```
!pip install yfinance
```

```
Requirement already satisfied: yfinance in /usr/local/lib/python3.11/dist-packages (0.2.54)
Requirement already satisfied: pandas>=1.3.0 in /usr/local/lib/python3.11/dist-packages (from yfinance) (2.2.2)
Requirement already satisfied: numpy>=1.16.5 in /usr/local/lib/python3.11/dist-packages (from yfinance) (1.26.4)
Requirement already satisfied: requests>=2.31 in /usr/local/lib/python3.11/dist-packages (from yfinance) (2.32.3)
Requirement already satisfied: multitasking>=0.0.7 in /usr/local/lib/python3.11/dist-packages (from yfinance) (0.0.11)
Requirement already satisfied: platformdirs>=2.0.0 in /usr/local/lib/python3.11/dist-packages (from yfinance) (4.3.6)
Requirement already satisfied: pytz>=2022.5 in /usr/local/lib/python3.11/dist-packages (from yfinance) (2025.1)
Requirement already satisfied: frozendict>=2.3.4 in /usr/local/lib/python3.11/dist-packages (from yfinance) (2.4.6)
Requirement already satisfied: peewee>=3.16.2 in /usr/local/lib/python3.11/dist-packages (from yfinance) (3.17.9)
Requirement already satisfied: beautifulsoup4>=4.11.1 in /usr/local/lib/python3.11/dist-packages (from yfinance) (4.13.3)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.11/dist-packages (from beautifulsoup4>=4.11.1->yfinance) (2.6)
Requirement already satisfied: typing-extensions>=4.0.0 in /usr/local/lib/python3.11/dist-packages (from beautifulsoup4>=4.11.1->yfinanc
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.3.0->yfinance) (2.8.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.3.0->yfinance) (2025.1)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests>=2.31->yfinance) (3.4.
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests>=2.31->yfinance) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests>=2.31->yfinance) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests>=2.31->yfinance) (2025.1.31)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->pandas>=1.3.0->yfinance
```

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


# Dataset Initializations
df1 = pd.DataFrame({
    "USN": ["1BM22CS" + str(i) for i in range (200, 211)] # 10 records
    "Name": [i for i in "abcdefghij"]
    "Marks": [random.randint(0, 100) for i in range(10)]
})
print(df1.head())

from sklearn.datasets import load_diabetes
df2 = pd.DataFram(load_diabetes())
print(df2.head())
```

```
# Step 2: Downloading Stock Market Data
# Define the ticker symbols for Indian companies
# Example: Reliance Industries (RELIANCE.NS), TCS (TCS.NS), Infosys (INFY.NS)
tickers = ["RELIANCE.NS", "TCS.NS", "INFY.NS"]
# Fetch historical data for the last 1 year
data = yf.download(tickers, start="2022-10-01", end="2023-10-01", group_by='ticker')
data = pd.DataFrame(data)
print("\n", type(data))
# Display the first 5 rows of the dataset
print("First 5 rows of the dataset:")
print(data.head())
```

```
[*********************100%***********************]  3 of 3 completed
 <class 'pandas.core.frame.DataFrame'>
First 5 rows of the dataset:
Ticker      RELIANCE.NS                                                \
Price             Open         High          Low        Close    Volume
Date
2022-10-03  1092.351750  1103.976348  1079.334004  1082.302979  11852723
2022-10-04  1095.229543  1104.456151  1091.735252  1102.263672   8948850
2022-10-06  1109.480507  1119.072468  1104.524564  1106.328735  13352162
2022-10-07  1102.925982  1116.286369  1102.925982  1111.010742   7714340
2022-10-10  1098.518071  1104.273344  1090.753117  1098.883545   6329527

Ticker          INFY.NS                                            \
Price              Open         High          Low        Close   Volume
Date
2022-10-03  1326.433859  1326.433859  1302.009439  1309.289795  4943169
2022-10-04  1333.667406  1345.456933  1328.312866  1342.779663  6631341
2022-10-06  1357.434175  1371.337353  1356.588691  1366.969116  6180672
2022-10-07  1358.702473  1369.505584  1352.878234  1363.258545  3994466
2022-10-10  1339.914233  1376.222095  1339.914233  1374.014526  5274677

Ticker           TCS.NS
Price              Open         High          Low        Close   Volume
```

```
Date
2022-10-03  2848.973126  2873.420027  2828.997020  2839.413086  1763331
2022-10-04  2882.219505  2946.951656  2875.608411  2940.435547  2145875
2022-10-06  2959.317293  2971.683453  2941.671687  2950.708496  1790816
2022-10-07  2946.380491  2953.610032  2908.996820  2915.465088  1939879
2022-10-10  2863.242252  2974.537715  2858.486036  2966.499756  3064063
```

