Release notes ×

Double-click (or enter) to edit

1

```
import pandas as pd
# Creating a sample dataset
data = {
    'OrderID': [1, 2, 3, 4, 5],
    'CustomerID': [101, 102, 101, 103, 104],
    'ProductID': [1001, 1002, 1003, 1001, 1002],
    'Quantity': [2, 1, 4, 2, 3],
    'TotalPrice': [50.0, 20.0, 80.0, 50.0, 60.0]
}
# Creating a DataFrame
df = pd.DataFrame(data)
# Displaying the DataFrame
print("Sample Dataset:\n", df)
# Performing analysis
# 1. Total revenue generated
total_revenue = df['TotalPrice'].sum()
print("\nTotal Revenue: $", total_revenue)
# 2. Number of unique customers
unique_customers = df['CustomerID'].nunique()
print("\nNumber of Unique Customers:", unique_customers)
# 3. Total quantity of products sold
total_quantity_sold = df['Quantity'].sum()
print("\nTotal Quantity Sold:", total_quantity_sold)
# 4. Average order value
average_order_value = df['TotalPrice'].mean()
print("\nAverage Order Value: $", average_order_value)
# 5. Orders per customer
orders_per_customer = df['CustomerID'].value_counts()
print("\nOrders per Customer:\n", orders_per_customer)
# 6. Total revenue per customer
revenue_per_customer = df.groupby('CustomerID')['TotalPrice'].sum(
print("\nTotal Revenue per Customer:\n", revenue_per_customer)
# 7. Total quantity sold per product
quantity_per_product = df.groupby('ProductID')['Quantity'].sum()
print("\nTotal Quantity Sold per Product:\n", quantity_per_product
```

```
Sample Dataset:
    OrderID  CustomerID  ProductID  Quantity  TotalPrice
0        1         101       1001         2        50.0
1        2         102       1002         1        20.0
2        3         101       1003         4        80.0
3        4         103       1001         2        50.0
4        5         104       1002         3        60.0

Total Revenue: $ 260.0

Number of Unique Customers: 4

Total Quantity Sold: 12

Average Order Value: $ 52.0

Orders per Customer:
 CustomerID
101    2
102    1
103    1
104    1
```

```
Name: count, dtype: int64

Total Revenue per Customer:
 CustomerID
101    130.0
102     20.0
103     50.0
104     60.0
Name: TotalPrice, dtype: float64

Total Quantity Sold per Product:
 ProductID
1001    4
1002    4
1003    4
Name: Quantity, dtype: int64
```

2

```
import pandas as pd
import matplotlib.pyplot as plt

# Define the data
data = {
    'SmokingStatus': ['Yes', 'No', 'Yes', 'No', 'Yes', 'No', 'Yes',
    'LungCancer': [True, False, True, False, True, True, True, Fals
}

# Create a DataFrame
df = pd.DataFrame(data)

# Map 'Yes' to 1 and 'No' to 0 in a new column 'SmokingStatus_Num'
smoking_map = {'Yes': 1, 'No': 0}
df['SmokingStatus_Num'] = df['SmokingStatus'].map(smoking_map)

# Calculate correlation coefficient between 'SmokingStatus_Num' and
correlation = df['SmokingStatus_Num'].corr(df['LungCancer'])
print("Correlation coefficient:", correlation)

# Plotting
plt.figure(figsize=(8, 6))
plt.scatter(df['SmokingStatus_Num'], df['LungCancer'], c='blue', al
plt.xlabel('Smoking Status (0=No, 1=Yes)')
plt.ylabel('Lung Cancer (True/False)')
plt.title('Smoking Status vs. Lung Cancer Incidence')
plt.grid(True)
plt.show()
```

Please follow our blog to see more information about new features, tips and tricks, and featured notebooks such as Analyzing a Bank Failure with Colab.

2024-05-13

- Code actions are now supported to automatically improve and refactor code. Code actions can be triggered by the keyboard shortcut "Ctrl/⌘ + ."
- Python package upgrades
  - bigframes 1.0.0 -> 1.5.0
  - google-cloud-aiplatform 1.47.0 -> 1.51.0
  - jax[tpu] 0.4.23 -> 0.4.26
- Python package inclusions
  - cudf 24.4.1

2024-04-15

- TPU v2 runtime is now available
- L4 runtime is now available for paid users
- New distributed fine-tuning Gemma tutorial on TPUs (GitHub)
- Symbol rename is now supported with keyboard shortcut F2
- Fixed bug causing inability to re-upload deleted files
- Fixed breaking bug in colabtools %upload_files_async
- Added syntax highlighting to %%writefile cells
- Cuda dependencies that come with Torch are cached for faster downloads for packages that require Torch and its dependencies (GitHub issue)
- Python package upgrades
  - bigframes 0.24.0 -> 1.0.0
  - duckdb 0.9.2 -> 0.10.1
  - google-cloud-aiplatform 1.43.0 -> 1.47.0
  - jax 0.4.23 -> 0.4.26

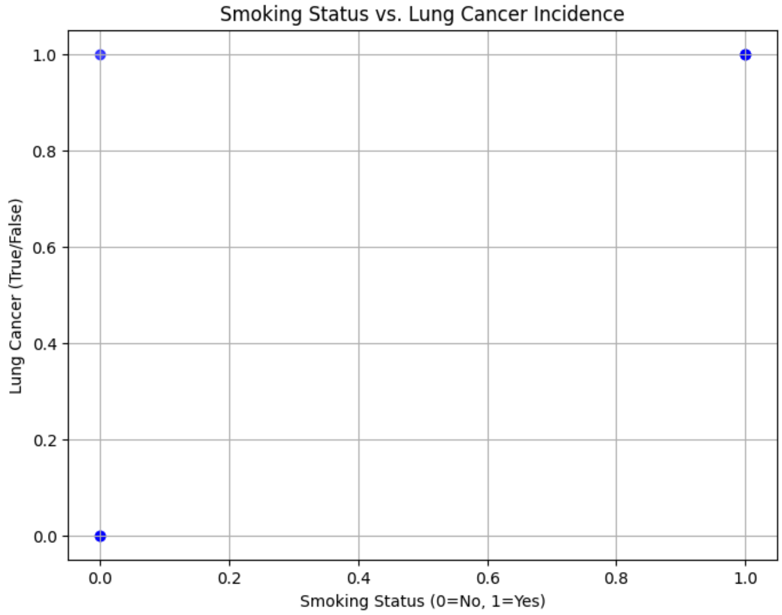
2024-03-13

- Fixed bug that sometimes caused UserSecrets to move / disappear
- Improved messaging for mounting drive in an unsupported environment (GitHub issue)
- Python package upgrades
  - torch 2.1.0 -> 2.2.1
  - torchaudio 2.1.0 -> 2.2.1
  - torchvision 0.16.0 -> 0.17.1
  - torchtext 0.16.0 -> 0.17.1
  - PyMC 5.7.2 -> 5.10.4
  - BigFrames 0.21.0 -> 0.24.0
  - google-cloud-aiplatform 1.42.1 -> 1.43.0
  - tornado 6.3.2 -> 6.3.3

2024-02-21

- Try out Gemma on Colab!
- Allow unicode in form text inputs

```
Correlation coefficient: 0.7745966692414833
```



3

```
import pandas as pd
import matplotlib.pyplot as plt
data = {
    'OrderID': [1000, 1001, 1002, 1003, 1004, 1005, 1006, 1007],
    'ProductCategory': ['Electronics', 'Electronics', 'Clothing', 'I
    'Sales': [500, 200, 300, 150, 700, 100, 400, 250]
}

df = pd.DataFrame(data)

categories = df['ProductCategory'].unique()
for category in categories:
    category_data = df[df['ProductCategory'] == category]
    plt.plot(category_data['OrderID'], category_data['Sales'], labe
plt.xlabel('Order ID')
plt.ylabel('Sales')
plt.title('Sales Trend by Product Category (Line Plot)')
plt.legend()
plt.grid(True)
plt.show()
colors = plt.cm.tab10(range(len(categories)))
category_sales = df.groupby('ProductCategory')['Sales'].sum()
plt.bar(category_sales.index, category_sales.values)
plt.xlabel('Product Category')
plt.ylabel('Total Sales')
plt.title('Total Sales per Product Category (Bar Plot)')
plt.xticks(rotation=45, ha='right')
plt.grid(axis='y')
plt.show()
```

- Display documentation and link to source when displaying functions
- Display image-like ndarrays as images
- Improved UX around quick charts and execution error suggestions
- Released Marketplace image for the month of February (GitHub issue)
- Python package upgrades
  - bigframes 0.19.2 -> 0.21.0
  - regex 2023.6.3 -> 2023.12.25
  - spacy 3.6.1 -> 3.7.4
  - beautifulsoup4 4.11.2 -> 4.12.3
  - tensorflow-probability 0.22.0 -> 0.23.0
  - google-cloud-language 2.9.1 -> 2.13.1
  - google-cloud-aiplatform 1.39.0 -> 1.42.1
  - transformers 4.35.2 -> 4..37.2
  - pyarrow 10.0.1 -> 14.0.2

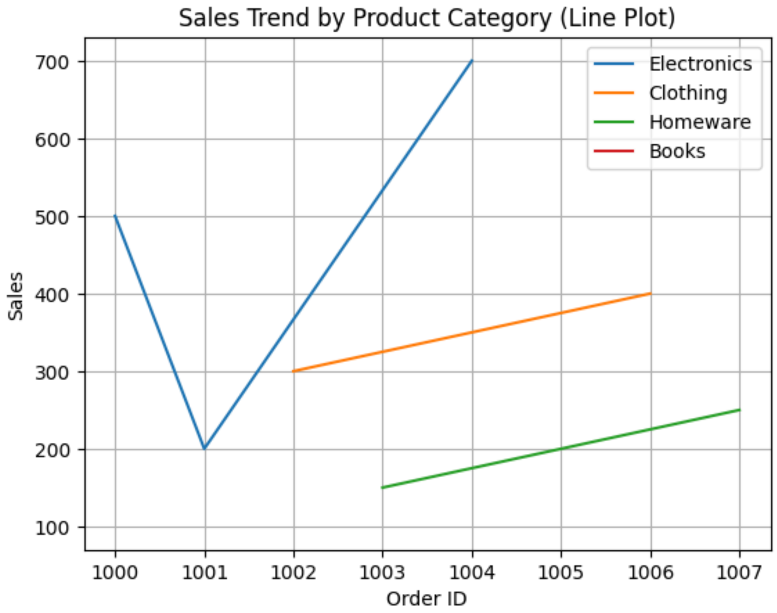
2024-01-29

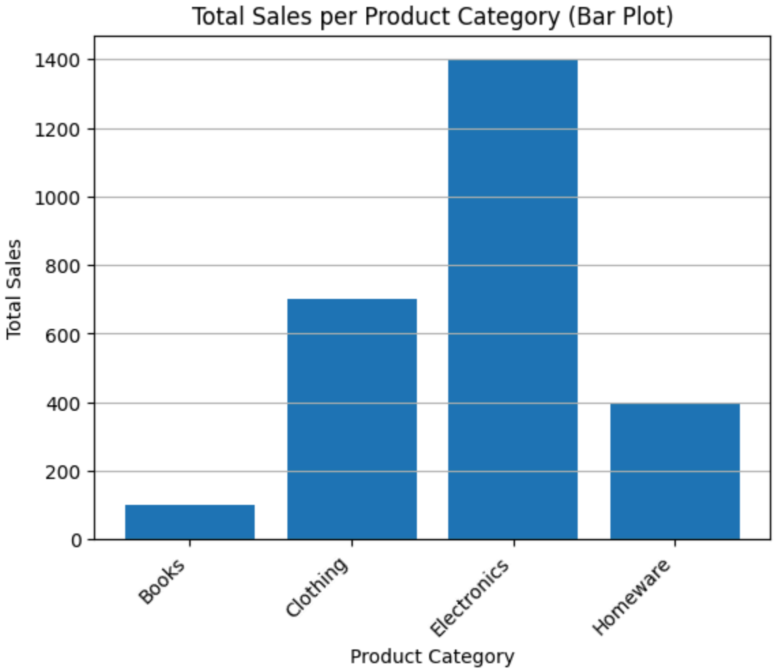
- New Kaggle Notebooks <> Colab updates! Now you can:
  - Import directly from Colab without having to download/re-upload
  - Upload via link, by pasting Google Drive or Colab URLs
  - Export & run Kaggle Notebooks on Colab with 1 click
- Try these notebooks that talk to Gemini:
  - Gemini and Stable Diffusion
  - Learning with Gemini and ChatGPT
  - Talk to Gemini with Google's Speech to Text API
  - Sell lemonade with Gemini and Sheets
  - Generate images with Gemini and Vertex
- Python package upgrades
  - google-cloud-aiplatform 1.38.1 -> 1.39.0
  - bigframes 0.18.0 -> 0.19.2
  - polars 0.17.3 -> 0.20.2
  - gdown 4.6.6 -> 4.7.3 (GitHub issue)
  - tensorflow-hub 0.15.0 -> 0.16.0
  - flax 0.7.5 -> 0.8.0
- Python package inclusions
  - sentencepiece 0.1.99

