

Domain Name: Data Analytics with Cognos

Project Sales Analysis

Phase 3

Data analysis by loading and preprocessing dataset

Team members

- Sabarish T K
- Sabith Ahamed S
- Keerthi Varman M
- Suresh L

Overview of the data

```
In [4]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_csv("statsfinal.csv")

print(df.shape)
```

(4600, 10)

```
In [5]: print(df.columns)

Index(['Unnamed: 0', 'Date', 'Q-P1', 'Q-P2', 'Q-P3', 'Q-P4', 'S-P1', 'S-P2',
       'S-P3', 'S-P4'],
      dtype='object')
```

```
In [6]: print(df.head(10))
```

```
      Unnamed: 0        Date   Q-P1   Q-P2   Q-P3   Q-P4      S-P1      S-P2 \
0          0  13-06-2010  5422  3725   576   907  17187.74  23616.50
1          1  14-06-2010  7047  779  3578  1574  22338.99  4938.86
2          2  15-06-2010  1572  2082   595  1145  4983.24  13199.88
3          3  16-06-2010  5657  2399  3140  1672  17932.69  15209.66
4          4  17-06-2010  3668  3207  2184   708  11627.56  20332.38
5          5  18-06-2010  2898  2539   311  1513  9186.66  16097.26
6          6  19-06-2010  6912  1470  1576  1608  21911.04  9319.80
7          7  20-06-2010  5209  2550  3415   842  16512.53  16167.00
8          8  21-06-2010  6322   852  3646  1377  20040.74  5401.68
9          9  22-06-2010  6865   414  3902   562  21762.05  2624.76
```

```
      S-P3      S-P4
0  3121.92  6466.91
1  19392.76 11222.62
2  3224.90  8163.85
3  17018.80 11921.36
4  11837.28  5048.04
5  1685.62  10787.69
6  8541.92  11465.04
7  18509.30  6003.46
8  19761.32  9818.01
9  21148.84  4007.06
```

```
In [8]: print(df.dtypes)
```

```
      Unnamed: 0        int64
      Date            object
      Q-P1           int64
      Q-P2           int64
      Q-P3           int64
      Q-P4           int64
      S-P1         float64
      S-P2         float64
      S-P3         float64
      S-P4         float64
      dtype: object
```

```
In [9]: print(df.info)
```

```

<bound method DataFrame.info of
S-P1      S-P2  \
0          0  13-06-2010  5422  3725  576  907  17187.74  23616.50
1          1  14-06-2010  7047  779  3578  1574  22338.99  4938.86
2          2  15-06-2010  1572  2082  595  1145  4983.24  13199.88
3          3  16-06-2010  5657  2399  3140  1672  17932.69  15209.66
4          4  17-06-2010  3668  3207  2184  708  11627.56  20332.38
...
4595      4595  30-01-2023  2476  3419  525  1359  7848.92  21676.46
4596      4596  31-01-2023  7446  841  4825  1311  23603.82  5331.94
4597      4597  01-02-2023  6289  3143  3588  474  19936.13  19926.62
4598      4598  02-02-2023  3122  1188  5899  517  9896.74  7531.92
4599      4599  03-02-2023  1234  3854  2321  406  3911.78  24434.36

           S-P3      S-P4
0    3121.92   6466.91
1   19392.76  11222.62
2    3224.90   8163.85
3   17018.80  11921.36
4   11837.28   5048.04
...
4595   2845.50   9689.67
4596   26151.50  9347.43
4597   19446.96  3379.62
4598   31972.58  3686.21
4599   12579.82  2894.78

```

[4600 rows x 10 columns]>

In [10]: `print(df.isnull().sum())`

```

Unnamed: 0      0
Date            0
Q-P1            0
Q-P2            0
Q-P3            0
Q-P4            0
S-P1            0
S-P2            0
S-P3            0
S-P4            0
dtype: int64

```

No Null Values

In [11]: `print(df.duplicated().sum())`

0

No Duplicate Values

In [12]: `print(df.describe())`

	Unnamed: 0	Q-P1	Q-P2	Q-P3	Q-P4	\
count	4600.000000	4600.000000	4600.000000	4600.000000	4600.000000	
mean	2299.500000	4121.849130	2130.281522	3145.740000	1123.500000	
std	1328.049949	2244.271323	1089.783705	1671.832231	497.385676	
min	0.000000	254.000000	251.000000	250.000000	250.000000	
25%	1149.750000	2150.500000	1167.750000	1695.750000	696.000000	
50%	2299.500000	4137.000000	2134.000000	3202.500000	1136.500000	
75%	3449.250000	6072.000000	3070.250000	4569.000000	1544.000000	
max	4599.000000	7998.000000	3998.000000	6000.000000	2000.000000	