```
print(f"Shape of dataset:\n{data.shape}")
print(f"Column names:\n{data.columns}")
print(f"Summary of Reliance Stats:\n{data['RELIANCE.NS'].describe()}")
```

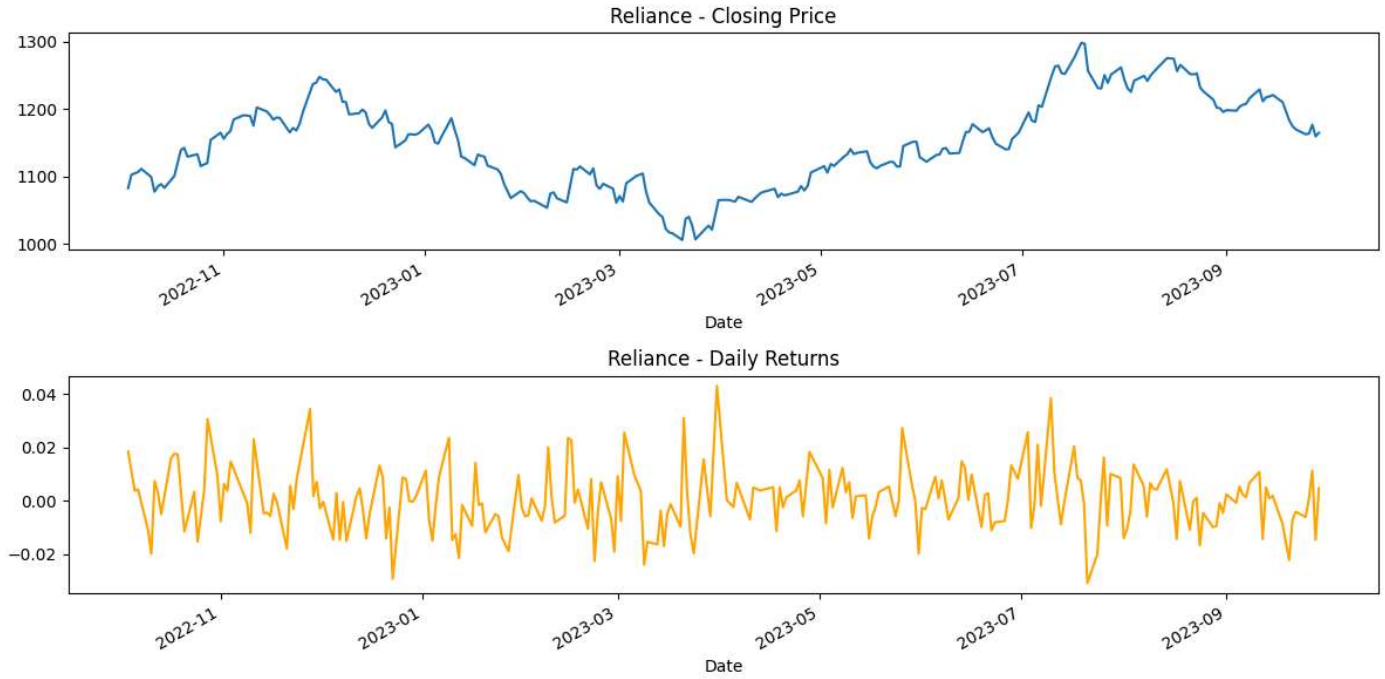
```
Shape of dataset:
(247, 15)
Column names:
MultiIndex([('RELIANCE.NS',   'Open'),
            ('RELIANCE.NS',   'High'),
            ('RELIANCE.NS',    'Low'),
            ('RELIANCE.NS',  'Close'),
            ('RELIANCE.NS', 'Volume'),
            (    'INFY.NS',   'Open'),
            (    'INFY.NS',   'High'),
            (    'INFY.NS',    'Low'),
            (    'INFY.NS',  'Close'),
            (    'INFY.NS', 'Volume'),
            (     'TCS.NS',   'Open'),
            (     'TCS.NS',   'High'),
            (     'TCS.NS',    'Low'),
            (     'TCS.NS',  'Close'),
            (     'TCS.NS', 'Volume')],
           names=['Ticker', 'Price'])
Summary of Reliance Stats:
Price         Open         High          Low        Close        Volume
count   247.000000   247.000000   247.000000   247.000000  2.470000e+02
mean   1151.113732  1159.809206  1140.728178  1150.085768  1.316652e+07
std      65.667213    66.649930    65.532728    66.499556  6.754099e+06
min    1011.732921  1014.016738   995.746138  1005.452393  3.370033e+06
25%    1102.777486  1107.310930  1088.640638  1101.247314  8.717141e+06
50%    1151.502781  1159.130787  1142.824338  1151.319946  1.158959e+07
75%    1198.585248  1204.999150  1189.185760  1197.370239  1.530302e+07
max    1292.642908  1304.518807  1277.569690  1298.055542  5.708188e+07
```

