## import the libraries

Generate | randomly select 5 items from a list | Close

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

## read dataset

Generate | randomly select 5 items from a list | Close

```
btc = pd.read_csv('BTC-USD.csv')
```

## 15 Basic functions

Double-click (or enter) to edit

```
# 1
print("1. First 5 rows:")
print(btc.head())
```

```
1. First 5 rows:
         Date        Open        High         Low       Close   Adj Close  \
0  2014-09-17  465.864014  468.174011  452.421997  457.334015  457.334015
1  2014-09-18  456.859985  456.859985  413.104004  424.440002  424.440002
2  2014-09-19  424.102997  427.834991  384.532013  394.795990  394.795990
3  2014-09-20  394.673004  423.295990  389.882996  408.903992  408.903992
4  2014-09-21  408.084991  412.425995  393.181000  398.821014  398.821014

     Volume
0  21056800
1  34483200
2  37919700
3  36863600
4  26580100
```

```
# 2
print("\n2. Last 5 rows:")
print(btc.tail())
```

```
2. Last 5 rows:
            Date          Open          High           Low         Close  \
2708  2022-02-15  42586.464844  44667.218750  42491.035156  44575.203125
2709  2022-02-16  44578.277344  44578.277344  43456.691406  43961.859375
2710  2022-02-17  43937.070313  44132.972656  40249.371094  40538.011719
2711  2022-02-18  40552.132813  40929.152344  39637.617188  40030.976563
2712  2022-02-19  40022.132813  40246.027344  40010.867188  40126.429688

         Adj Close       Volume
2708  44575.203125  22721659051
2709  43961.859375  19792547657
2710  40538.011719  26246662813
2711  40030.976563  23310007704
2712  40126.429688  22263900160
```

```
# 3
print("\n3. Dataset shape:")
print(btc.shape)
```

```
3. Dataset shape:
(2713, 7)
```

```
# 4
print("\n4. Column names:")
print(btc.columns)
```

```
4. Column names:
Index(['Date', 'Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume'], dtype='object')
```

```
# 5
print("\n5. Data types:")
print(btc.dtypes)
```

```
5. Data types:
Date          object
Open         float64
High         float64
Low          float64
Close        float64
Adj Close    float64
Volume         int64
dtype: object
```

```
# 6
print("\n6. Summary statistics:")
print(btc.describe())
```

```
6. Summary statistics:
                Open          High           Low         Close     Adj Close  \
count   2713.000000   2713.000000   2713.000000   2713.000000   2713.000000
mean   11311.041069  11614.292482  10975.555057  11323.914637  11323.914637
std    16106.428891  16537.390649  15608.572560  16110.365010  16110.365010
min      176.897003    211.731003    171.509995    178.102997    178.102997
25%      606.396973    609.260986    604.109985    606.718994    606.718994
50%     6301.569824   6434.617676   6214.220215   6317.609863   6317.609863
75%    10452.399414  10762.644531  10202.387695  10462.259766  10462.259766
max    67549.734375  68789.625000  66382.062500  67566.828125  67566.828125

             Volume
count  2.713000e+03
mean   1.470462e+10
std    2.001627e+10
min    5.914570e+06
25%    7.991080e+07
50%    5.098183e+09
75%    2.456992e+10
max    3.509679e+11
```

```
# 7
print("\n7. Missing values:")
print(btc.isnull().sum())
```

```
7. Missing values:
Date         0
Open         0
High         0
Low          0
Close        0
Adj Close    0
Volume       0
dtype: int64
```

```
# 8
print("\n8. Unique dates:")
print(btc['Date'].nunique())
```

```
8. Unique dates:
2713
```

```
# 9
print("\n9. Memory usage:")
print(btc.memory_usage())
```

```
9. Memory usage:
Index          132
Date         21704
Open         21704
High         21704
Low          21704
Close        21704
```

```
Adj Close    21704
Volume       21704
dtype: int64
```

```
# 10
print("\n10. Correlation matrix:")
# Select only numeric columns before calculating correlation
print(btc.select_dtypes(include=np.number).corr())
```

```
10. Correlation matrix:
                Open      High       Low     Close  Adj Close    Volume
Open        1.000000  0.999535  0.999103  0.998839   0.998839  0.728537
High        0.999535  1.000000  0.999046  0.999489   0.999489  0.732137
Low         0.999103  0.999046  1.000000  0.999399   0.999399  0.720922
Close       0.998839  0.999489  0.999399  1.000000   1.000000  0.727443
Adj Close   0.998839  0.999489  0.999399  1.000000   1.000000  0.727443
Volume      0.728537  0.732137  0.720922  0.727443   0.727443  1.000000
```

```
# 11
print("\n11. Number of unique values per column:")
print(btc.nunique())
```

```
11. Number of unique values per column:
Date         2713
Open         2709
High         2710
Low          2712
Close        2710
Adj Close    2710
Volume       2713
dtype: int64
```

```
# 12
print("\n12. Index information:")
print(btc.index)
```

```
12. Index information:
RangeIndex(start=0, stop=2713, step=1)
```

```
# 13
print("\n13. Sample of 3 random rows:")
print(btc.sample(3))
```

```
13. Sample of 3 random rows:
            Date          Open          High           Low         Close  \
2526  2021-08-17  45936.457031  47139.570313  44512.417969  44695.359375
2210  2020-10-05  10676.529297  10793.507813  10634.600586  10793.339844
1413  2018-07-31   8181.200195   8181.529785   7696.930176   7780.439941

         Adj Close       Volume
2526  44695.359375  33451362600
2210  10793.339844  47537578009
1413   7780.439941   5287530000
```

```
# 14
print("\n14. Count of non-NA values:")
print(btc.count())
```

```
14. Count of non-NA values:
Date         2713
Open         2713
High         2713
Low          2713
Close        2713
Adj Close    2713
Volume       2713
dtype: int64
```