	S-P1	S-P2	S-P3	S-P4
count	4600.000000	4600.000000	4600.000000	4600.000000
mean	13066.261743	13505.984848	17049.910800	8010.555000
std	7114.340094	6909.228687	9061.330694	3546.359869
min	805.180000	1591.340000	1355.000000	1782.500000
25%	6817.085000	7403.535000	9190.965000	4962.480000
50%	13114.290000	13529.560000	17357.550000	8103.245000
75%	19248.240000	19465.385000	24763.980000	11008.720000
max	25353.660000	25347.320000	32520.000000	14260.000000

Data Cleaning

In [13]: #Data Cleaning

```
print(df.sample(2))
```

	Unnamed: 0	Date	Q-P1	Q-P2	Q-P3	Q-P4	S-P1	S-P2	\
2198	2198	29-06-2016	7530	696	3976	1993	23870.10	4412.64	
919	919	22-12-2012	6502	1305	5079	1911	20611.34	8273.70	

	S-P3	S-P4
2198	21549.92	14210.09
919	27528.18	13625.43

In [14]: from datetime import datetime as dt

```
df[df["Date"] == "31-9-2010"]
```

Out[14]:

	Unnamed: 0	Date	Q- P1	Q- P2	Q- P3	Q- P4	S-P1	S-P2	S-P3	S-P4
109	109	31-9- 2010	4986	342	4978	558	15805.62	2168.28	26980.76	3978.54

In [15]: df['Date'] = pd.to_datetime(df['Date'], errors='coerce')

```
C:\Users\sbrsh\AppData\Local\Temp\ipykernel_5656\2263964175.py:1: UserWarning: Parsing dates in %d-%m-%Y format when dayfirst=False (the default) was specified. Pass `dayfirst=True` or specify a format to silence this warning.
df['Date'] = pd.to_datetime(df['Date'], errors='coerce')
```

In [16]: df[df['Date'].isnull()]

Out[16]:

	Unnamed: 0	Date	Q- P1	Q- P2	Q- P3	Q- P4	S-P1	S-P2	S-P3	S-P4
109	109	NaT	4986	342	4978	558	15805.62	2168.28	26980.76	3978.54
170	170	NaT	4632	3930	523	1581	14683.44	24916.20	2834.66	11272.53
473	473	NaT	2242	401	5926	789	7107.14	2542.34	32118.92	5625.57
534	534	NaT	325	3476	4588	1771	1030.25	22037.84	24866.96	12627.23
836	836	NaT	1003	256	1346	1449	3179.51	1623.04	7295.32	10331.37
897	897	NaT	2509	2666	4146	593	7953.53	16902.44	22471.32	4228.09
1200	1200	NaT	597	709	5470	1994	1892.49	4495.06	29647.40	14217.22
1261	1261	NaT	7681	1235	347	1087	24348.77	7829.90	1880.74	7750.31
1564	1564	NaT	5333	833	3494	618	16905.61	5281.22	18937.48	4406.34
1625	1625	NaT	3870	2779	3246	1290	12267.90	17618.86	17593.32	9197.70
1928	1928	NaT	3583	2111	4225	1401	11358.11	13383.74	22899.50	9989.13
1989	1989	NaT	7516	3423	3116	458	23825.72	21701.82	16888.72	3265.54
2291	2291	NaT	7891	741	2280	1068	25014.47	4697.94	12357.60	7614.84
2352	2352	NaT	2457	3144	533	1184	7788.69	19932.96	2888.86	8441.92
2655	2655	NaT	3512	2851	4072	1597	11133.04	18075.34	22070.24	11386.61
2716	2716	NaT	6094	3798	5849	881	19317.98	24079.32	31701.58	6281.53
3019	3019	NaT	1727	2645	5715	1295	5474.59	16769.30	30975.30	9233.35
3080	3080	NaT	7360	2974	2717	1127	23331.20	18855.16	14726.14	8035.51
3383	3383	NaT	3195	2525	5918	1003	10128.15	16008.50	32075.56	7151.39
3444	3444	NaT	2660	2674	2732	934	8432.20	16953.16	14807.44	6659.42
3746	3746	NaT	4713	1227	4065	403	14940.21	7779.18	22032.30	2873.39
3807	3807	NaT	870	3463	798	851	2757.90	21955.42	4325.16	6067.63
4110	4110	NaT	3511	2609	1543	853	11129.87	16541.06	8363.06	6081.89
4171	4171	NaT	506	3333	3897	574	1604.02	21131.22	21121.74	4092.62
4474	4474	NaT	6964	1873	5481	1336	22075.88	11874.82	29707.02	9525.68
4535	4535	NaT	4600	2006	3796	1426	14582.00	12718.04	20574.32	10167.38

