

Introduction

The objective of this project is to analyze flight booking data from Easemytrip, focusing on routes between India's top six metro cities. This analysis aims to uncover patterns in flight prices, durations, and airline frequencies, providing insights for both consumers and airlines.

Data Overview

- **Dataset:** The analysis utilizes a dataset containing 300,261 entries and 11 features, sourced from `Clean_Dataset.csv`.
- **Features:** Key features include airline, source city, destination city, departure time, arrival time, price, and duration

Methodology

We utilized several key libraries for this analysis:

Pandas for data manipulation,

```
import pandas as pd
```

NumPy for numerical operations, and

```
import numpy as np
```

Matplotlib along with **Seaborn** for data visualization.

```
import matplotlib.pyplot as plt
```

```
import seaborn as sns
```

These tools allowed us to effectively analyze and visualize flight booking data.

Data Loading

The dataset was loaded into a Pandas DataFrame for analysis .

```
data=pd.read_csv('Clean_Dataset.csv')
```

Initial Data Inspection

The first five rows were reviewed to understand the dataset's structure.

```
#print first five rows of data
```

```
data.head()
```

The screenshot shows a Google Colab notebook with the following content:

```
[1] import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

data=pd.read_csv('Clean_Dataset.csv')

#print first five rows of data
data.head()
```

Unnamed: 0	airline	flight	source_city	departure_time	stops	arrival_time	destination_city	class	duration	days_left	price	
0	0	SpiceJet	SG-8709	Delhi	Evening	zero	Night	Mumbai	Economy	2.17	1.0	5953.0
1	1	SpiceJet	SG-8157	Delhi	Early_Morning	zero	Morning	Mumbai	Economy	2.33	1.0	5953.0
2	2	AirAsia	I5-764	Delhi	Early_Morning	zero	Early_Morning	Mumbai	Economy	2.17	1.0	5956.0
3	3	Vistara	UK-995	Delhi	Morning	zero	Afternoon	Mumbai	Economy	2.25	1.0	5955.0
4	4	Vistara	UK-963	Delhi	Morning	zero	Morning	Mumbai	Economy	2.33	1.0	5955.0

Next steps: [Generate code with data](#) [View recommended plots](#) [New interactive sheet](#)

```
[4] #print last five rows of data
data.tail()
```

completed at 12:04 AM

Initial Data Inspection

The last five rows were reviewed to understand the dataset's structure.

```
#print last five rows of data
```

```
data.tail()
```

The screenshot shows a Google Colab notebook with two code cells. The first cell, labeled [4], contains the code `#print last five rows of data` and `data.tail()`. The output is a table with 12 columns: `Unnamed: 0`, `airline`, `flight`, `source_city`, `departure_time`, `stops`, `arrival_time`, `destination_city`, `class`, `duration`, `days_left`, and `price`. The second cell, labeled [5], contains the code `#Cleaning the data for missing values, null values` and `data.isnull().sum()`. The output shows the count of null values for each column: `Unnamed: 0` (0), `airline` (0), `flight` (0), and `source_city` (0).

Unnamed: 0	airline	flight	source_city	departure_time	stops	arrival_time	destination_city	class	duration	days_left	price
91550	91550	Vistara UK-852	Bangalore	Morning	one	Night	Delhi	Economy	13.50	38.0	4111.0
91551	91551	Vistara UK-864	Bangalore	Evening	one	Morning	Delhi	Economy	13.58	38.0	4111.0
91552	91552	Vistara UK-854	Bangalore	Evening	one	Morning	Delhi	Economy	13.67	38.0	4111.0
91553	91553	Vistara UK-858	Bangalore	Early_Morning	one	Night	Delhi	Economy	14.00	38.0	4111.0
91554	91554	Vistara UK-854	Bangalore	Evening	one	M	NaN	NaN	NaN	NaN	NaN

Data Cleaning

Missing Values: Checked for null values and removed any rows with missing data.

```
#Cleaning the data for missing values, null values
data.isnull().sum()
data.dropna(inplace=True)
```

Index Column:

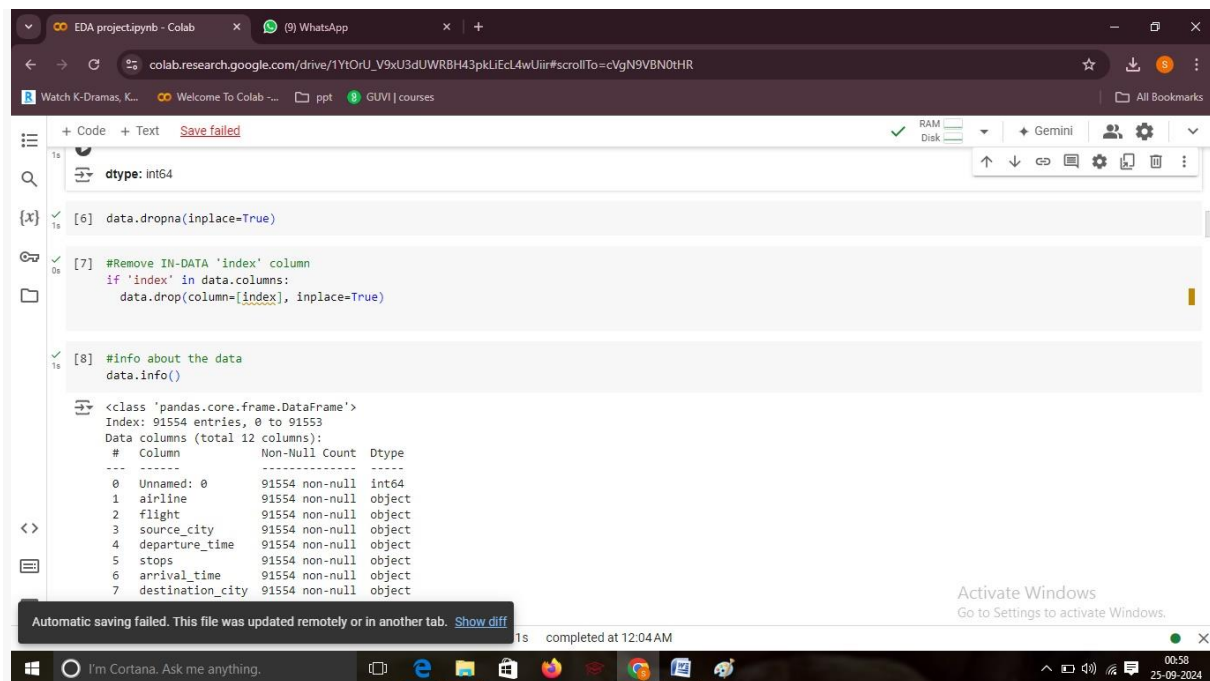
Removed the 'index' column if it was present.

```
#Remove IN-DATA 'index' column
if 'index' in data.columns:
    data.drop(column=[index], inplace=True)
```

Dataset Insights

Used `info()` and `describe()` to gather information about the data types and statistical properties.

```
#info about the data
data.info()
```



```
dtype: int64
```

```
[6] data.dropna(inplace=True)
```

```
[7] #Remove IN-DATA 'index' column
    if 'index' in data.columns:
        data.drop(column=[index], inplace=True)
```

