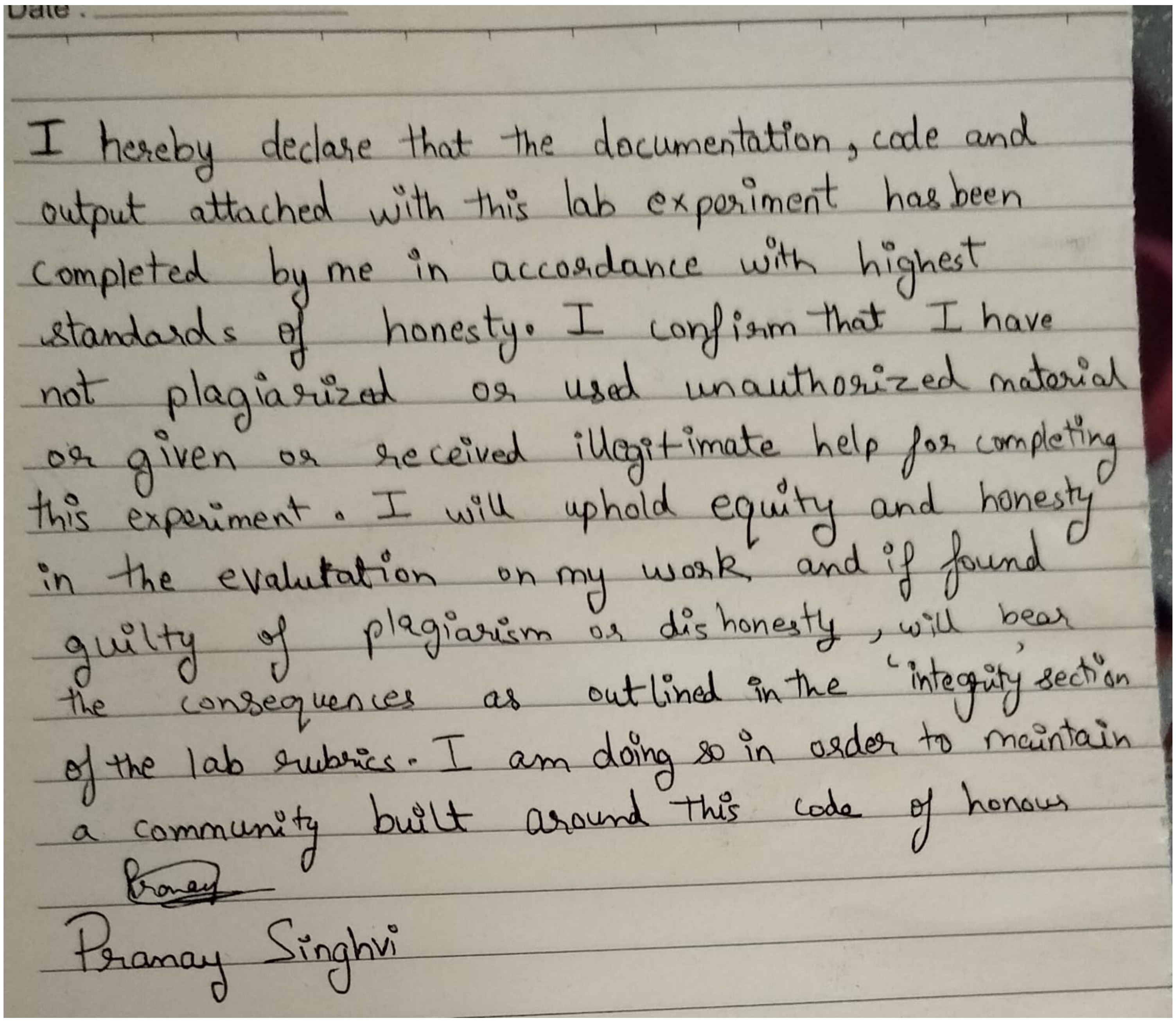
Experiment 4

Name	Pranay Singhvi
UID	2021300126

HONOR PLEDGE	 Date : I hereby declare that the documentation, code and output attached with this lab experiment has been completed by me in accordance with highest standards of honesty. I confirm that I have not plagiarized or used unauthorized material or given or received illegitimate help for completing this experiment. I will uphold equity and honesty in the evalutation on my work and if found guilty of plagiarism or dishonesty, will bear the consequences as outlined in the 'integrity section of the lab rubrics. I am doing so in order to maintain a community built around this code of honour Pranay Pranay Singhvi
PROBLEM STATEMENT	**Dealing with Time Series Data :** ● Resample a time series to a different time frequency(eg. Daily, monthly) One up sampling and one down sampling and OHLC sampling required to be done. For up sampling use FFill ● Shift a time series forward and backward in time Use naive shifts, and shift using frequency ● Compute moving averages or rolling sums over a time series Apply 3 moving window functions to your dataset
THEORY	**Resampling Time Series** • **Purpose**: Resampling changes the granularity of a time series. You might resample to analyze trends at a lower level (e.g., hourly data to daily trends) or aggregate to a higher level (e.g., daily to monthly performance). • **Upsampling**: Increasing the frequency of data (e.g., daily to hourly). Techniques include: ▪ **Forward Fill (FFill)**: Propagates the last valid observation forward to fill in missing values. Useful when changes between points are expected to be minimal. ▪ **Interpolation**: Estimates missing values using existing data points (e.g., linear interpolation). This is more complex but generally smoother than FFill. • **Downsampling**: Decreasing the frequency of data (e.g., hourly to daily). Techniques include: ▪ **Aggregation**: Calculate summary statistics (average, sum, max, min) within each new time window. ▪ **OHLC Sampling**: Preserves Open, High, Low, and Close values of a times series. This is mainly used in financial data analysis. **Shifting Time Series** • **Purpose**: Shifting is crucial for creating lagged features. Lagged features allow you to incorporate historical trends into forecasting and models. • **Naive Shift**: Simple shifting by a defined number of periods. For example, shifting by one period in a daily time series