In [17]: `# Filling the NaT values with average of time``df['Date'].fillna(df["Date"].mean(), inplace=True)`In [18]: `df['Date'].isnull().sum()`

```
Out[18]: 0
```

```
In [19]: df.dtypes
```

```
Out[19]: Unnamed: 0           int64
Date            datetime64[ns]
Q-P1            int64
Q-P2            int64
Q-P3            int64
Q-P4            int64
S-P1            float64
S-P2            float64
S-P3            float64
S-P4            float64
dtype: object
```

```
In [20]: #fetching month,day of week, weekday
df["month"] = df["Date"].dt.month_name()
df["day"] = df["Date"].dt.day_name()
df["dayoftheweek"] = df["Date"].dt.weekday
df["year"] = df["Date"].dt.year
df.sample()
```

```
Out[20]:      Unnamed: 0      Date    Q-P1    Q-P2    Q-P3    Q-P4      S-P1      S-P2      S-P3      S-P4      mo
3763        3763  2020-10-17  4509   498   1618   1733  14293.53  3157.32  8769.56  12356.29  Octc
```

```
◀ ▶
```

```
In [21]: ## Droping column unnamed as it is not usefull for us
df.drop(columns=["Unnamed: 0"], inplace=True)
df.sample()
```

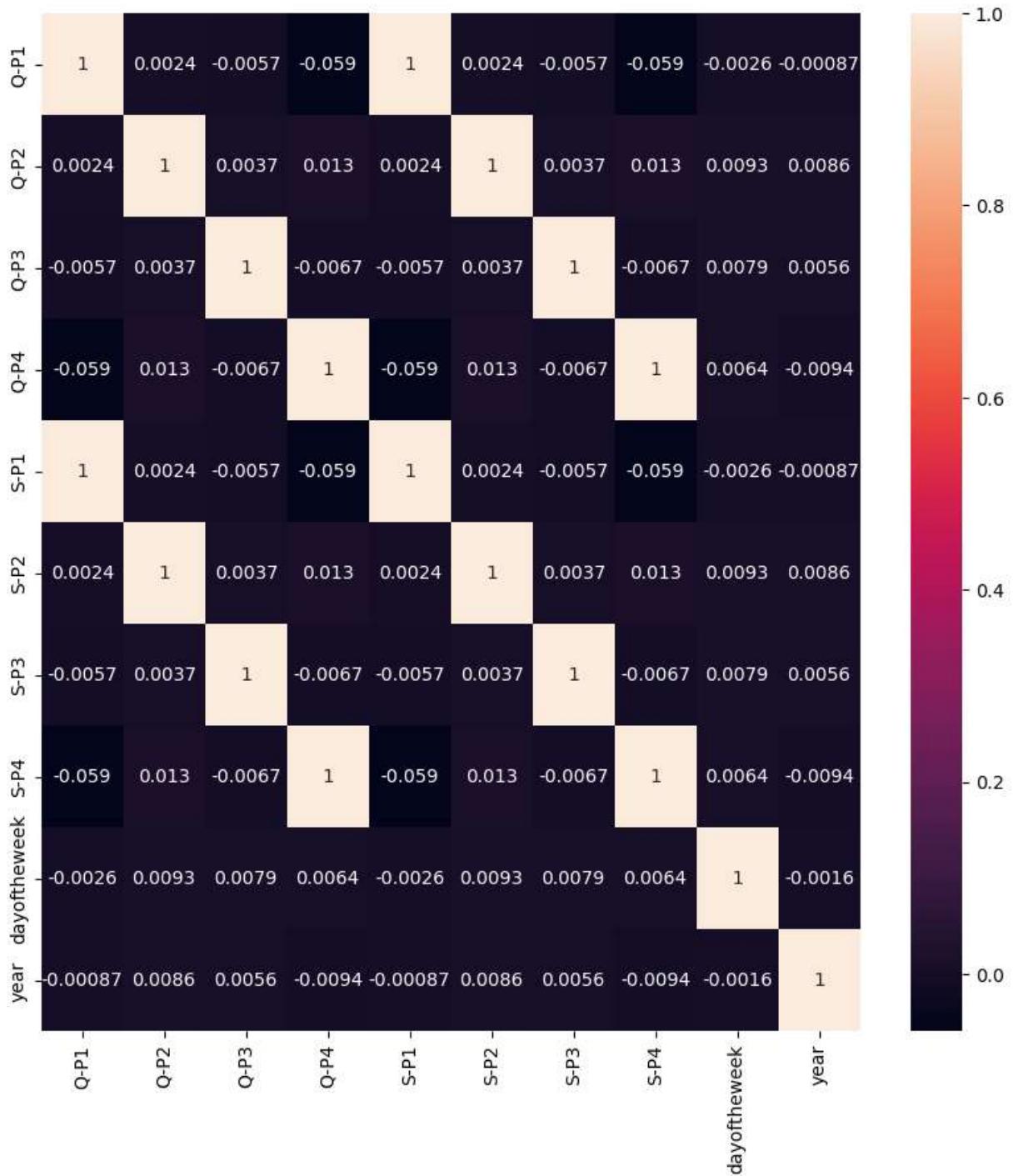
```
Out[21]:      Date    Q-P1    Q-P2    Q-P3    Q-P4      S-P1      S-P2      S-P3      S-P4      month      day      da
2858  2018-04-22  2881  2904   955   427  9132.77  18411.36  5176.1  3044.51      April  Sunday
```

```
◀ ▶
```

```
In [29]: plt.figure(figsize=(10,10))
correlation = df.select_dtypes(include=['number'])

sns.heatmap(correlation.corr(), annot=True)