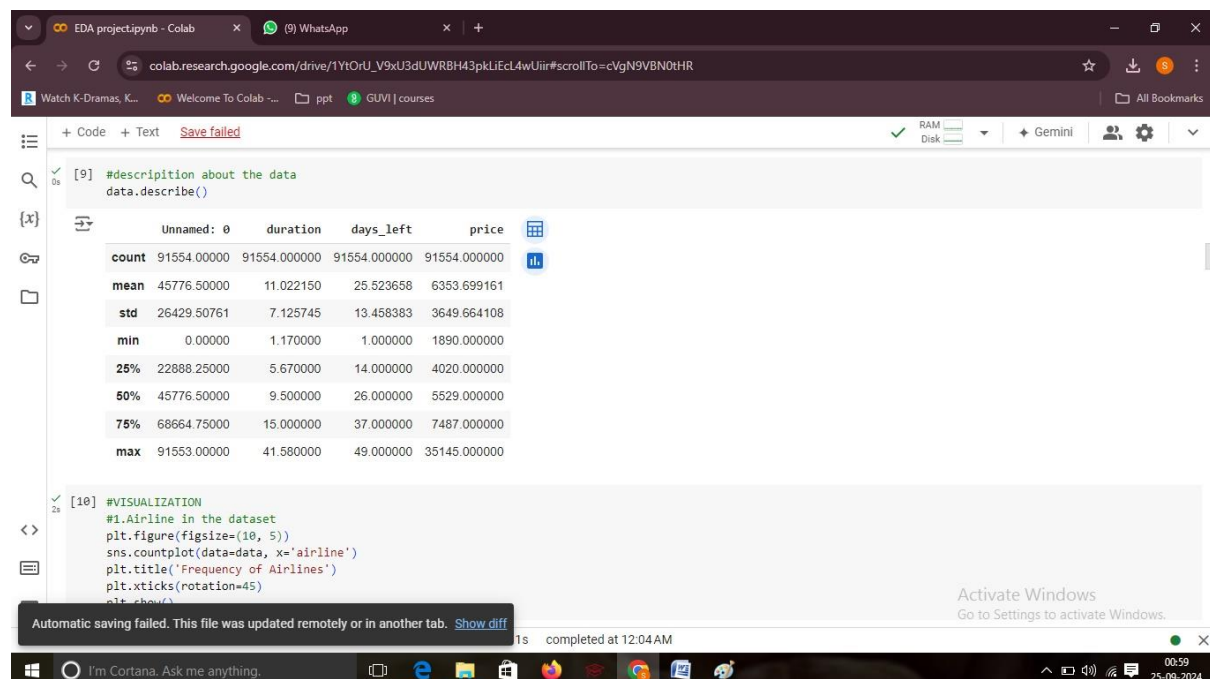
```
[8] #info about the data
    data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 91554 entries, 0 to 91553
Data columns (total 12 columns):
#   Column          Non-Null Count  Dtype
---  -
0   Unnamed: 0       91554 non-null  int64
1   airline          91554 non-null  object
2   flight           91554 non-null  object
3   source_city      91554 non-null  object
4   departure_time   91554 non-null  object
5   stops            91554 non-null  object
6   arrival_time     91554 non-null  object
7   destination_city 91554 non-null  object
```

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)

1s completed at 12:04 AM

```
#description about the data
data.describe()
```



```
[9] #description about the data
    data.describe()
```

	Unnamed: 0	duration	days_left	price
count	91554.00000	91554.000000	91554.000000	91554.000000
mean	45776.50000	11.022150	25.523658	6353.699161
std	26429.50761	7.125745	13.458383	3649.664108
min	0.00000	1.170000	1.000000	1890.000000
25%	22888.25000	5.670000	14.000000	4020.000000
50%	45776.50000	9.500000	26.000000	5529.000000
75%	68664.75000	15.000000	37.000000	7487.000000
max	91553.00000	41.580000	49.000000	35145.000000

```
[10] #VISUALIZATION
#1.Airline in the dataset
plt.figure(figsize=(10, 5))
sns.countplot(data=data, x='airline')
plt.title('Frequency of Airlines')
plt.xticks(rotation=45)
plt.show()
```

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)

1s completed at 12:04 AM

Visualizations

Airline Frequency

A count plot visualized the distribution of flights across different airlines.