Generate | 10 random numbers using numpy | Close

```
# 15
print("\n15. Value counts for Volume (top 5):")
print(btc['Volume'].value_counts().head())
```

```
15. Value counts for Volume (top 5):
Volume
```

```
22263900160    1
21056800       1
34483200       1
37919700       1
36863600       1
Name: count, dtype: int64
```

## Data Cleaning

```
# 1
# Check for missing values
print("Missing values before cleaning:")
print(btc.isnull().sum())
```

```
Missing values before cleaning:
Date         0
Open         0
High         0
Low          0
Close        0
Adj Close    0
Volume       0
dtype: int64
```

```
# Drop rows with any missing values
btc_clean = btc.dropna()
```

```
# Convert Date to datetime format
btc_clean['Date'] = pd.to_datetime(btc_clean['Date'])
```

```
# 2
# Verify data types
print("\nData types after cleaning:")
print(btc_clean.dtypes)
```

```
Data types after cleaning:
Date         datetime64[ns]
Open                float64
High                float64
Low                 float64
Close               float64
Adj Close           float64
Volume                int64
dtype: object
```

```
Start coding or generate with AI.
```

```
# 3
# Check for duplicates
print("\nNumber of duplicate rows:", btc_clean.duplicated().sum())
```

```
Number of duplicate rows: 0
```

```
# 4
# Remove potential outliers using z-score (for Close price)
z_scores = np.abs(stats.zscore(btc_clean['Close']))
btc_clean = btc_clean[(z_scores < 3)]

print("\nDataset shape after cleaning:", btc_clean.shape)
```

```
Dataset shape after cleaning: (2609, 7)
```

## Data Filtering

```
#1
# Filter for prices above $1000
high_prices = btc_clean[btc_clean['Close'] > 1000]
print("Number of days with price > $1000:", len(high_prices))
```

```
Number of days with price > $1000: 1736
```

```
# 2
# Filter for specific year (2017)
btc_2017 = btc_clean[btc_clean['Date'].dt.year == 2017]
print("\n2017 data shape:", btc_2017.shape)
```

```
2017 data shape: (365, 7)
```

