# Week 1 - Exploring Data Basics

(Project 1: Sales Data Analysis )

**Objective:**

The goal of this project is to analyze sales data by:

1. Cleaning and preprocessing the dataset.
2. Calculating metrics such as total and average sales.
3. Visualizing sales trends over time using line plots.

**Code Highlights:**

**1. Data Loading and Preprocessing**:

Python code:

import pandas as pd

file\_path = '/path/to/your/file.csv' # Replace with the actual file path

data = pd.read\_csv(file\_path)

data['Order Date'] = pd.to\_datetime(data['Order Date'], format='%d/%m/%Y')

data['Ship Date'] = pd.to\_datetime(data['Ship Date'], format='%d/%m/%Y')

data['Postal Code'].fillna('Unknown', inplace=True)

* This section loads the dataset and ensures that the date columns are properly formatted for time-based analysis.
* Missing values in the Postal Code column are filled with "Unknown."

**2. Calculating Metrics**:

Python code:

total\_sales = data['Sales'].sum()

average\_sales = data['Sales'].mean()

print(f"Total Sales: ${total\_sales:,.2f}")

print(f"Average Sales: ${average\_sales:,.2f}")

* This calculates and displays key metrics like total sales and average sales per transaction.

**3. Sales Trends Visualization**:

Python code:

import matplotlib.pyplot as plt

sales\_trends = data.groupby('Order Date')['Sales'].sum()

plt.figure(figsize=(12, 6))

plt.plot(sales\_trends.index, sales\_trends.values, label='Daily Sales', color='blue', linewidth=2)

plt.title('Sales Trends Over Time', fontsize=16)

plt.xlabel('Order Date', fontsize=12)

plt.ylabel('Total Sales ($)', fontsize=12)

plt.grid(alpha=0.4)

plt.legend()

plt.tight\_layout()

plt.show()

* This plots a line chart showing daily sales trends over time, revealing seasonal patterns or anomalies.

**How It Works:**

* **Data Loading**: The dataset is read into a DataFrame using Pandas, a Python library for data manipulation.
* **Preprocessing**: Dates are converted to a standard format, and missing values are handled to ensure the dataset is clean.
* **Analysis**: Metrics like total and average sales are calculated using basic Pandas operations.
* **Visualization**: A time-series plot is created to display trends in sales, enabling insights into seasonal performance.