2024-01-08

- Avoid nested scrollbars for large outputs by using `google.colab.output.no_vertical_scr` Example notebook
- Fix bug where downloading models from Hugging Face could freeze
- Python package upgrades
  - huggingface-hub 0.19.4 -> 0.20.2
  - bigframes 0.17.0 -> 0.18.0

2023-12-18





4

```
import pandas as pd
import matplotlib.pyplot as plt
data = {
    'Month': ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug
    'Temperature': [5, 7, 12, 18, 22, 25, 27, 26, 22, 18, 12, 7],
    'Rainfall': [30, 25, 40, 50, 70, 80, 60, 50, 40, 30, 20, 25]
}
df = pd.DataFrame(data)
plt.figure(figsize=(10, 6))
```

- Expanded access to AI coding has arrived in Colab across 175 locales for all tiers of Colab users
- Improvements to display of ML-based inline completions (for eligible Pro/Pro+ users)
- Started a series of notebooks highlighting Gemini API capabilities
- Enable ⌘/Ctrl+L to select the full line in an editor
- Fixed bug where we weren't correctly formatting output from multiple execution results
- Python package upgrades
  - CUDA 11.8 to CUDA 12.2
  - tensorflow 2.14.0 -> 2.15.0
  - tensorboard 2.14.0 -> 2.15.0
  - keras 2.14.0 -> 2.15.0
  - Nvidia drivers 525.105.17 -> 535.104.05
  - tensorflow-gcs-config 2.14.0 -> 2.15.0
  - bigframes 0.13.0 -> 0.17.0
  - geemap 0.28.2 -> 0.29.6
  - pyarrow 9.0.0 -> 10.0.1
  - google-generativeai 0.2.2 -> 0.3.1
  - jax 0.4.20 —> 0.4.23
  - jaxlib 0.4.20 —> 0.4.23
- Python package inclusions
  - kagglehub 0.1.4
  - google-cloud-aiplatform 1.38.1

2023-11-27

- Removed warning when calling `await` to make it render as code
- Added "Run selection" to the cell context menu
- Added highlighting for the `%%python` cell magic
- Launched AI coding features for Pro/Pro+ users in more locales
- Python package upgrades
  - bigframes 0.12.0 -> 0.13.0
- Python package inclusions
  - transformers 4.35.2
  - google-generativeai 0.2.2

2023-11-08

- Launched Secrets, for safe storage of private keys on Colab (tweet)
- Fixed issue where TensorBoard would not load (#3990)
- Python package upgrades
  - lightgbm 4.0.0 -> 4.1.0
  - bigframes 0.10.0 -> 0.12.0
  - bokeh 3.2.2 -> 3.3.0
  - duckdb 0.8.1 -> 0.9.1
  - numba 0.56.4 -> 0.58.1
  - tweepy 4.13.0 -> 4.14.0
  - jax 0.4.16 -> 0.4.20
  - jaxlib 0.4.16 -> 0.4.20

2023-10-23

```
plt.plot(df['Month'], df['Temperature'], label='Temperature (°C)', 
plt.ylabel('Temperature (°C)')

ax2 = plt.twinx()
plt.plot(df['Month'], df['Rainfall'], label='Rainfall (mm)', marker
plt.ylabel('Rainfall (mm)', color='blue')

plt.xlabel('Month')
plt.title('Monthly Temperature and Rainfall Trends')
plt.xticks(rotation=45, ha='right')
plt.legend()
plt.grid(True, which='both', linestyle='--', linewidth=0.5)
plt.show()

plt.figure(figsize=(8, 6))
plt.scatter(df['Temperature'], df['Rainfall'], c='green', alpha=0.7
plt.xlabel('Temperature (°C)')
plt.ylabel('Rainfall (mm)')
plt.title('Temperature vs. Rainfall (Scatter Plot)')
plt.grid(True)
plt.show()
```

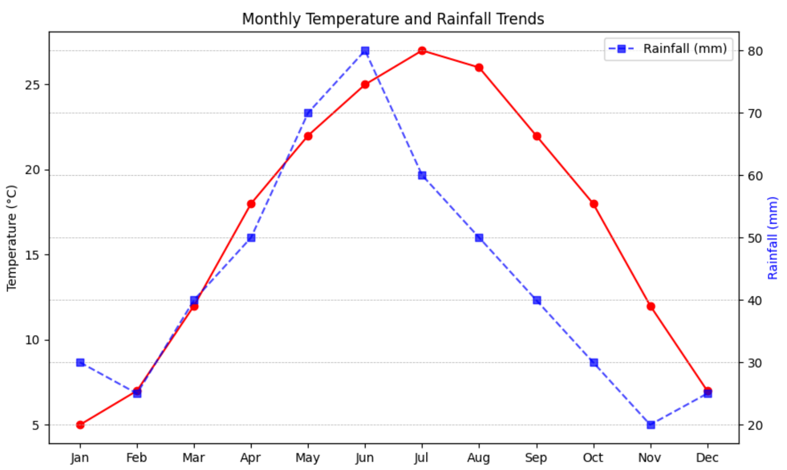
- Updated the **Open notebook** dialog for better usability and support for smaller screen sizes
- Added smart paste support for data from Google Sheets for R notebooks
- Enabled showing release notes in a tab
- Launched AI coding features for Pro/Pro+ users in Australia AU Canada CA India IN and Japan JP (tweet)
- Python package upgrades
  - earthengine-api 0.1.357 -> 0.1.375
  - flax 0.7.2 -> 0.7.4
  - geemap 0.27.4 -> 0.28.2
  - jax 0.4.14 -> 0.4.16
  - jaxlib 0.4.14 -> 0.4.16
  - keras 2.13.1 -> 2.14.0
  - tensorboard 2.13.0 -> 2.14.1
  - tensorflow 2.13.0 -> 2.14.0
  - tensorflow-gcs-config 2.13.0 -> 2.14.0
  - tensorflow-hub 0.14.0 -> 0.15.0
  - tensorflow-probability 0.20.1 -> 0.22.0
  - torch 2.0.1 -> 2.1.0
  - torchaudio 2.0.2 -> 2.1.0
  - torchtext 0.15.2 -> 0.16.0
  - torchvision 0.15.2 -> 0.16.0
  - xgboost 1.7.6 -> 2.0.0
- Python package inclusions
  - bigframes 0.10.0
  - malloy 2023.1056

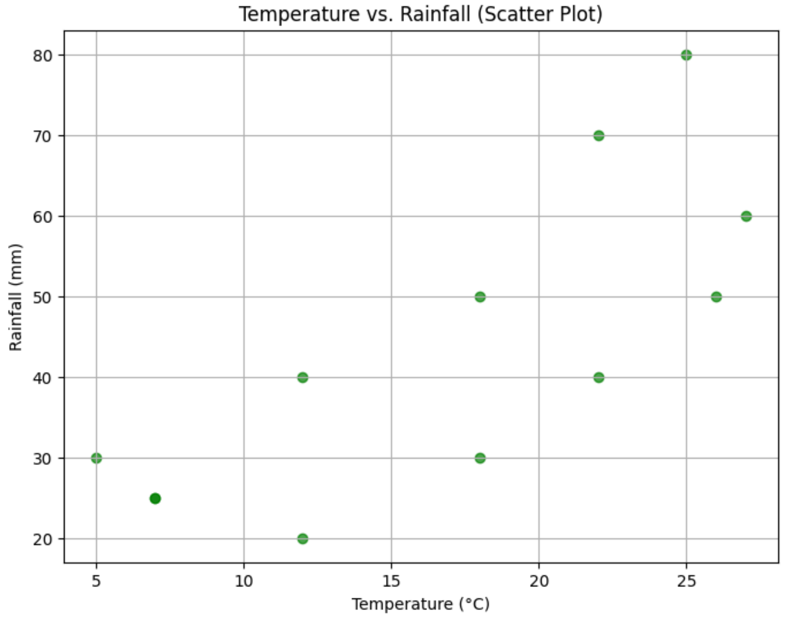
2023-09-22

- Added the ability to scope an AI generated suggestion to a specific Pandas dataframe (tweet)
- Added Colab link previews to Docs (tweet)
- Added smart paste support for data from Google Sheets
- Increased font size of dropdowns in interactive forms
- Improved rendering of the notebook when printing
- Python package upgrades
  - tensorflow 2.12.0 -> 2.13.0
  - tensorboard 2.12.3 -> 2.13.0
  - keras 2.12.0 -> 2.13.1
  - tensorflow-gcs-config 2.12.0 -> 2.13.
  - scipy 1.10.1-> 1.11.2
  - cython 0.29.6 -> 3.0.2
- Python package inclusions
  - geemap 0.26.0

2023-08-18

- Added "Change runtime type" to the menu in the connection button
- Improved auto-reconnection to an already running notebook (#3764)
- Increased the specs of our highmem machines for Pro users
- Fixed `add-apt-repository` command on Ubuntu 22.04 runtime (#3867)
- Python package upgrades
  - bokeh 2.4.3 -> 3.2.2





5

```
import matplotlib.pyplot as plt

months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', '
sales = [1000, 1200, 1500, 1800, 2100, 2300, 2000, 1800, 1600, 1400
```

  - cmake 3.25.2 -> 3.27.2
  - cryptography 3.4.8 -> 41.0.3
  - dask 2022.12.1 -> 2023.8.0
  - distributed 2022.12.1 -> 2023.8.0
  - earthengine-api 0.1.358 -> 0.1.364
  - flax 0.7.0 -> 0.7.2
  - ipython-sql 0.4.0 -> 0.5.0
  - jax 0.4.13 -> 0.4.14
  - jaxlib 0.4.13 -> 0.4.14
  - lightgbm 3.3.5 -> 4.0.0
  - mkl 2019.0 -> 2023.2.0
  - notebook 6.4.8 -> 6.5.5
  - numpy 1.22.4 -> 1.23.5
  - opencv-python 4.7.0.72 -> 4.8.0.76
  - pillow 8.4.0 -> 9.4.0
  - plotly 5.13.1 -> 5.15.0
  - prettytable 0.7.2 -> 3.8.0
  - pytensor 2.10.1 -> 2.14.2
  - spacy 3.5.4 -> 3.6.1
  - statsmodels 0.13.5 -> 0.14.0
  - xarray 2022.12.0 -> 2023.7.0
- Python package inclusions
  - PyDrive2 1.6.3

2023-07-21

- Launched auto-plotting for dataframes, available using the chart button that shows up alongside datatables (post)

- Added a menu to the table of contents to support running a section or collapsing/expanding sections (post)

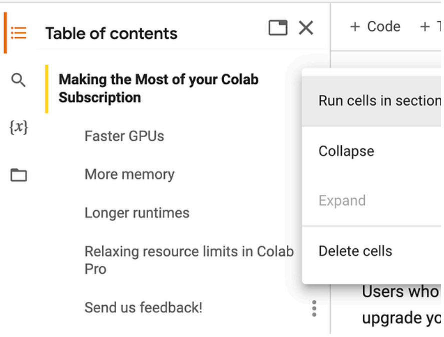
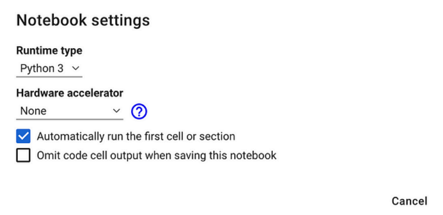


Table of contents

Making the Most of your Colab Subscription
- Faster GPUs
- More memory
- Longer runtimes
- Relaxing resource limits in Colab Pro
- Send us feedback!