## Grouping

```
# 1
# Group by year and get average closing price
yearly_avg = btc_clean.groupby(btc_clean['Date'].dt.year)['Close'].mean()
print("Yearly average closing prices:")
print(yearly_avg)
```

```
Yearly average closing prices:
Date
2014      363.693085
2015      272.453381
2016      568.492407
2017     4006.033629
2018     7572.298947
2019     7395.246282
2020    11116.378092
2021    42661.042460
2022    41345.687735
Name: Close, dtype: float64
```

```
# 2
# Group by month in 2017 and get max volume
btc_2017 = btc_clean[btc_clean['Date'].dt.year == 2017]
monthly_max_vol = btc_2017.groupby(btc_2017['Date'].dt.month)['Volume'].max()
print("\nMonthly max volume in 2017:")
print(monthly_max_vol)
```

```
Monthly max volume in 2017:
Date
1       510199008
2       407220000
3       706598976
4       580444032
5      2406700032
6      2569530112
7      2249260032
8      3764239872
9      4148069888
10     3615480064
11    11568799744
12    22197999616
Name: Volume, dtype: int64
```

## Sorting

```
# 1
# Sort by highest closing price
highest_prices = btc_clean.sort_values('Close', ascending=False)
print("Top 5 highest closing prices:")
print(highest_prices[['Date', 'Close']].head())
```

```
Top 5 highest closing prices:
           Date         Close
2414 2021-04-27  55033.117188
2579 2021-10-09  54968.222656
2415 2021-04-28  54824.703125
2365 2021-03-09  54824.117188
2628 2021-11-27  54815.078125
```

```
# 2
# Sort by date (chronological order)
btc_sorted = btc_clean.sort_values('Date')
print("\nFirst 5 dates in chronological order:")
print(btc_sorted[['Date', 'Close']].head())
```

```
First 5 dates in chronological order:
        Date       Close
```

```
0 2014-09-17  457.334015
1 2014-09-18  424.440002
2 2014-09-19  394.795990
3 2014-09-20  408.903992
4 2014-09-21  398.821014
```

## Aggregations

```
# Calculate various aggregations for Close price
close_stats = {
    'Minimum': btc_clean['Close'].min(),
    'Maximum': btc_clean['Close'].max(),
    'Mean': btc_clean['Close'].mean(),
    'Median': btc_clean['Close'].median(),
    'Mode': btc_clean['Close'].mode()[0],
    'Variance': btc_clean['Close'].var(),
    'Std Dev': btc_clean['Close'].std()
}

print("Close Price Statistics:")
for stat, value in close_stats.items():
    print(f"{stat}: {value:.2f}")
```

```
Close Price Statistics:
Minimum: 178.10
Maximum: 55033.12
Mean: 9406.60
Median: 5725.59
Mode: 236.15
Variance: 173628838.32
Std Dev: 13176.83
```