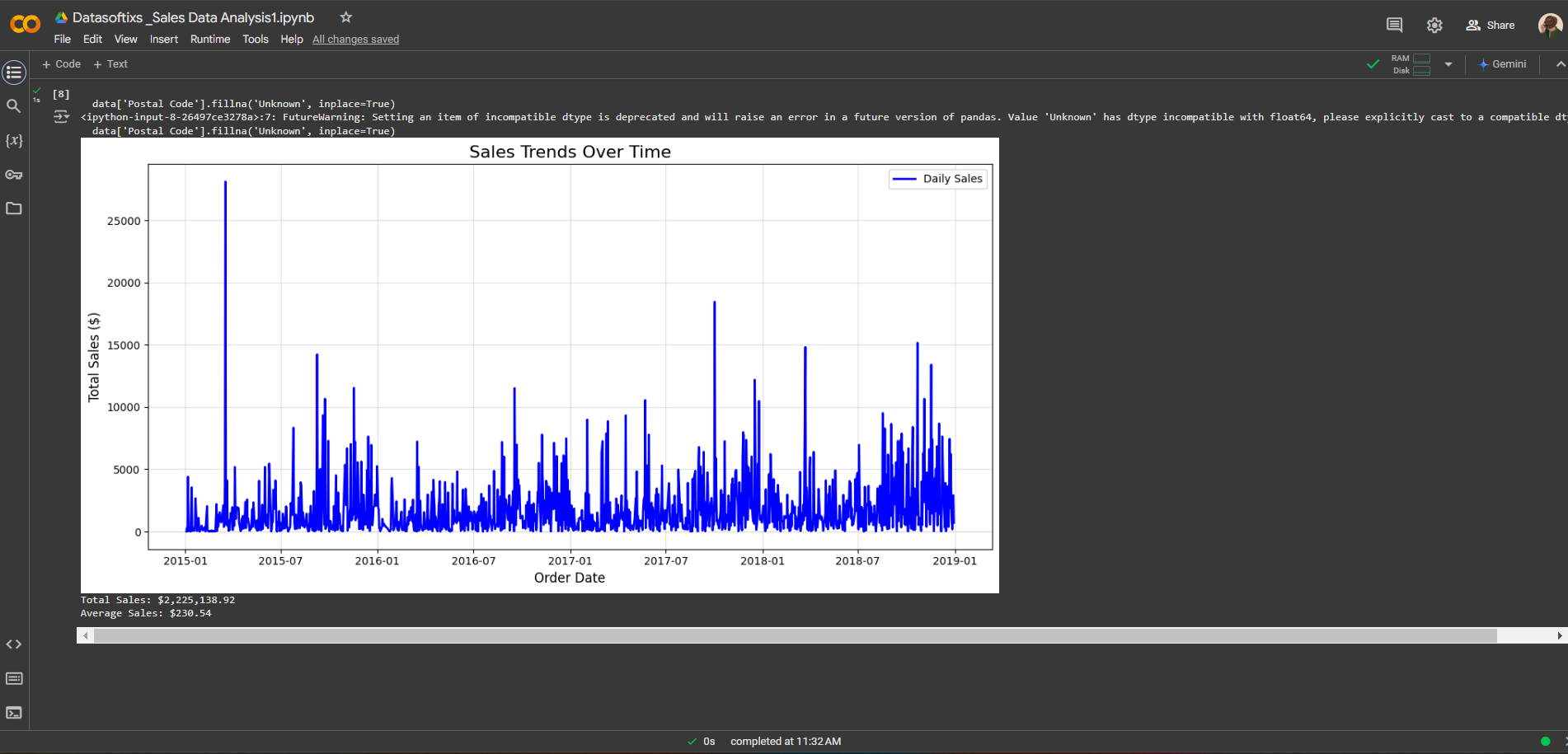
**Tools and Technologies Used:**

* **Programming Language**: Python
* **Libraries**:
* **Pandas**: For data loading, preprocessing, and analysis.
* **Matplotlib**: For data visualization (line charts for trends).
* **File Format**: CSV (Comma-Separated Values)
* **Environment**: Jupyter Notebook or any Python IDE.

**Key Insights:**

* **Total Sales**: Aggregated from all transactions in the dataset.
* **Average Sales**: Useful for understanding per-transaction performance.
* **Sales Trends**: Time-series visualization reveals seasonal peaks and troughs in sales.

**Output:**



# Week 2 - Data Manipulation & Visualization

(Project 3 : Weather Data Analysis and Visualization)

**Objective:**

To analyze weather trends and anomalies using a dataset that includes parameters such as temperature, humidity, and location, and to create insightful visualizations.

**Key Sections of Code:**

**1. Data Loading and Preparation**

Python code:

file\_path = '/mnt/data/weather\_data.csv'

weather\_data = pd.read\_csv(file\_path)

weather\_data['Date\_Time'] = pd.to\_datetime(weather\_data['Date\_Time'])

* The dataset is loaded using pandas, and the Date\_Time column is converted into a datetime format to facilitate time-series analysis.

**2. Grouping and Resampling Data**

Python code:

monthly\_data = weather\_data.set\_index('Date\_Time').groupby('Location').resample('ME').mean().reset\_index()

* The data is grouped by location and resampled to compute monthly averages for easier trend visualization.

**3. Visualization: Temperature Trends**

Python code:

plt.figure(figsize=(14, 7))

sns.lineplot(data=monthly\_data, x='Date\_Time', y='Temperature\_C', hue='Location', linewidth=1.5)

plt.title('Temperature Trends Over Time by Location')

plt.xlabel('Date Time')

plt.ylabel('Temperature (°C)')

plt.legend(title='Location')

plt.tight\_layout()

plt.show()

* A line plot showing temperature trends over time, grouped by location.

