```
from google.colab import files


import pandas as pd

df = pd.read_csv("infy_stock.csv")
df.head()
```

	Date	Symbol	Series	Prev Close	Open	High	Low	Last	Close	VWAP	Volume	Turnover	Trades	Deliverable Volume	%Deli
0	2015-01-01	INFY	EQ	1972.55	1968.95	1982.00	1956.9	1971.00	1974.40	1971.34	500691	9.870306e+13	14908	258080	
1	2015-01-02	INFY	EQ	1974.40	1972.00	2019.05	1972.0	2017.95	2013.20	2003.25	1694580	3.394669e+14	54166	1249104	
2	2015-01-05	INFY	EQ	2013.20	2009.90	2030.00	1977.5	1996.00	1995.90	2004.59	2484256	4.979911e+14	82694	1830962	

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
df.info()
df.describe()
df.columns
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 248 entries, 0 to 247
Data columns (total 15 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Date                248 non-null    object
 1   Symbol              248 non-null    object
 2   Series              248 non-null    object
 3   Prev Close          248 non-null    float64
 4   Open                248 non-null    float64
 5   High                248 non-null    float64
 6   Low                 248 non-null    float64
 7   Last                248 non-null    float64
 8   Close               248 non-null    float64
 9   VWAP                248 non-null    float64
 10  Volume              248 non-null    int64
 11  Turnover            248 non-null    float64
 12  Trades              248 non-null    int64
 13  Deliverable Volume  248 non-null    int64
 14  %Deliverble         248 non-null    float64
dtypes: float64(9), int64(3), object(3)
memory usage: 29.2+ KB
Index(['Date', 'Symbol', 'Series', 'Prev Close', 'Open', 'High', 'Low', 'Last',
       'Close', 'VWAP', 'Volume', 'Turnover', 'Trades', 'Deliverable Volume',
       '%Deliverble'],
      dtype='object')
```

```
df.isnull().sum()
df.duplicated().sum()
```

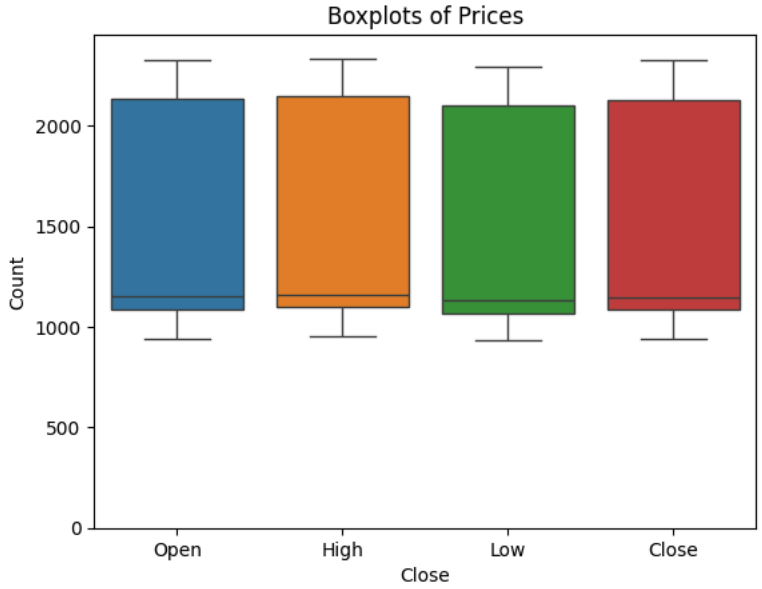
```
np.int64(0)
```

```
import matplotlib.pyplot as plt
import seaborn as sns

sns.histplot(df['Close'], kde=True)
plt.title('Distribution of Closing Price')

sns.boxplot(data=df[['Open', 'High', 'Low', 'Close']])
plt.title('Boxplots of Prices')
```