## Additional Queries

```
# 1. Days with highest price volatility (difference between High and Low)
btc_clean['Volatility'] = btc_clean['High'] - btc_clean['Low']
print("Top 5 most volatile days:")
print(btc_clean.sort_values('Volatility', ascending=False)[['Date', 'Volatility']].head())
```

```
Top 5 most volatile days:
            Date    Volatility
2436 2021-05-19  12864.621094
2635 2021-12-04  11030.062500
2547 2021-09-07   9568.558594
2351 2021-02-23   8914.339844
2429 2021-05-12   8788.828125
```

```
# 2. Percentage change from previous day
btc_clean['Daily_Change'] = btc_clean['Close'].pct_change() * 100
print("\nLargest single day percentage gains:")
print(btc_clean.sort_values('Daily_Change', ascending=False)[['Date', 'Daily_Change']].head())
```

```

Largest single day percentage gains:
            Date  Daily_Change
1177 2017-12-07     25.247169
1037 2017-07-20     23.936087
1176 2017-12-06     19.928334
2336 2021-02-08     18.746474
2010 2020-03-19     18.187756
```

```
# 3. Days where closing price was higher than opening price
up_days = btc_clean[btc_clean['Close'] > btc_clean['Open']]
print("\nNumber of days closing higher than opening:", len(up_days))
```

```

Number of days closing higher than opening: 1408
```

```
# 4. Average volume by year
yearly_volume = btc_clean.groupby(btc_clean['Date'].dt.year)['Volume'].mean()
print("\nAverage yearly volume:")
print(yearly_volume)
```

```

Average yearly volume:
```

```
Date
2014    2.383690e+07
2015    3.390557e+07
2016    8.592451e+07
2017    2.382867e+09
2018    6.063552e+09
2019    1.673049e+10
2020    3.302327e+10
2021    4.676478e+10
2022    2.718684e+10
Name: Volume, dtype: float64
```

```
# 5. Correlation between volume and price change
correlation = btc_clean['Volume'].corr(btc_clean['Daily_Change'])
print("\nCorrelation between volume and daily change:", correlation)
```

```
Correlation between volume and daily change: 0.009254312543180031
```

```
# 6. Days with highest trading volume
print("\nTop 5 days by trading volume:")
print(btc_clean.sort_values('Volume', ascending=False)[['Date', 'Volume']].head())
```

```
Top 5 days by trading volume:
           Date        Volume
2354 2021-02-26  350967941479
2436 2021-05-19  126358098747
2308 2021-01-11  123320567399
2326 2021-01-29  117894572511
2351 2021-02-23  106102492824
```

```
# 7. Monthly average price in 2017
btc_2017 = btc_clean[btc_clean['Date'].dt.year == 2017]
monthly_avg_2017 = btc_2017.groupby(btc_2017['Date'].dt.month)['Close'].mean()
print("\n2017 monthly average prices:")
print(monthly_avg_2017)
```

```
2017 monthly average prices:
Date
1       914.916159
2      1062.533672
3      1129.365228
4      1206.641007
5      1895.383529
6      2636.204346
7      2519.418386
8      3880.989998
9      4064.836312
10     5360.071604
11     7813.132975
12    15294.270980
Name: Close, dtype: float64
```

```
# 8. Price range for each year
yearly_range = btc_clean.groupby(btc_clean['Date'].dt.year)['Close'].agg(['min', 'max'])
print("\nYearly price ranges:")
print(yearly_range)
```

```
Yearly price ranges:
                min           max
Date
2014    310.737000    457.334015
2015    178.102997    465.321014
2016    364.330994    975.921021
2017    777.757019  19497.400391
2018   3236.761719  17527.000000
2019   3399.471680  13016.231445
2020   4970.788086  29001.720703
2021  29374.152344  55033.117188
2022  35030.250000  47686.812500
```

```
# 9. Days where price doubled from previous day
doubled_days = btc_clean[btc_clean['Daily_Change'] > 100]
print("\nDays where price more than doubled:")
print(doubled_days[['Date', 'Daily_Change']])
```

```
Days where price more than doubled:
Empty DataFrame
Columns: [Date, Daily_Change]
Index: []
```

```
# 10. Rolling 7-day average price
btc_clean['7_Day_Avg'] = btc_clean['Close'].rolling(window=7).mean()
print("\n7-day rolling average (last 5 days):")
print(btc_clean[['Date', 'Close', '7_Day_Avg']].tail())
```

```
7-day rolling average (last 5 days):
           Date         Close     7_Day_Avg
2708 2022-02-15  44575.203125  43130.850446
2709 2022-02-16  43961.859375  43077.002232
2710 2022-02-17  40538.011719  42644.559152
2711 2022-02-18  40030.976563  42304.993304
2712 2022-02-19  40126.429688  42002.416295
```

```
# 11. Quarterly performance
btc_clean['Quarter'] = btc_clean['Date'].dt.quarter
quarterly_perf = btc_clean.groupby([btc_clean['Date'].dt.year, 'Quarter'])['Close'].last().unstack()
print("\nQuarterly closing prices:")
print(quarterly_perf)
```

```
Quarterly closing prices:
Quarter             1             2             3             4
Date
2014              NaN           NaN    386.944000    320.192993
2015       244.223999    263.071991    236.059998    430.566986
2016       416.729004    673.336975    609.734985    963.742981