- Added an option to automatically run the first cell or section, available under Edit -> Notebook settings (post)


Notebook settings

Runtime type: Python 3

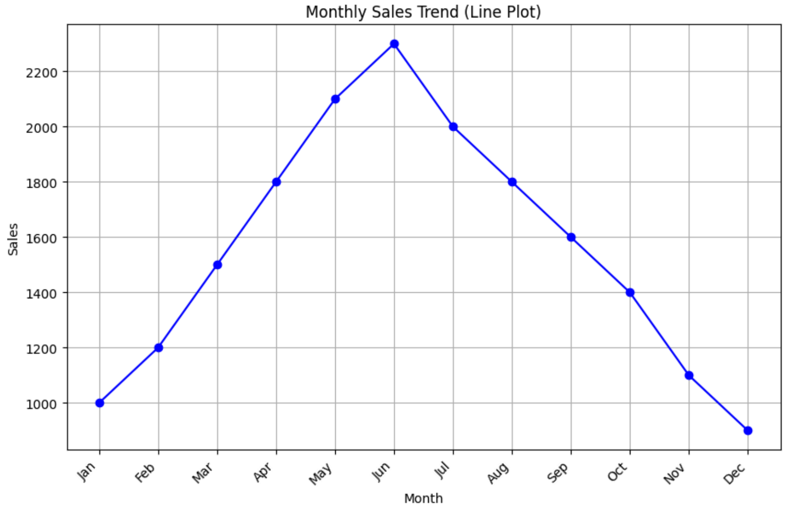
Hardware accelerator: None

☑ Automatically run the first cell or section

☐ Omit code cell output when saving this notebook

- Launched Pro/Pro+ to Algeria, Argentina, Chile, Ecuador, Egypt, Ghana, Kenya, Malaysia, Nepal, Nigeria, Peru, Rwanda, Saudi Arabia, South Africa, Sri Lanka, Tunisia, and Ukraine (tweet)
- Added a command, "Toggle tab moves focus" for toggling tab trapping in the editor (Tools -> Command palette, "Toggle tab moves focus")
- Fixed issue where `files.upload()` was sometimes returning an incorrect

```
plt.figure(figsize=(10, 6))
plt.plot(months, sales, marker='o', linestyle='-', color='blue')
plt.xlabel('Month')
plt.ylabel('Sales')
plt.title('Monthly Sales Trend (Line Plot)')
plt.xticks(rotation=45, ha='right')
plt.grid(True)
plt.show()
```



6

filename (#1550)

- Fixed f-string syntax highlighting bug (#3802)
- Disabled ambiguous characters highlighting for commonly used LaTeX characters (#3648)
- Upgraded Ubuntu from 20.04 LTS to 22.04 LTS
- Updated the Colab Marketplace VM image
- Python package upgrades:
  - autograd 1.6.1 -> 1.6.2
  - drivefs 76.0 -> 77.0
  - flax 0.6.11 -> 0.7.0
  - earthengine-api 0.1.357 -> 0.1.358
  - GDAL 3.3.2->3.4.3
  - google-cloud-bigquery-storage 2.20.0 -> 2.22.2
  - gspread-dataframe 3.0.8 -> 3.3.1
  - holidays 0.27.1 -> 0.29
  - jax 0.4.10 -> jax 0.4.13
  - jaxlib 0.4.10 -> jax 0.4.13
  - jupyterlab-widgets 3.0.7 -> 3.0.8
  - nbformat 5.9.0 -> 5.9.1
  - opencv-python-headless 4.7.0.72 -> 4.8.0.74
  - pygame 2.4.0 -> 2.5.0
  - spacy 3.5.3 -> 3.5.4
  - SQLAlchemy 2.0.16 -> 2.0.19
  - tabulate 0.8.10 -> 0.9.0
  - tensorflow-hub 0.13.0 -> 0.14.0

2023-06-23

- Launched AI coding features to subscribed users starting with Pro+ users in the US (tweet, post)
- Added the Kernel Selector in the Notebook Settings (tweet)
- Fixed double space trimming issue in markdown #3766
- Fixed run button indicator not always centered #3609
- Fixed inconsistencies for automatic indentation on multi-line #3697
- Upgraded Python from 3.10.11 to 3.10.12
- Python package updates:
  - duckdb 0.7.1 -> 0.8.1
  - earthengine-api 0.1.350 -> 0.1.357
  - flax 0.6.9 -> 0.6.11
  - google-cloud-bigquery 3.9.0 -> 3.10.0
  - google-cloud-bigquery-storage 2.19.1 -> 2.20.0
  - grpcio 1.54.0 -> 1.56.0
  - holidays 0.25 -> 0.27.1
  - nbformat 5.8.0 -> 5.9.0
  - prophet 1.1.3 -> 1.1.4
  - pydata-google-auth 1.7.0 -> 1.8.0
  - spacy 3.5.2 -> 3.5.3
  - tensorboard 2.12.2 -> 2.12.3
  - xgboost 1.7.5 -> 1.7.6
- Python package inclusions:
  - gcsfs 2023.6.0
  - geopandas 0.13.2
  - google-cloud-bigquery-connection 1.12.0
  - google-cloud-functions 1.13.0

```
import pandas as pd

# Assuming property_data is your DataFrame or load it from CSV
# property_data = pd.read_csv("property_data.csv")

# Example DataFrame creation (replace with your actual data loadi
property_data = pd.DataFrame({
    'property_id': [1, 2, 3, 4, 5],
    'location': ['A', 'B', 'A', 'C', 'B'],
    'number_of_bedrooms': [3, 4, 5, 3, 4],
    'area_in_square_feet': [1500, 1800, 2000, 1700, 2200],
    'listing_price': [200000, 250000, 300000, 220000, 280000]
})

# 1. Average listing price of properties in each location
average_price_by_location = property_data.groupby('location')['lis
average_price_by_location.columns = ['location', 'average_listing_

# 2. Number of properties with more than four bedrooms
properties_more_than_four_bedrooms = property_data[property_data['
number_of_properties_more_than_four_bedrooms = pd.DataFrame({'numb

# 3. Property with the largest area
property_with_largest_area = property_data.loc[property_data['area

# Printing the results
print("Average listing price of properties in each location:")
print(average_price_by_location)
print()

print("Number of properties with more than four bedrooms:")
print(number_of_properties_more_than_four_bedrooms)
print()

print("Property with the largest area:")
print(property_with_largest_area)
```

```
Average listing price of properties in each location:
  location  average_listing_price
0        A               250000.0
1        B               265000.0
2        C               220000.0

Number of properties with more than four bedrooms:
   number_of_properties_more_than_four_bedrooms
0                                             1

Property with the largest area:
  property_id location  number_of_bedrooms  area_in_square_feet
4           5        B                   4                 2200
```

7

```
import matplotlib.pyplot as plt
import pandas as pd

# Example DataFrame creation (replace with your actual data loading
sales_data = pd.DataFrame({
    'date': pd.date_range(start='2024-01-01', periods=30),
    'sales': [100, 120, 130, 110, 150, 140, 160, 170, 180, 200,
              220, 230, 250, 240, 260, 270, 280, 300, 320, 330,
              350, 340, 360, 370, 380, 400, 420, 430, 450, 440]
```

  - grpc-google-iam-v1 0.12.6
  - multidict 6.0.4
  - tensorboard-data-server 0.7.1

2023-06-02

- Released the new site colab.google
- Published Colab's Docker runtime image to us-docker.pkg.dev/colab-images/public/runtime (tweet, instructions)
- Launched support for Google children accounts (tweet)
- Launched DagsHub integration (tweet, post)
- Upgraded to Monaco Editor Version 0.37.1
- Fixed various Vim keybinding bugs
- Fixed issue where the N and P letters sometimes couldn't be typed (#3664)
- Fixed rendering support for compositional inputs (#3660, #3679)
- Fixed lag in notebooks with lots of cells (#3676)
- Improved support for R by adding a Runtime type notebook setting (Edit -> Notebook settings)
- Improved documentation for connecting to a local runtime (Connect -> Connect to a local runtime)
- Python package updates:
  - holidays 0.23 -> 0.25
  - jax 0.4.8 -> 0.4.10
  - jaxlib 0.4.8 -> 0.4.10
  - pip 23.0.1 -> 23.1.2
  - tensorflow-probability 0.19.0 -> 0.20.1
  - torch 2.0.0 -> 2.0.1
  - torchaudio 2.0.1 -> 2.0.2
  - torchdata 0.6.0 -> 0.6.1
  - torchtext 0.15.1 -> 0.15.2
  - torchvision 0.15.1 -> 0.15.2
  - tornado 6.2 -> 6.3.1

2023-05-05

- Released GPU type selection for paid users, allowing them to choose a preferred NVidia GPU
- Upgraded R from 4.2.3 to 4.3.0
- Upgraded Python from 3.9.16 to 3.10.11
- Python package updates:
  - attrs 22.2.0 -> attrs 23.1.0
  - earthengine-api 0.1.349 -> earthengine-api 0.1.350
  - flax 0.6.8 -> 0.6.9
  - grpcio 1.53.0 -> 1.54.0
  - nbclient 0.7.3 -> 0.7.4
  - tensorflow-datasets 4.8.3 -> 4.9.2
  - termcolor 2.2.0 -> 2.3.0
  - zict 2.2.0 -> 3.0.0

2023-04-14

- Python package updates:
  - google-api-python-client 2.70.0 -> 2.84.0
  - google-auth-oauthlib 0.4.6 -> 1.0.0
  - google-cloud-bigquery 3.4.2 -> 3.9.0
  - google-cloud-datastore 2.11.1 -> 2.15.1

```
})

# Plotting
fig, axs = plt.subplots(3, 1, figsize=(10, 15))

# Line plot
axs[0].plot(sales_data['date'], sales_data['sales'], color='blue')
axs[0].set_xlabel('Date')
axs[0].set_ylabel('Sales')
axs[0].set_title('Monthly Sales Over Time')
axs[0].tick_params(axis='x', rotation=45)
axs[0].grid(True)

# Scatter plot
axs[1].scatter(sales_data['date'], sales_data['sales'], color='gree
axs[1].set_xlabel('Date')
axs[1].set_ylabel('Sales')
axs[1].set_title('Monthly Sales Scatter Plot')
axs[1].tick_params(axis='x', rotation=45)
axs[1].grid(True)

# Bar plot
monthly_sales = sales_data.groupby(sales_data['date'].dt.strftime('
monthly_sales.plot(kind='bar', ax=axs[2], color='red')
axs[2].set_xlabel('Month')
axs[2].set_ylabel('Sales')
axs[2].set_title('Monthly Sales Bar Plot')
axs[2].tick_params(axis='x', rotation=45)
axs[2].grid(axis='y')

plt.tight_layout()
plt.show()
```

  - google-cloud-firestore 2.7.3 -> 2.11.0
  - google-cloud-language 2.6.1 -> 2.9.1
  - google-cloud-storage 2.7.0 -> 2.8.0
  - google-cloud-translate 3.8.4 -> 3.11.1
  - networkx 3.0 -> 3.1
  - notebook 6.3.0 -> 6.4.8
  - jax 0.4.7 -> 0.4.8
  - pandas 1.4.4 -> 1.5.3
  - spacy 3.5.1 -> 3.5.2
  - SQLAlchemy 1.4.47 -> 2.0.9
  - xgboost 1.7.4 -> 1.7.5

2023-03-31

- Improve bash ! syntax highlighting (GitHub issue)
- Fix bug where VIM keybindings weren't working in the file editor
- Upgraded R from 4.2.2 to 4.2.3
- Python package updates:
  - arviz 0.12.1 --> 0.15.1
  - astropy 4.3.1 --> 5.2.2
  - dopamine-rl 1.0.5 --> 4.0.6
  - gensim 3.6.0 --> 4.3.1
  - ipykernel 5.3.4 -> 5.5.6
  - ipython 7.9.0 -> 7.34.0
  - jax 0.4.4 -> 0.4.7
  - jaxlib 0.4.4 -> 0.4.7
  - jupyter_core 5.2.0 -> 5.3.0
  - keras 2.11.0 -> 2.12.0
  - lightgbm 2.2.3 -> 3.3.5
  - matplotlib 3.5.3 -> 3.7.1
  - nltk 3.7 -> 3.8.1
  - opencv-python 4.6.0.66 -> 4.7.0.72
  - plotly 5.5.0 -> 5.13.1
  - pymc 4.1.4 -> 5.1.2
  - seaborn 0.11.2 -> 0.12.2
  - spacy 3.4.4 -> 3.5.1
  - sympy 1.7.1 -> 1.11.1
  - tensorboard 2.11.2 -> 2.12.0
  - tensorflow 2.11.0 -> 2.12.0
  - tensorflow-estimator 2.11.0 -> 2.12.0
  - tensorflow-hub 0.12.0 -> 0.13.0
  - torch 1.13.1 -> 2.0.0
  - torchaudio 0.13.1 -> 2.0.1
  - torchtext 0.14.1 -> 0.15.1
  - torchvision 0.14.1 -> 0.15.1

