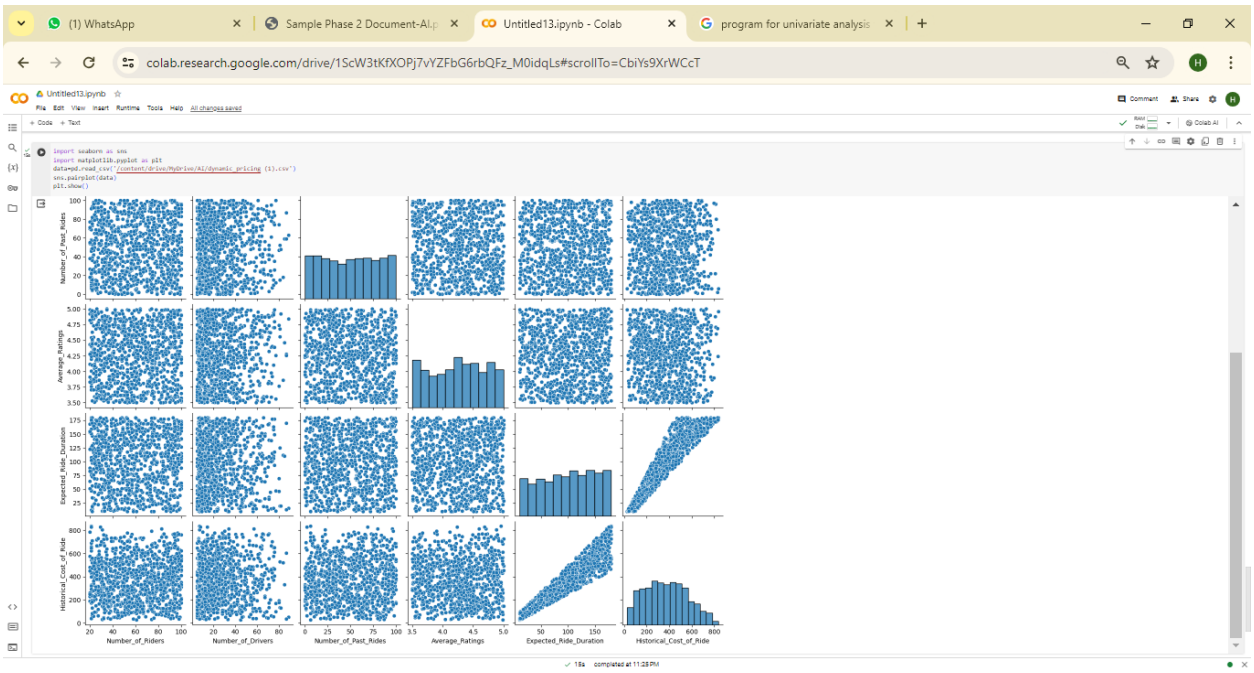
Multivariate Analysis examines the relationships between multiple variables simultaneously, offering a comprehensive understanding of complex patterns within the dataset. It uncovers interactions and dependencies through techniques like regression, principal component analysis, and clustering, enabling 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()
```

Output:





Assumed Scenario:

Scenario: The project seeks to provide personalized content recommendations to users based on their historical interactions and preferences.

Objective: The goal is to improve user engagement and satisfaction by delivering relevant and tailored content recommendations.

Target Audience: The target audience includes users of digital platforms who are interested in receiving personalized content recommendations across various domains.

Conclusion:

In conclusion, dynamic pricing offers businesses the ability to adapt pricing strategies in real-time, maximizing revenue and competitiveness. However, successful implementation requires careful consideration of factors like data quality, algorithm selection, and ethical considerations. With ongoing monitoring and optimization, businesses can leverage dynamic pricing to stay competitive, responsive, and deliver value to customers.