aligns today's data point with yesterday's. • **Shifting Using Frequency**: Leverages the underlying frequency of your time series for more precise shifts. You might shift data forward by two months if working with monthly data. **Moving Averages and Rolling Sums** • **Purpose**: Smoothing time series to reduce noise and reveal underlying trends. These techniques give you insights into trend direction and momentum. • **Moving Averages (MA)**:

  - **Simple Moving Average (SMA)**: Averages a fixed number of past data points, giving them equal weight. As time progresses, the calculation window 'moves' through the data.
  - **Weighted Moving Average (WMA)**: Allows for greater weighting of more recent data points, which emphasizes current trends.
  - **Exponential Moving Average (EMA)**: Applies exponential weighting that decreases over time. This allows EMAs to react more quickly to recent changes in the data.
- **Rolling Sums**: Calculated over a specified window, providing the cumulative sum of values within that window. For example, a 30-day rolling sum displays the total over the past 30 days.

1.Import Library

In [ ]:

```
import pandas as pd
import numpy as np
```

In [ ]:

```
df = pd.read_csv("multiTimeline (1).csv")
df
```

Out[ ]:

	Week	Asha Bhosle	Mohammed Rafi	Kishore Kumar	Kumar Sanu	Alka Yagnik
0	2/10/19	16	56	63	59	20
1	2/17/19	15	51	60	58	21
2	2/24/19	14	45	55	54	21
3	3/3/19	14	52	62	60	23
4	3/10/19	16	51	69	54	21
...	...	...	...	...	...	...
255	12/31/23	15	37	47	29	18
256	1/7/24	14	36	41	29	17
257	1/14/24	12	34	42	27	16
258	1/21/24	11	31	39	25	15
259	1/28/24	12	31	42	28	19

260 rows × 6 columns

2. Set index to "Week"

In [ ]:

```
df["Week"] = pd.to_datetime(df["Week"], format="%m/%d/%y")
df.set_index("Week", inplace=True)

print("Original Dataset:")
df
```

```
Original Dataset:
```

Out[ ]:

	Asha Bhosle	Mohammed Rafi	Kishore Kumar	Kumar Sanu	Alka Yagnik
Week					
2019-02-10	16	56	63	59	20
2019-02-17	15	51	60	58	21
2019-02-24	14	45	55	54	21
2019-03-03	14	52	62	60	23
2019-03-10	16	51	69	54	21
...	...	...	...	...	...
2023-12-31	15	37	47	29	18
2024-01-07	14	36	41	29	17
2024-01-14	12	34	42	27	16
2024-01-21	11	31	39	25	15
2024-01-28	12	31	42	28	19

260 rows × 5 columns

Resample to daily frequency (up-sampling with forward fill)

In [ ]:

```
daily_data = df.resample('D').ffill()

print("\nUp-sampled Daily Dataset with Forward Fill:")
daily_data
```

```
Up-sampled Daily Dataset with Forward Fill:
```

Out[ ]:

	Asha Bhosle	Mohammed Rafi	Kishore Kumar	Kumar Sanu	Alka Yagnik
Week					
2019-02-10	16	56	63	59	20
2019-02-11	16	56	63	59	20
2019-02-12	16	56	63	59	20
2019-02-13	16	56	63	59	20
2019-02-14	16	56	63	59	20
...	...	...	...	...	...
2024-01-24	11	31	39	25	15
2024-01-25	11	31	39	25	15
2024-01-26	11	31	39	25	15
2024-01-27	11	31	39	25	15
2024-01-28	12	31	42	28	19

1814 rows × 5 columns

Resample to monthly frequency (down-sampling with OHLC)

In [ ]:

```
monthly_data = df.resample('ME').ohlc()

# Display the down-sampled monthly dataset
print("\nDown-sampled Monthly Dataset with OHLC:")
monthly_data.head()
```

```
Down-sampled Monthly Dataset with OHLC:
```

Out[ ]:

	Asha Bhosle				Mohammed Rafi				Kishore Kumar				Kumar Sanu				Alka Yagnik			
	open	high	low	close	open	high	low	close	open	high	low	close	open	high	low	close	open	high	low	close
Week																				
2019-02-28	16	16	14	14	56	56	45	45	63	63	55	55	59	59	54	54	20	21	20	21
2019-03-31	14	16	13	15	52	52	47	48	62	69	59	63	60	60	48	53	23	23	18	21
2019-04-30	13	15	13	14	48	53	47	53	65	81	62	70	48	51	46	46	21	23	21	23
2019-05-31	14	14	10	14	44	55	44	55	75	83	72	76	51	78	51	78	23	27	23	27
2019-06-30	17	17	15	15	58	59	55	55	77	94	76	76	59	66	52	52	27	30	22	29

Shift the time series forward and backward using naive shifts

In [ ]:

```
shifted_forward = df.shift(2)

# Display the shifted datasets
print("\nShifted Forward Dataset:")
shifted_forward
```

```
Shifted Forward Dataset:
```

Out[ ]:

	Asha Bhosle	Mohammed Rafi	Kishore Kumar	Kumar Sanu	Alka Yagnik
Week					
2019-02-10	NaN	NaN	NaN	NaN	NaN
2019-02-17	NaN	NaN	NaN	NaN	NaN
2019-02-24	16.0	56.0	63.0	59.0	20.0
2019-03-03	15.0	51.0	60.0	58.0	21.0
2019-03-10	14.0	45.0	55.0	54.0	21.0
...	...	...	...	...	...
2023-12-31	15.0	37.0	46.0	32.0	18.0
2024-01-07	14.0	40.0	49.0	31.0	16.0
2024-01-14	15.0	37.0	47.0	29.0	18.0
2024-01-21	14.0	36.0	41.0	29.0	17.0
2024-01-28	12.0	34.0	42.0	27.0	16.0

260 rows × 5 columns

In [ ]:

```
shifted_backward = df.shift(-2)
print("\nShifted Backward Dataset:")
shifted_backward
```

```
Shifted Backward Dataset:
```

Out[ ]:

	Asha Bhosle	Mohammed Rafi	Kishore Kumar	Kumar Sanu	Alka Yagnik
Week					
2019-02-10	14.0	45.0	55.0	54.0	21.0
2019-02-17	14.0	52.0	62.0	60.0	23.0
2019-02-24	16.0	51.0	69.0	54.0	21.0
2019-03-03	16.0	52.0	64.0	53.0	22.0
2019-03-10	13.0	47.0	59.0	48.0	18.0
...	...	...	...	...	...
2023-12-31	12.0	34.0	42.0	27.0	16.0
2024-01-07	11.0	31.0	39.0	25.0	15.0
2024-01-14	12.0	31.0	42.0	28.0	19.0
2024-01-21	NaN	NaN	NaN	NaN	NaN
2024-01-28	NaN	NaN	NaN	NaN	NaN

260 rows × 5 columns

Shift the time series forward and backward using frequency

In [ ]:

```
shifted_forward_freq = df.shift(2, freq='W')

# Display the frequency-shifted datasets
print("\nFrequency Shifted Forward Dataset:")
shifted_forward_freq
```

```
Frequency Shifted Forward Dataset:
```

Out[ ]:

	Asha Bhosle	Mohammed Rafi	Kishore Kumar	Kumar Sanu	Alka Yagnik
Week					
2019-02-24	16	56	63	59	20
2019-03-03	15	51	60	58	21
2019-03-10	14	45	55	54	21
2019-03-17	14	52	62	60	23
2019-03-24	16	51	69	54	21
...	...	...	...	...	...
2024-01-14	15	37	47	29	18
2024-01-21	14	36	41	29	17
2024-01-28	12	34	42	27	16
2024-02-04	11	31	39	25	15
2024-02-11	12	31	42	28	19