2023-03-10

- Added the Colab editor shortcuts example notebook
- Fixed triggering of @-mention and email autocomplete for large comments (GitHub issue)
- Added View Resources to the Runtime menu
- Made file viewer images fit the view by default, resizing to original size on click
- When in VIM mode, enable copy as well as allowing propagation to monaco-vim to escape visual mode (GitHub issue)
- Upgraded CUDA 11.6.2 -> 11.8.0 and cuDNN 8.4.0.27 -> 8.7.0.84
- Upgraded Nvidia drivers 525.78.01 -> 530.30.02
- Upgraded Python 3.8.10 -> 3.9.16
- Python package updates:

  - beautifulsoup4 4.6.3 -> 4.9.3
  - bokeh 2.3.3 -> 2.4.3
  - debugpy 1.0.0 -> 1.6.6
  - Flask 1.1.4 -> 2.2.3
  - jax 0.3.25 -> 0.4.4
  - jaxlib 0.3.25 -> 0.4.4
  - Jinja2 2.11.3 -> 3.1.2
  - matplotlib 3.2.2 -> 3.5.3
  - nbconvert 5.6.1 -> 6.5.4
  - pandas 1.3.5 -> 1.4.4
  - pandas-datareader 0.9.0 -> 0.10.0
  - pandas-profiling 1.4.1 -> 3.2.0
  - Pillow 7.1.2 -> 8.4.0
  - plotnine 0.8.0 -> 0.10.1
  - scikit-image 0.18.3 -> 0.19.3
  - scikit-learn 1.0.2 -> 1.2.2
  - scipy 1.7.3 -> 1.10.1
  - setuptools 57.4.0 -> 63.4.3
  - sklearn-pandas 1.8.0 -> 2.2.0
  - statsmodels 0.12.2 -> 0.13.5
  - urllib3 1.24.3 -> 1.26.14
  - Werkzeug 1.0.1 -> 2.2.3
  - wrapt 1.14.1 -> 1.15.0
  - xgboost 0.90 -> 1.7.4
  - xlrd 1.2.0 -> 2.0.1

2023-02-17

- Show graphs of RAM and disk usage in notebook toolbar
- Copy cell links directly to the clipboard instead of showing a dialog when clicking on the link icon in the cell toolbar
- Updated the Colab Marketplace VM image
- Upgraded CUDA to 11.6.2 and cuDNN to 8.4.0.27
- Python package updates:
  - tensorflow 2.9.2 -> 2.11.0
  - tensorboard 2.9.1 -> 2.11.2
  - keras 2.9.0 -> 2.11.0
  - tensorflow-estimator 2.9.0 -> 2.11.0
  - tensorflow-probability 0.17.0 -> 0.19.0
  - tensorflow-gcs-config 2.9.0 -> 2.11.0
  - earthengine-api 0.1.339 -> 0.1.341
  - flatbuffers 1.12 -> 23.1.21
  - platformdirs 2.6.2 -> 3.0.0
  - pydata-google-auth 1.6.0 -> 1.7.0
  - python-utils 3.4.5 -> 3.5.2
  - tenacity 8.1.0 -> 8.2.1
  - tifffile 2023.1.23.1 -> 2023.2.3
  - notebook 5.7.16 -> 6.3.0
  - tornado 6.0.4 -> 6.2
  - aiohttp 3.8.3 -> 3.8.4
  - charset-normalizer 2.1.1 -> 3.0.1
  - fastai 2.7.0 -> 2.7.1
  - soundfile 0.11.0 -> 0.12.1
  - typing-extensions 4.4.0 -> 4.5.0
  - widgetsnbextension 3.6.1 -> 3.6.2
  - pydantic 1.10.4 -> 1.10.5
  - zipp 3.12.0 -> 3.13.0
  - numpy 1.21.6 -> 1.22.4
  - drivefs 66.0 -> 69.0
  - gdal 3.0.4 -> 3.3.2 GitHub issue
- Added libudunits2-dev for smoother R package installs GitHub issue

2023-02-03

8

```
import pandas as pd

# Example DataFrame creation (replace with your actual data loadir
sales_data = pd.DataFrame({
    'customer_id': [1, 2, 3, 4, 5],
    'customer_age': [25, 30, 35, 40, 45],  # Assuming this is the
    'purchase_date': pd.date_range(start='2024-05-01', periods=5)
})

# Filtering data for the past month
current_month_sales = sales_data[sales_data['purchase_date'].dt.mc

# Frequency distribution of customer ages
age_frequency_distribution = current_month_sales['customer_age'].v

print("Frequency distribution of customer ages in the past month:"
print(age_frequency_distribution)
```

```
Frequency distribution of customer ages in the past month:
Series([], Name: count, dtype: int64)
```

9

```
import numpy as np

# Assuming student_scores is your NumPy array containing student s
student_scores = np.array([
    [85, 90, 88, 92],  # Math scores
    [78, 85, 80, 88],  # Science scores
    [90, 92, 85, 89],  # English scores
    [88, 85, 90, 86]   # History scores
])

# Calculate the average score for each subject
average_scores = np.mean(student_scores, axis=0)

# Identify the subject with the highest average score
highest_average_score_subject = np.argmax(average_scores)

# Printing the results
print("Average score for each subject:", average_scores)
print("Subject with the highest average score:", highest_average_s
```

```
Average score for each subject: [85.25 88.   85.75 88.75]
Subject with the highest average score: 3
```

- Improved tooltips for pandas series to show common statistics about the series object
- Made the forms dropdown behave like an autocomplete box when it allows input
- Updated the nvidia driver from 460.32.03 to 510.47.03
- Python package updates:
    - absl-py 1.3.0 -> 1.4.0
    - bleach 5.0.1 -> 6.0.0
    - cachetools 5.2.1 -> 5.3.0
    - cmdstanpy 1.0.8 -> 1.1.0
    - dnspython 2.2.1 -> 2.3.0
    - fsspec 2022.11.0 -> 2023.1.0
    - google-cloud-bigquery-storage 2.17.0 -> 2.18.1
    - holidays 0.18 -> 0.19
    - jupyter-core 5.1.3 -> 5.2.0
    - packaging 21.3 -> 23.0
    - prometheus-client 0.15.0 -> 0.16.0
    - pyct 0.4.8 -> 0.5.0
    - pydata-google-auth 1.5.0 -> 1.6.0
    - python-slugify 7.0.0 -> 8.0.0
    - sqlalchemy 1.4.46 -> 2.0.0
    - tensorflow-io-gcs-filesystem 0.29.0 -> 0.30.0
    - tifffile 2022.10.10 -> 2023.1.23.1
    - zipp 3.11.0 -> 3.12.0
    - Pinned sqlalchemy to version 1.4.46

2023-01-12

- Added support for @-mention and email autocomplete in comments
- Improved errors when GitHub notebooks can't be loaded
- Increased color contrast for colors used for syntax highlighting in the code editor
- Added terminal access for custom GCE VM runtimes
- Upgraded Ubuntu from 18.04 LTS to 20.04 LTS (GitHub issue)
- Python package updates:
    - GDAL 2.2.2 -> 2.2.3.
    - NumPy from 1.21.5 to 1.21.6.
    - attrs 22.1.0 -> 22.2.0
    - chardet 3.0.4 -> 4.0.0
    - cloudpickle 1.6.0 -> 2.2.0
    - filelock 3.8.2 -> 3.9.0
    - google-api-core 2.8.2 -> 2.11.0
    - google-api-python-client 1.12.11 -> 2.70.0
    - google-auth-httplib2 0.0.3 -> 0.1.0
    - google-cloud-bigquery 3.3.5 -> 3.4.1
    - google-cloud-datastore 2.9.0 -> 2.11.0
    - google-cloud-firestore 2.7.2 -> 2.7.3
    - google-cloud-storage 2.5.0 -> 2.7.0
    - holidays 0.17.2 -> holidays 0.18
    - importlib-metadata 5.2.0 -> 6.0.0
    - networkx 2.8.8 -> 3.0
    - opencv-python-headless 4.6.0.66 -> 4.7.0.68
    - pip 21.1.3 -> 22.04
    - pip-tools 6.2.0 -> 6.6.2
    - prettytable 3.5.0 -> 3.6.0
    - requests 2.23.0 -> 2.25.1
    - termcolor 2.1.1 -> 2.2.0

10

```
import numpy as np

# Assuming sales_data is your NumPy array containing sales data
sales_data = np.array([
    [50, 60, 70],  # Sales data for product 1: price1, price2, pri
    [45, 55, 65],  # Sales data for product 2: price1, price2, pri
    [55, 65, 75]   # Sales data for product 3: price1, price2, pri
])

# Calculate the average price of all products
average_price = np.mean(sales_data)

# Printing the result
print("Average price of all products sold in the past month:", ave
```

```
Average price of all products sold in the past month: 60.0
```

11

```
import numpy as np

# Assuming response_times is your NumPy array containing response
response_times = np.array([10, 15, 20, 25, 30, 35, 40, 45, 50, 55]

# Calculate the percentiles
percentiles = np.percentile(response_times, [25, 50, 75])

# Printing the results
print("25th percentile (Q1):", percentiles[0])
print("50th percentile (Q2 or median):", percentiles[1])
print("75th percentile (Q3):", percentiles[2])
```

```
25th percentile (Q1): 21.25
50th percentile (Q2 or median): 32.5
75th percentile (Q3): 43.75
```

12

```
import numpy as np

# Assuming recovery_times is your NumPy array containing recovery
recovery_times = np.array([3, 5, 7, 9, 11, 13, 15, 17, 19, 21])

# Calculate the percentiles
percentiles = np.percentile(recovery_times, [10, 50, 90])

# Printing the results
print("10th percentile:", percentiles[0])
print("50th percentile (median):", percentiles[1])
print("90th percentile:", percentiles[2])
```

```
10th percentile: 4.8
50th percentile (median): 12.0
90th percentile: 19.2
```

  - torch 1.13.0 -> 1.13.1
  - torchaudio 0.13.0 -> 0.13.1
  - torchtext 0.14.0-> 0.14.1
  - torchvision 0.14.0 -> 0.14.1

2022-12-06

- Made fallback runtime version available until mid-December (GitHub issue)
- Upgraded to Python 3.8 (GitHub issue)
- Python package updates:
  - jax from 0.3.23 to 0.3.25, jaxlib from 0.3.22 to 0.3.25
  - pyarrow from 6.0.1 to 9.0.0
  - torch from 1.12.1 to 1.13.0
  - torchaudio from 0.12.1 to 0.13.0
  - torchvision from 0.13.1 to 0.14.0
  - torchtext from 0.13.1 to 0.14.0
  - xlrd from 1.1.0 to 1.2.0
  - DriveFS from 62.0.1 to 66.0.3
- Made styling of markdown tables in outputs match markdown tables in text cells
- Improved formatting for empty interactive table rows
- Fixed syntax highlighting for variables with names that contain Python keywords (GitHub issue)

2022-11-11

- Added more dark editor themes for Monaco (when in dark mode, "Editor colorization" appears as an option in the Editor tab of the Tools → Settings dialog)
- Fixed bug where collapsed forms were deleted on mobile GitHub issue
- Python package updates:
  - rpy2 from 3.4.0 to 3.5.5 (GitHub issue)
  - notebook from 5.5.0 to 5.7.16
  - tornado from 5.1.1 to 6.0.4
  - tensorflow_probability from 0.16.0 to 0.17.0
  - pandas-gbq from 0.13.3 to 0.17.9
  - protobuf from 3.17.3 to 3.19.6
  - google-api-core[grpc] from 1.31.5 to 2.8.2
  - google-cloud-bigquery from 1.21.0 to 3.3.5
  - google-cloud-core from 1.0.1 to 2.3.2
  - google-cloud-datastore from 1.8.0 to 2.9.0
  - google-cloud-firestore from 1.7.0 to 2.7.2
  - google-cloud-language from 1.2.0 to 2.6.1
  - google-cloud-storage from 1.18.0 to 2.5.0
  - google-cloud-translate from 1.5.0 to 3.8.4

2022-10-21

- Launched a single-click way to get from BigQuery to Colab to further explore query results (announcement)
- Launched Pro, Pro+, and Pay As You Go to 19 additional countries: Austria, Belgium, Bulgaria, Croatia, Cyprus, Czechia, Denmark, Estonia, Finland,

13

```
import numpy as np
from scipy import stats

# Assuming purchase_amounts is your NumPy array containing purchas
purchase_amounts = np.array([20, 30, 40, 50, 50, 60, 70, 70, 70, 8

# Calculate the mean (average) purchase amount
mean_purchase_amount = np.mean(purchase_amounts)

# Identify the mode of the purchase amounts
mode_purchase_amount = stats.mode(purchase_amounts)

# Printing the results
print("Mean (average) purchase amount:", mean_purchase_amount)
print("Mode of purchase amounts:", mode_purchase_amount.mode.item(
```

```
Mean (average) purchase amount: 54.0
Mode of purchase amounts: 70
```