**4. Visualization: Temperature vs. Humidity**

Python code:

plt.figure(figsize=(10, 6))

sns.scatterplot(data=weather\_data, x='Humidity\_pct', y='Temperature\_C', hue='Location', alpha=0.7)

plt.title('Temperature vs. Humidity by Location')

plt.xlabel('Humidity (%)')

plt.ylabel('Temperature (°C)')

plt.legend(title='Location')

plt.tight\_layout()

plt.show()

* A scatter plot analyzing the relationship between temperature and humidity for each location.

**5. Extreme Temperature Analysis**

Python code:

extreme\_thresholds = (weather\_data['Temperature\_C'] < 0) | (weather\_data['Temperature\_C'] > 35)

extreme\_counts = weather\_data[extreme\_thresholds].groupby('Location').size().sort\_values(ascending=False)

plt.figure(figsize=(10, 6))

extreme\_counts.plot(kind='bar', color='skyblue', edgecolor='black')

plt.title('Extreme Temperature Counts by Location')

plt.xlabel('Location')

plt.ylabel('Count of Extreme Temperatures')

plt.xticks(rotation=45)

plt.tight\_layout()

plt.show()

* Highlights the frequency of extreme temperatures (e.g., below 0°C or above 35°C) across locations.

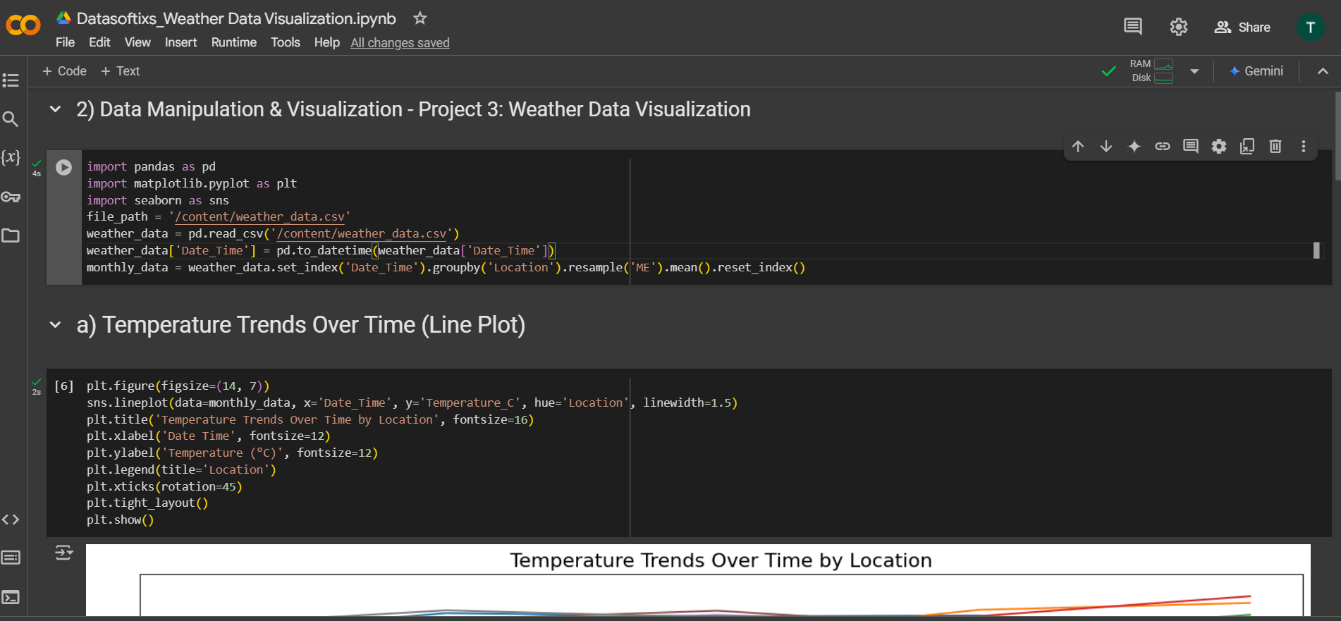
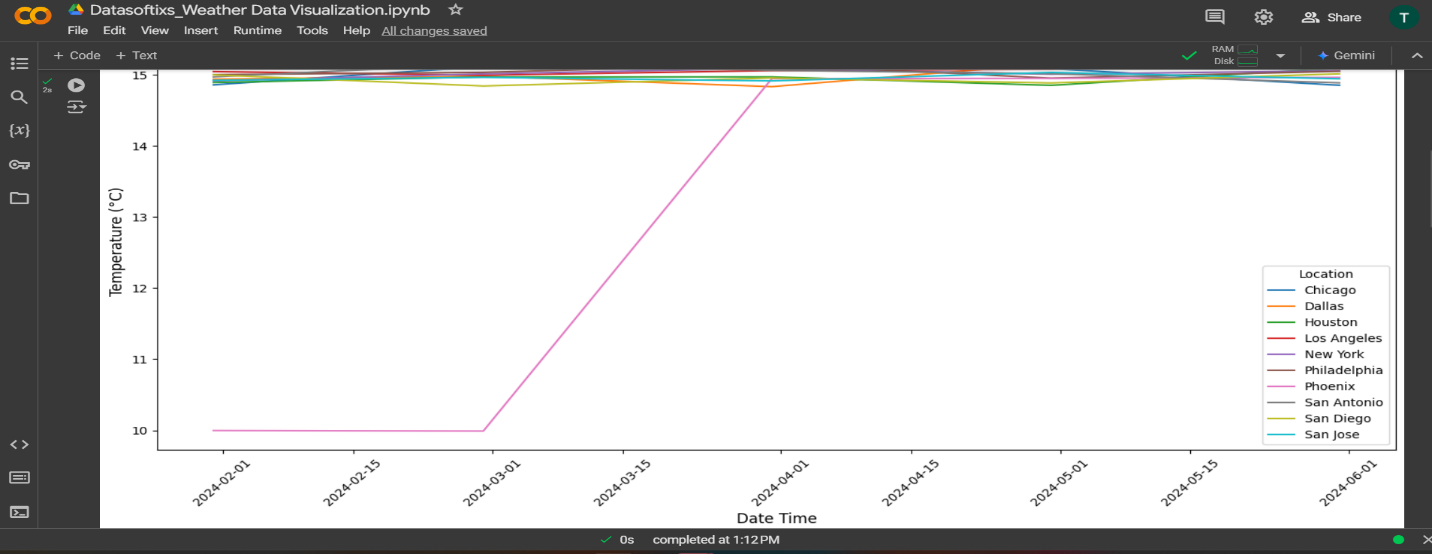
**How the Project Works:**