```
Text(0.5, 1.0, 'Boxplots of Prices')
```




```
# Example: Predict 'Close' price
X = df.drop(['Close'], axis=1)
y = df['Close']


df.select_dtypes(include='object').columns  # Check categorical columns
```

```
Index(['Date', 'Symbol', 'Series'], dtype='object')
```

```
df = pd.get_dummies(df, drop_first=True)


from sklearn.preprocessing import StandardScaler
import numpy as np  # Assuming your data is in a NumPy array

# Example: Creating some sample data for X
X = np.array([[1, 2],
              [3, 4],
              [5, 6]])

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

print(X_scaled)
```

```
[[-1.22474487 -1.22474487]
 [ 0.          0.        ]
 [ 1.22474487  1.22474487]]
```

```
from sklearn.model_selection import train_test_split
import numpy as np  # Assuming your data is in NumPy arrays

# Assuming X_scaled is already defined (from the previous step)
# Example: Creating some sample target data for y
y = np.array([0, 1, 0])  # Corresponding target values for the example X

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

print("X_train shape:", X_train.shape)
print("X_test shape:", X_test.shape)
print("y_train shape:", y_train.shape)
print("y_test shape:", y_test.shape)
```

```
X_train shape: (2, 2)
X_test shape: (1, 2)
y_train shape: (2,)
y_test shape: (1,)
```

```
from sklearn.linear_model import LinearRegression
import numpy as np  # Make sure NumPy is imported

# Assuming X_train and y_train are already defined

# Check the shape of y_train
```

```
print("Shape of y_train before reshaping:", y_train.shape)

# Reshape y_train to be a 1D array if it's not already
if len(y_train.shape) > 1 and y_train.shape[1] > 1:
    y_train = y_train.reshape(-1)  # Reshape to a single column

model = LinearRegression()
model.fit(X_train, y_train)

print("Shape of y_train after reshaping:", y_train.shape)
print("Linear Regression model trained successfully!")
```

```
Shape of y_train before reshaping: (2,)
Shape of y_train after reshaping: (2,)
Linear Regression model trained successfully!
```

```
y_test = y_test.astype(float)
y_pred = y_pred.astype(float)
```

```
sample = X_test[0].reshape(1, -1)
model.predict(sample)
```

```
array([2.])
```

```
import pandas as pd
from sklearn.preprocessing import StandardScaler
import numpy as np # Assuming X was a NumPy array initially

# Let's assume your original training data X looked something like this
X = pd.DataFrame({
    'Open': [1600, 1650, 1700],
    'High': [1620, 1680, 1720],
    'Low': [1590, 1630, 1695],
    'Volume': [100000, 110000, 120000],
    'Another_Feature': [10, 20, 30] # Example of another feature
})

scaler = StandardScaler()
scaler.fit(X)
X_scaled = scaler.transform(X)

new_input = pd.DataFrame([{
    'Open': 1700,
    'High': 1720,
    'Low': 1695,
    'Volume': 123456,
    'Another_Feature': 25 # Make sure all original columns are present
}])