14

Greece, Hungary, Latvia, Lithuania, Norway, Portugal, Romania, Slovakia, Slovenia, and Sweden (tweet)

- Updated jax from 0.3.17 to 0.3.23, jaxlib from 0.3.15 to 0.3.22, TensorFlow from 2.8.2 to 2.9.2, CUDA from 11.1 to 11.2, and cuDNN from 8.0 to 8.1 (backend-info)
- Added a `readonly` option to `drive.mount`
- Fixed bug where Xarray was not working (GitHub issue)
- Modified Markdown parsing to ignore block quote symbol within MathJax (GitHub issue)

2022-09-30

- Launched Pay As You Go, allowing premium GPU access without requiring a subscription
- Added vim and tcllib to our runtime image
- Fixed bug where open files were closed on kernel disconnect (GitHub issue)
- Fixed bug where the play button/execution indicator was not clickable when scrolled into the cell output (GitHub issue)
- Updated the styling for form titles so that they avoid obscuring the code editor
- Created a GitHub repo, backend-info, with the latest apt-list.txt and pip-freeze.txt files for the Colab runtime (GitHub issue)
- Added `files.upload_file(filename)` to upload a file from the browser to the runtime with a specified filename

2022-09-16

- Upgraded pymc from 3.11.0 to 4.1.4, jax from 0.3.14 to 0.3.17, jaxlib from 0.3.14 to 0.3.15, fsspec from 2022.8.1 to 2022.8.2
- Modified our save flow to avoid persisting Drive filenames as titles in notebook JSON
- Updated our Terms of Service
- Modified the `Jump to Cell` command to locate the cursor at the end of the command palette input (`Jump to cell` in Tools → Command palette in a notebook with section headings)
- Updated the styling of the Drive notebook comment UI
- Added support for terminating your runtime from code: `python from google.colab import runtime runtime.unassign()`
- Added regex filter support to the Recent notebooks dialog
- Inline google.colab.files.upload JS to fix `files.upload()` not working (GitHub issue)

2022-08-26

- Upgraded PyYAML from 3.13 to 6.0 (GitHub issue), drivefs from 61.0.3 to 62.0.1
- Upgraded TensorFlow from 2.8.2 to 2.9.1 and ipywidgets from 7.7.1 to 8.0.1 but

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats

# Creating a DataFrame with the provided data
data = {
    'age': [23, 23, 27, 27, 39, 41, 47, 49, 50, 52, 54, 54, 56, 57
    'fat_percentage': [9.5, 26.5, 7.8, 17.8, 31.4, 25.9, 27.4, 27.
}

df = pd.DataFrame(data)

# Calculating mean, median, and standard deviation
mean_age = df['age'].mean()
median_age = df['age'].median()
std_dev_age = df['age'].std()

mean_fat_percentage = df['fat_percentage'].mean()
median_fat_percentage = df['fat_percentage'].median()
std_dev_fat_percentage = df['fat_percentage'].std()

print("Age:")
print("Mean:", mean_age)
print("Median:", median_age)
print("Standard Deviation:", std_dev_age)

print("\nFat Percentage:")
print("Mean:", mean_fat_percentage)
print("Median:", median_fat_percentage)
print("Standard Deviation:", std_dev_fat_percentage)

# Drawing boxplots
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
sns.boxplot(y=df['age'])
plt.title('Boxplot of Age')

plt.subplot(1, 2, 2)
sns.boxplot(y=df['fat_percentage'])
plt.title('Boxplot of Fat Percentage')

plt.tight_layout()
plt.show()

# Drawing scatter plot
plt.figure(figsize=(8, 6))
sns.scatterplot(x='age', y='fat_percentage', data=df)
plt.title('Scatter Plot of Age vs Fat Percentage')
plt.xlabel('Age')
plt.ylabel('Fat Percentage')
plt.show()

# Drawing Q-Q plot
plt.figure(figsize=(8, 6))
stats.probplot(df['fat_percentage'], dist="norm", plot=plt)
plt.title('Q-Q Plot of Fat Percentage')
plt.xlabel('Theoretical quantiles')
plt.ylabel('Ordered Values')
plt.show()
```

rolled both back due to a number of user reports (GitHub issue, GitHub issue)

- Stop persisting inferred titles in notebook JSON (GitHub issue)
- Fix bug in background execution which affected some Pro+ users (GitHub issue)
- Fix bug where `Download as .py` incorrectly handled text cells ending in a double quote
- Fix bug for Pro and Pro+ users where we weren't honoring the preference (Tools → Settings) to use a temporary scratch notebook as the default landing page
- Provide undo/redo for scratch cells
- When writing ipynb files, serialize empty multiline strings as `[]` for better consistency with JupyterLab

2022-08-11

- Upgraded ipython from 5.5.0 to 7.9.0, fbprophet 0.7 to prophet 1.1, tensorflow-datasets from 4.0.1 to 4.6.0, drivefs from 60.0.2 to 61.0.3, pytorch from 1.12.0 to 1.12.1, numba from 0.51 to 0.56, and lxml from 4.2.0 to 4.9.1
- Loosened our `requests` version requirement (GitHub issue)
- Removed support for TensorFlow 1
- Added Help → Report Drive abuse for Drive notebooks
- Fixed indentation for Python lines ending in [
- Modified styling of tables in Markdown to left-align them rather than centering them
- Fixed special character replacement when copying interactive tables as Markdown
- Fixed ansi 8-bit color parsing (GitHub issue)
- Configured logging to preempt transitive imports and other loading from implicitly configuring the root logger
- Modified forms to use a value of `None` instead of causing a parse error when clearing raw and numeric-typed form fields

2022-07-22

- Update scipy from 1.4.1 to 1.7.3, drivefs from 59.0.3 to 60.0.2, pytorch from 1.11 to 1.12, jax & jaxlib from 0.3.8 to 0.3.14, opencv-python from 4.1.2.30 to 4.6.0.66, spaCy from 3.3.1 to 3.4.0, and dlib from 19.18.0 to 19.24.0
- Fix `Open in tab` doc link which was rendering incorrectly (GitHub issue)
- Add a preference for the default tab orientation to the Site section of the settings menu under Tools → Settings
- Show a warning for `USE_AUTH_EPHEM` usage when running authenticate_user on a TPU runtime (code)

2022-07-01

- Add a preference for code font to the settings menu under Tools → Settings
- Update drivefs from 58.0.3 to 59.0.3 and spacy from 2.2.4 to 3.3.1

- Allow display_data and execute_result text outputs to wrap, matching behavior of JupyterLab (does not affect stream outputs/print statements).
- Improve LSP handling of some magics, esp. %%writefile (GitHub issue).
- Add a FAQ entry about the mount Drive button behavior and include link buttons for each FAQ entry.
- Fix bug where the notebook was sometimes hidden behind other tabs on load when in single pane view.
- Fix issue with inconsistent scrolling when an editor is in multi-select mode.
- Fix bug where clicking on a link in a form would navigate away from the notebook
- Show a confirmation dialog before performing Replace all from the Find and replace pane.

2022-06-10

- Update drivefs from 57.0.5 to 58.0.3 and tensorflow from 2.8.0 to 2.8.2
- Support more than 100 repos in the GitHub repo selector shown in the open dialog and the clone to GitHub dialog
- Show full notebook names on hover in the open dialog
- Improve the color contrast for links, buttons, and the `ipywidgets.Accordion` widget in dark mode

2022-05-20

- Support URL params for linking to some common pref settings: force_theme=dark, force_corgi_mode=1, force_font_size=14. Params forced by URL are not persisted unless saved using Tools → Settings.
- Add a class `markdown-google-sans` to allow Markdown to render in Google Sans
- Update monaco-vim from 0.1.19 to 0.3.4
- Update drivefs from 55.0.3 to 57.0.5, jax from 0.3.4 to 0.3.8, and jaxlib from 0.3.2 to 0.3.7

2022-04-29

- Added 🦀 mode (under Miscellaneous in Tools → Settings)
- Added "Disconnect and delete runtime" option to the menu next to the Connect button
- Improved rendering of filter options in an interactive table
- Added git-lfs to the base image
- Updated torch from 1.10.0 to 1.11.0, jupyter-core from 4.9.2 to 4.10.0, and cmake from 3.12.0 to 3.22.3
- Added more details to our FAQ about unsupported uses (using proxies, downloading torrents, etc.)
- Fixed issue with apt-get dependencies

2022-04-15

- Add an option in the file browser to show hidden files.

15

```
import numpy as np
student_scores=np.array([[90,88,91,92],
[76,88,80,90],
[77,88,90,78],
[78,88,87,89]])
avg=np.mean(student_scores,axis=0)
high=np.argmax(avg)
highest=['maths','science','english','History'][high]
print(avg)
print(highest)
```

```
[80.25 88.    87.    87.25]
science
```

16

```
import numpy as np
sales_data=np.array([[200,400,500],[600,700,900],[400,800,400]])
avg=np.mean(sales_data)
print(avg)
```

```
544.4444444444445
```

17

```
import numpy as np
data=np.array([10000,20000,25000,30000])
total=np.sum(data)
print(total)
q1=data[0]
q4=data[-1]
per=((q4-q1)/q1)*100
print(per)
```

- Upgrade gdown from 4.2.0 to 4.4.0, google-api-core[grpc] from 1.26.0 to 1.31.5, and pytz from 2018.4 to 2022.1

2022-03-25

- Launched Pro/Pro+ to 12 additional countries: Australia, Bangladesh, Colombia, Hong Kong, Indonesia, Mexico, New Zealand, Pakistan, Philippines, Singapore, Taiwan, and Vietnam
- Added `google.colab.auth.authenticate_serv` to support using Service Account keys
- Update jax from 0.3.1 to 0.3.4 & jaxlib from 0.3.0 to 0.3.2
- Fixed an issue with Twitter previews of notebooks shared as GitHub Gists

2022-03-10

- Launched Pro/Pro+ to 10 new countries: Ireland, Israel, Italy, Morocco, the Netherlands, Poland, Spain, Switzerland, Turkey, and the United Arab Emirates
- Launched support for scheduling notebooks for Pro+ users
- Fixed bug in interactive datatables where filtering by number did not work
- Finished removing the python2 kernelspec

2022-02-25

- Made various accessibility improvements to the header
- Fix bug with forms run:auto where a form field change would trigger multiple runs
- Minor updates to the bigquery example notebook and snippet
- Include background execution setting in the sessions dialog for Pro+ users
- Update tensorflow-probability from 0.15 to 0.16
- Update jax from 0.2.25 to 0.3.1 & jaxlib from 0.1.71 to 0.3.0

2022-02-11

- Improve keyboard navigation for the open dialog
- Fix issue where nvidia-smi stopped reporting resource utilization for some users who were modifying the version of nvidia used
- Update tensorflow from 2.7 to 2.8, keras from 2.7 to 2.8, numpy from 1.19.5 to 1.21.5, tables from 3.4.4 to 3.7.0

2022-02-04

- Improve UX for opening content alongside your notebook, such as files opened from the file browser. This includes a multi-pane view and drag-drop support
- Better Twitter previews when sharing example Colab notebooks and notebooks opened from GitHub Gists
- Update pandas from 1.1.5 to 1.3.5
- Update openpyxl from 2.5.9 to 3.0.0 and pyarrow from 3.0.0 to 6.0.0
- Link to the release notes from the Help menu

```
85000
200.0
```

18

```
import pandas as pd
import numpy as np
property_data=pd.DataFrame({'property_ID':[1,2,3,4,5],
                'location':['A','B','C','D','B'],
                'no of bedrooms':[1,2,3,4,5],
                'area in square':[2000, 2500, 3000, 1800, 3500],
                'listing price':[200,390,300,400,890]})

avg=property_data.groupby('location')['listing price'].mean()
avg.columns=['location','average lisitng price']
print(avg)
no_of_bedrooms=(property_data['no of bedrooms']>4).sum()
print("/n number of bedrooms with more than four bedrooms:",no_of_
larger=property_data.loc[property_data['area in square'].idxmax()]
print(larger)
n
```