```
reliance_data = data['RELIANCE.NS']
reliance_data["daily_returns"] = reliance_data["Close"].pct_change()
print("Daily Returns:")
print(reliance_data["daily_returns"])
```

```
Daily Returns:
Date
2022-10-03         NaN
2022-10-04    0.018443
2022-10-06    0.003688
2022-10-07    0.004232
2022-10-10   -0.010915
                ...
2023-09-25   -0.006157
2023-09-26    0.000876
2023-09-27    0.011270
2023-09-28   -0.014690
2023-09-29    0.004670
Name: daily_returns, Length: 247, dtype: float64
<ipython-input-12-25188281a49c>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-cc
  reliance_data["daily_returns"] = reliance_data["Close"].pct_change()
```

```
plt.figure(figsize=(12, 6))
plt.subplot(2,1,1)
reliance_data["Close"].plot(title="Reliance - Closing Price")
plt.subplot(2, 1, 2)
reliance_data["daily_returns"].plot(title='Reliance - Daily Returns', color="orange")
plt.tight_layout()
plt.show()
```




```
tickers = ["HDFCBANK.NS", "ICICIBANK.NS", "KOTAKBANK.NS"]
data = yf.download(tickers, start="2024-01-01", end="2024-12-30", group_by='ticker')
data = pd.DataFrame(data)
print("\n", type(data))
print(data.describe())
```

```
[*********************100%***********************]  3 of 3 completed
 <class 'pandas.core.frame.DataFrame'>
Ticker KOTAKBANK.NS                                                    \
Price          Open         High          Low        Close        Volume
count    244.000000   244.000000   244.000000   244.000000  2.440000e+02
mean    1771.245907  1787.548029  1754.395105  1770.792347  5.736598e+06
std       62.189675    61.978802    62.765980    62.594747  5.388927e+06
min     1581.266899  1586.161558  1542.159736  1545.006592  1.824890e+05
25%     1733.974927  1754.131905  1719.028421  1736.297058  3.300380e+06
50%     1769.500000  1789.450012  1758.099976  1773.681030  4.307680e+06
75%     1809.925018  1826.998164  1789.912506  1808.155670  6.159475e+06
max     1935.000000  1942.000000  1909.599976  1934.699951  6.617908e+07

Ticker  HDFCBANK.NS                                                    \
Price          Open         High          Low        Close        Volume
count    244.000000   244.000000   244.000000   244.000000  2.440000e+02
mean    1601.375295  1615.443664  1588.221245  1601.898968  2.119658e+07
std      134.648125   134.183203   132.796819   133.748372  2.133860e+07
min     1357.463183  1372.754374  1345.180951  1365.404785  8.798460e+05
25%     1475.316358  1494.072805  1460.259509  1474.564087  1.274850e+07
50%     1627.724976  1638.350037  1616.000000  1625.950012  1.686810e+07
75%     1696.474976  1711.425018  1679.250000  1697.062531  2.295014e+07
max     1877.699951  1880.000000  1858.550049  1871.750000  2.226710e+08