1. **Data Preparation**:
   * Loads and preprocesses the dataset to ensure it is structured for time-series analysis.
   * Groups and aggregates data by location and time intervals.
2. **Visualizations**:
   * Line plots for temperature trends provide insights into seasonal or periodic fluctuations.
   * Scatter plots reveal relationships between temperature and other factors (like humidity).
   * Bar charts identify anomalies such as extreme temperature counts.
3. **Insights**:
   * Helps understand climatic behavior across locations.
   * Facilitates anomaly detection for specific locations or periods.

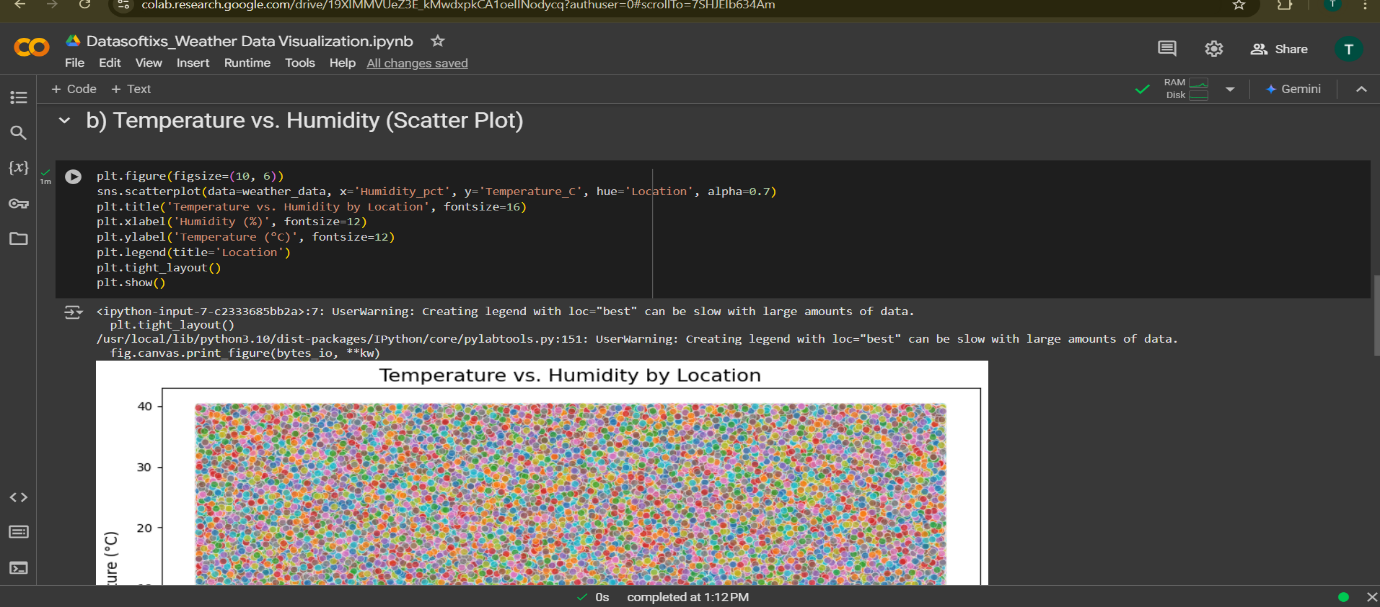
**Tools and Technologies Used:**

* **Programming Language**: Python
* **Libraries**:
  + **pandas**: Data manipulation and analysis.
  + **matplotlib**: Basic plotting and charting.
  + **seaborn**: Advanced data visualization with a focus on aesthetics.
* **Data**:
  + CSV file containing weather parameters across different locations and times.
* **Output :**