```
location
A    200.0
B    640.0
C    300.0
D    400.0
Name: listing price, dtype: float64
/n number of bedrooms with more than four bedrooms: 1
property_ID          5
location             B
no of bedrooms       5
area in square    3500
listing price      890
Name: 4, dtype: object
-------------------------------------------------------------
--------------
NameError                                 Traceback (most
recent call last)
<ipython-input-19-80ef42a7c006> in <cell line: 16>()
     14 larger=property_data.loc[property_data['area in
square'].idxmax()]
     15 print(larger)
---> 16 n
```

Next steps: Explain error

19

2022-01-28

- Add a copy button to data tables
- Python LSP support for better completions and code diagnostics. This can be configured in the Editor Settings (Tools → Settings)
- Update gspread examples in our documentation
- Update gdown from 3.6 to 4.2

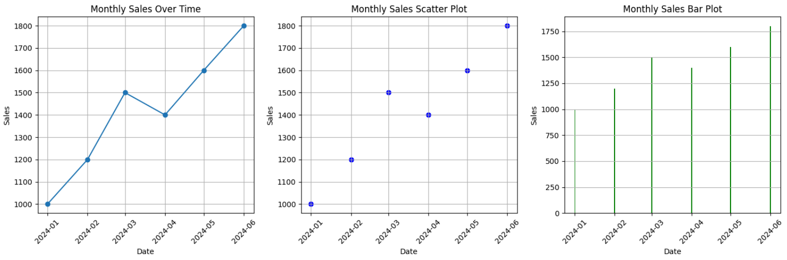
2022-01-21

- New documentation for the `google.colab` package
- Show GPU RAM in the resource usage tab
- Improved security for mounting Google Drive which disallows mounting Drive from accounts other than the one currently executing the notebook

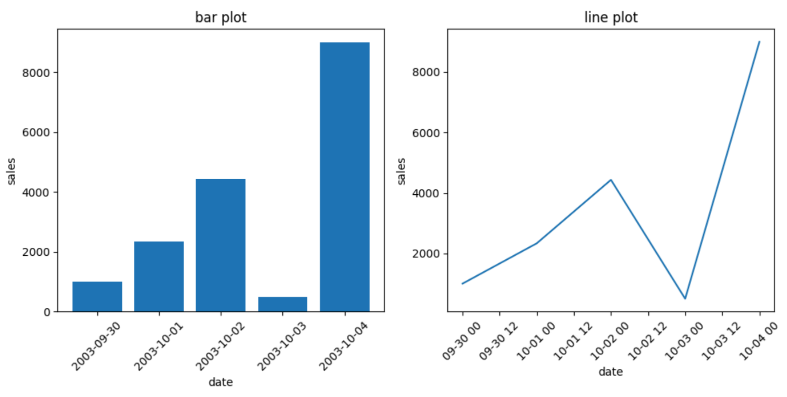
2022-01-14

- Add a preference (Tools → Settings) to use a temporary scratch notebook as the default landing page
- Fix bug where `/` and `:` weren't working in VIM mode
- Update gspread from 3.0 to 3.4
- Update the Colab Marketplace VM image

```
import matplotlib.pyplot as plt
import pandas as pd

# Sample data (replace with your actual data)
dates = pd.date_range(start='2024-01-01', end='2024-06-01', freq='
sales = [1000, 1200, 1500, 1400, 1600, 1800]

# Create DataFrame
sales_df = pd.DataFrame({'Date': dates, 'Sales': sales})

# Plotting
plt.figure(figsize=(15, 5))

# Line plot
plt.subplot(1, 3, 1)
plt.plot(sales_df['Date'], sales_df['Sales'], marker='o', linestyl
plt.title('Monthly Sales Over Time')
plt.xlabel('Date')
plt.ylabel('Sales')
plt.xticks(rotation=45)
plt.grid(True)

# Scatter plot
plt.subplot(1, 3, 2)
plt.scatter(sales_df['Date'], sales_df['Sales'], color='blue')
plt.title('Monthly Sales Scatter Plot')
plt.xlabel('Date')
plt.ylabel('Sales')
plt.xticks(rotation=45)
plt.grid(True)

# Bar plot
plt.subplot(1, 3, 3)
plt.bar(sales_df['Date'], sales_df['Sales'], color='green')
plt.title('Monthly Sales Bar Plot')
plt.xlabel('Date')
plt.ylabel('Sales')
plt.xticks(rotation=45)
plt.grid(axis='y')

plt.tight_layout()
plt.show()
```



```
import matplotlib.pyplot as plt
import pandas as pd
dates=pd.date_range(start='2003-09-30',end='2003-10-04')
sales=[1000,2333,4434,500,9000]
sales_df=pd.DataFrame({'date':dates,'sales':sales})
plt.figure(figsize=(10,5))
plt.subplot(1,2,1)
plt.bar(sales_df['date'],sales_df['sales'])
plt.xlabel("date")
plt.ylabel("sales")
plt.xticks(rotation=45)

plt.title("bar plot")
plt.subplot(1,2,2)
plt.plot(sales_df['date'],sales_df['sales'])
plt.xlabel("date")
plt.ylabel("sales")
plt.xticks(rotation=45)
plt.title("line plot")
plt.tight_layout()
plt.show()
```



21

```
import numpy as np

data = np.array([10000, 20000, 25000, 30000])

# Calculate the total sales for the year
total_sales = np.sum(data)
print("Total sales for the year:", total_sales)

# Extract sales for the first and fourth quarters
q1_sales = data[0]
q4_sales = data[-1]

# Calculate the percentage increase from the first quarter to the
percentage_increase = ((q4_sales - q1_sales) / q1_sales) * 100
print("Percentage increase in sales from the first quarter to the
     Total sales for the year: 85000
     Percentage increase in sales from the first quarter to the fou
```

22

```
import numpy as np

fuel_efficiency = np.array([20, 30, 40, 23])

# Calculate the average fuel efficiency across all car models
average_fuel_efficiency = np.mean(fuel_efficiency)
print("Average fuel efficiency:", average_fuel_efficiency)

# Select the fuel efficiency values for two car models
fuel_efficiency_car1 = fuel_efficiency[0]
fuel_efficiency_car2 = fuel_efficiency[2]

# Calculate the percentage improvement in fuel efficiency between
percentage_improvement = ((fuel_efficiency_car2 - fuel_efficiency_
print("Percentage improvement in fuel efficiency between two car m
```

```
Average fuel efficiency: 28.25
Percentage improvement in fuel efficiency between two car mode
```

23

```
items_price = [6, 5, 10, 2]
quantities = [3, 2, 4, 1]
discount = 10
tax = 8

# Calculate the total cost before any discount or tax
total_cost = sum(price * quantity for price, quantity in zip(items

# Calculate the discount amount
discount_amt = (discount / 100) * total_cost

# Calculate the total cost after applying the discount
total_cost_after_discount = total_cost - discount_amt

# Calculate the tax amount
tax_amt = (tax / 100) * total_cost_after_discount

# Calculate the total cost after applying the tax
total_cost_after_tax = total_cost_after_discount + tax_amt

print("Total cost after tax:", total_cost_after_tax)
print("Total cost after discount:", total_cost_after_discount)
```

```
Total cost after tax: 68.04
Total cost after discount: 63.0
```

24

```
import pandas as pd

order_data = pd.DataFrame({
    'CustomerID': [1, 2, 1, 3, 2],
    'OrderDate': ['2022-01-01', '2022-01-02', '2022-01-01', '2022-
    'ProductName': ['ProductA', 'ProductB', 'ProductA', 'ProductC'
    'OrderQuantity': [3, 5, 2, 1, 4]
})

# 1. Total number of orders made by each customer
total_orders_per_customer = order_data.groupby('CustomerID').size(

# 2. Average order quantity for each product
avg_order_quantity_per_product = order_data.groupby('ProductName')

# 3. Earliest and latest order dates in the dataset
earliest_order_date = order_data['OrderDate'].min()
latest_order_date = order_data['OrderDate'].max()

print("Total number of orders made by each customer:")
print(total_orders_per_customer)
print("\nAverage order quantity for each product:")
print(avg_order_quantity_per_product)
print("\nEarliest order date:", earliest_order_date)
print("Latest order date:", latest_order_date)
```

```
Total number of orders made by each customer:
CustomerID
1    2
2    2
3    1
dtype: int64

Average order quantity for each product:
ProductName
ProductA    2.5
ProductB    4.5
ProductC    1.0
Name: OrderQuantity, dtype: float64

Earliest order date: 2022-01-01
Latest order date: 2022-01-03
```

25

```
import pandas as pd

sales_data = pd.DataFrame({
    'ProductID': [1, 2, 1, 3, 2, 3, 4, 5, 4, 5],
    'ProductName': ['ProductA', 'ProductB', 'ProductA', 'ProductC'
    'QuantitySold': [10, 15, 8, 20, 12, 18, 5, 25, 6, 22]
})

# Group by product name and sum the quantities sold for each produ
product_sales = sales_data.groupby('ProductName')['QuantitySold'].

# Sort the summed quantities in descending order to find the top-s
sorted_sales = product_sales.sort_values(ascending=False)

# Select the top 5 products
top_5_products = sorted_sales.head(5)

print("Top 5 products sold the most in the past month:")
print(top_5_products)
```

```
Top 5 products sold the most in the past month:
ProductName
ProductE    47
ProductC    38
ProductB    27
ProductA    18
ProductD    11
Name: QuantitySold, dtype: int64
```

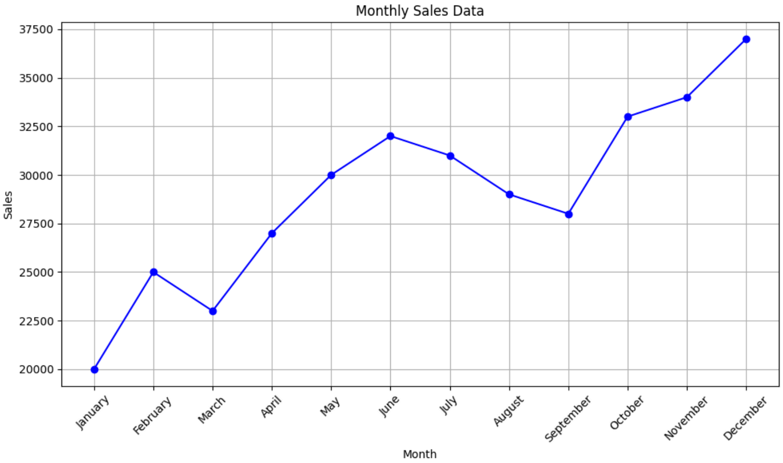
26

```
import pandas as pd

# Example: Loading the property_data DataFrame from a CSV file
# Make sure to replace 'path_to_your_csv_file.csv' with the actual 
# property_data = pd.read_csv('path_to_your_csv_file.csv')

# Sample data for demonstration purposes (Remove this when using ac
property_data = pd.DataFrame({
    'property_id': [1, 2, 3, 4, 5],
    'location': ['Location A', 'Location B', 'Location A', 'Locatio
    'bedrooms': [3, 5, 4, 2, 6],
    'area_sqft': [1500, 2000, 1800, 1200, 2500],
    'listing_price': [300000, 500000, 450000, 250000, 600000]
})

# 1. Average listing price of properties in each location
average_listing_price = property_data.groupby('location')['listing_

# 2. Number of properties with more than four bedrooms
properties_with_more_than_four_bedrooms = property_data[property_da

# 3. Property with the largest area
property_with_largest_area = property_data.loc[property_data['area_

# Display the results
print("Average listing price of properties in each location:")
print(average_listing_price)

print("\nNumber of properties with more than four bedrooms:")
print(properties_with_more_than_four_bedrooms)

print("\nProperty with the largest area:")
print(property_with_largest_area)
```

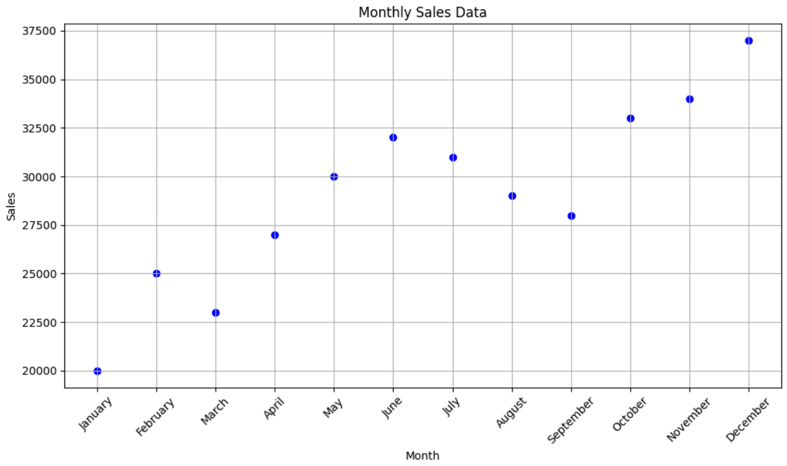
```
Average listing price of properties in each location:
location
Location A    375000.0
Location B    550000.0
Location C    250000.0
Name: listing_price, dtype: float64