2017      1071.790039   2480.840088   4338.709961  14156.400391
2018      6973.529785   6404.000000   6625.560059   3742.700439
2019      4105.404297  10817.155273   8293.868164   7193.599121
2020      6438.644531   9137.993164  10784.491211  29001.720703
2021     51704.160156  35040.835938  43790.894531  46306.445313
2022     40126.429688           NaN           NaN           NaN
```

```
# 12. Days with lowest volatility
print("\nTop 5 least volatile days:")
print(btc_clean.sort_values('Volatility')[['Date', 'Volatility']].head())
```

```
Top 5 least volatile days:
          Date  Volatility
382 2015-10-04    1.028000
253 2015-05-28    1.172012
262 2015-06-06    1.339996
365 2015-09-17    1.359009
368 2015-09-20    1.455001
```

```
# 13. Price change by year
annual_change = btc_clean.groupby(btc_clean['Date'].dt.year)['Close'].agg(['first', 'last'])
annual_change['Change'] = (annual_change['last'] - annual_change['first']) / annual_change['first'] * 100
print("\nAnnual price changes:")
print(annual_change)
```

```
Annual price changes:
              first          last       Change
Date
2014    457.334015    320.192993   -29.987059
2015    314.248993    430.566986    37.014595
2016    434.334015    963.742981   121.889824
2017    998.325012  14156.400391  1318.015198
2018  13657.200195   3742.700439   -72.595405
2019   3843.520020   7193.599121    87.161745
2020   7200.174316  29001.720703   302.791925
2021  29374.152344  46306.445313    57.643512
2022  47686.812500  40126.429688   -15.854242
```

```
# 14. Volume vs Price scatter
high_volume_days = btc_clean[btc_clean['Volume'] > btc_clean['Volume'].quantile(0.9)]
print("\nAverage price on high volume days:", high_volume_days['Close'].mean())
```

```
Average price on high volume days: 28415.67222864751
```

```
# 15. Price momentum (3-day)
btc_clean['3_Day_Momentum'] = btc_clean['Close'].diff(3)
print("\nRecent 3-day momentum values:")
print(btc_clean[['Date', 'Close', '3_Day_Momentum']].tail())
```

```
Recent 3-day momentum values:
           Date         Close  3_Day_Momentum
2708 2022-02-15  44575.203125     2330.734375
2709 2022-02-16  43961.859375     1764.343750
2710 2022-02-17  40538.011719    -2048.906250
2711 2022-02-18  40030.976563    -4544.226562
2712 2022-02-19  40126.429688    -3835.429687
```

```
# 16. Price bands (20-day)
btc_clean['20_Day_MA'] = btc_clean['Close'].rolling(window=20).mean()
btc_clean['Upper_Band'] = btc_clean['20_Day_MA'] + (btc_clean['Close'].rolling(window=20).std() * 2)
btc_clean['Lower_Band'] = btc_clean['20_Day_MA'] - (btc_clean['Close'].rolling(window=20).std() * 2)
print("\nPrice bands (last 5 days):")
print(btc_clean[['Date', 'Close', '20_Day_MA', 'Upper_Band', 'Lower_Band']].tail())
```

```
Price bands (last 5 days):
           Date         Close     20_Day_MA    Upper_Band    Lower_Band
2708 2022-02-15  44575.203125  40877.074414  46372.734271  35381.414557
2709 2022-02-16  43961.859375  41218.255664  46582.262366  35854.248963
2710 2022-02-17  40538.011719  41355.939649  46485.040528  36226.838769
2711 2022-02-18  40030.976563  41450.579492  46396.260739  36504.898246
2712 2022-02-19  40126.429688  41561.020899  46267.374226  36854.667571
```

```
# 17. Relative Strength Index (14-day)
delta = btc_clean['Close'].diff()
gain = delta.where(delta > 0, 0)
loss = -delta.where(delta < 0, 0)
avg_gain = gain.rolling(window=14).mean()
avg_loss = loss.rolling(window=14).mean()
rs = avg_gain / avg_loss
btc_clean['RSI'] = 100 - (100 / (1 + rs))
print("\nRSI values (last 5 days):")
print(btc_clean[['Date', 'Close', 'RSI']].tail())
```

```
RSI values (last 5 days):
           Date         Close        RSI
2708 2022-02-15  44575.203125  71.108013
2709 2022-02-16  43961.859375  77.730362
2710 2022-02-17  40538.011719  60.666638
2711 2022-02-18  40030.976563  43.885888
2712 2022-02-19  40126.429688  44.547513
```

```
# 18. Days closing above opening by more than 10%
big_up_days = btc_clean[(btc_clean['Close'] - btc_clean['Open']) / btc_clean['Open'] > 0.1]
print("\nDays closing >10% above open:", len(big_up_days))
```

```
Days closing >10% above open: 47
```

```
# 19. Monthly returns
monthly_returns = btc_clean.groupby([btc_clean['Date'].dt.year, btc_clean['Date'].dt.month])['Close'].last().pct_change()
print("\nMonthly returns (last 5 months):")
print(monthly_returns.tail())
```

```
Monthly returns (last 5 months):
Date  Date
2021  10      0.250753
      11      0.000794
      12     -0.155224
2022  1      -0.168947
      2       0.042702
Name: Close, dtype: float64
```

```
# 20. Volatility clusters
btc_clean['High_Volatility'] = btc_clean['Volatility'] > btc_clean['Volatility'].quantile(0.9)
print("\nHigh volatility days count:", btc_clean['High_Volatility'].sum())
```