Ticker ICICIBANK.NS
Price          Open         High          Low        Close        Volume
count    244.000000   244.000000   244.000000   244.000000  2.440000e+02
mean    1161.723560  1173.687900  1151.318979  1162.751791  1.539172e+07
std      104.905646   105.668229   105.083015   105.520481  9.503609e+06
min      965.637027   979.567116   961.869473   971.387512  1.007022e+06
25%     1073.818215  1085.368782  1067.386038  1075.107086  1.014533e+07
50%     1169.443635  1178.450012  1157.361521  1165.470703  1.291768e+07
75%     1248.512512  1261.399994  1236.649963  1250.812531  1.755770e+07
max     1344.900024  1362.349976  1340.050049  1346.099976  7.325777e+07
```

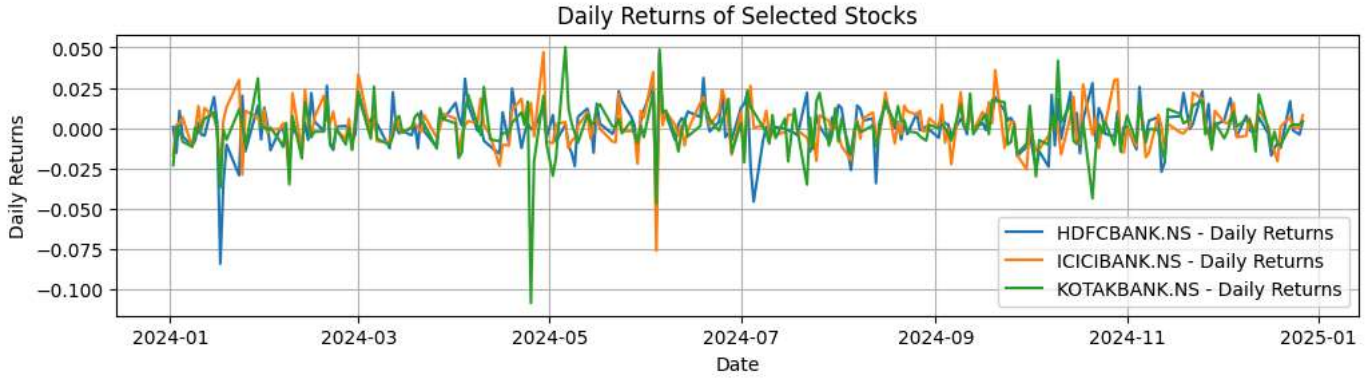
```
print(data.columns)
```

```
MultiIndex([('KOTAKBANK.NS',    'Open'),
            ('KOTAKBANK.NS',    'High'),
            ('KOTAKBANK.NS',     'Low'),
```

```
                    ('KOTAKBANK.NS',  'Close'),
                    ('KOTAKBANK.NS', 'Volume'),
                    ( 'HDFCBANK.NS',   'Open'),
                    ( 'HDFCBANK.NS',   'High'),
                    ( 'HDFCBANK.NS',    'Low'),
                    ( 'HDFCBANK.NS',  'Close'),
                    ( 'HDFCBANK.NS', 'Volume'),
                    ('ICICIBANK.NS',   'Open'),
                    ('ICICIBANK.NS',   'High'),
                    ('ICICIBANK.NS',    'Low'),
                    ('ICICIBANK.NS',  'Close'),
                    ('ICICIBANK.NS', 'Volume')],
                   names=['Ticker', 'Price'])


for t in tickers:
    # Calculate daily returns first
    data[t, "daily_returns"] = data[t]["Close"].pct_change()
    # Then, print them
    print(f"Daily Returns for {t}:")
    print(data[t]["daily_returns"])
plt.figure(figsize=(12, 6))
plt.subplot(2, 1, 1)
for t in tickers:
    plt.plot(data[t].index, data[t]["daily_returns"], label=f"{t} - Daily Returns")
plt.xlabel("Date")
plt.ylabel("Daily Returns")
plt.title("Daily Returns of Selected Stocks")
plt.grid(True)
plt.legend()
plt.show()
plt.subplot(2, 1, 2)
for t in tickers:
    plt.plot(data[t].index, data[t]["Close"], label=f"{t} - Close Price")
# plt.axhline(y=0, color='r', linestyle='--')
plt.xlabel("Date")
plt.ylabel("Close Price")
plt.title("Close Price of Selected Stocks")
plt.grid(True)
plt.legend()
plt.show()
```