Number of properties with more than four bedrooms:
2

Property with the largest area:
property_id               5
location         Location B
bedrooms                  6
area_sqft              2500
listing_price        600000
Name: 4, dtype: object
```

27

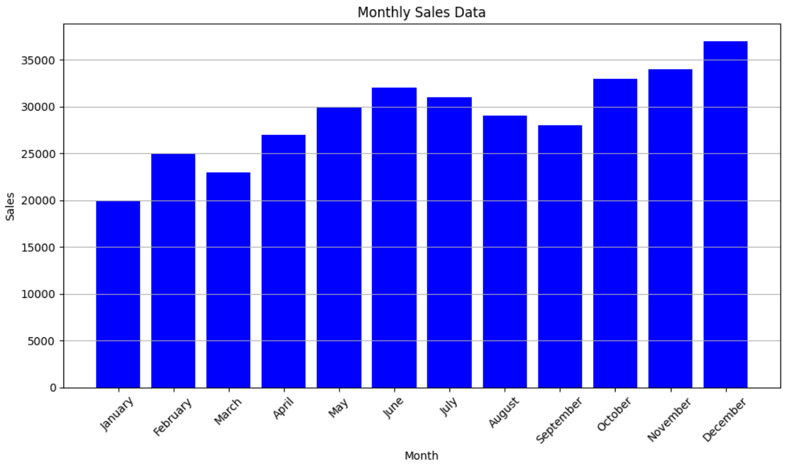
```
import matplotlib.pyplot as plt

# Sample monthly sales data
months = ['January', 'February', 'March', 'April', 'May', 'June',
          'July', 'August', 'September', 'October', 'November', 'D
sales = [20000, 25000, 23000, 27000, 30000, 32000, 31000, 29000, 2

# Create a line plot
plt.figure(figsize=(10, 6))
plt.plot(months, sales, marker='o', linestyle='-', color='b')
plt.title('Monthly Sales Data')
plt.xlabel('Month')
plt.ylabel('Sales')
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()  # Adjust layout to make room for rotated x-axi
plt.show()
```



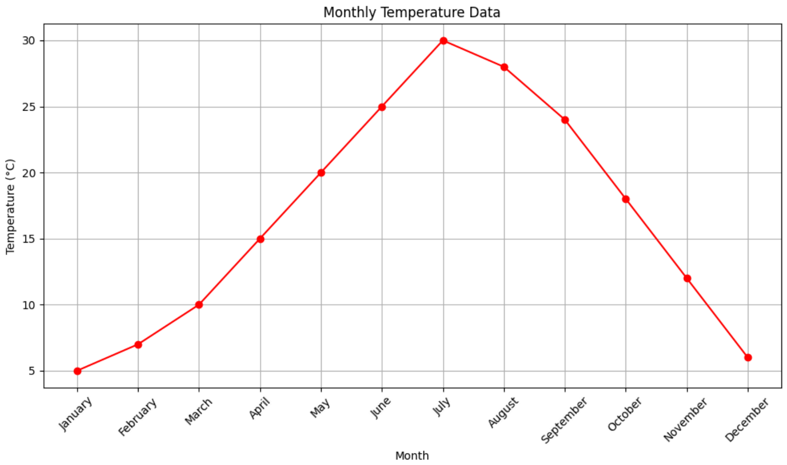
```
import matplotlib.pyplot as plt

# Sample monthly sales data
months = ['January', 'February', 'March', 'April', 'May', 'June',
          'July', 'August', 'September', 'October', 'November', 'D
sales = [20000, 25000, 23000, 27000, 30000, 32000, 31000, 29000, 2

# Create a scatter plot
plt.figure(figsize=(10, 6))
plt.scatter(months, sales, color='b')
plt.title('Monthly Sales Data')
plt.xlabel('Month')
plt.ylabel('Sales')
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()  # Adjust layout to make room for rotated x-axi
plt.show()
```



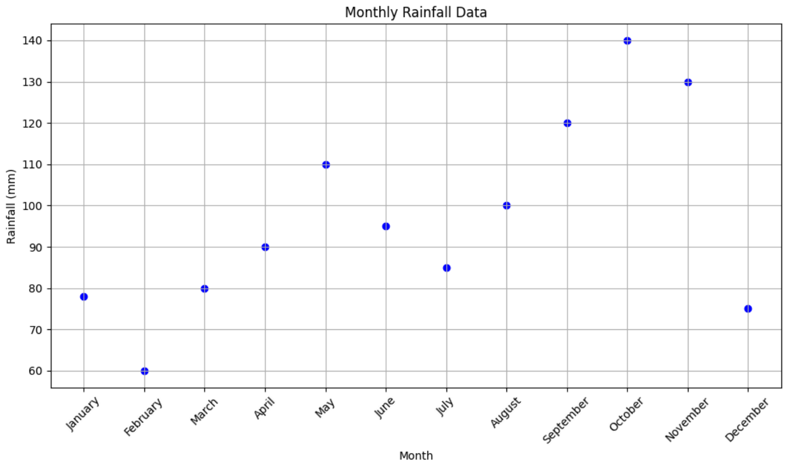
```
import matplotlib.pyplot as plt

# Sample monthly sales data
months = ['January', 'February', 'March', 'April', 'May', 'June',
          'July', 'August', 'September', 'October', 'November', 'D
sales = [20000, 25000, 23000, 27000, 30000, 32000, 31000, 29000, 2

# Create a bar plot
plt.figure(figsize=(10, 6))
plt.bar(months, sales, color='b')
plt.title('Monthly Sales Data')
plt.xlabel('Month')
plt.ylabel('Sales')
plt.grid(axis='y')
plt.xticks(rotation=45)
plt.tight_layout()  # Adjust layout to make room for rotated x-axi
plt.show()
```



```
import matplotlib.pyplot as plt

# Sample monthly temperature data (in degrees Celsius)
months = ['January', 'February', 'March', 'April', 'May', 'June',
          'July', 'August', 'September', 'October', 'November', 'D
temperature = [5, 7, 10, 15, 20, 25, 30, 28, 24, 18, 12, 6]

# Create a line plot for the monthly temperature data
plt.figure(figsize=(10, 6))
plt.plot(months, temperature, marker='o', linestyle='-', color='r'
plt.title('Monthly Temperature Data')
plt.xlabel('Month')
plt.ylabel('Temperature (°C)')
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()  # Adjust layout to make room for rotated x-axi
plt.show()
```

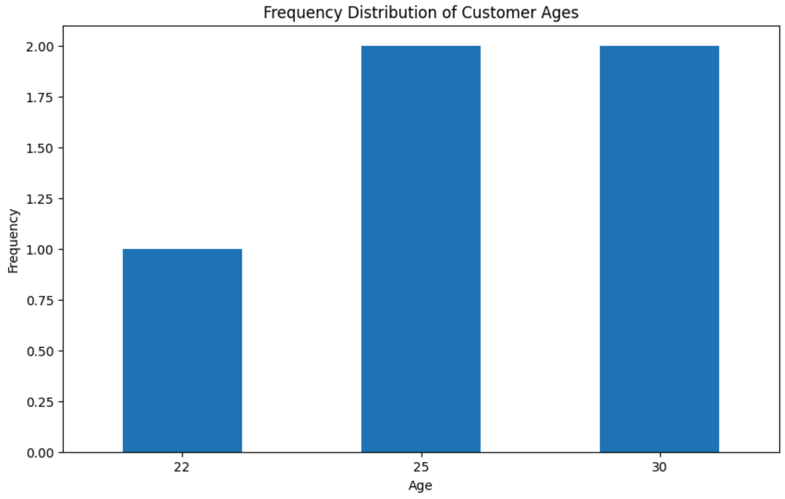


```
import matplotlib.pyplot as plt

# Sample monthly rainfall data (in mm)
months = ['January', 'February', 'March', 'April', 'May', 'June',
          'July', 'August', 'September', 'October', 'November', 'D
rainfall = [78, 60, 80, 90, 110, 95, 85, 100, 120, 140, 130, 75]

# Create a scatter plot for the monthly rainfall data
plt.figure(figsize=(10, 6))
plt.scatter(months, rainfall, color='b')
plt.title('Monthly Rainfall Data')
plt.xlabel('Month')
plt.ylabel('Rainfall (mm)')
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()  # Adjust layout to make room for rotated x-axi
plt.show()
```



```
import pandas as pd
import matplotlib.pyplot as plt

# Sample data frame initialization (replace with your actual data)
data = {
    'customer_id': [1, 2, 3, 4, 5],
    'age': [25, 30, 22, 30, 25],
    'purchase_amount': [100, 150, 200, 100, 300],
    'purchase_date': ['2023-05-01', '2023-05-02', '2023-05-03', '2
}

df = pd.DataFrame(data)

# Calculate the frequency distribution of ages
age_distribution = df['age'].value_counts().sort_index()

# Display the frequency distribution
print(age_distribution)

# Plot the distribution for visualization
plt.figure(figsize=(10, 6))
age_distribution.plot(kind='bar')
plt.title('Frequency Distribution of Customer Ages')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.xticks(rotation=0)
plt.show()
```

```
age
22    1
25    2
30    2
Name: count, dtype: int64
```



34

```
import pandas as pd

# Sample DataFrame (replace with your actual DataFrame)
data = {
    'post_id': [1, 2, 3, 4, 5],
    'likes': [10, 5, 15, 10, 20]
}

df = pd.DataFrame(data)

# Calculate frequency distribution of likes
like_distribution = df['likes'].value_counts().sort_index()

# Print the frequency distribution
print("Frequency distribution of likes:")
print(like_distribution)
```

```
Frequency distribution of likes:
likes
5     1
10    2
15    1
20    1
Name: count, dtype: int64
```

35

```
from collections import Counter
import re

# Sample data (replace with your actual data)
reviews = [
    "This product is great! I love it.",
    "The quality of this product is excellent.",
    "Not satisfied with the product. It broke after a week."
]

# Function to preprocess text (remove punctuation and convert to l
def preprocess_text(text):
    text = text.lower()  # Convert to lowercase
    text = re.sub(r'[^\w\s]', '', text)  # Remove punctuation
    return text

# Tokenize and preprocess the reviews
words = []
for review in reviews:
    review = preprocess_text(review)
    words.extend(review.split())

# Calculate frequency distribution of words
word_counts = Counter(words)

# Print the frequency distribution
print("Frequency distribution of words:")
for word, frequency in word_counts.items():
    print(f"{word}: {frequency} times")
```

```
Frequency distribution of words:
this: 2 times
product: 3 times
is: 2 times
great: 1 times
i: 1 times
```

```
love: 1 times
it: 2 times
the: 2 times
quality: 1 times
of: 1 times
excellent: 1 times
not: 1 times
satisfied: 1 times
with: 1 times
broke: 1 times
after: 1 times
a: 1 times
week: 1 times
```

36

```
# Function to plot bar graph of top N words
def plot_top_words(word_counts, top_n):
    top_words = word_counts.most_common(top_n)
    df_top_words = pd.DataFrame(top_words, columns=['Word', 'Freque
    plt.figure(figsize=(10, 6))
    plt.bar(df_top_words['Word'], df_top_words['Frequency'], color=
    plt.xlabel('Words')
    plt.ylabel('Frequency')
    plt.title(f'Top {top_n} Most Frequent Words')
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()

# Load dataset from CSV file
def load_dataset(file_path):
    df = pd.read_csv(file_path)
    return df

# Main function to analyze feedback dataset
def analyze_feedback(data_file, top_n):
    # Load dataset
    df = load_dataset(data_file)

    # Preprocess text data
    all_words = []
    for feedback in df['feedback']:
        preprocessed_words = preprocess_text(feedback)
        all_words.extend(preprocessed_words)

    # Calculate frequency distribution of words
    word_counts = Counter(all_words)

    # Display the top N most frequent words
    print(f"Top {top_n} most frequent words:")
    for word, frequency in word_counts.most_common(top_n):
        print(f"{word}: {frequency} times")

