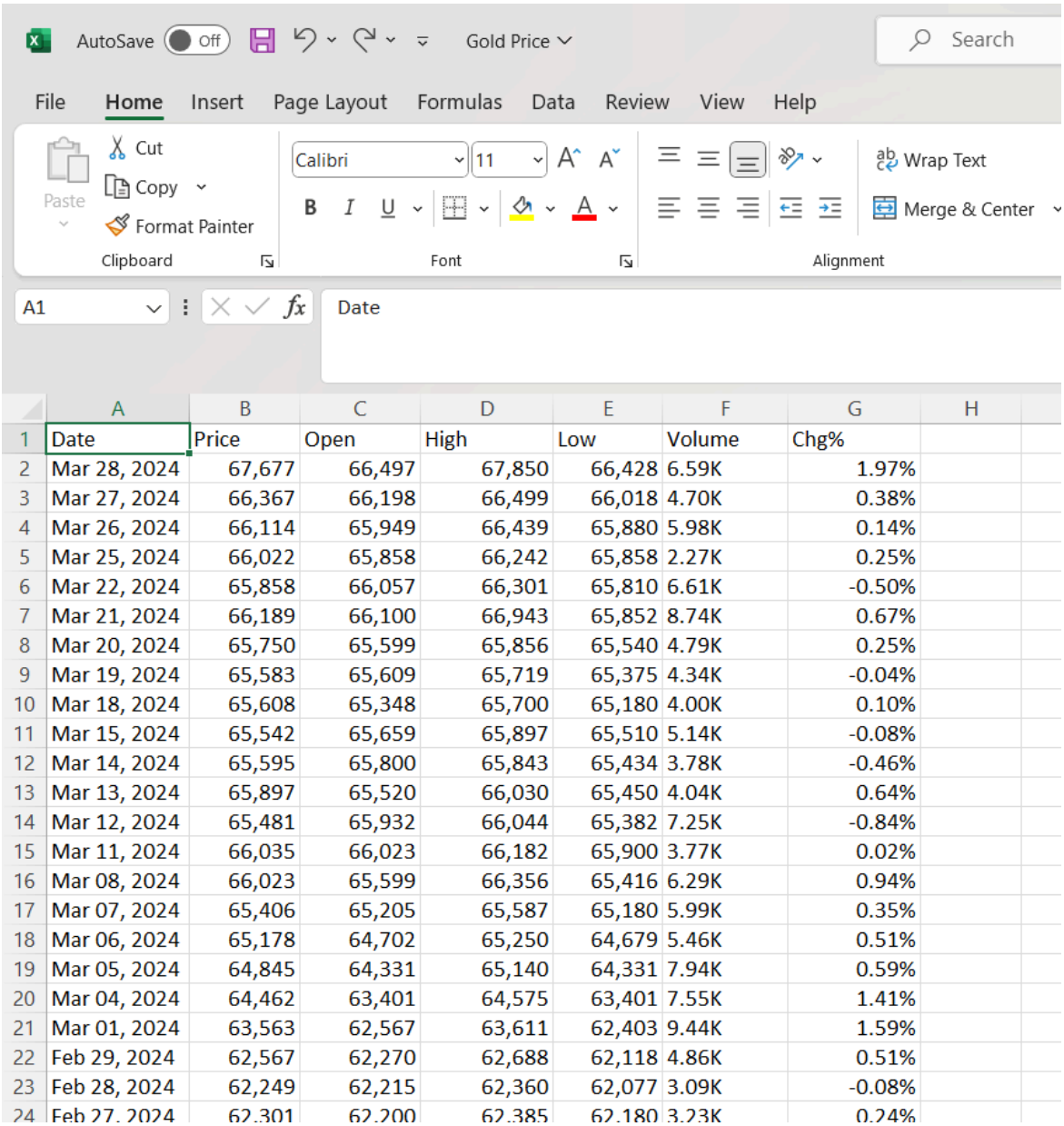
Gold Price Prediction and Recommendation

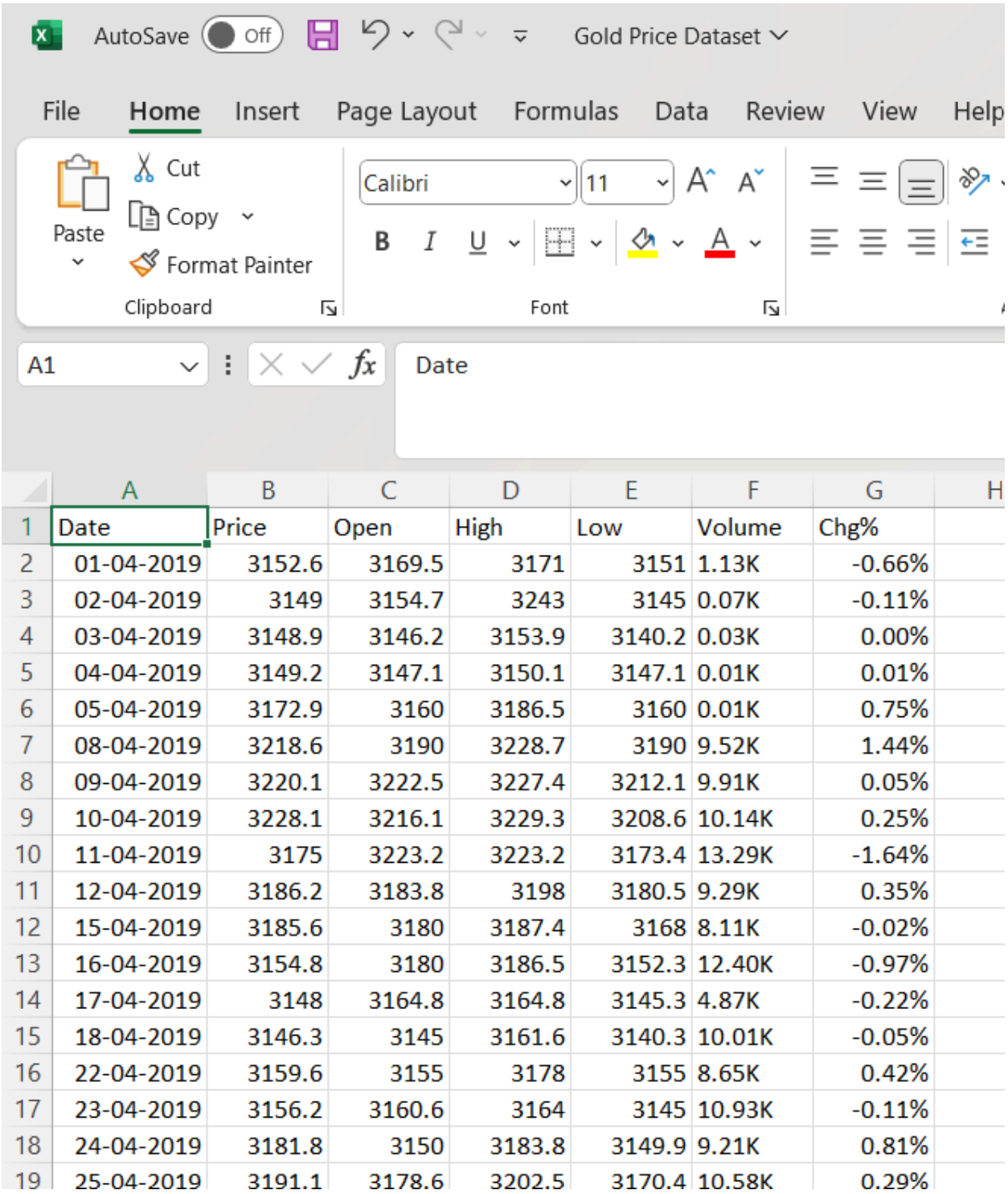
Submitted by : Athira K P
Data Science

Original gold price csv file:



	A	B	C	D	E	F	G	H
1	Date	Price	Open	High	Low	Volume	Chg%	
2	Mar 28, 2024	67,677	66,497	67,850	66,428	6.59K	1.97%	
3	Mar 27, 2024	66,367	66,198	66,499	66,018	4.70K	0.38%	
4	Mar 26, 2024	66,114	65,949	66,439	65,880	5.98K	0.14%	
5	Mar 25, 2024	66,022	65,858	66,242	65,858	2.27K	0.25%	
6	Mar 22, 2024	65,858	66,057	66,301	65,810	6.61K	-0.50%	
7	Mar 21, 2024	66,189	66,100	66,943	65,852	8.74K	0.67%	
8	Mar 20, 2024	65,750	65,599	65,856	65,540	4.79K	0.25%	
9	Mar 19, 2024	65,583	65,609	65,719	65,375	4.34K	-0.04%	
10	Mar 18, 2024	65,608	65,348	65,700	65,180	4.00K	0.10%	
11	Mar 15, 2024	65,542	65,659	65,897	65,510	5.14K	-0.08%	
12	Mar 14, 2024	65,595	65,800	65,843	65,434	3.78K	-0.46%	
13	Mar 13, 2024	65,897	65,520	66,030	65,450	4.04K	0.64%	
14	Mar 12, 2024	65,481	65,932	66,044	65,382	7.25K	-0.84%	
15	Mar 11, 2024	66,035	66,023	66,182	65,900	3.77K	0.02%	
16	Mar 08, 2024	66,023	65,599	66,356	65,416	6.29K	0.94%	
17	Mar 07, 2024	65,406	65,205	65,587	65,180	5.99K	0.35%	
18	Mar 06, 2024	65,178	64,702	65,250	64,679	5.46K	0.51%	
19	Mar 05, 2024	64,845	64,331	65,140	64,331	7.94K	0.59%	
20	Mar 04, 2024	64,462	63,401	64,575	63,401	7.55K	1.41%	
21	Mar 01, 2024	63,563	62,567	63,611	62,403	9.44K	1.59%	
22	Feb 29, 2024	62,567	62,270	62,688	62,118	4.86K	0.51%	
23	Feb 28, 2024	62,249	62,215	62,360	62,077	3.09K	-0.08%	
24	Feb 27, 2024	62,301	62,200	62,385	62,180	3.23K	0.24%	