```

```
Out[29]: <Axes: >
```



```
In [30]: for i in df.columns:
    print(i,"-----",df[i].unique())
```

```

Date ----- <DatetimeArray>
['2010-06-13 00:00:00', '2010-06-14 00:00:00', '2010-06-15 00:00:00',
 '2010-06-16 00:00:00', '2010-06-17 00:00:00', '2010-06-18 00:00:00',
 '2010-06-19 00:00:00', '2010-06-20 00:00:00', '2010-06-21 00:00:00',
 '2010-06-22 00:00:00',
...
'2023-01-25 00:00:00', '2023-01-26 00:00:00', '2023-01-27 00:00:00',
'2023-01-28 00:00:00', '2023-01-29 00:00:00', '2023-01-30 00:00:00',
'2023-01-31 00:00:00', '2023-02-01 00:00:00', '2023-02-02 00:00:00',
'2023-02-03 00:00:00']
Length: 4575, dtype: datetime64[ns]
Q-P1 ----- [5422 7047 1572 ... 1227 3122 1234]
Q-P2 ----- [3725 779 2082 ... 3404 841 3143]
Q-P3 ----- [ 576 3578 595 ... 4825 3588 5899]
Q-P4 ----- [ 907 1574 1145 ... 1161 1151 1112]
S-P1 ----- [17187.74 22338.99 4983.24 ... 3889.59 9896.74 3911.78]
S-P2 ----- [23616.5 4938.86 13199.88 ... 21581.36 5331.94 19926.62]
S-P3 ----- [ 3121.92 19392.76 3224.9 ... 26151.5 19446.96 31972.58]
S-P4 ----- [ 6466.91 11222.62 8163.85 ... 8277.93 8206.63 7928.56]
month ----- ['June' 'July' 'August' 'September' 'October' 'November' 'December'
 'January' 'February' 'March' 'April' 'May']
day ----- ['Sunday' 'Monday' 'Tuesday' 'Wednesday' 'Thursday' 'Friday' 'Saturday']
dayoftheweek ----- [6 0 1 2 3 4 5]
year ----- [2010 2016 2011 2012 2013 2014 2015 2017 2018 2019 2020 2021 2022 202
3]