    # Plot bar graph of top N words
    plot_top_words(word_counts, top_n)

# Example usage
if __name__ == "__main__":
    data_file = 'data.csv'  # Replace with your actual CSV file pat
    top_n = int(input("Enter the number of top words to display and
    analyze_feedback(data_file, top_n)
```

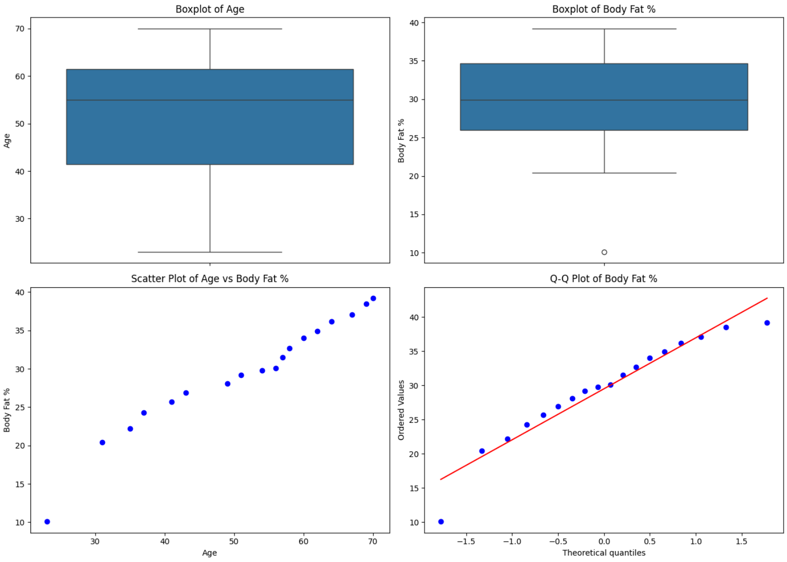
```
Enter the number of top words to display and visualize: 8
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most
recent call last)
<ipython-input-38-956847bdef5a> in <cell line: 42>()
     43     data_file = 'data.csv'  # Replace with your
actual CSV file path
     44     top_n = int(input("Enter the number of top words
to display and visualize: "))
---> 45     analyze_feedback(data_file, top_n)

                        6 frames
/usr/local/lib/python3.10/dist-packages/pandas/io/common.py
in get_handle(path_or_buf, mode, encoding, compression,
memory_map, is_text, errors, storage_options)
    857         if ioargs.encoding and "b" not in
ioargs.mode:
    858             # Encoding
--> 859             handle = open(
```

Next steps: Explain error

37

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import scipy.stats as stats
data = {
    'Age': [23, 31, 35, 37, 41, 43, 49, 51, 54, 56, 57, 58, 60, 62
    '%Fat': [10.1, 20.4, 22.2, 24.3, 25.7, 26.9, 28.1, 29.2, 29.8,
}
df = pd.DataFrame(data)
mean_age = df['Age'].mean()
median_age = df['Age'].median()
std_age = df['Age'].std()

mean_fat = df['%Fat'].mean()
median_fat = df['%Fat'].median()
std_fat = df['%Fat'].std()

print(f'Mean Age: {mean_age:.2f}, Median Age: {median_age}, Std De
print(f'Mean %Fat: {mean_fat:.2f}, Median %Fat: {median_fat}, Std

plt.figure(figsize=(14, 10))
plt.subplot(2, 2, 1)
sns.boxplot(y=df['Age'])
plt.title('Boxplot of Age')
plt.ylabel('Age')
plt.subplot(2, 2, 2)
sns.boxplot(y=df['%Fat'])
plt.title('Boxplot of Body Fat %')
plt.ylabel('Body Fat %')
plt.subplot(2, 2, 3)
plt.scatter(df['Age'], df['%Fat'], color='blue')
plt.title('Scatter Plot of Age vs Body Fat %')
plt.xlabel('Age')
plt.ylabel('Body Fat %')
plt.subplot(2, 2, 4)
stats.probplot(df['%Fat'], dist="norm", plot=plt)
plt.title('Q-Q Plot of Body Fat %')
plt.tight_layout()
plt.show()
```

```
Mean Age: 51.50, Median Age: 55.0, Std Dev Age: 13.79
Mean %Fat: 29.49, Median %Fat: 29.950000000000003, Std Dev %Fa
```



38

```
import numpy as np
daily_sales = np.array([100, 120, 90, 110, 130, 95, 105, 115, 125, 
105])
mean_sales = np.mean(daily_sales)
variance_sales = np.var(daily_sales)
print("Variance of Daily Sales:", variance_sales)
import numpy as np
study_hours = np.array([2, 3, 1, 4, 5, 2, 3, 4, 5, 1])
exam_scores = np.array([65, 70, 60, 75, 80, 65, 70, 75, 80, 60])
covariance_study_exam = np.cov(study_hours, exam_scores)[0, 1]
print("Covariance between Study Hours and Exam Scores:", covariance_
```

```
Variance of Daily Sales: 156.66666666666666
Covariance between Study Hours and Exam Scores: 11.1111111111
```

39

```
import numpy as np
monthly_expenses = np.array([
[50000, 55000, 60000, 52000, 58000],
[40000, 42000, 38000, 41000, 45000],
[30000, 32000, 31000, 30000, 33000]
])
variance_expenses = np.var(monthly_expenses, axis=1)
print("Variance of Monthly Expenses for Each Department:")
print(variance_expenses)
covariance_matrix_expenses = np.cov(monthly_expenses)
print("\nCovariance Matrix of Monthly Expenses:")
print(covariance_matrix_expenses)
```

```
Variance of Monthly Expenses for Each Department:
[13600000.  5360000.  1360000.]

Covariance Matrix of Monthly Expenses:
[[17000000.   500000.  3500000.]
 [  500000.  6700000.  2450000.]
 [ 3500000.  2450000.  1700000.]]
```

40

```
import numpy as np
response_times = np.array([20, 25, 30, 35, 40, 45, 50, 55, 60, 65,
percentiles_25th = np.percentile(response_times, 25)
percentiles_50th = np.percentile(response_times, 50)
percentiles_75th = np.percentile(response_times, 75)
print("25th Percentile:", percentiles_25th)
print("50th Percentile (Median):", percentiles_50th)
print("75th Percentile:", percentiles_75th)
import numpy as np
recovery_times = np.array([10, 15, 20, 25, 30, 35, 40, 45, 50, 55,
percentiles_10th = np.percentile(recovery_times, 10)
percentiles_50th = np.percentile(recovery_times, 50)
percentiles_90th = np.percentile(recovery_times, 90)
print("10th Percentile:", percentiles_10th)
print("50th Percentile (Median):", percentiles_50th)
print("90th Percentile:", percentiles_90th)
```

```
25th Percentile: 32.5
50th Percentile (Median): 45.0
75th Percentile: 57.5
10th Percentile: 15.0
50th Percentile (Median): 35.0
90th Percentile: 55.0
```

41

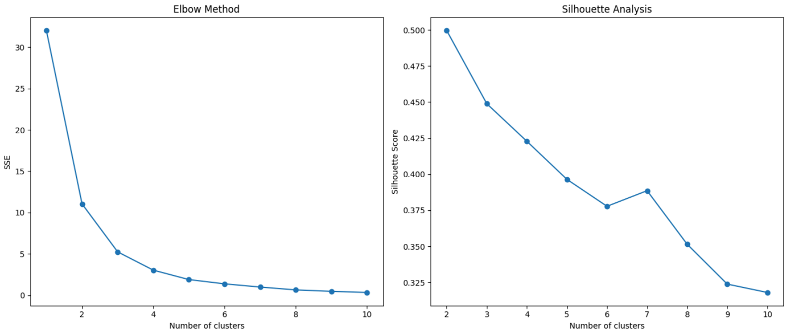
```
import numpy as np
daily_temperatures = np.array([25.5, 26.0, 24.8, 25.2, 25.7, 26.5,
29.5, 24.0, 26.5, 27.8])
variance_temperatures = np.var(daily_temperatures)
print("Variance of Daily Temperatures:", variance_temperatures)
z_scores = (daily_temperatures - np.mean(daily_temperatures)) / np
z_score_threshold = 2
potential_outliers = np.abs(z_scores) > z_score_threshold
print("\nPotential Outliers:")
print(daily_temperatures[potential_outliers])
```

```
Variance of Daily Temperatures: 2.021155555555556

Potential Outliers:
[29.5]
```

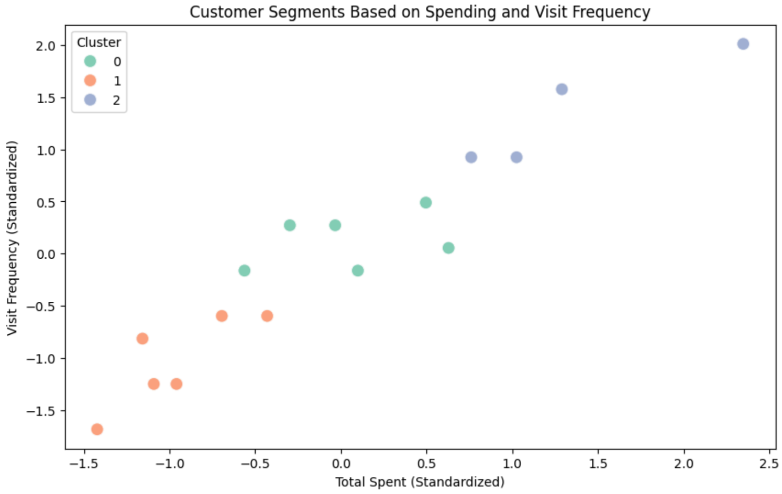
```
42
```

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

# Step 1: Simulated transaction data for multiple stores
data = {
    'CustomerID': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 1
    'TotalSpent': [500, 150, 2000, 350, 3000, 700, 1200, 400, 800, 
    'VisitFrequency': [5, 3, 15, 7, 20, 8, 12, 5, 10, 12, 18, 11, 8
}

# Create a DataFrame
df = pd.DataFrame(data)

# Step 2: Data Preprocessing
# No missing values to handle in this simulated data

# Normalize the data
scaler = StandardScaler()
df[['TotalSpent', 'VisitFrequency']] = scaler.fit_transform(df[['To

# Step 3: Determine the optimal number of clusters using Elbow Meth
sse = []
silhouette_scores = []
K_range = range(1, 11)

for k in K_range:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(df[['TotalSpent', 'VisitFrequency']])
    sse.append(kmeans.inertia_)
    if k > 1:
        score = silhouette_score(df[['TotalSpent', 'VisitFrequency'
        silhouette_scores.append(score)

# Plot Elbow Method
plt.figure(figsize=(14, 6))
plt.subplot(1, 2, 1)
plt.plot(K_range, sse, marker='o')
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('SSE')

# Plot Silhouette Analysis
plt.subplot(1, 2, 2)
plt.plot(range(2, 11), silhouette_scores, marker='o')
plt.title('Silhouette Analysis')
plt.xlabel('Number of clusters')
plt.ylabel('Silhouette Score')

plt.tight_layout()
plt.show()

# Step 4: Apply K-Means Clustering with the optimal number of clust
optimal_clusters = 3
kmeans = KMeans(n_clusters=optimal_clusters, random_state=42)
df['Cluster'] = kmeans.fit_predict(df[['TotalSpent', 'VisitFrequenc

# Step 5: Visualize the clusters
plt.figure(figsize=(10, 6))
plt.figure(figsize=(10, 6))
sns.scatterplot(x='TotalSpent', y='VisitFrequency', hue='Cluster',
plt.title('Customer Segments Based on Spending and Visit Frequency'
plt.xlabel('Total Spent (Standardized)')
plt.ylabel('Visit Frequency (Standardized)')
plt.legend(title='Cluster')
plt.show()

# Step 6: Print the cluster centers (de-normalize to get original v
cluster_centers = scaler.inverse_transform(kmeans.cluster_centers_)
print("Cluster Centers (Original Scale):")
print(cluster_centers)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmear
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmear
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmear
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmear
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmear
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmear
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmear
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmear
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmear
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmear
  warnings.warn(
```



```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmear
  warnings.warn(
```



```
Cluster Centers (Original Scale):
[[1266.66666667   11.33333333]
 [ 500.            6.        ]
 [2250.           17.        ]]
```

43

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_
import seaborn as sns
import matplotlib.pyplot as plt

# Load the dataset
# Assuming the dataset is named 'treatment_data.csv'
# df = pd.read_csv('treatment_data.csv')

# Sample data creation for illustration purposes
np.random.seed(42)
df = pd.DataFrame({
    'age': np.random.randint(20, 80, 100),
    'gender': np.random.choice(['Male', 'Female'], 100),
    'blood_pressure': np.random.randint(80, 180, 100),
    'cholesterol': np.random.randint(150, 300, 100),
    'outcome': np.random.choice(['Good', 'Bad'], 100)
})

# Encode categorical variables (if any)
df['gender'] = df['gender'].map({'Male': 0, 'Female': 1})
df['outcome'] = df['outcome'].map({'Good': 1, 'Bad': 0})

# Splitting the data into features (X) and target (y)
X = df.drop('outcome', axis=1)
y = df['outcome']

# Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=

# Standardizing the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Building the KNN model
knn = KNeighborsClassifier(n_neighbors=5)  # You can choose a diffe
knn.fit(X_train, y_train)

# Making predictions on the test set
y_pred = knn.predict(X_test)

# Evaluating the model
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
```