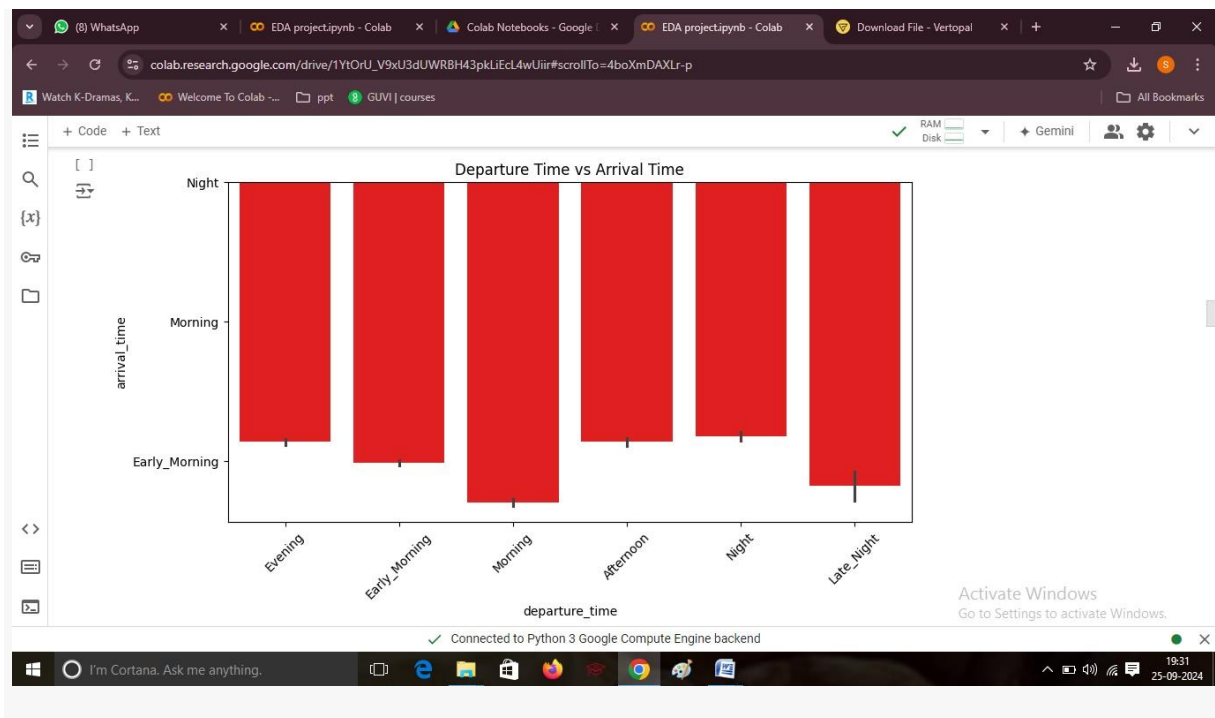
```
#VISUALIZATION
#1.Airline in the dataset
plt.figure(figsize=(10, 5))
sns.countplot(data=data, x='airline')
plt.title('Frequency of Airlines')
plt.xticks(rotation=45)
plt.show()
```



Departure vs. Arrival Time

A bar plot displayed the relationship between departure and arrival times.

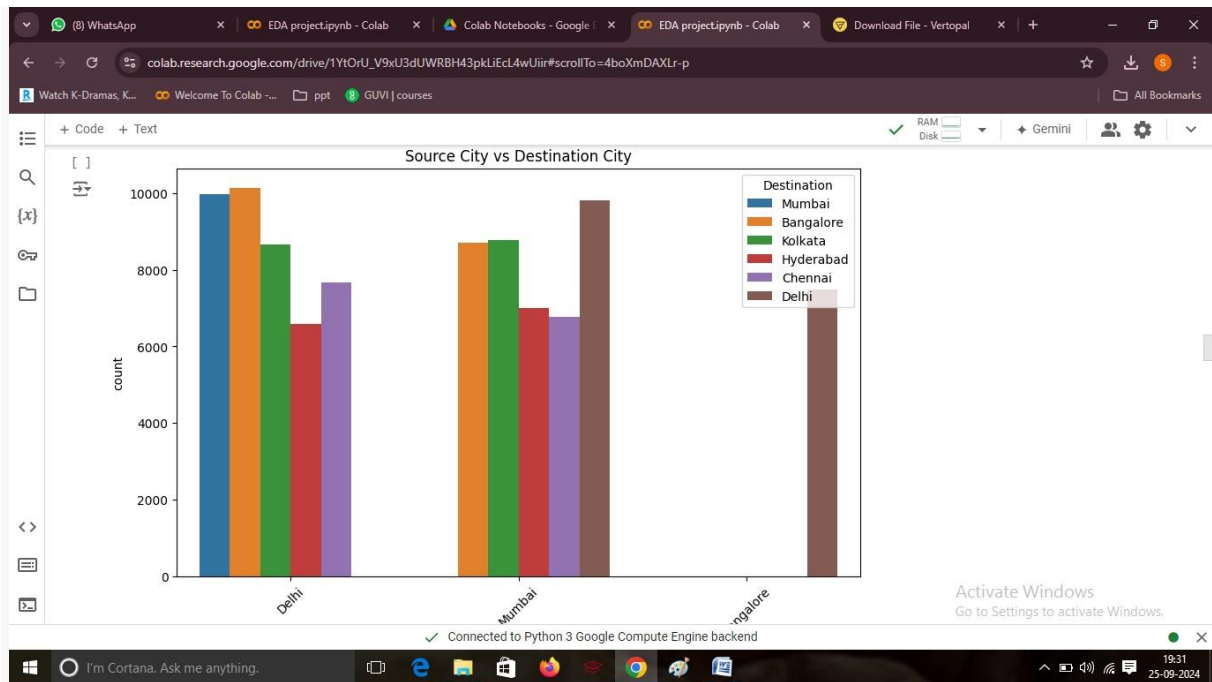
```
#2.Departure Time Against Arrival Time
plt.figure(figsize=(10, 5))
sns.barplot(data=data, x='departure_time',
y='arrival_time', color='red')
plt.title('Departure Time vs Arrival Time')
plt.xticks(rotation=45)
plt.show()
```