**1)**

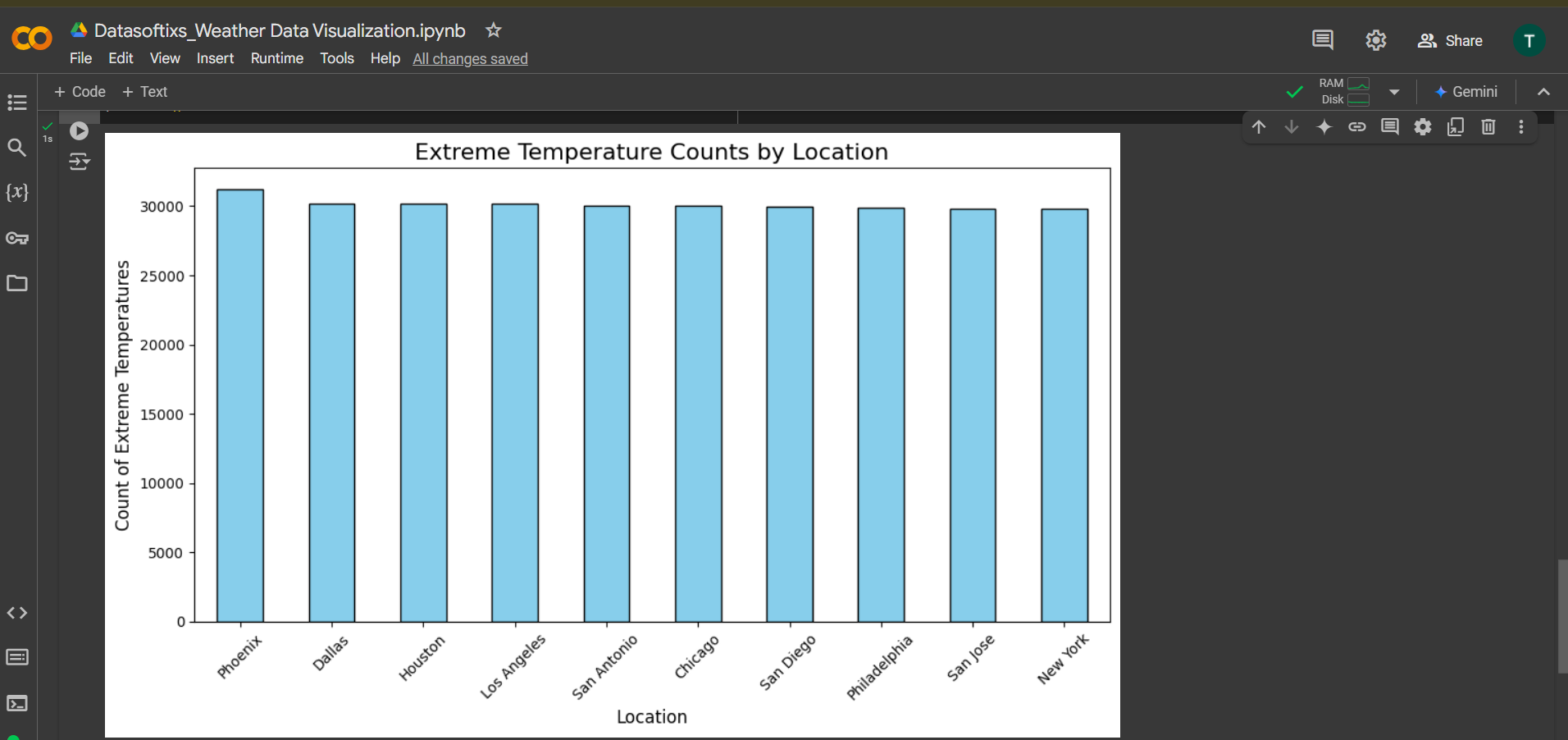
**** 2) 

**3)**



**4)**

5)



# Week 3 - Machine Learning Foundations

## (Project 5: Predicting House Prices)

**Overview**

This project involves predicting house prices based on various features such as size, location, and other property attributes. Using machine learning, the model was trained to analyze historical data and make accurate price predictions for new houses.

**Code Highlights**

**1. Data Preprocessing**

Python code:

preprocessor = ColumnTransformer(

transformers=[

('cat', OneHotEncoder(drop='first'), categorical\_cols)

],

remainder='passthrough'

)

* Used OneHotEncoder to convert categorical variables like mainroad, airconditioning, and others into numerical format.
* Numerical columns, such as area and bedrooms, were retained as-is.

**2. Splitting and Training**

Important snippet:

Python Code:

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

pipeline = Pipeline(steps=[

('preprocessor', preprocessor),

('model', LinearRegression())

])

pipeline.fit(X\_train, y\_train)

* Divided the dataset into training (80%) and testing (20%) subsets for effective evaluation.
* Created a pipeline combining preprocessing and training using LinearRegression.

**3. Evaluation Metrics**

Key evaluation code:

Python Code:

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

r2 = r2\_score(y\_test, y\_pred)

* Evaluated model performance using:
  + **Mean Absolute Error (MAE)**: ₹970,043.40
  + **R² Score**: 0.653 (explains 65.3% of price variance).