```

Visualizing discrete numeric values

```

In [31]: products = ['Q-P1', 'Q-P2', 'Q-P3', 'Q-P4']

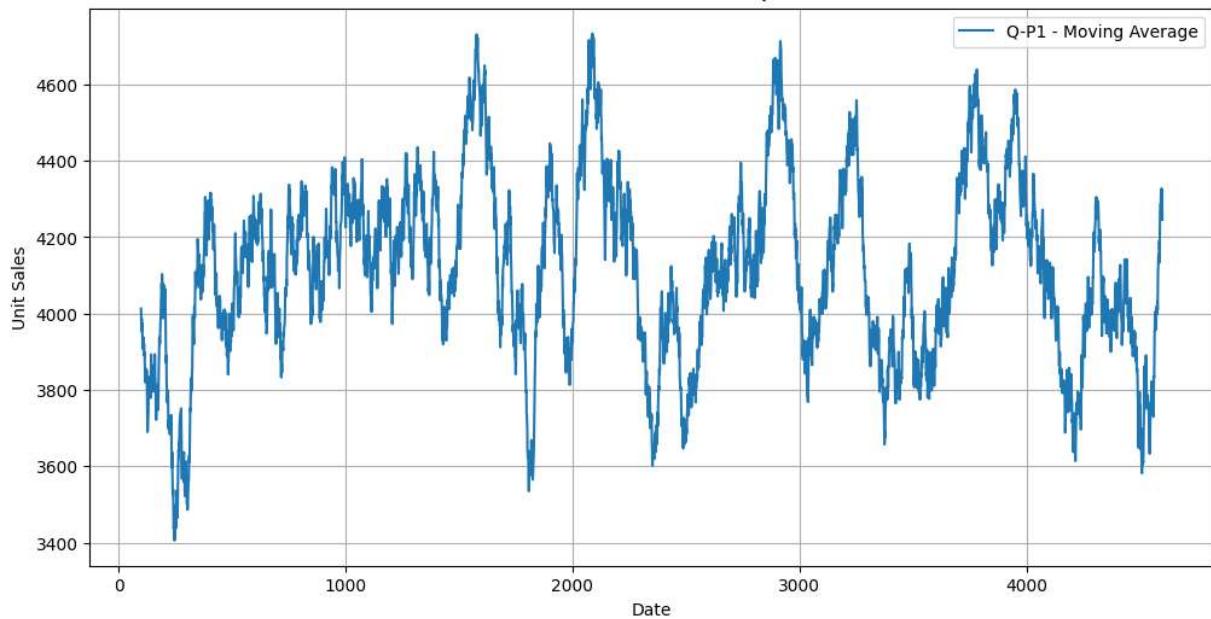
for product in products:
    product_data = df[product]

    moving_average = product_data.rolling(window=100).mean()

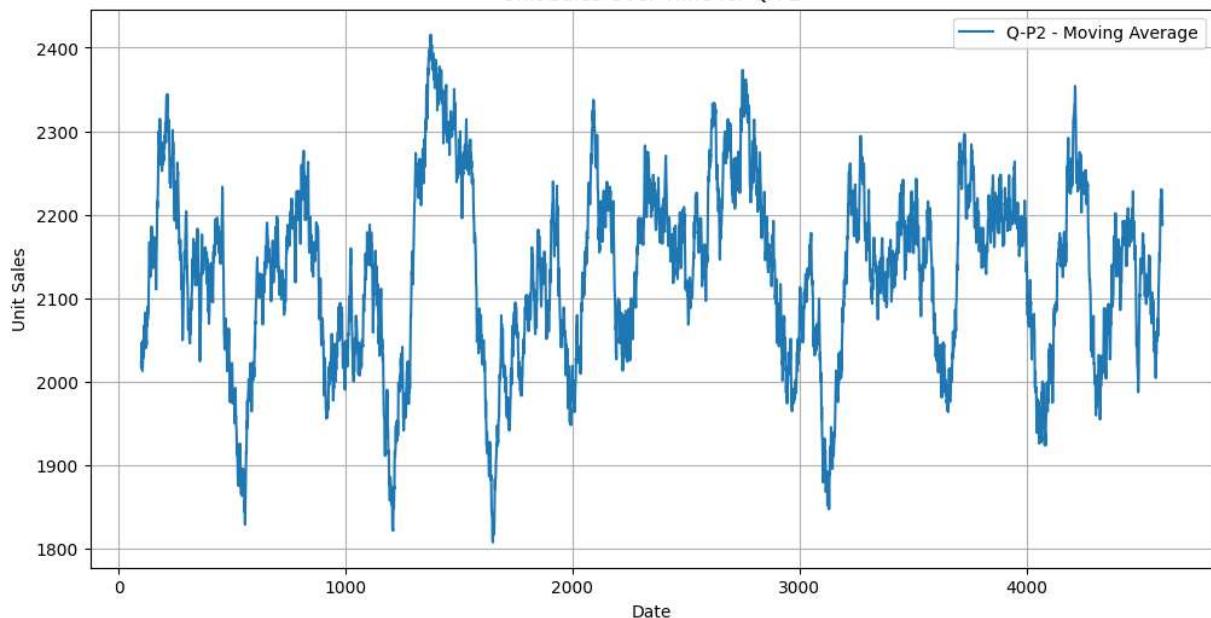
    plt.figure(figsize=(12, 6))