Source vs. Destination Cities

A count plot compared the source cities to destination cities, revealing popular routes.

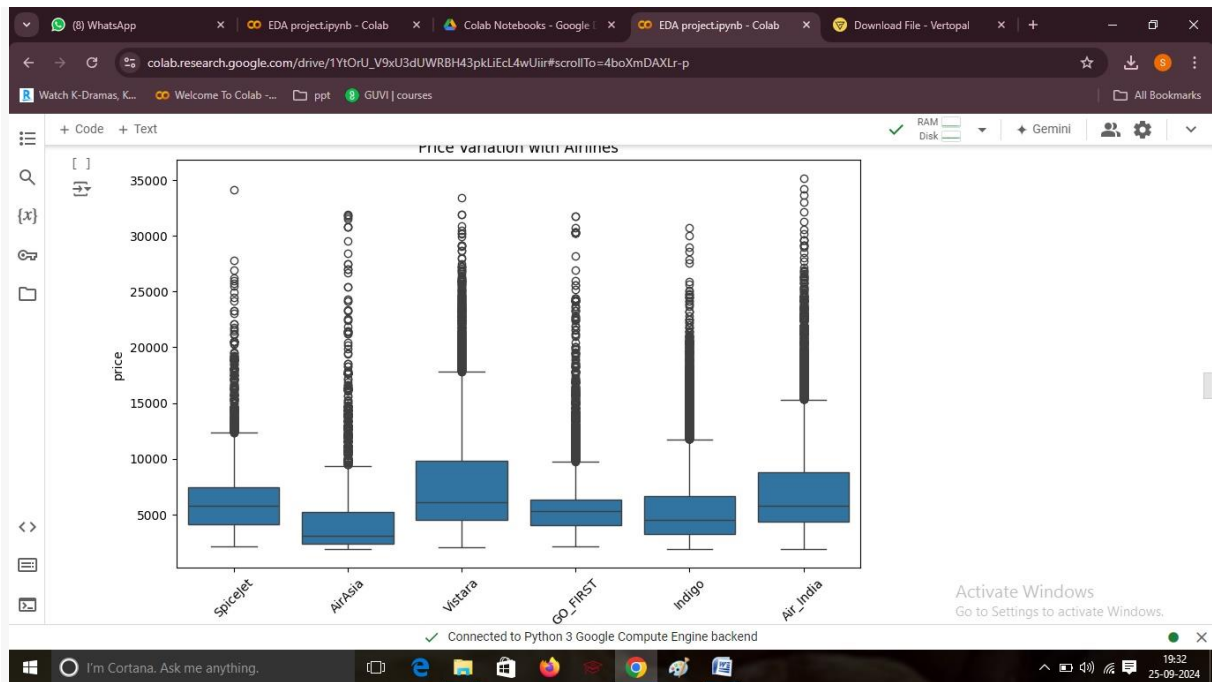
```
#3.Source City Against Destination City
plt.figure(figsize=(10, 6))
sns.countplot(data=data, x='source_city', hue='destination_city')
plt.title('Source City vs Destination City')
plt.xticks(rotation=45)
plt.legend(title='Destination')
plt.show()
```