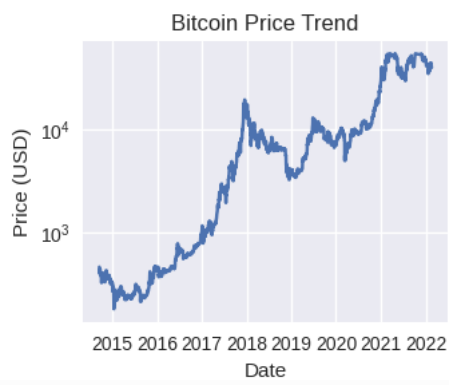
```
High volatility days count: 261
```

## Data Visualization

```
# Set style
# Use an available seaborn style from matplotlib
```

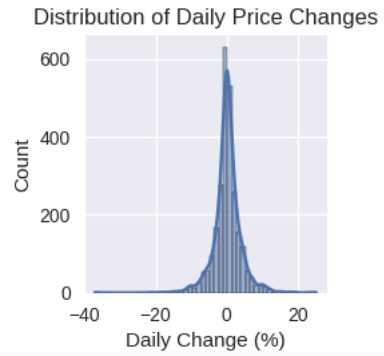
```
plt.style.use('seaborn-v0_8')
plt.figure(figsize=(12, 6))

# 1. Price Trend Over Time
plt.subplot(2, 3, 1)
plt.plot(btc_clean['Date'], btc_clean['Close'])
plt.title('Bitcoin Price Trend')
plt.xlabel('Date')
plt.ylabel('Price (USD)')
plt.yscale('log')  # Log scale for better visualization
```

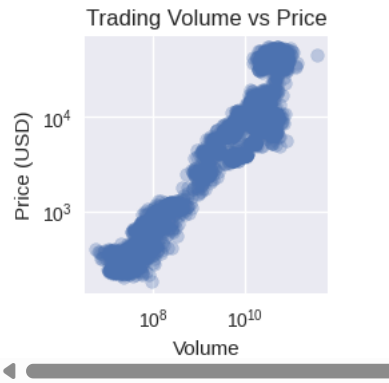



```
# 2. Daily Price Changes Distribution
plt.subplot(2, 3, 2)
sns.histplot(btc_clean['Daily_Change'].dropna(), bins=50, kde=True)
plt.title('Distribution of Daily Price Changes')
plt.xlabel('Daily Change (%)')
```

```
Text(0.5, 0, 'Daily Change (%)')
```



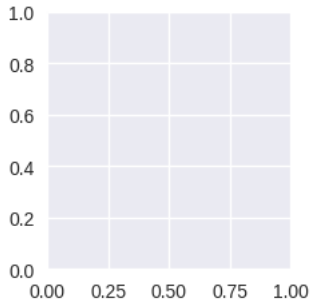

```
# 3. Volume vs Price Scatter
plt.subplot(2, 3, 3)
plt.scatter(btc_clean['Volume'], btc_clean['Close'], alpha=0.3)
plt.title('Trading Volume vs Price')
plt.xlabel('Volume')
plt.ylabel('Price (USD)')
plt.yscale('log')
plt.xscale('log')
```