260 rows × 5 columns

In [ ]:

```
shifted_backward_freq = df.shift(-2, freq='W')
print("\nFrequency Shifted Backward Dataset:")
shifted_backward_freq
```

```
Frequency Shifted Backward Dataset:
```

Out[ ]:

Week	Asha Bhosle	Mohammed Rafi	Kishore Kumar	Kumar Sanu	Alka Yagnik
2019-01-27	16	56	63	59	20
2019-02-03	15	51	60	58	21
2019-02-10	14	45	55	54	21
2019-02-17	14	52	62	60	23
2019-02-24	16	51	69	54	21
...	...	...	...	...	...
2023-12-17	15	37	47	29	18
2023-12-24	14	36	41	29	17
2023-12-31	12	34	42	27	16
2024-01-07	11	31	39	25	15
2024-01-14	12	31	42	28	19

260 rows × 5 columns

Compute moving averages using three different window functions

In [ ]:

```
rolling_mean_3 = df.rolling(window=3).mean()

# Display the computed moving averages
print("\nMoving Average (Window=3):")
rolling_mean_3
```

```
Moving Average (Window=3):
```

Out[ ]:

Week	Asha Bhosle	Mohammed Rafi	Kishore Kumar	Kumar Sanu	Alka Yagnik
2019-02-10	NaN	NaN	NaN	NaN	NaN
2019-02-17	NaN	NaN	NaN	NaN	NaN
2019-02-24	15.000000	50.666667	59.333333	57.000000	20.666667
2019-03-03	14.333333	49.333333	59.000000	57.333333	21.666667
2019-03-10	14.666667	49.333333	62.000000	56.000000	21.666667
...	...	...	...	...	...
2023-12-31	14.666667	38.000000	47.333333	30.666667	17.333333
2024-01-07	14.333333	37.666667	45.666667	29.666667	17.000000
2024-01-14	13.666667	35.666667	43.333333	28.333333	17.000000
2024-01-21	12.333333	33.666667	40.666667	27.000000	16.000000
2024-01-28	11.666667	32.000000	41.000000	26.666667	16.666667

260 rows × 5 columns

In [ ]:

```
rolling_sum_4 = df.rolling(window=4).sum()
print("\nRolling Sum (Window=4):")
rolling_sum_4
```

```
Rolling Sum (Window=4):
```

Out[ ]:

	Asha Bhosle	Mohammed Rafi	Kishore Kumar	Kumar Sanu	Alka Yagnik
Week					
2019-02-10	NaN	NaN	NaN	NaN	NaN
2019-02-17	NaN	NaN	NaN	NaN	NaN
2019-02-24	NaN	NaN	NaN	NaN	NaN
2019-03-03	59.0	204.0	240.0	231.0	85.0
2019-03-10	59.0	199.0	246.0	226.0	86.0
...	...	...	...	...	...
2023-12-31	60.0	149.0	187.0	124.0	70.0
2024-01-07	58.0	150.0	183.0	121.0	69.0
2024-01-14	55.0	147.0	179.0	116.0	67.0
2024-01-21	52.0	138.0	169.0	110.0	66.0
2024-01-28	49.0	132.0	164.0	109.0	67.0

260 rows × 5 columns

In [ ]:

```
rolling_expanding = df.expanding().mean()
print("\nExpanding Mean:")
rolling_expanding
```

```
Expanding Mean:
```

Out[ ]:

	Asha Bhosle	Mohammed Rafi	Kishore Kumar	Kumar Sanu	Alka Yagnik
Week					
2019-02-10	16.000000	56.000000	63.000000	59.000000	20.000000
2019-02-17	15.500000	53.500000	61.500000	58.500000	20.500000
2019-02-24	15.000000	50.666667	59.333333	57.000000	20.666667
2019-03-03	14.750000	51.000000	60.000000	57.750000	21.250000
2019-03-10	15.000000	51.000000	61.800000	57.000000	21.200000
...	...	...	...	...	...
2023-12-31	16.074219	52.890625	63.160156	41.542969	22.898438
2024-01-07	16.066148	52.824903	63.073930	41.494163	22.875486
2024-01-14	16.050388	52.751938	62.992248	41.437984	22.848837
2024-01-21	16.030888	52.667954	62.899614	41.374517	22.818533
2024-01-28	16.015385	52.584615	62.819231	41.323077	22.803846

260 rows × 5 columns

CONCLUSION	I learned how to change the timeframe of my data, look at values from the past, and find smoother trends with moving averages. These tools help me understand how my data changes over time and make better predictions for the future.