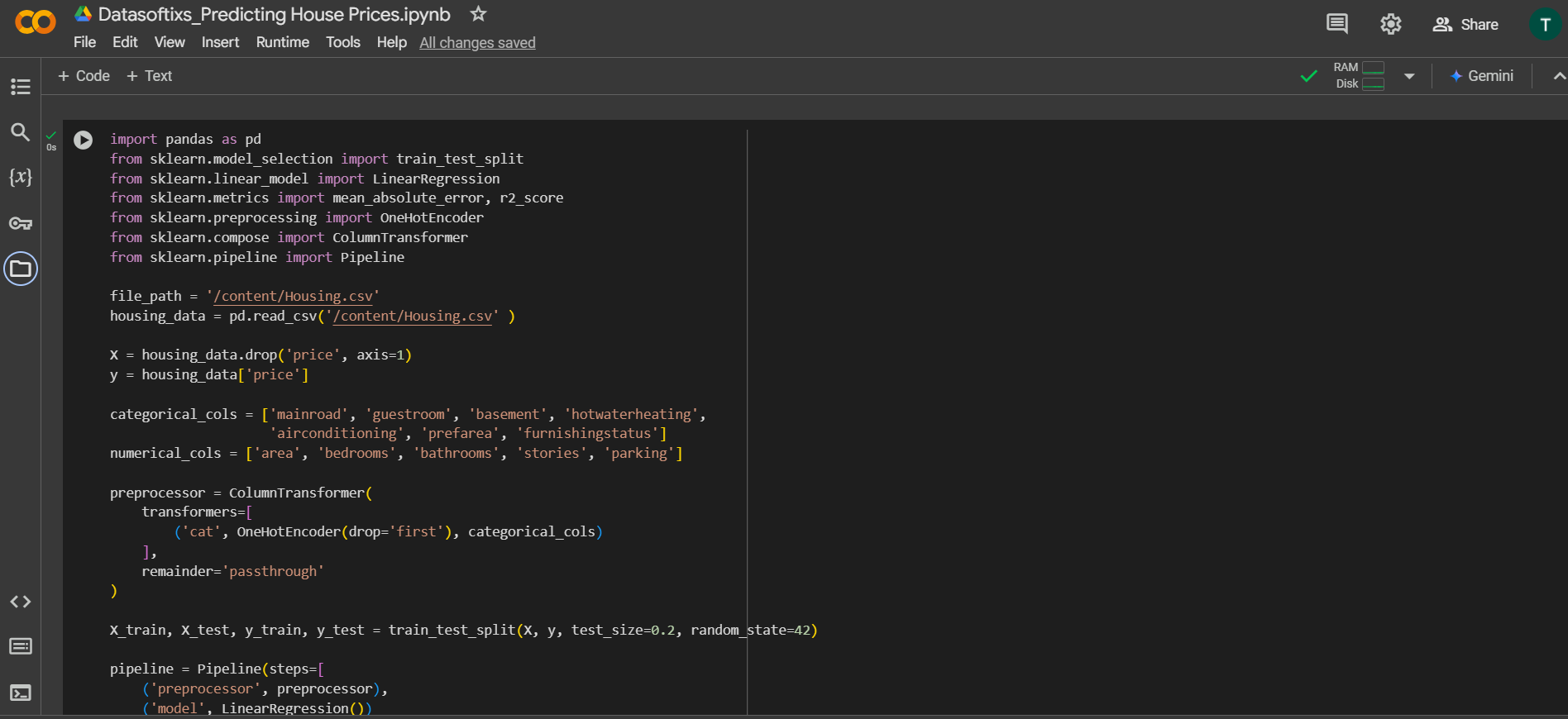
**Explanation**

1. **Data Loading**: The dataset was loaded into a pandas DataFrame, and columns were separated into target (price) and features.
2. **Preprocessing**:
   * Categorical columns were transformed using one-hot encoding.
   * A ColumnTransformer was used to apply preprocessing steps in an efficient manner.
3. **Model Training**: A LinearRegression model was trained using the preprocessed data to learn the relationship between features and house prices.
4. **Prediction**: The model predicted house prices on unseen test data.
5. **Evaluation**: Performance was measured using metrics like MAE and R², providing insights into the model’s accuracy and reliability.

**Tools and Technologies Used**

* **Languages**: Python
* **Libraries**:
  + **pandas**: For data manipulation and analysis.
  + **scikit-learn**: For preprocessing, model training, and evaluation.
* **Machine Learning Model**: Linear Regression
* **Tools**:
  + Jupyter Notebook or IDE for development.
  + CSV file for storing and processing housing data

**Output:**

****