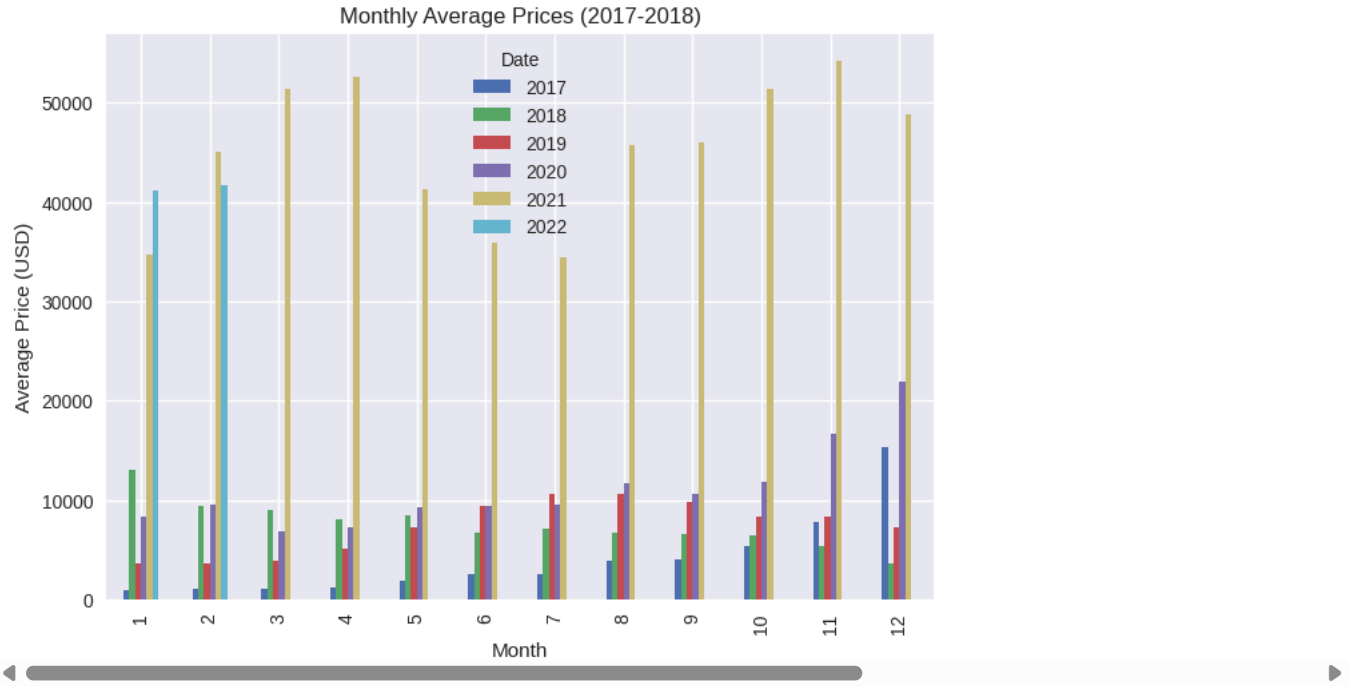



```
# 4. Monthly Average Prices (2017-2018)
btc_recent = btc_clean[btc_clean['Date'].dt.year >= 2017]
monthly_avg = btc_recent.groupby([btc_recent['Date'].dt.year, btc_recent['Date'].dt.month])['Close'].mean().unstack(0)
plt.subplot(2, 3, 4)
monthly_avg.plot(kind='bar')
plt.title('Monthly Average Prices (2017-2018)')
```

```
plt.xlabel('Month')
plt.ylabel('Average Price (USD)')
```

```
Text(0, 0.5, 'Average Price (USD)')
```




```
# 5. Moving Averages
plt.subplot(2, 3, 5)
plt.plot(btc_clean['Date'], btc_clean['Close'], label='Daily Close')
plt.plot(btc_clean['Date'], btc_clean['20_Day_MA'], label='20-Day MA')
plt.plot(btc_clean['Date'], btc_clean['Upper_Band'], label='Upper Band', linestyle='--')
plt.plot(btc_clean['Date'], btc_clean['Lower_Band'], label='Lower Band', linestyle='--')
plt.title('Price with Moving Averages')
plt.xlabel('Date')
plt.ylabel('Price (USD)')
plt.yscale('log')
plt.legend()

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