Price Variation by Airline

A box plot illustrated how ticket prices varied across different airlines.

```
#4.Price Variation with Airlines
plt.figure(figsize=(10, 6))
sns.boxplot(data=data, x='airline', y='price')
plt.title('Price Variation with Airlines')
plt.xticks(rotation=45)
plt.show()
```



Ticket Price Trends

Line plots explored the relationship between ticket prices, departure time, and arrival time.

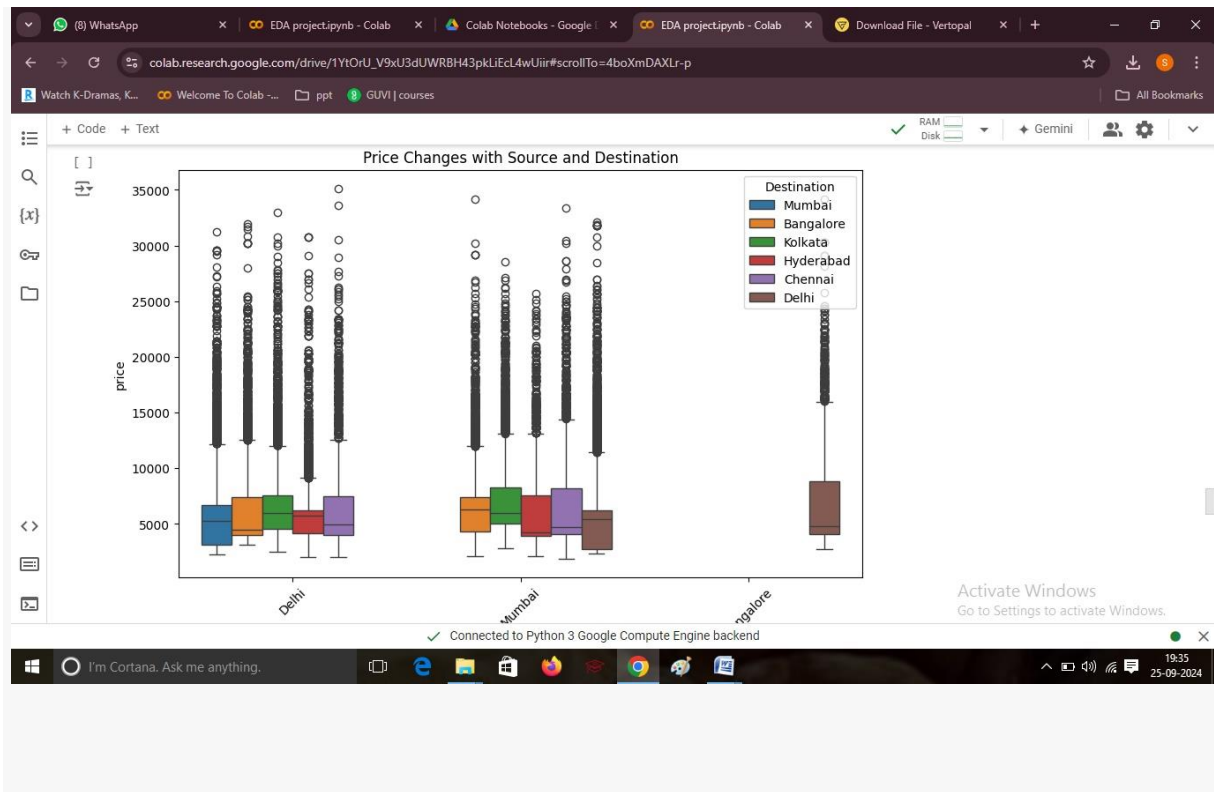
```
#5.Ticket Price vs Departure and Arrival Time
plt.figure(figsize=(12, 6))
sns.lineplot(data=data, x='departure_time', y='price',
label='departure_price', color='blue')
sns.lineplot(data=data, x='arrival_time', y='price',
label='arrival_price', color='orange')
plt.title('Ticket Price vs Departure and Arrival Time')
plt.xticks(rotation=45)
plt.legend()
plt.show()
```




Price Changes with Source and Destination

A box plot visualized price changes based on source and destination cities.\

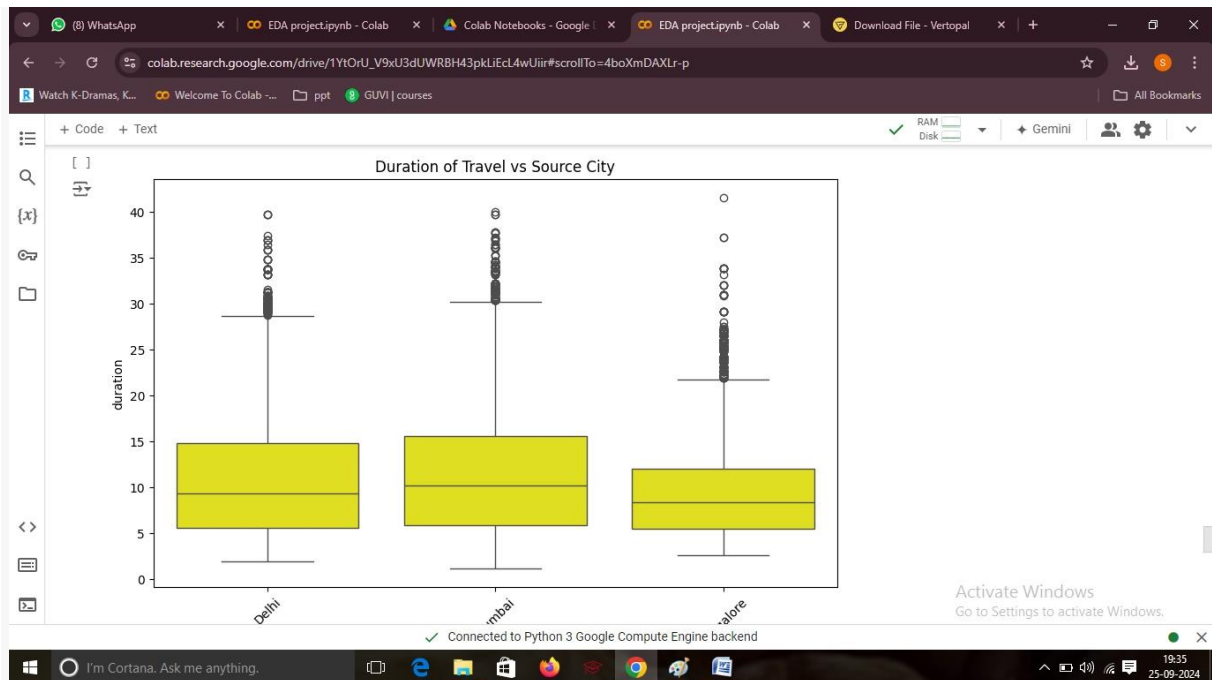
```
#6.Price changes with Source and Destination
plt.figure(figsize=(10, 6))
sns.boxplot(data=data, x='source_city', y='price',
            hue='destination_city')
plt.title('Price Changes with Source and Destination')
plt.xticks(rotation=45)
plt.legend(title='Destination')
plt.show()
```



Duration of Travel vs. Source City

A box plot represented travel durations for different source cities.

```
#7.Duration of travel vs city
plt.figure(figsize=(10, 6))
sns.boxplot(data=data, x='source_city', y='duration',color='yellow')
plt.title('Duration of Travel vs Source City')
plt.xticks(rotation=45)
plt.show()
```



High price with class type for city.

The heatmap illustrates average flight prices by source city and class type, highlighting pricing variations across different classes.

```
#8.High price with class type for city
average_price = data.groupby(['source_city',
                              'class'])['price'].mean().unstack()
plt.figure(figsize=(12, 8))
sns.heatmap(average_price, annot=True, fmt=".2f", cmap='YlGnBu',
            linewidths=0.5)
plt.title('Heatmap of Average Prices by Class Type for Each City')
plt.xlabel('Class Type')
plt.ylabel('Source City')
plt.show()
```