    plt.plot(df.index, moving_average, label=f'{product} - Moving Average')
    plt.xlabel('Date')
    plt.ylabel('Unit Sales')
    plt.title(f'Unit Sales Over Time for {product}')
    plt.grid(True)
    plt.legend()
    plt.show()

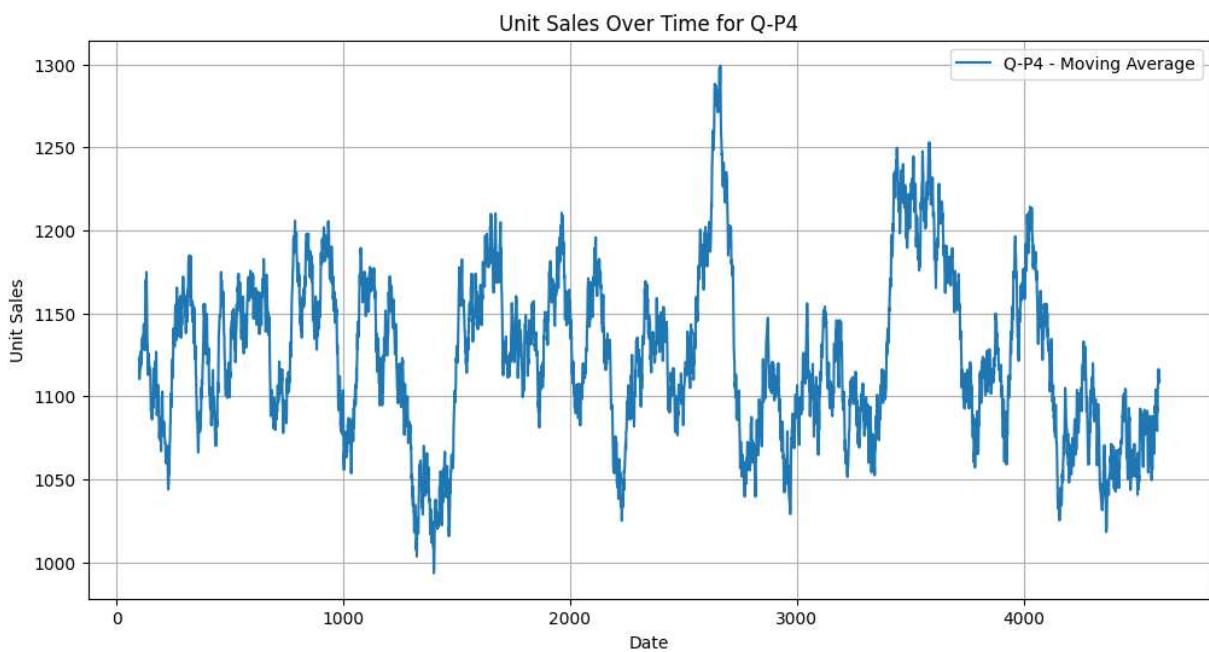
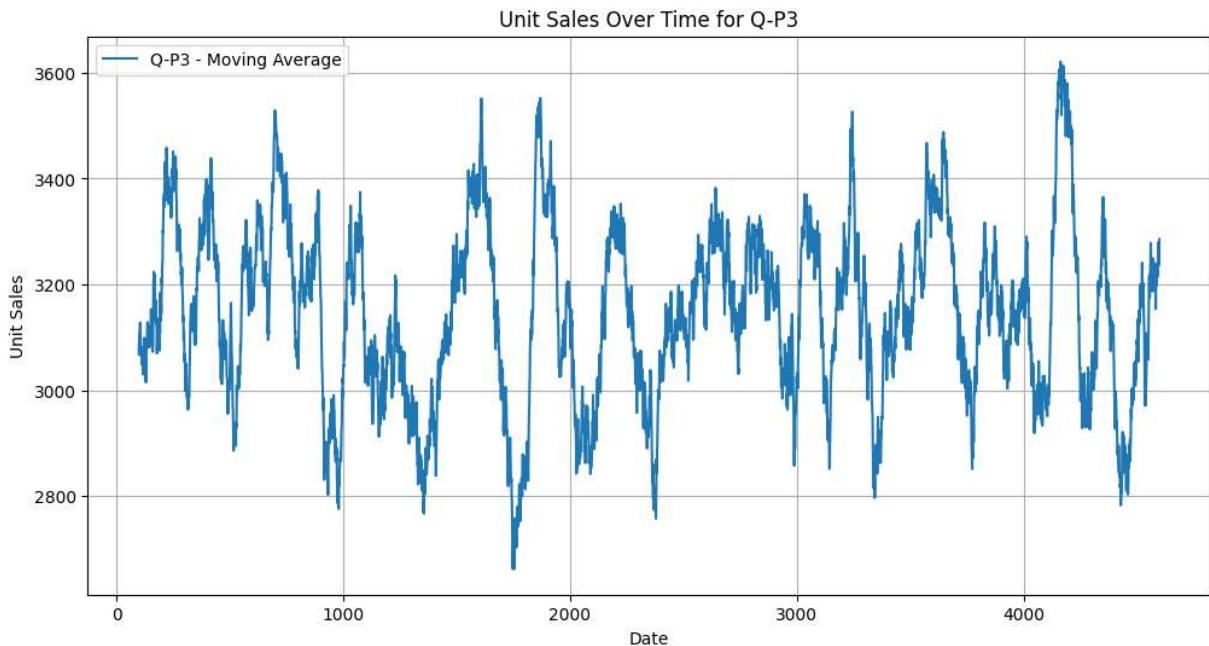
```