new_input_scaled = scaler.transform(new_input)
print(new_input_scaled)
```

```
[[1.22474487 1.13554995 1.30963107 1.6480167  0.61237244]]
```

```
print("Shape of new_input_scaled:", new_input_scaled.shape)
```

```
Shape of new_input_scaled: (1, 5)
```

```
!pip install gradio
```

```
  Downloading uvicorn-0.34.2-py3-none-any.whl.metadata (6.5 kB)
Requirement already satisfied: fsspec in /usr/local/lib/python3.11/dist-packages (from gradio-client==1.10.0->gradio) (2025.3.2)
Requirement already satisfied: websockets<16.0,>=10.0 in /usr/local/lib/python3.11/dist-packages (from gradio-client==1.10.0->grad
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.11/dist-packages (from anyio<5.0,>=3.0->gradio) (3.10)
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.11/dist-packages (from anyio<5.0,>=3.0->gradio) (1.3.1)
Requirement already satisfied: certifi in /usr/local/lib/python3.11/dist-packages (from httpx>=0.24.1->gradio) (2025.4.26)
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.11/dist-packages (from httpx>=0.24.1->gradio) (1.0.9)
Requirement already satisfied: h11>=0.16 in /usr/local/lib/python3.11/dist-packages (from httpcore==1.*->httpx>=0.24.1->gradio) (0
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.28.1->gradio) (3.18.0)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.28.1->gradio) (2.32.3)
Requirement already satisfied: tqdm>=4.42.1 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.28.1->gradio) (4.6
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas<3.0,>=1.0->gradio) (
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas<3.0,>=1.0->gradio) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas<3.0,>=1.0->gradio) (2025.2)
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.11/dist-packages (from pydantic<2.12,>=2.0->gradio
Requirement already satisfied: pydantic-core==2.33.2 in /usr/local/lib/python3.11/dist-packages (from pydantic<2.12,>=2.0->gradio)
Requirement already satisfied: typing-inspection>=0.4.0 in /usr/local/lib/python3.11/dist-packages (from pydantic<2.12,>=2.0->grad
Requirement already satisfied: click>=8.0.0 in /usr/local/lib/python3.11/dist-packages (from typer<1.0,>=0.12->gradio) (8.1.8)
Requirement already satisfied: shellingham>=1.3.0 in /usr/local/lib/python3.11/dist-packages (from typer<1.0,>=0.12->gradio) (1.5
Requirement already satisfied: rich>=10.11.0 in /usr/local/lib/python3.11/dist-packages (from typer<1.0,>=0.12->gradio) (13.9.4)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->pandas<3.0,>=1.0-
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.11/dist-packages (from rich>=10.11.0->typer<1.0,>=0
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.11/dist-packages (from rich>=10.11.0->typer<1.0,>
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests->huggingface-hub
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests->huggingface-hub>=0.28
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.11/dist-packages (from markdown-it-py>=2.2.0->rich>=10.11.0->t
Downloading gradio-5.29.0-py3-none-any.whl (54.1 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 54.1/54.1 MB 11.3 MB/s eta 0:00:00
Downloading gradio_client-1.10.0-py3-none-any.whl (322 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 322.9/322.9 kB 15.7 MB/s eta 0:00:00
Downloading aiofiles-24.1.0-py3-none-any.whl (15 kB)
Downloading fastapi-0.115.12-py3-none-any.whl (95 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 95.2/95.2 kB 4.3 MB/s eta 0:00:00
Downloading groovy-0.1.2-py3-none-any.whl (14 kB)
Downloading python_multipart-0.0.20-py3-none-any.whl (24 kB)
Downloading ruff-0.11.9-py3-none-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (11.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 11.5/11.5 MB 22.5 MB/s eta 0:00:00
Downloading safehttpx-0.1.6-py3-none-any.whl (8.7 kB)
Downloading semantic_version-2.10.0-py2.py3-none-any.whl (15 kB)
Downloading starlette-0.46.2-py3-none-any.whl (72 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 72.0/72.0 kB 5.5 MB/s eta 0:00:00
Downloading tomlkit-0.13.2-py3-none-any.whl (37 kB)
```

```
def predict_price(open_price, high_price, low_price, volume):
    input_df = pd.DataFrame([{
        'Open': open_price,
        'High': high_price,
        'Low': low_price,
        'Volume': volume
    }])
    input_scaled = scaler.transform(input_df)
    prediction = model.predict(input_scaled)
    return prediction[0]
```

```
import gradio as gr

interface = gr.Interface(
    fn=predict_price,
    inputs=["number", "number", "number", "number"],
    outputs="number",
    title="Infy Stock Price Predictor",
    description="Enter stock data to predict closing price"
)
interface.launch()
```

```
It looks like you are running Gradio on a hosted a Jupyter notebook. For the Gradio app to work, sharing must be enabled. Automatica

Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://38a393097f33de6ada.gradio.live

This share link expires in 1 week. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the working
```