The cleaned gold price dataset in CSV format:

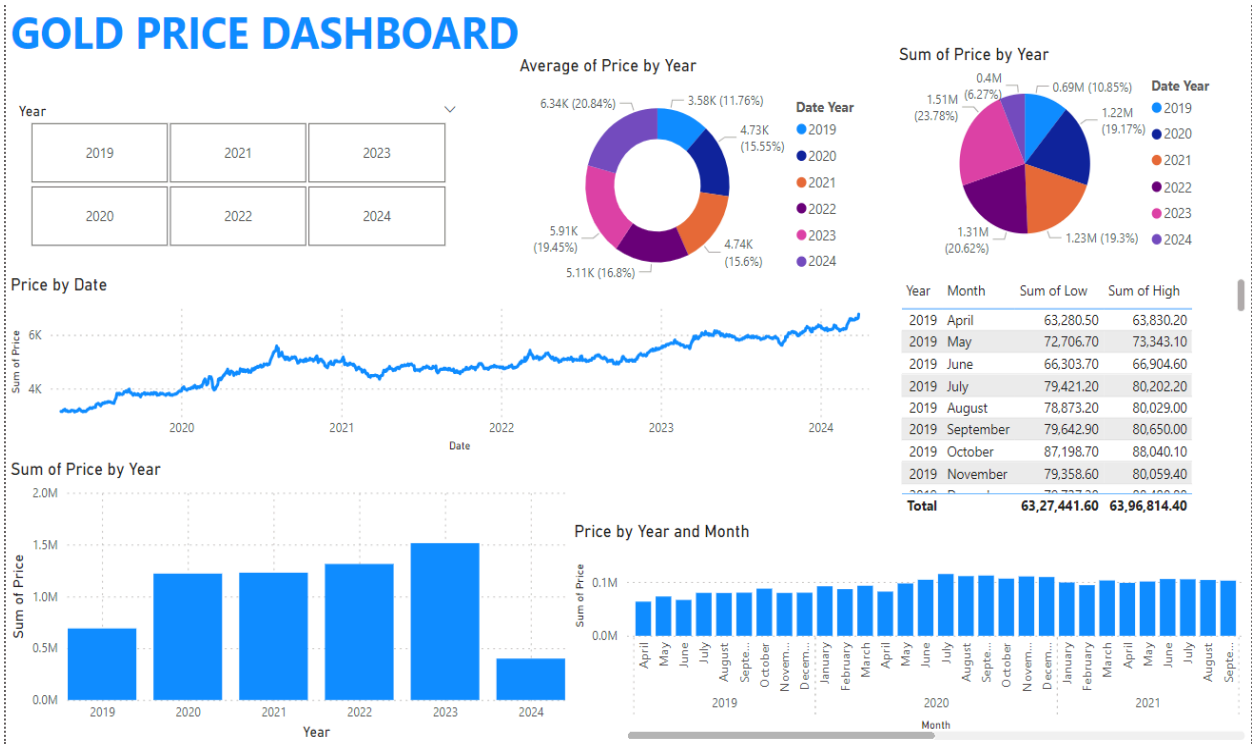


	A	B	C	D	E	F	G
1	Date	Price	Open	High	Low	Volume	Chg%
2	01-04-2019	3152.6	3169.5	3171	3151	1.13K	-0.66%
3	02-04-2019	3149	3154.7	3243	3145	0.07K	-0.11%
4	03-04-2019	3148.9	3146.2	3153.9	3140.2	0.03K	0.00%
5	04-04-2019	3149.2	3147.1	3150.1	3147.1	0.01K	0.01%
6	05-04-2019	3172.9	3160	3186.5	3160	0.01K	0.75%
7	08-04-2019	3218.6	3190	3228.7	3190	9.52K	1.44%
8	09-04-2019	3220.1	3222.5	3227.4	3212.1	9.91K	0.05%
9	10-04-2019	3228.1	3216.1	3229.3	3208.6	10.14K	0.25%
10	11-04-2019	3175	3223.2	3223.2	3173.4	13.29K	-1.64%
11	12-04-2019	3186.2	3183.8	3198	3180.5	9.29K	0.35%
12	15-04-2019	3185.6	3180	3187.4	3168	8.11K	-0.02%
13	16-04-2019	3154.8	3180	3186.5	3152.3	12.40K	-0.97%
14	17-04-2019	3148	3164.8	3164.8	3145.3	4.87K	-0.22%
15	18-04-2019	3146.3	3145	3161.6	3140.3	10.01K	-0.05%
16	22-04-2019	3159.6	3155	3178	3155	8.65K	0.42%
17	23-04-2019	3156.2	3160.6	3164	3145	10.93K	-0.11%
18	24-04-2019	3181.8	3150	3183.8	3149.9	9.21K	0.81%
19	25-04-2019	3191.1	3178.6	3202.5	3170.4	10.58K	0.29%

Cleaning–

- Change the format of Date column
- Sort the date column in ascending order (latest in last)

Dashboard Creation In power BI

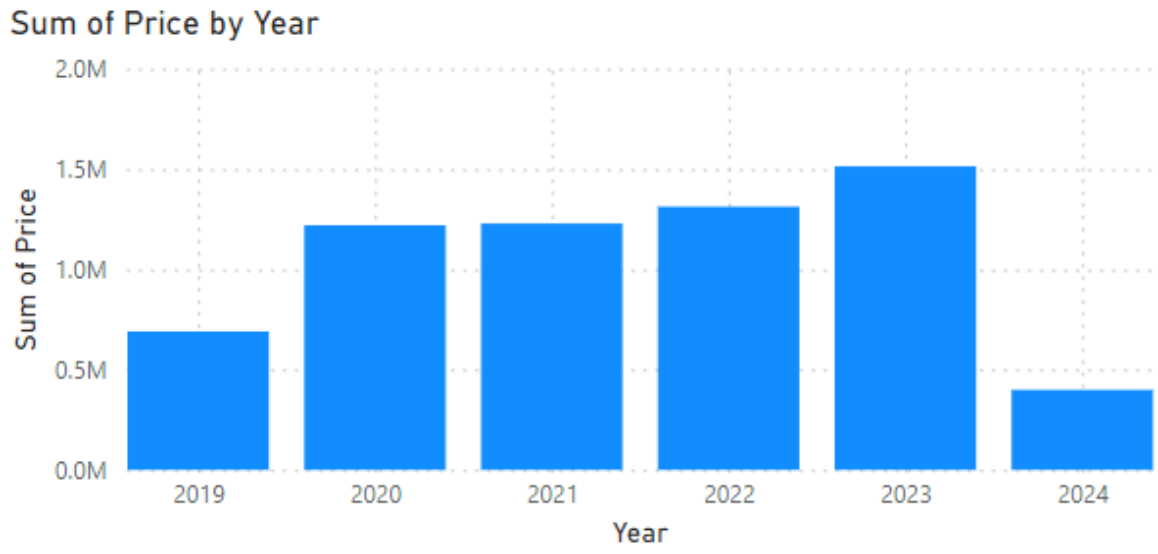


This dashboard includes a bar chart showing the price over the years, another bar chart showing the price over the years and months, a pie chart comparing the price versus years, a donut chart displaying the average price versus years, and a line chart showing the price versus dates. Additionally, there is a table showing the highest and lowest prices of gold for each year in a specific month. A filter is also provided for selecting a particular year to view the dashboard for that year.

Line graph of Price over Date

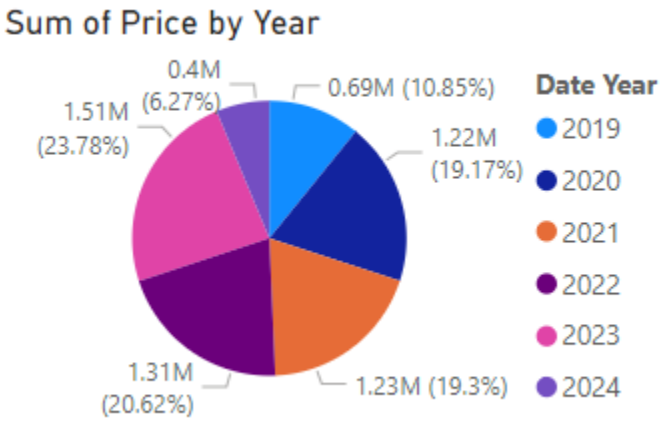
From this line graph of price over date, we can observe that the price of gold is increasing steadily year by year.

Bar graph of price over Year



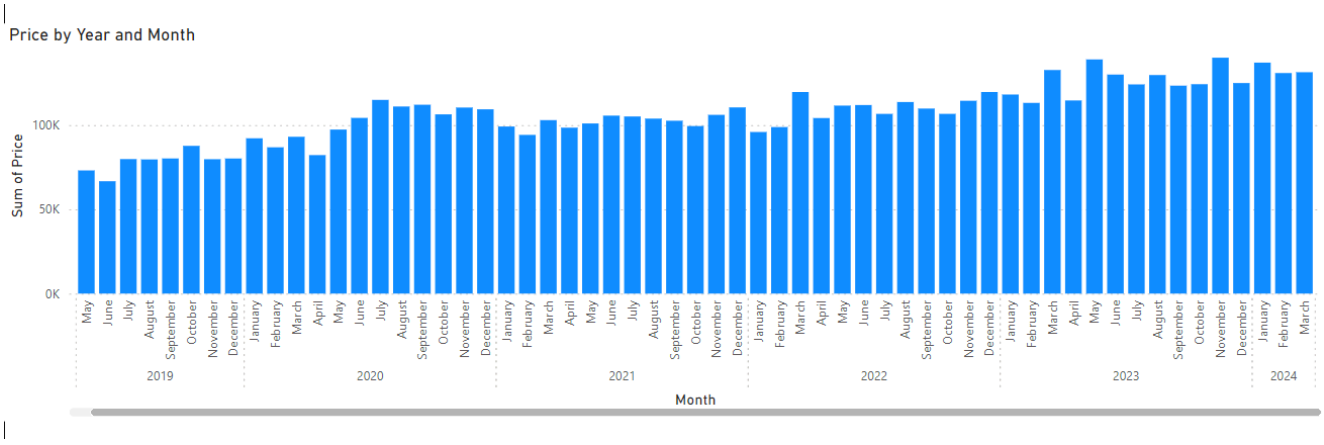
From this graph illustrating the price trend from April 2019 to March 2024, it is clear that the gold price was highest in the year 2023

Pie chart of year vs sum of price

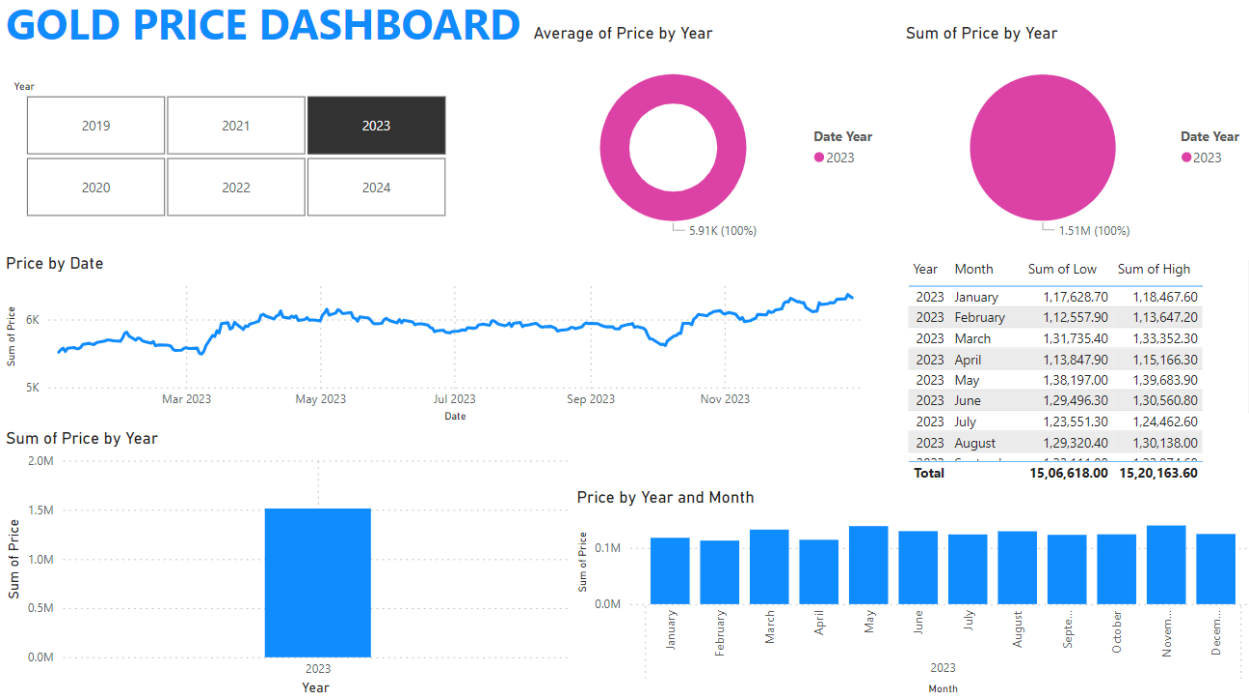


In this pie chart, we can also observe that the year 2023 has the highest price of gold.

Bar chart of year,month over price



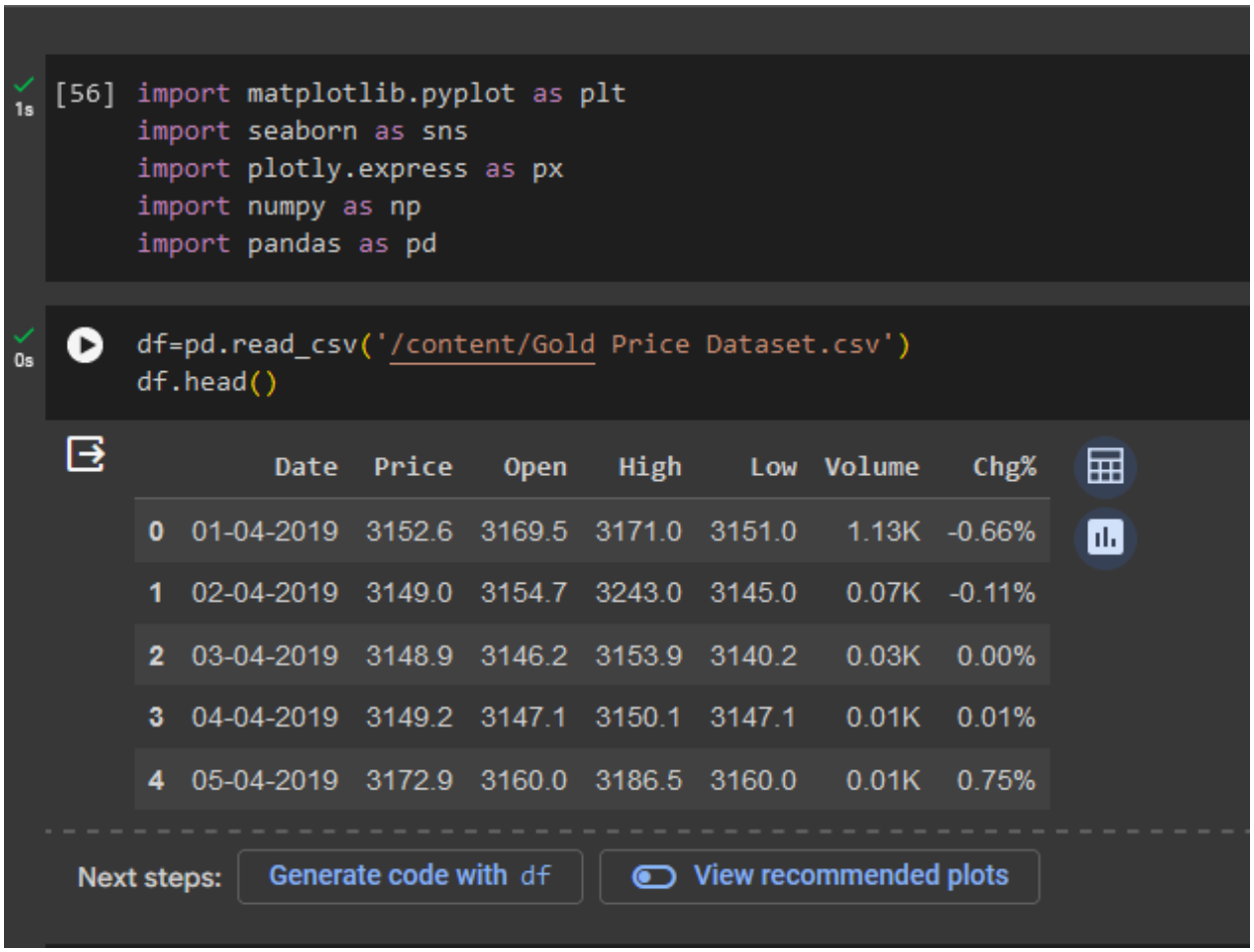
In the year 2023, the months of March, May, and September had the highest gold prices, and the prices continued to increase steadily thereafter.



This dashboard allows users to select the year 2023 from the filter, causing all the graphs and charts to change accordingly.