Unit Sales Over Time for Q-P1



Unit Sales Over Time for Q-P2





```
In [37]: df1 = df

try:
    df1['Date'] = pd.to_datetime(df1['Date'], format='%d-%m-%Y', errors='coerce')
except pd.errors.ParserError:
    pass

df1 = df1.dropna(subset=['Date'])
df1['Date'] = pd.to_datetime(df1['Date'], format='%d-%m-%Y')
df1.set_index('Date', inplace=True)

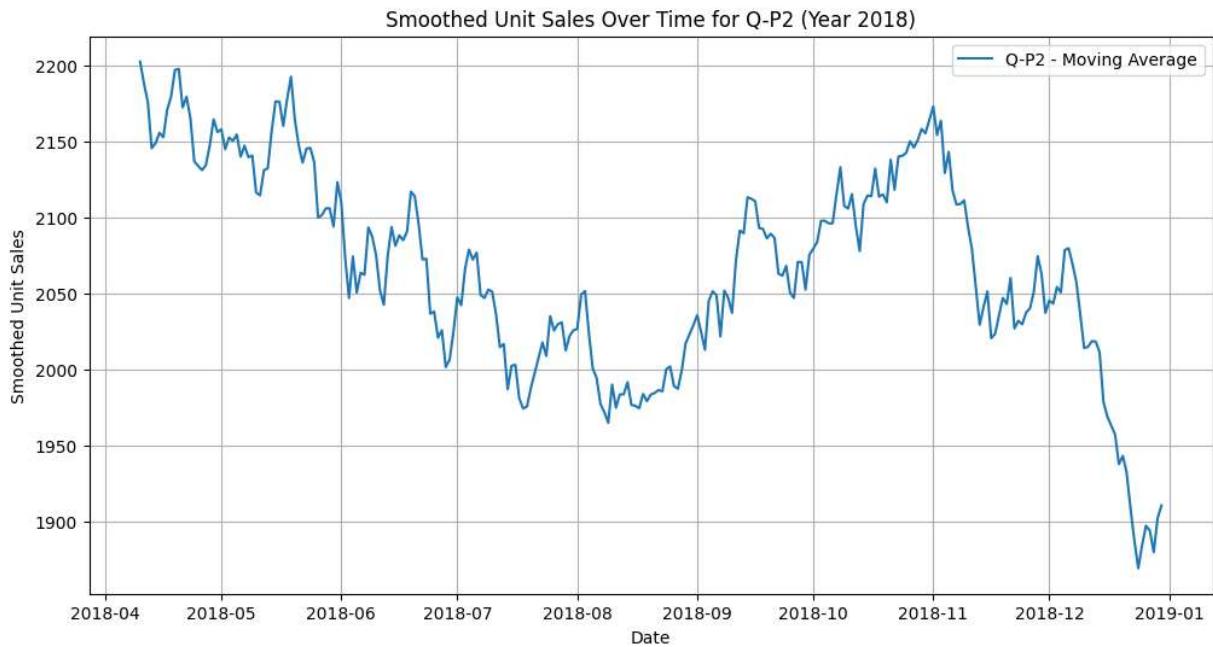
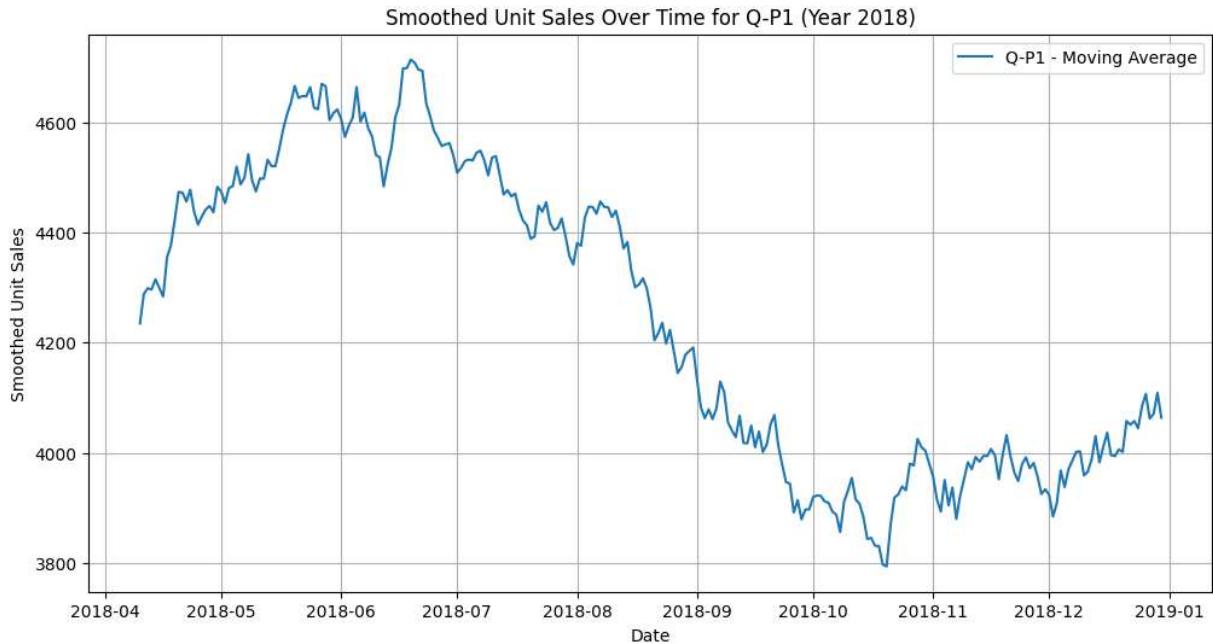
products = ['Q-P1', 'Q-P2', 'Q-P3', 'Q-P4']
for product in products:
    product_data = df1[product]['2018']

    moving_average = product_data.rolling(window=100).mean()
```