```
Daily Returns for HDFCBANK.NS:
Date
2024-01-01         NaN
2024-01-02    0.000589
2024-01-03   -0.015420
2024-01-04    0.010730
2024-01-05   -0.005116
                ...
2024-12-20   -0.012267
2024-12-23    0.016653
2024-12-24   -0.001610
2024-12-26   -0.004088
2024-12-27    0.004188
Name: daily_returns, Length: 244, dtype: float64
Daily Returns for ICICIBANK.NS:
Date
2024-01-01         NaN
2024-01-02   -0.017160
2024-01-03    0.001833
2024-01-04    0.003150
2024-01-05    0.006635
                ...
2024-12-20    0.001166
2024-12-23    0.006520
2024-12-24    0.000347
2024-12-26    0.000077
2024-12-27    0.007862
Name: daily_returns, Length: 244, dtype: float64
Daily Returns for KOTAKBANK.NS:
Date
2024-01-01         NaN
2024-01-02   -0.023099
2024-01-03    0.000456
2024-01-04   -0.001233
2024-01-05   -0.008586
                ...
2024-12-20   -0.010527
2024-12-23    0.001032
2024-12-24    0.002120
2024-12-26    0.002144
2024-12-27    0.004051
Name: daily_returns, Length: 244, dtype: float64
```



Start coding or generate with AI.

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

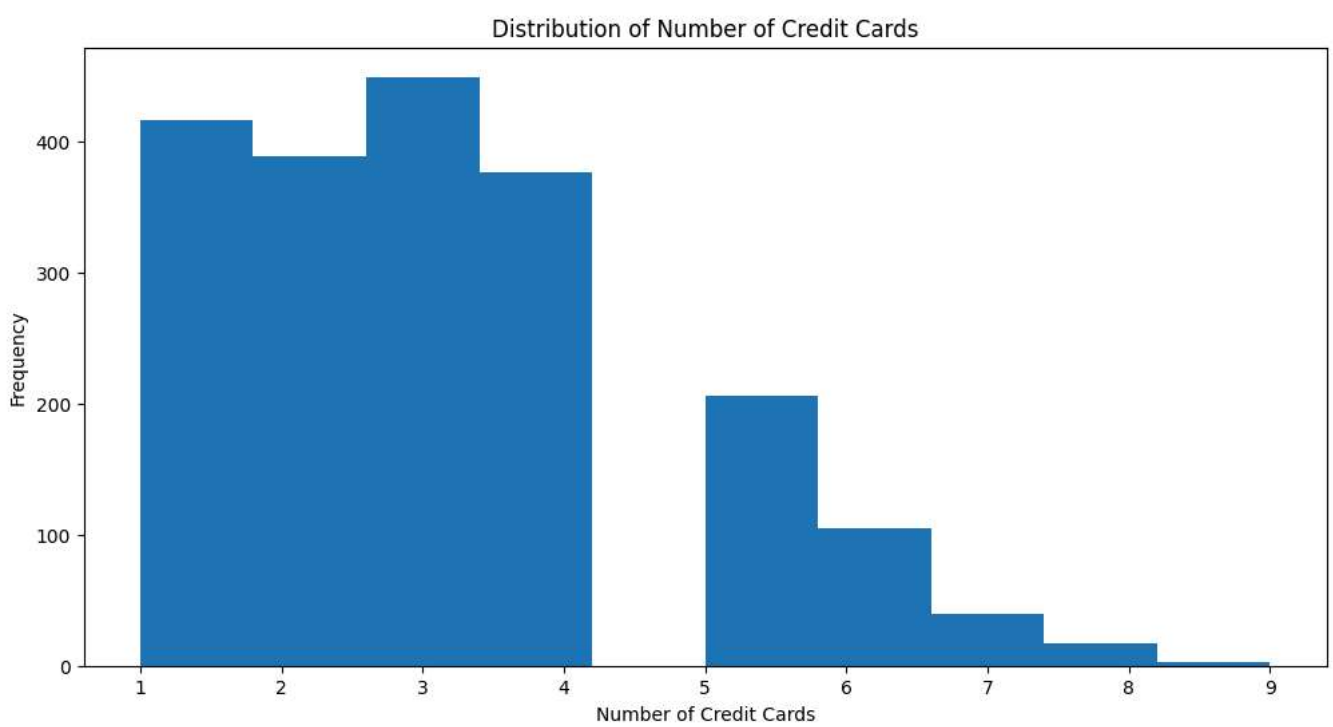
```
import pandas as pd
df = pd.read_csv("/content/drive/My Drive/users_data.csv")
print(df.head())
```

```
     id  current_age  retirement_age  birth_year  birth_month  gender  \
0   825           53              66        1966           11  Female
1  1746           53              68        1966           12  Female
2  1718           81              67        1938           11  Female
3   708           63              63        1957            1  Female
4  1164           43              70        1976            9    Male

                  address  latitude  longitude per_capita_income  \
0             462 Rose Lane     34.15    -117.76           $29278
1    3606 Federal Boulevard     40.76     -73.74           $37891
2           766 Third Drive     34.02    -117.89           $22681
3          3 Madison Street     40.71     -73.99          $163145
4  9620 Valley Stream Drive     37.76    -122.44           $53797

  yearly_income total_debt  credit_score  num_credit_cards
0        $59696    $127613           787                 5
1        $77254    $191349           701                 5
2        $33483       $196           698                 5
3       $249925    $202328           722                 4
4       $109687    $183855           675                 1
```

```
import matplotlib.pyplot as plt
plt.figure(figsize=(12, 6))
df["num_credit_cards"].plot(kind="hist", bins=10, title="Distribution of Number of Credit Cards")
plt.xlabel("Number of Credit Cards")
plt.ylabel("Frequency")
plt.show()
```




```
plt.plot(df["current_age"], df["yearly_income"],kind)
plt.title("Age vs. Yearly Income")
plt.xlabel("Age")
plt.ylabel("Yearly Income")
plt.grid(True)
plt.show()
```