Gold Price Dashboard using Matplotlib and seaborn:

Link : Gold Price Dashboard in colab



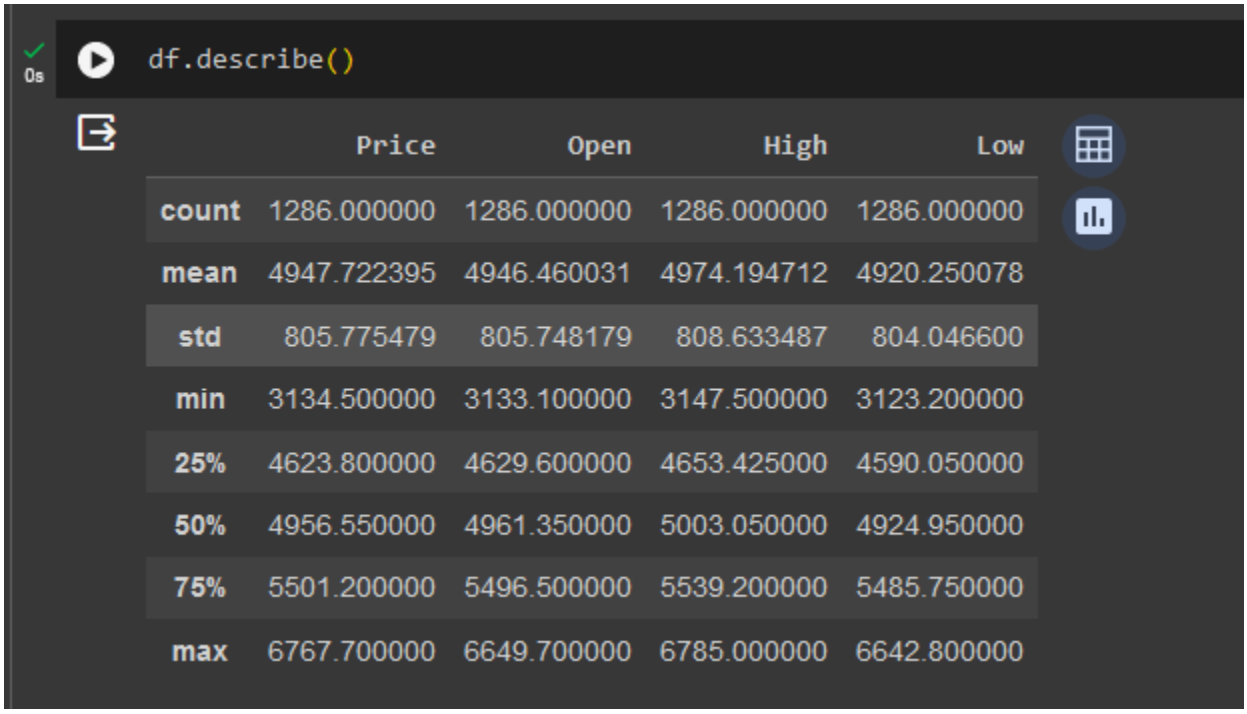
```
[56] import matplotlib.pyplot as plt
     import seaborn as sns
     import plotly.express as px
     import numpy as np
     import pandas as pd
```

```
df=pd.read_csv('/content/Gold Price Dataset.csv')
df.head()
```

	Date	Price	Open	High	Low	Volume	Chg%
0	01-04-2019	3152.6	3169.5	3171.0	3151.0	1.13K	-0.66%
1	02-04-2019	3149.0	3154.7	3243.0	3145.0	0.07K	-0.11%
2	03-04-2019	3148.9	3146.2	3153.9	3140.2	0.03K	0.00%
3	04-04-2019	3149.2	3147.1	3150.1	3147.1	0.01K	0.01%
4	05-04-2019	3172.9	3160.0	3186.5	3160.0	0.01K	0.75%

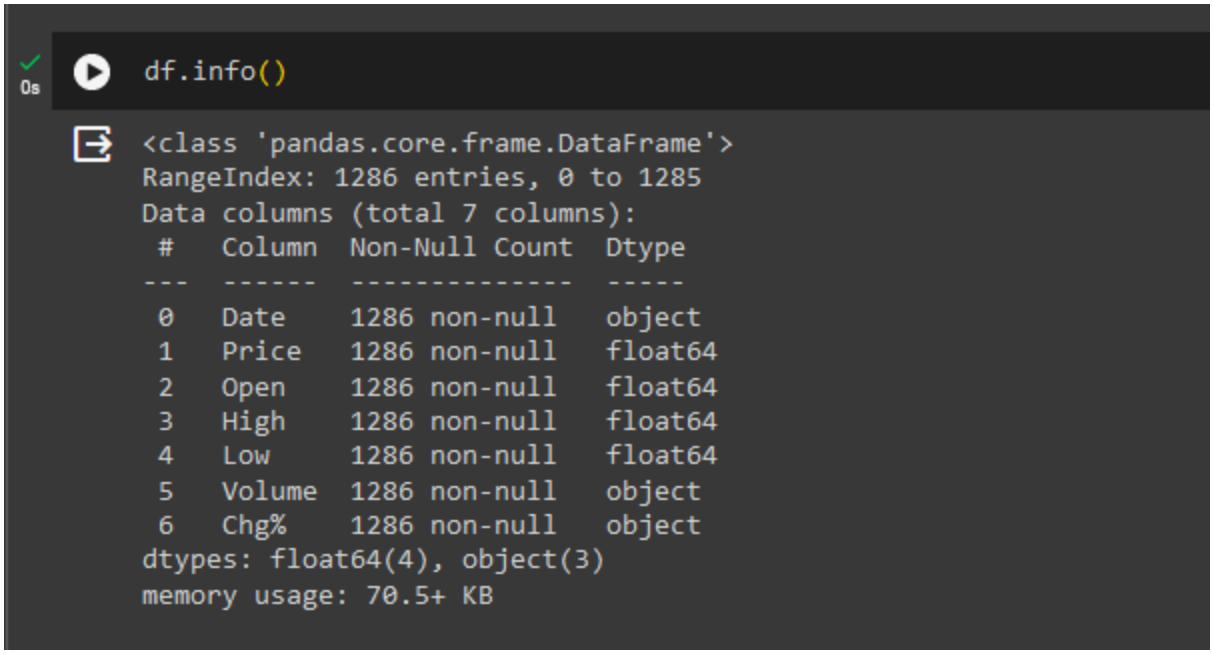
Next steps: Generate code with df View recommended plots

To create different dashboards using the matplotlib and seaborn libraries, we first need to import the required libraries and upload the dataset. Next, we can read the dataset and use the **`head()`** method to display the first 5 rows of the dataset.



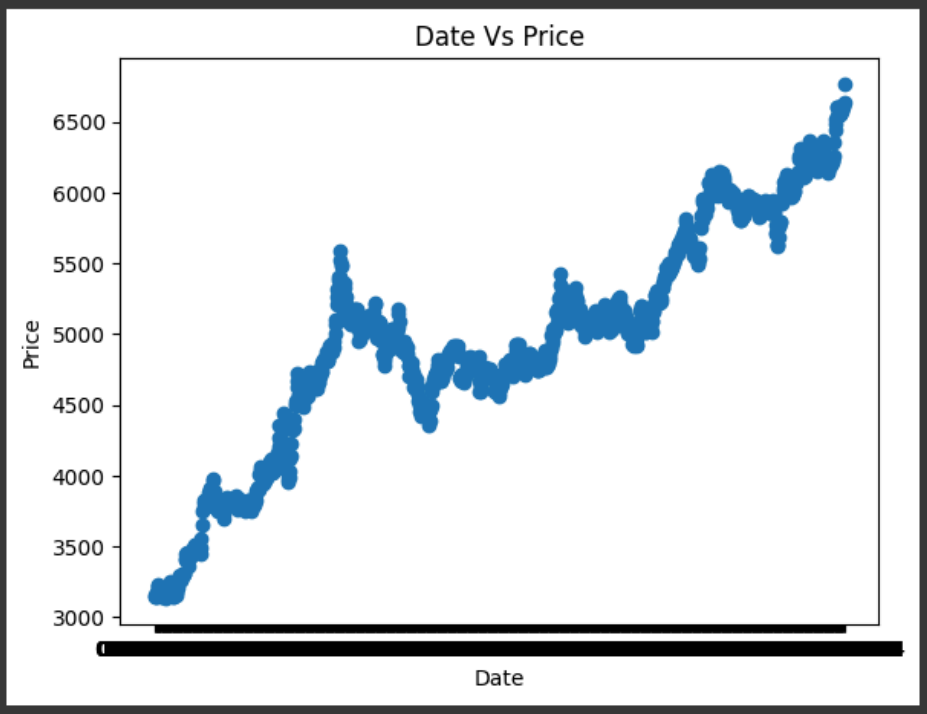
	Price	Open	High	Low
count	1286.000000	1286.000000	1286.000000	1286.000000
mean	4947.722395	4946.460031	4974.194712	4920.250078
std	805.775479	805.748179	808.633487	804.046600
min	3134.500000	3133.100000	3147.500000	3123.200000
25%	4623.800000	4629.600000	4653.425000	4590.050000
50%	4956.550000	4961.350000	5003.050000	4924.950000
75%	5501.200000	5496.500000	5539.200000	5485.750000
max	6767.700000	6649.700000	6785.000000	6642.800000

To get a summary of the numerical columns in the dataset using the **`describe()`** method, which provides statistics such as count, mean, standard deviation, minimum, maximum, and various quantiles for each numerical column.



To get a concise summary of a dataset, including the number of non-null values in each column, the data type of each column, and memory usage, you can use the **`info()`** method. which can be helpful for quickly understanding the structure of your dataset.

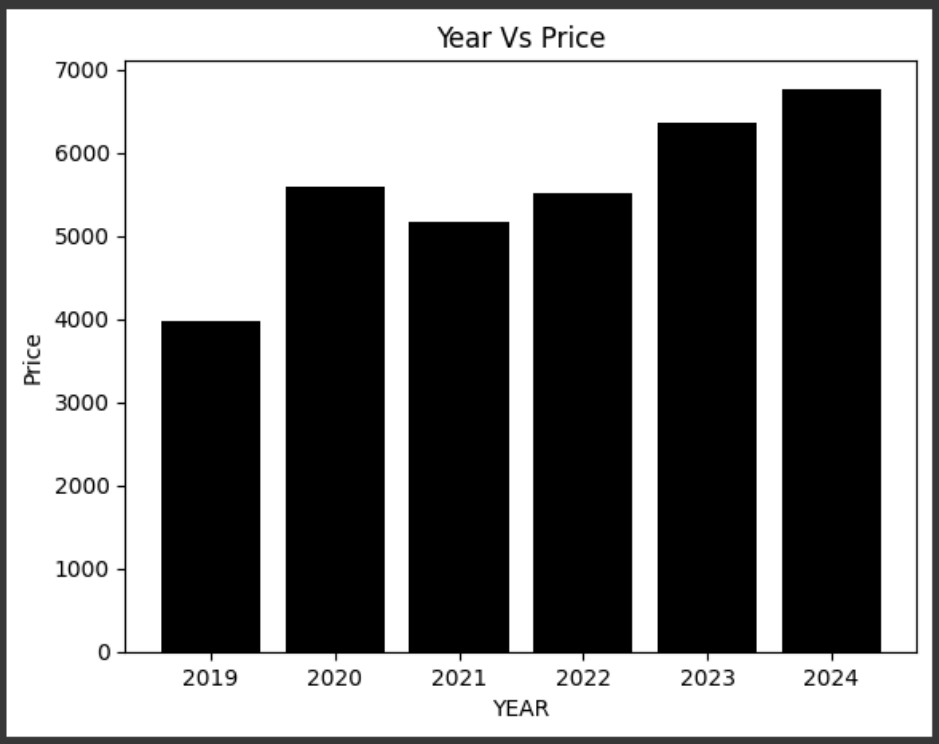
```
[60] plt.scatter(df['Date'],df['Price'])
     plt.title("Date Vs Price")
     plt.xlabel('Date')
     plt.ylabel('Price')
     plt.show()
```



Here is a line graph using matplotlib to visualize the price of gold over time. In this graph, the x-axis represents the dates, and the y-axis represents the price of gold. The graph shows the trend of increasing gold prices over time. Matplotlib is a popular data visualization library in Python used to create static, animated, and interactive visualizations in a variety of formats. It provides a wide range of functionalities for creating plots and charts to visualize data in a clear and effective manner.

```
df['Date'] = pd.to_datetime(df['Date'], format="%d-%m-%Y")
df['year'] = df['Date'].dt.year
df['month'] = df['Date'].dt.month_name()
```

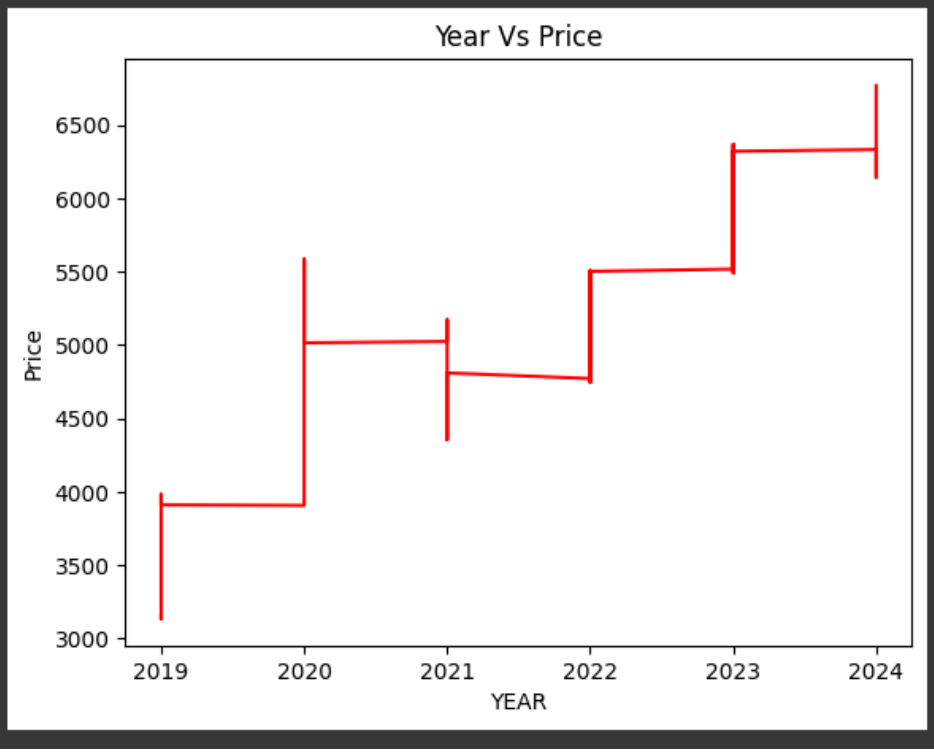
```
plt.bar(df['year'],df['Price'],color='black')
plt.title("Year Vs Price")
plt.xlabel('YEAR')
plt.ylabel('Price')
plt.show()
```



Here is a bar graph showing the price of gold over different years. Initially, the 'Date' column in our gold price dataset is converted to a date format. From this date, the year and month columns are extracted. The graph has the year on the x-axis and the gold price on the y-axis, illustrating the increase in gold price year by year.

```
df['Date'] = pd.to_datetime(df['Date'], format="%d-%m-%Y")
df['year'] = df['Date'].dt.year
df['month'] = df['Date'].dt.month_name()
```

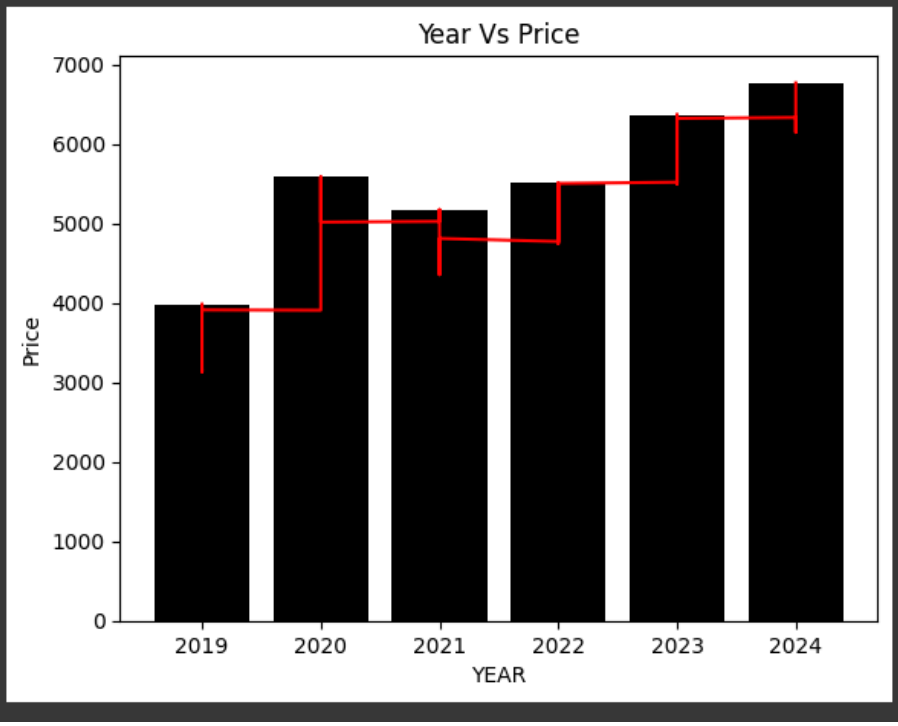
```
plt.plot(df['year'],df['Price'],color='red')
plt.title("Year Vs Price")
plt.xlabel('YEAR')
plt.ylabel('Price')
plt.show()
```



This is a line graph showing the price of gold over the years. From 2019 to 2021, the price of gold fluctuates, showing both increases and decreases at different time periods. However, after 2021, the trend in the graph shows a consistent increase in the price of gold.

```
df['Date'] = pd.to_datetime(df['Date'], format="%d-%m-%Y")
df['year'] = df['Date'].dt.year
df['month'] = df['Date'].dt.month_name()
```

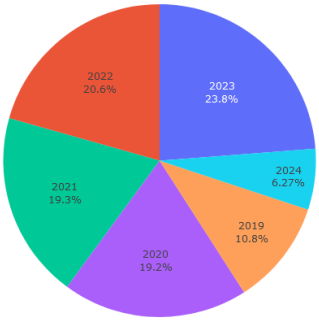
```
[64] plt.bar(df['year'],df['Price'],color='black')
     plt.plot(df['year'],df['Price'],color='red')
     plt.title("Year Vs Price")
     plt.xlabel('YEAR')
     plt.ylabel('Price')
     plt.show()
```



This graph combines a line graph and a bar graph, where the x-axis represents the years and the y-axis represents the price of gold. The graph illustrates that the price of gold is increasing steadily over time.

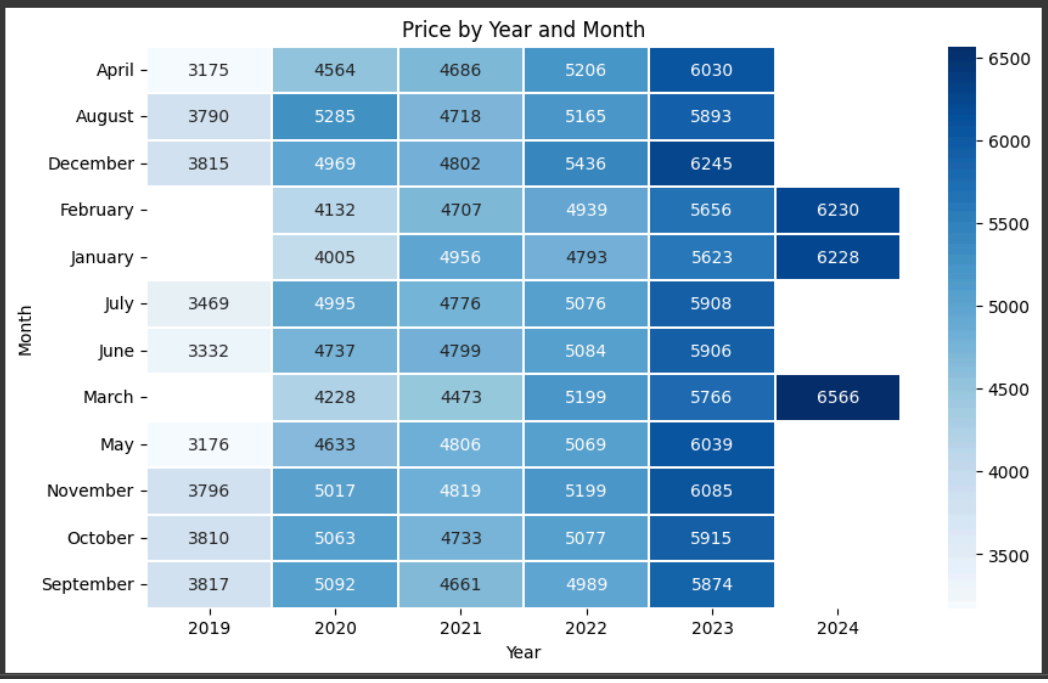
```
df['Date'] = pd.to_datetime(df['Date'], format="%d-%m-%Y")
df['year'] = df['Date'].dt.year
df['month'] = df['Date'].dt.month_name()
```

```
df = df.sort_values(by=['year', 'month'], ascending=True)
plt.figure(figsize=(6,2))
fig = px.pie(df, values='Price', names='year')
fig.update_traces(textposition='inside', textinfo='percent+label')
fig.show()
```



This is a pie chart created using Plotly Express, a high-level data visualization library in Python known for its ability to generate interactive plots and charts with minimal code. The chart illustrates the distribution of gold prices over different years, clearly displaying an increasing trend from 2019 to 2024. The percentage labels in the chart help visualize this upward trend in gold prices over the specified years.

```
df_agg = df.groupby(['year', 'month']).agg({'Price': 'mean'}).reset_index()
df_pivot = df_agg.pivot(index='month', columns='year', values='Price')
plt.figure(figsize=(10, 6))
sns.heatmap(df_pivot, annot=True, fmt='.0f', linewidth=0.2, cmap='Blues')
plt.title('Price by Year and Month')
plt.xlabel('Year')
plt.ylabel('Month')
plt.show()
```



This is a heatmap showing the gold prices over different months and years. A heatmap is a data visualization technique that uses color to represent the magnitude of a value in a two-dimensional (2D) matrix. It is often used to visualize the relationships or patterns in complex data sets. Initially, the data is grouped by month and year, aggregating the monthly gold prices using the mean value. Then, a pivot table is created with the months as the index, years as the columns, and the gold prices as the values. The heatmap uses color intensity to indicate the price levels, where darker shades of blue represent higher prices in specific years and months.

Gold Price Prediction using Linear Regression :

Gold price prediction year
Link : Gold price prediction(year)

```
[19] import pandas as pd
     import matplotlib.pyplot as plt
     from sklearn.linear_model import LinearRegression
     import warnings
     warnings.filterwarnings('ignore', message='X does not have valid feature names')
```

```
[20] # Read the CSV file into a DataFrame
     df = pd.read_csv('/content/Gold Price Dataset.csv')
     df.head()
```

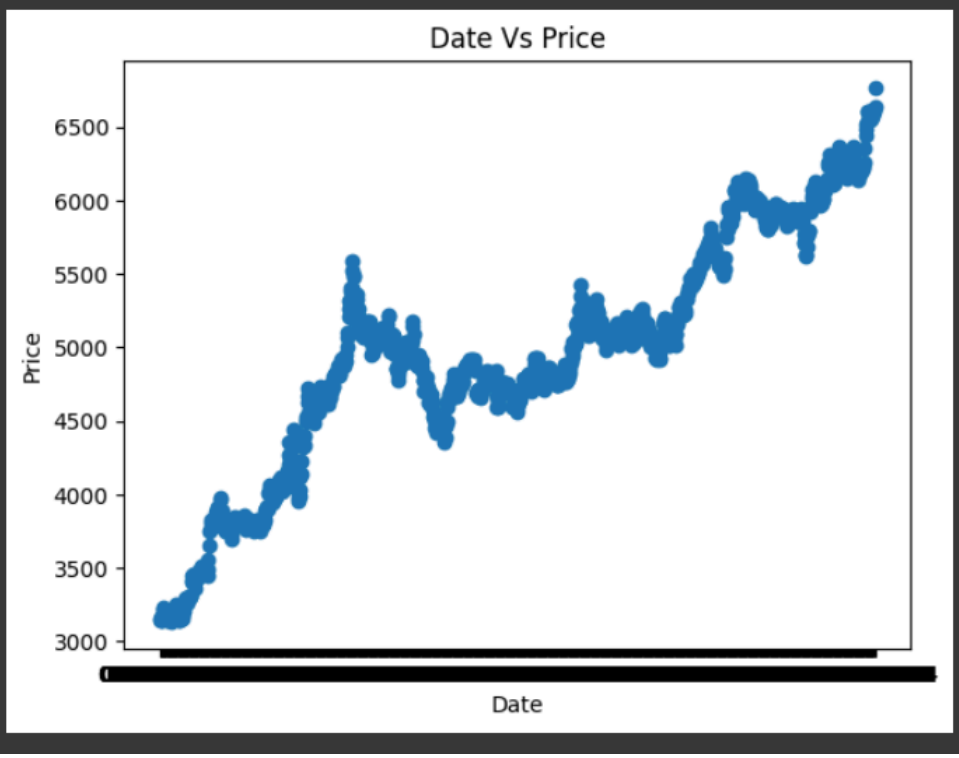
	Date	Price	Open	High	Low	Volume	Chg%
0	01-04-2019	3152.6	3169.5	3171.0	3151.0	1.13K	-0.66%
1	02-04-2019	3149.0	3154.7	3243.0	3145.0	0.07K	-0.11%
2	03-04-2019	3148.9	3146.2	3153.9	3140.2	0.03K	0.00%
3	04-04-2019	3149.2	3147.1	3150.1	3147.1	0.01K	0.01%
4	05-04-2019	3172.9	3160.0	3186.5	3160.0	0.01K	0.75%

Next steps: Generate code with df | View recommended plots

This code predicts the price of gold for a specific year using linear regression. It starts by importing necessary libraries: pandas for reading the dataset, matplotlib for plotting graphs, and sklearn.linear_model for the linear regression model. The warnings module is also imported to handle warnings in the code.

Next, the code reads a CSV file containing gold price data using pandas' **`read_csv()`** function. It then displays the first five rows of the dataset using the **`head()`** method to provide a glimpse of the data structure.

```
plt.scatter(df['Date'],df['Price'])
plt.title("Date Vs Price")
plt.xlabel('Date')
plt.ylabel('Price')
plt.show()
```



Here is a line graph using matplotlib to visualize the price of gold over time. In this graph, the x-axis represents the dates, and the y-axis represents the price of gold. The graph shows the trend of increasing gold prices over time

```
[21] # Drop unnecessary columns
     df = df.drop('Open', axis=1)
     df = df.drop('Low', axis=1)
     df = df.drop('Volume', axis=1)
     df = df.drop('High', axis=1)
     df = df.drop('Chg%', axis=1)
```

```
[22] # Convert the 'Date' column to a datetime object
     df['Date'] = pd.to_datetime(df['Date'], format='%d-%m-%Y')
```

```
[23] # Extract the year from the date column
     df['year'] = df['Date'].dt.year
     df=df.drop('Date', axis=1)
     df.head()
```

	Price	year
0	3152.6	2019
1	3149.0	2019
2	3148.9	2019
3	3149.2	2019
4	3172.9	2019

Next steps: Generate code with df View recommended plots

First, the code uses the **`drop()`** method to remove unwanted columns from the dataset. Next, it converts the 'Date' column to a date data type to ensure proper handling of dates.In the following code, the year is extracted from the 'Date' column, and the 'Date' column is dropped from the dataset. Finally, the **`head()`** method is used to display the first five rows of the updated dataset.

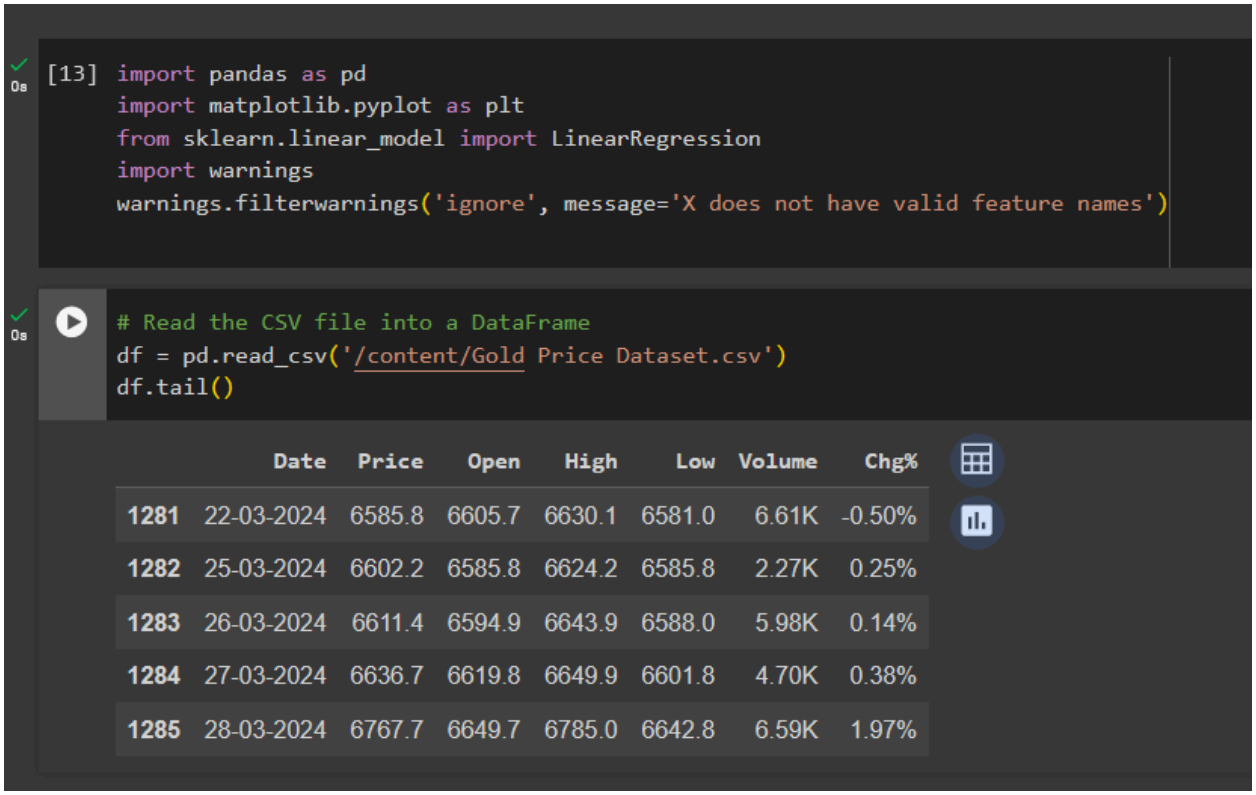
First, an instance of the LinearRegression class from scikit-learn's linear_model module is created. This instance is used to perform linear regression.

Next, the **`fit()`** method is used to fit the linear regression model to the data. The **`fit()`** method takes two arguments: the independent variable (year) and the target variable (gold price). This method trains the model to learn the relationship between the year and the price of gold.

Finally, the **`predict()`** method is used to make a prediction for the year 2025. This method returns the predicted price of gold for the year 2025 based on the learned relationship from the training data

Gold Price Prediction(Date)

Link : [Gold price prediction(date)]



```
[13] import pandas as pd
     import matplotlib.pyplot as plt
     from sklearn.linear_model import LinearRegression
     import warnings
     warnings.filterwarnings('ignore', message='X does not have valid feature names')
```

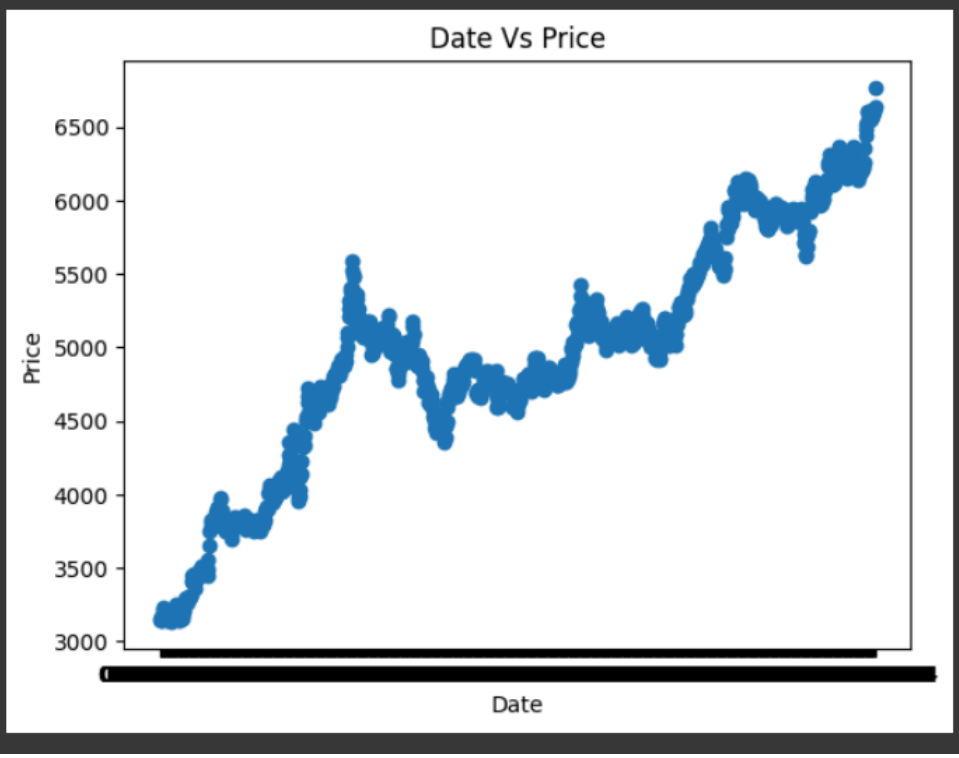
```
# Read the CSV file into a DataFrame
df = pd.read_csv('/content/Gold Price Dataset.csv')
df.tail()
```

	Date	Price	Open	High	Low	Volume	Chg%
1281	22-03-2024	6585.8	6605.7	6630.1	6581.0	6.61K	-0.50%
1282	25-03-2024	6602.2	6585.8	6624.2	6585.8	2.27K	0.25%
1283	26-03-2024	6611.4	6594.9	6643.9	6588.0	5.98K	0.14%
1284	27-03-2024	6636.7	6619.8	6649.9	6601.8	4.70K	0.38%
1285	28-03-2024	6767.7	6649.7	6785.0	6642.8	6.59K	1.97%

This code predicts the price of gold for a specific Date using linear regression. It starts by importing necessary libraries: pandas for reading the dataset, matplotlib for plotting graphs, and sklearn.linear_model for the linear regression model. The warnings module is also imported to handle warnings in the code.

Next, the code reads a CSV file containing gold price data using pandas' **`read_csv()`** function. It then displays the last five rows of the dataset using the **`tail()`** method to provide a glimpse of the data structure.

```
plt.scatter(df['Date'],df['Price'])
plt.title("Date Vs Price")
plt.xlabel('Date')
plt.ylabel('Price')
plt.show()
```



Here is a line graph using matplotlib to visualize the price of gold over time. In this graph, the x-axis represents the dates, and the y-axis represents the price of gold. The graph shows the trend of increasing gold prices over time

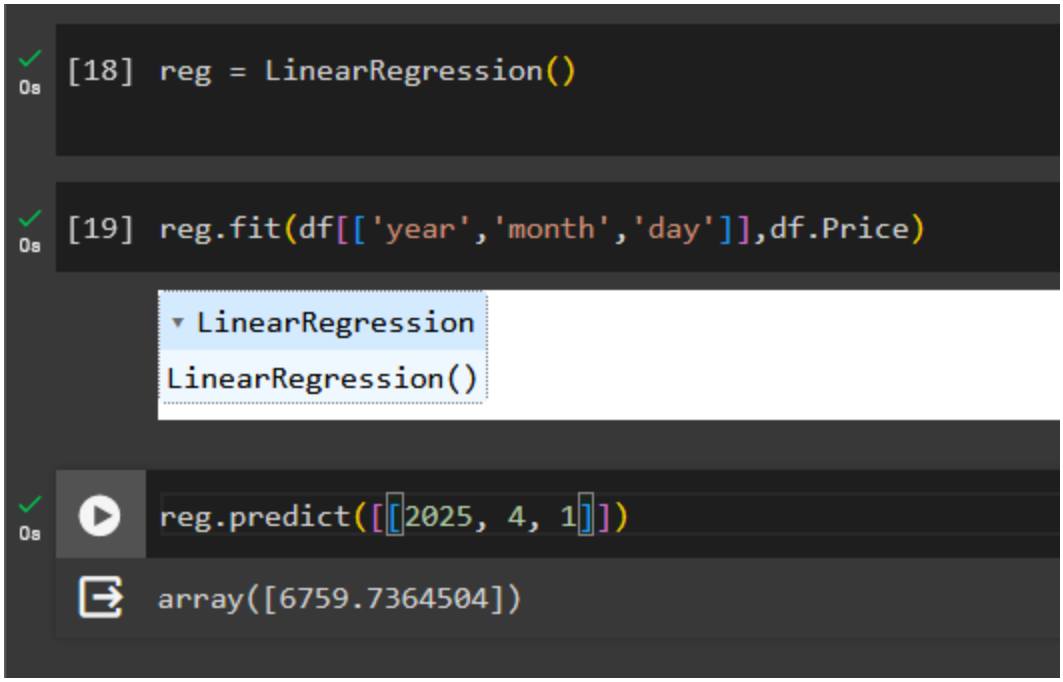
```
[16] # Drop unnecessary columns
     df = df.drop('Open', axis=1)
     df = df.drop('Low', axis=1)
     df = df.drop('Volume', axis=1)
     df = df.drop('High', axis=1)
     df = df.drop('Chg%', axis=1)
```

```
# Convert the 'Date' column to a datetime object
df['Date'] = pd.to_datetime(df['Date'], format='%d-%m-%Y')

df['year'] = df['Date'].dt.year
df['month'] = df['Date'].dt.month
df['day'] = df['Date'].dt.day
df=df.drop('Date', axis=1)
df
```

	Price	year	month	day
0	3152.6	2019	4	1
1	3149.0	2019	4	2
2	3148.9	2019	4	3
3	3149.2	2019	4	4
4	3172.9	2019	4	5
...	...	...	...	...
1281	6585.8	2024	3	22
1282	6602.2	2024	3	25
1283	6611.4	2024	3	26

First, the code uses the `drop()` method to remove unwanted columns from the dataset. Next, it converts the 'Date' column to a date data type to ensure proper handling of dates.In the following code, the year ,Month and Date is extracted from the 'Date' column, and the 'Date' column is dropped from the dataset. Finally, display the updated dataset.



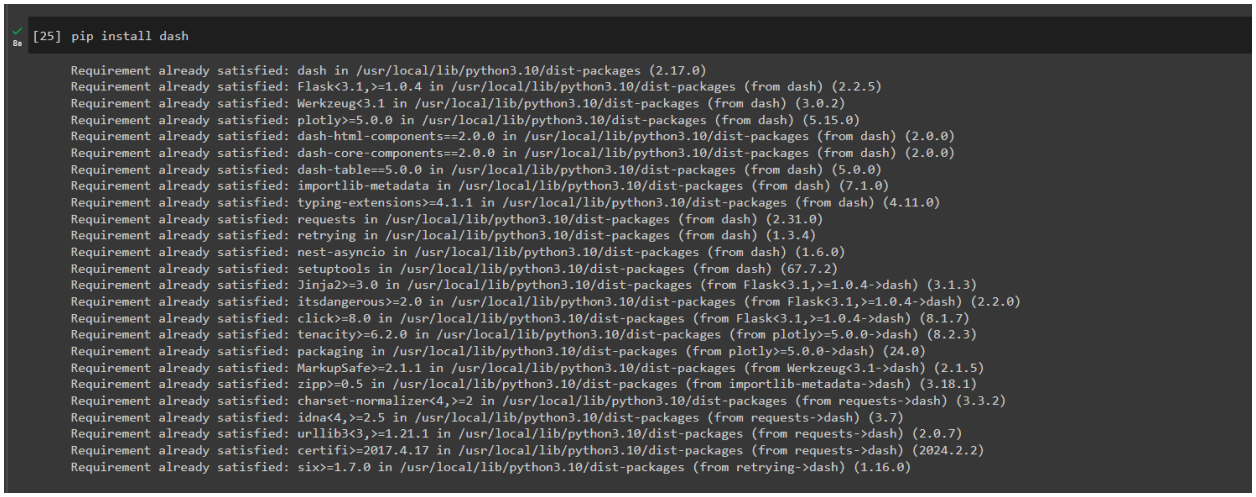
First, an instance of the LinearRegression class from scikit-learn's linear_model module is created. This instance is used to perform linear regression.

Next, the **`fit()`** method is used to fit the linear regression model to the data. The **`fit()`** method takes two arguments: the independent variable (year,month,day) and the target variable (gold price). This method trains the model to learn the relationship between the year and the price of gold.

Finally, the **`predict()`** method is used to make a prediction for a specific date, in this case, April 1, 2025. The method returns the predicted price of gold for the specified date based on the learned relationship from the training data

Gold Price Dashboard With Dash app

LINK:[Gold price Dashboard using Dash]



```
[25] pip install dash

Requirement already satisfied: dash in /usr/local/lib/python3.10/dist-packages (2.17.0)
Requirement already satisfied: Flask<3.1,>=1.0.4 in /usr/local/lib/python3.10/dist-packages (from dash) (2.2.5)
Requirement already satisfied: Werkzeug<3.1 in /usr/local/lib/python3.10/dist-packages (from dash) (3.0.2)
Requirement already satisfied: plotly>=5.0.0 in /usr/local/lib/python3.10/dist-packages (from dash) (5.15.0)
Requirement already satisfied: dash-html-components==2.0.0 in /usr/local/lib/python3.10/dist-packages (from dash) (2.0.0)
Requirement already satisfied: dash-core-components==2.0.0 in /usr/local/lib/python3.10/dist-packages (from dash) (2.0.0)
Requirement already satisfied: dash-table==5.0.0 in /usr/local/lib/python3.10/dist-packages (from dash) (5.0.0)
Requirement already satisfied: importlib-metadata in /usr/local/lib/python3.10/dist-packages (from dash) (7.1.0)
Requirement already satisfied: typing-extensions>=4.1.1 in /usr/local/lib/python3.10/dist-packages (from dash) (4.11.0)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from dash) (2.31.0)
Requirement already satisfied: retrying in /usr/local/lib/python3.10/dist-packages (from dash) (1.3.4)
Requirement already satisfied: nest-asyncio in /usr/local/lib/python3.10/dist-packages (from dash) (1.6.0)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from dash) (67.7.2)
Requirement already satisfied: Jinja2>=3.0 in /usr/local/lib/python3.10/dist-packages (from Flask<3.1,>=1.0.4->dash) (3.1.3)
Requirement already satisfied: itsdangerous>=2.0 in /usr/local/lib/python3.10/dist-packages (from Flask<3.1,>=1.0.4->dash) (2.2.0)
Requirement already satisfied: click>=8.0 in /usr/local/lib/python3.10/dist-packages (from Flask<3.1,>=1.0.4->dash) (8.1.7)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from plotly>=5.0.0->dash) (8.2.3)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from plotly>=5.0.0->dash) (24.0)
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.10/dist-packages (from Werkzeug<3.1->dash) (2.1.5)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.10/dist-packages (from importlib-metadata->dash) (3.18.1)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->dash) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->dash) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->dash) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->dash) (2024.2.2)
Requirement already satisfied: six>=1.7.0 in /usr/local/lib/python3.10/dist-packages (from retrying->dash) (1.16.0)
```

To create an interactive dashboard of gold prices over dates, we can use the Dash library, which is specifically designed for building web applications with Python. After installing Dash, we can proceed with creating your interactive dashboard.

```
import dash
from dash import dcc, html
from dash.dependencies import Input, Output
import plotly.express as px
import pandas as pd


# Sample data for demonstration
data = pd.read_csv('/content/Gold Price Dataset.csv')
df = pd.DataFrame(data)


# Convert the 'Date' column to datetime format
df['Date'] = pd.to_datetime(df['Date'], format='%d-%m-%Y')

# Get unique years from the 'Date' column
years = df['Date'].dt.year.unique()
```

To create an interactive dashboard of gold prices over dates, first, import the required libraries including pandas, dash core components, dash html components, and from dash.dependencies import Input and Output, as well as plotly express

Pandas is used to read CSV datasets. Dash is a Python framework for building web applications. Dash core components create interactive elements like sliders, dropdown and graphs. Dash HTML components build the layout with HTML tags. The dependencies module in Dash handles inputs and outputs for callbacks. plotly express is a library for creating interactive plots and charts.

First, the pandas library is used to read a CSV dataset, and the **`head()`** method is applied to display the first 5 rows from the dataset. Next, the 'Date' column is converted to a date data type, and the year is extracted from the 'Date' column.

```python
app = dash.Dash(__name__)

# Define the layout of the dashboard
app.layout = html.Div([
    html.H1('Gold Price Dashboard'),
    html.Label('Select Year:'),
    dcc.Dropdown(
        id='year-dropdown',
        options=[{'label': str(year), 'value': year} for year in years],
        value=None,
        clearable=False
    ),
    html.Label('Select Data Range:'),
    dcc.DatePickerRange(
        id='date-range',
        start_date=df['Date'].min(),
        end_date=df['Date'].max(),
    ),
    dcc.Graph(id='line-chart')
])
```

First, create a Dash application instance. Then, define the dashboard layout using HTML components. Include a heading ('H1') for 'Gold Price Dashboard', a label for the year dropdown, and a dropdown component with options for each year. Add a label for the date range picker and create the date range picker component with default start and end dates. Finally, add a graph component to display the line chart.

```python
# Define callback to update the line chart based on dropdown and date range selection
@app.callback(
    Output('line-chart', 'figure'),
    [Input('year-dropdown', 'value'),
     Input('date-range', 'start_date'),
     Input('date-range', 'end_date')]
)
def update_line_chart(selected_year, start_date, end_date):
    if selected_year is None:
        filtered_df = df[(df['Date'] >= start_date) & (df['Date'] <= end_date)]
        fig = px.line(filtered_df, x='Date', y='Price', title='Price over Time')
    else:
        filtered_df = df[(df['Date'].dt.year == selected_year) & (df['Date'] >= start_date) & (df['Date'] <= end_date)]
        fig = px.line(filtered_df, x='Date', y='Price', title=f'Price over Time for {selected_year}')
    return fig

# Run the app
if __name__ == '__main__':
    app.run_server(debug=True)
```

The `@app.callback` decorator sets up a function to update the 'line-chart' component whenever the year, start date, or end date changes. The function returns a figure (fig) that updates the chart. The last part of the code runs the Dash app in debug mode if the script is executed directly. This callback function creates an interactive chart showing gold prices over time, letting users select a year and date range for the data display.