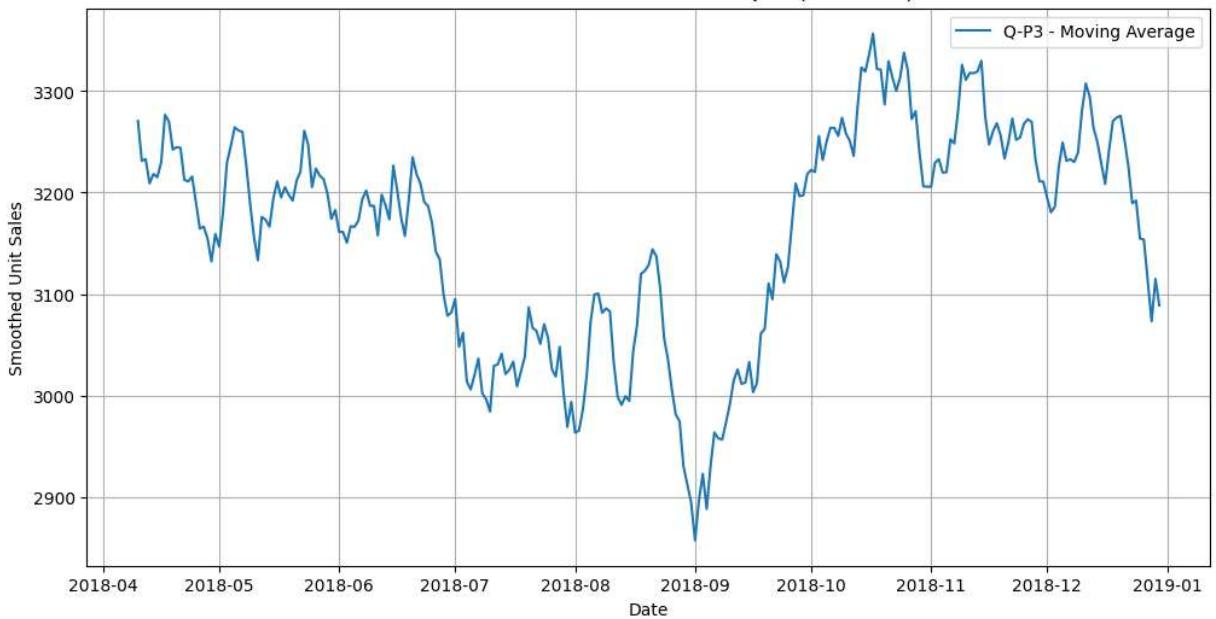
```

# Create a clearer plot of the smoothed data
plt.figure(figsize=(12, 6))
plt.plot(product_data.index, moving_average, label=f'{product} - Moving Average')
plt.xlabel('Date')
plt.ylabel('Smoothed Unit Sales')
plt.title(f'Smoothed Unit Sales Over Time for {product} (Year 2018)')
plt.grid(True)
plt.legend()
plt.show()

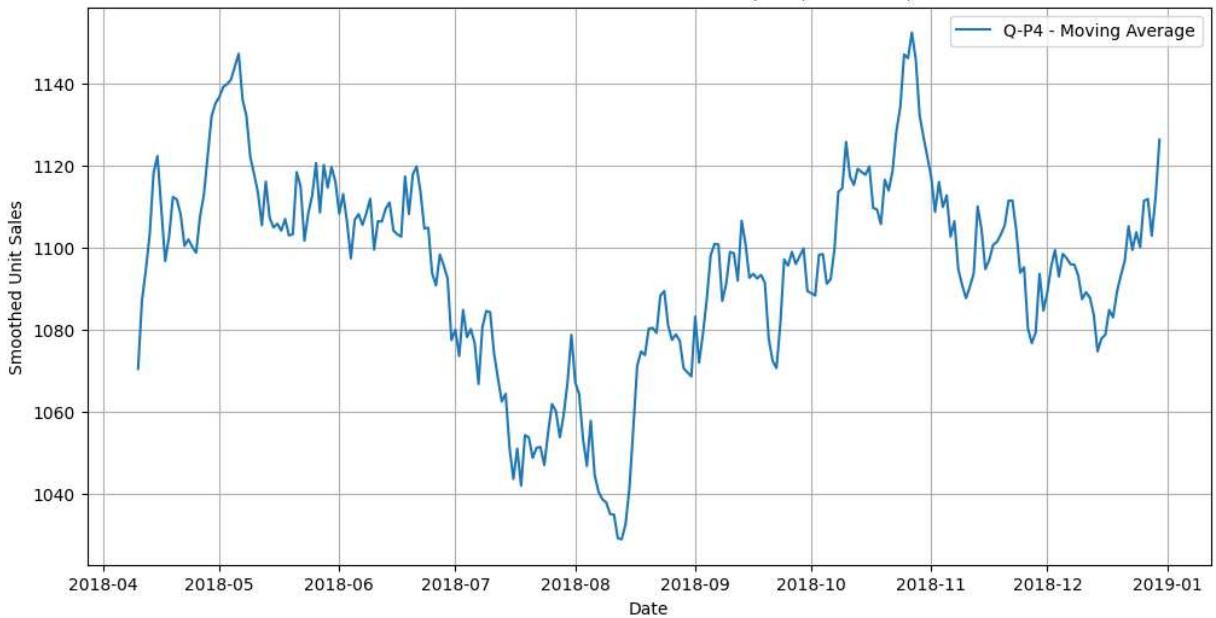
```



Smoothed Unit Sales Over Time for Q-P3 (Year 2018)



Smoothed Unit Sales Over Time for Q-P4 (Year 2018)



In []: