Product Sales Data

Help the company in finding trends and insights

REC corp LTD. is small-scaled business venture established in India. They have been selling FOUR PRODUCTS for OVER TEN YEARS .

The products are:

* P1
* P2
* P3
* P4

They have collected data from their retail centers and organized it into a small csv file , which has been given to you.

**The excel file contains about 8 numerical parameters :**

* Q1- Total unit sales of product 1
* Q2- Total unit sales of product 2
* Q3- Total unit sales of product 3
* Q4- Total unit sales of product 4
* S1- Total revenue from product 1
* S2- Total revenue from product 2
* S3- Total revenue from product 3
* S4- Total revenue from product 4

Step 1: Import libraries

In [1]:

*# import the important packages*

import pandas as pd *# library used for data manipulation and analysis*

import numpy as np *# library used for working with arrays*

import matplotlib.pyplot as plt *# library for plots and visualizations*

import seaborn as sns *# library for visualizations*

%matplotlib inline

*# To ignore warnings*

import warnings

warnings.filterwarnings("ignore")

Step 2: Loading the datasets

In [2]:

*#if you open in Kaggle editor*

data = pd.read\_csv('/kaggle/input/product-sales-data/statsfinal.csv')

*#if you open in juypter notebook*

*# data = pd.read\_csv('statsfinal.csv')*

In [3]:

*# Checking the first 5 and last 5 rows of the dataset*

data.head(-1)

Out[3]:

|  | Unnamed: 0 | Date | Q-P1 | Q-P2 | Q-P3 | Q-P4 | S-P1 | S-P2 | S-P3 | S-P4 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 13-06-2010 | 5422 | 3725 | 576 | 907 | 17187.74 | 23616.50 | 3121.92 | 6466.91 |
| 1 | 1 | 14-06-2010 | 7047 | 779 | 3578 | 1574 | 22338.99 | 4938.86 | 19392.76 | 11222.62 |
| 2 | 2 | 15-06-2010 | 1572 | 2082 | 595 | 1145 | 4983.24 | 13199.88 | 3224.90 | 8163.85 |
| 3 | 3 | 16-06-2010 | 5657 | 2399 | 3140 | 1672 | 17932.69 | 15209.66 | 17018.80 | 11921.36 |
| 4 | 4 | 17-06-2010 | 3668 | 3207 | 2184 | 708 | 11627.56 | 20332.38 | 11837.28 | 5048.04 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 4594 | 4594 | 29-01-2023 | 1227 | 3044 | 5510 | 1896 | 3889.59 | 19298.96 | 29864.20 | 13518.48 |
| 4595 | 4595 | 30-01-2023 | 2476 | 3419 | 525 | 1359 | 7848.92 | 21676.46 | 2845.50 | 9689.67 |
| 4596 | 4596 | 31-01-2023 | 7446 | 841 | 4825 | 1311 | 23603.82 | 5331.94 | 26151.50 | 9347.43 |
| 4597 | 4597 | 01-02-2023 | 6289 | 3143 | 3588 | 474 | 19936.13 | 19926.62 | 19446.96 | 3379.62 |
| 4598 | 4598 | 02-02-2023 | 3122 | 1188 | 5899 | 517 | 9896.74 | 7531.92 | 31972.58 | 3686.21 |

In [4]:

*# drop the first column*

data = data.drop(columns=['Unnamed: 0'])

Step 3: Checking the info of the training data

In [5]:

data.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 4600 entries, 0 to 4599

Data columns (total 9 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Date 4600 non-null object

1 Q-P1 4600 non-null int64

2 Q-P2 4600 non-null int64

3 Q-P3 4600 non-null int64

4 Q-P4 4600 non-null int64

5 S-P1 4600 non-null float64

6 S-P2 4600 non-null float64

7 S-P3 4600 non-null float64

8 S-P4 4600 non-null float64

dtypes: float64(4), int64(4), object(1)

memory usage: 323.6+ KB

Step 4: Check for missing values

In [6]:

data.isnull().sum()

Out[6]:

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

**Observations:**

* we have no missing data

**Note:**

* No missing values in a dataset is not common.
* while working with fresh data, you will have to do a ton of cleaning, this will result in some missing or lost data.
* Look into "feature engineering" and "missing value handling" for ways to resolve this issues.

Step 5: EDA

EDA: Exploratory data analysis [Link](https://www.itl.nist.gov/div898/handbook/eda/section1/eda11.htm)

Lets extract the year, month and Day from the date

In [7]:

*# Extract year from the 'Day' 'Month' 'year' from the 'Date' column using a lambda function*

*# We need to get the year from the data to analyse sales year to year*

data['Day'] = data['Date'].apply(lambda x: x.split('-')[0])

data['Month'] = data['Date'].apply(lambda x: x.split('-')[1])

data['Year'] = data['Date'].apply(lambda x: x.split('-')[2])

data

Out[7]:

|  | Date | Q-P1 | Q-P2 | Q-P3 | Q-P4 | S-P1 | S-P2 | S-P3 | S-P4 | Day | Month | Year |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 13-06-2010 | 5422 | 3725 | 576 | 907 | 17187.74 | 23616.50 | 3121.92 | 6466.91 | 13 | 06 | 2010 |
| 1 | 14-06-2010 | 7047 | 779 | 3578 | 1574 | 22338.99 | 4938.86 | 19392.76 | 11222.62 | 14 | 06 | 2010 |
| 2 | 15-06-2010 | 1572 | 2082 | 595 | 1145 | 4983.24 | 13199.88 | 3224.90 | 8163.85 | 15 | 06 | 2010 |
| 3 | 16-06-2010 | 5657 | 2399 | 3140 | 1672 | 17932.69 | 15209.66 | 17018.80 | 11921.36 | 16 | 06 | 2010 |
| 4 | 17-06-2010 | 3668 | 3207 | 2184 | 708 | 11627.56 | 20332.38 | 11837.28 | 5048.04 | 17 | 06 | 2010 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 4595 | 30-01-2023 | 2476 | 3419 | 525 | 1359 | 7848.92 | 21676.46 | 2845.50 | 9689.67 | 30 | 01 | 2023 |
| 4596 | 31-01-2023 | 7446 | 841 | 4825 | 1311 | 23603.82 | 5331.94 | 26151.50 | 9347.43 | 31 | 01 | 2023 |
| 4597 | 01-02-2023 | 6289 | 3143 | 3588 | 474 | 19936.13 | 19926.62 | 19446.96 | 3379.62 | 01 | 02 | 2023 |
| 4598 | 02-02-2023 | 3122 | 1188 | 5899 | 517 | 9896.74 | 7531.92 | 31972.58 | 3686.21 | 02 | 02 | 2023 |
| 4599 | 03-02-2023 | 1234 | 3854 | 2321 | 406 | 3911.78 | 24434.36 | 12579.82 | 2894.78 | 03 | 02 | 2023 |

In [8]:

data\_reduced = data.query("Year != '2010' and Year != '2023'")

Graph our TOTAL & MEAN unit sold for each product using a histogram.

In [9]:

*#Create a function that allows us to plot a bar chart for the 4 products*

def plot\_bar\_chart(df, columns, stri, str1, val):

*# Aggregate sales for each product by year, by sum or mean*

if val == 'sum':

sales\_by\_year = df.groupby('Year')[columns].sum().reset\_index()

elif val == 'mean':

sales\_by\_year = df.groupby('Year')[columns].mean().reset\_index()

*# Melt the data to make it easier to plot*

sales\_by\_year\_melted = pd.melt(sales\_by\_year, id\_vars='Year', value\_vars=columns, var\_name='Product', value\_name='Sales')

*# Create a bar chart*

plt.figure(figsize=(20,4))

sns.barplot(data=sales\_by\_year\_melted, x='Year', y='Sales', hue='Product') *#,palette="cividis")*

plt.xlabel('Year')

plt.ylabel(stri)

plt.title(f'**{**stri**}** by **{**str1**}**')

plt.xticks(rotation=45)

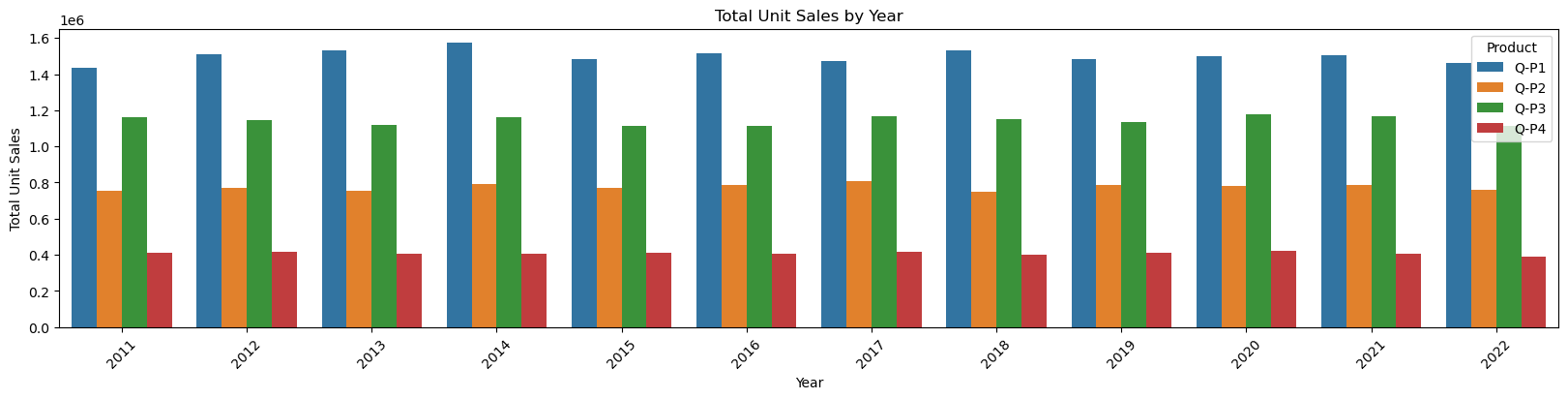
plt.show()

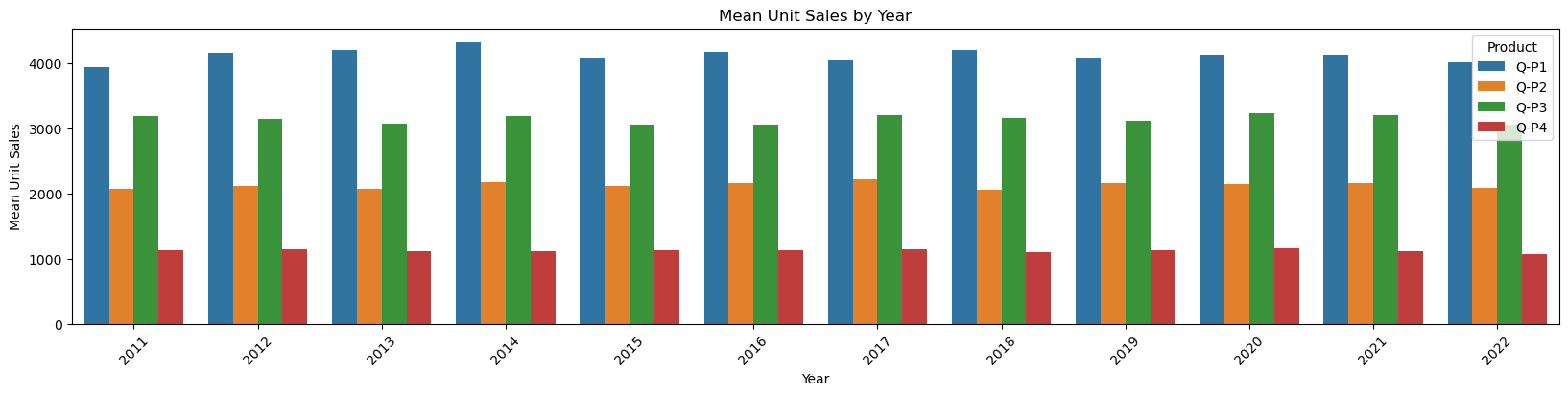
In [10]:

*#use the plot\_bar\_chart function, enter the Unit Sales Columns and the Unit Sales string*

plot\_bar\_chart(data\_reduced, ['Q-P1', 'Q-P2', 'Q-P3', 'Q-P4'],'Total Unit Sales', 'Year', 'sum')

plot\_bar\_chart(data\_reduced, ['Q-P1', 'Q-P2', 'Q-P3', 'Q-P4'],'Mean Unit Sales', 'Year', 'mean')





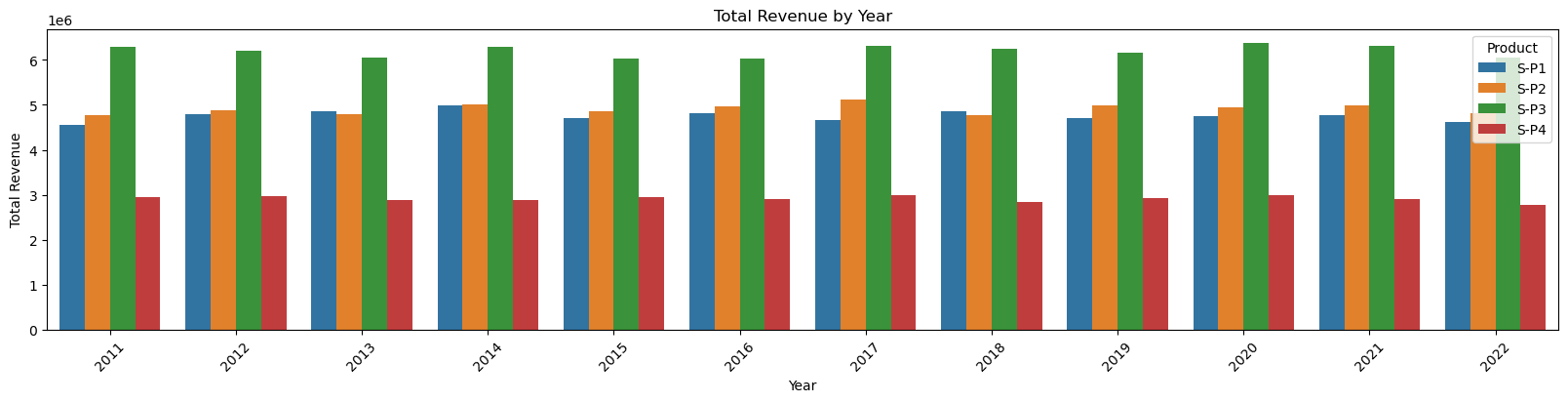
Graph our TOTAL & MEAN revenue of sales for each product using a historgram.

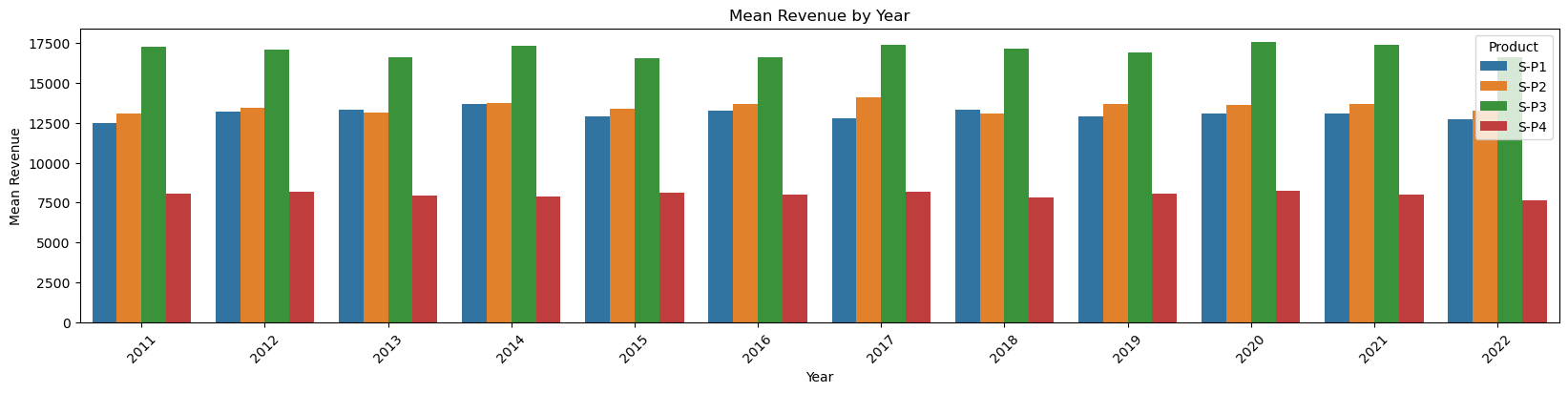
In [11]:

*#use the plot\_bar\_chart function, enter the Revenue Columns and the Revenue string*

plot\_bar\_chart(data\_reduced, ['S-P1', 'S-P2', 'S-P3', 'S-P4'], 'Total Revenue', 'Year', 'sum')

plot\_bar\_chart(data\_reduced, ['S-P1', 'S-P2', 'S-P3', 'S-P4'], 'Mean Revenue', 'Year', 'mean')





**Observation**

* We can observe that P3 brought in the most revenue. This could be as a result of multiple things:
  + P3 was sold for higher than the rest, as it had the second highest unit sales for each year.
* We can observe than P1 AND P2 brought in similar revenues for each year. With P2 bringing in slightly more.
  + P1 despite having the most unit sold, brought in the second lowest revenue each year.

In [12]:

data

Out[12]:

|  | Date | Q-P1 | Q-P2 | Q-P3 | Q-P4 | S-P1 | S-P2 | S-P3 | S-P4 | Day | Month | Year |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 13-06-2010 | 5422 | 3725 | 576 | 907 | 17187.74 | 23616.50 | 3121.92 | 6466.91 | 13 | 06 | 2010 |
| 1 | 14-06-2010 | 7047 | 779 | 3578 | 1574 | 22338.99 | 4938.86 | 19392.76 | 11222.62 | 14 | 06 | 2010 |
| 2 | 15-06-2010 | 1572 | 2082 | 595 | 1145 | 4983.24 | 13199.88 | 3224.90 | 8163.85 | 15 | 06 | 2010 |
| 3 | 16-06-2010 | 5657 | 2399 | 3140 | 1672 | 17932.69 | 15209.66 | 17018.80 | 11921.36 | 16 | 06 | 2010 |
| 4 | 17-06-2010 | 3668 | 3207 | 2184 | 708 | 11627.56 | 20332.38 | 11837.28 | 5048.04 | 17 | 06 | 2010 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 4595 | 30-01-2023 | 2476 | 3419 | 525 | 1359 | 7848.92 | 21676.46 | 2845.50 | 9689.67 | 30 | 01 | 2023 |
| 4596 | 31-01-2023 | 7446 | 841 | 4825 | 1311 | 23603.82 | 5331.94 | 26151.50 | 9347.43 | 31 | 01 | 2023 |
| 4597 | 01-02-2023 | 6289 | 3143 | 3588 | 474 | 19936.13 | 19926.62 | 19446.96 | 3379.62 | 01 | 02 | 2023 |
| 4598 | 02-02-2023 | 3122 | 1188 | 5899 | 517 | 9896.74 | 7531.92 | 31972.58 | 3686.21 | 02 | 02 | 2023 |
| 4599 | 03-02-2023 | 1234 | 3854 | 2321 | 406 | 3911.78 | 24434.36 | 12579.82 | 2894.78 | 03 | 02 | 2023 |

Trend in sales of all four products during certain months

In [13]:

*# Create a figure and axis*

def month\_plot():

fig, ax = plt.subplots()

*# Plot the sales data for each product by month*

data\_reduced.groupby('Month')[['Q-P1', 'Q-P2', 'Q-P3', 'Q-P4']].sum().plot(ax=ax)

*# Set the x-axis limits to only show up to December*

ax.set\_xlim(left=0, right=13)

*# Set the axis labels and title*

ax.set\_xlabel('Month')

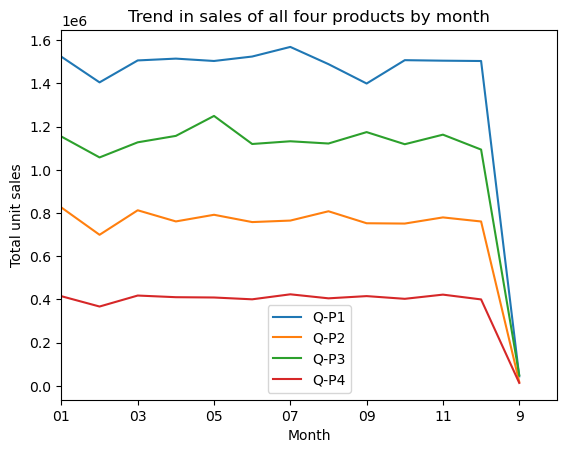
ax.set\_ylabel('Total unit sales')

ax.set\_title('Trend in sales of all four products by month')

*# Show the plot*

plt.show()

month\_plot()



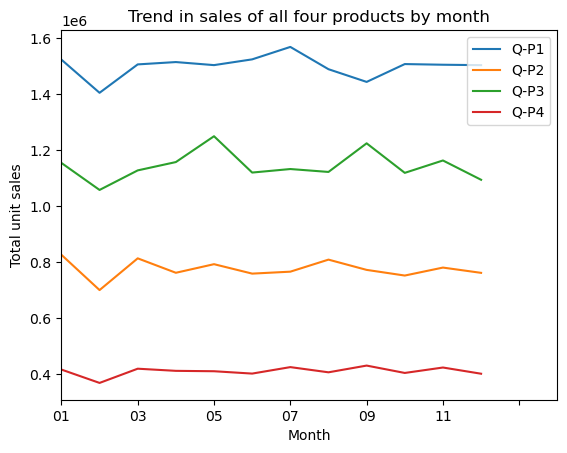
In [14]:

*# Replace all entries of '9' in the Month column with '09'*

data\_reduced['Month'] = data['Month'].replace('9', '09')

In [15]:

month\_plot()



Estimate for each product the unit of sales that could be sold on 31st of Dec, if all their retail centers were kept open.

In [16]:

*#get the 31st day for each month in each year. Note: not every month has 31 days*

def month\_31\_data(df, months):

m31\_data = df[df['Month'].isin(months) & (df['Day'] == '31')]

return m31\_data

\_31\_months = month\_31\_data(data\_reduced, ['01', '02', '03', '04', '05', '06', '07', '08', '09', '10', '11', '12'])

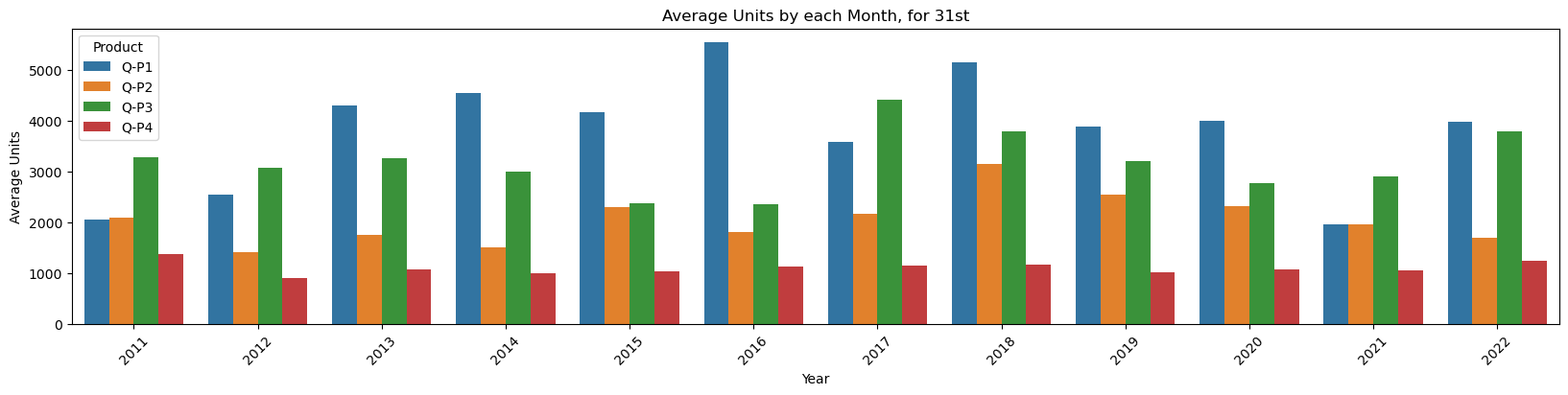
\_31\_months

Out[16]:

|  | Date | Q-P1 | Q-P2 | Q-P3 | Q-P4 | S-P1 | S-P2 | S-P3 | S-P4 | Day | Month | Year |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 231 | 31-01-2011 | 939 | 3325 | 1863 | 1612 | 2976.63 | 21080.50 | 10097.46 | 11493.56 | 31 | 01 | 2011 |
| 290 | 31-03-2011 | 464 | 2220 | 421 | 1663 | 1470.88 | 14074.80 | 2281.82 | 11857.19 | 31 | 03 | 2011 |
| 351 | 31-05-2011 | 1507 | 2980 | 3816 | 1202 | 4777.19 | 18893.20 | 20682.72 | 8570.26 | 31 | 05 | 2011 |
| 412 | 31-07-2011 | 4336 | 744 | 4717 | 667 | 13745.12 | 4716.96 | 25566.14 | 4755.71 | 31 | 07 | 2011 |
| 442 | 31-08-2011 | 4548 | 1484 | 1596 | 1974 | 14417.16 | 9408.56 | 8650.32 | 14074.62 | 31 | 08 | 2011 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 4352 | 31-05-2022 | 3669 | 2710 | 3067 | 1593 | 11630.73 | 17181.40 | 16623.14 | 11358.09 | 31 | 05 | 2022 |
| 4413 | 31-07-2022 | 1437 | 833 | 1867 | 1270 | 4555.29 | 5281.22 | 10119.14 | 9055.10 | 31 | 07 | 2022 |
| 4443 | 31-08-2022 | 1035 | 1639 | 3658 | 841 | 3280.95 | 10391.26 | 19826.36 | 5996.33 | 31 | 08 | 2022 |
| 4474 | 31-9-2022 | 6964 | 1873 | 5481 | 1336 | 22075.88 | 11874.82 | 29707.02 | 9525.68 | 31 | 09 | 2022 |
| 4535 | 31-11-2022 | 4600 | 2006 | 3796 | 1426 | 14582.00 | 12718.04 | 20574.32 | 10167.38 | 31 | 11 | 2022 |

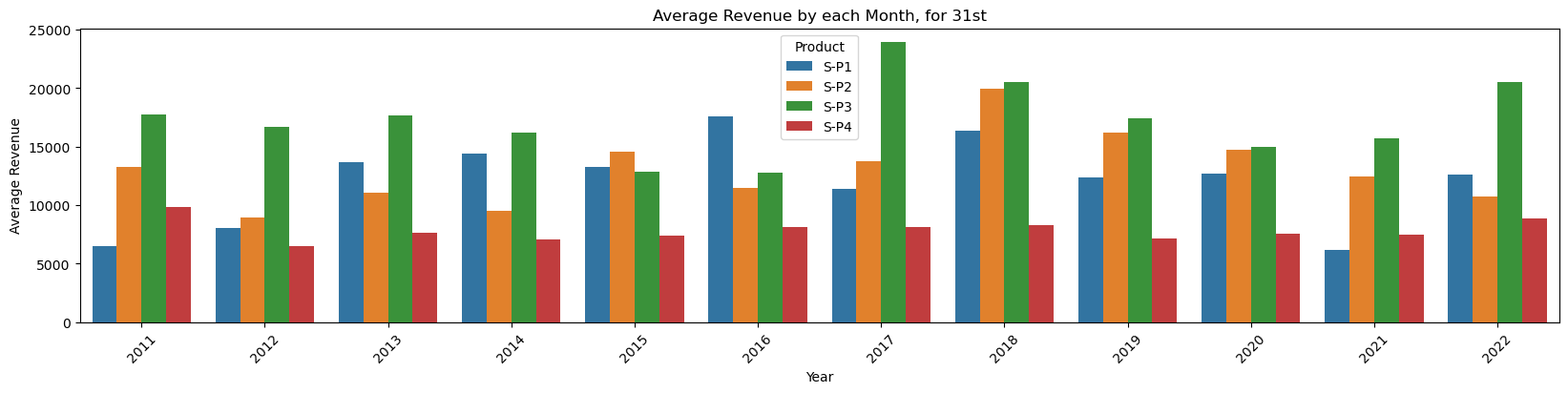
In [17]:

plot\_bar\_chart(\_31\_months, ['Q-P1', 'Q-P2', 'Q-P3', 'Q-P4'], 'Average Units', 'each Month, for 31st', 'mean')



In [18]:

plot\_bar\_chart(\_31\_months, ['S-P1', 'S-P2', 'S-P3', 'S-P4'], 'Average Revenue', 'each Month, for 31st', 'mean')



In [19]:

*# gives us the average for all the 31st days across all years for each product*

def avg\_on\_31st(df, product):

df\_31 = df[df['Day'] == '31']

avg\_sales = df\_31[product].mean()

return avg\_sales

In [20]:

*# Average for Unit Sales*

avg\_on\_31st(data\_reduced, ['Q-P1', 'Q-P2', 'Q-P3', 'Q-P4']).round(2)

Out[20]:

Q-P1 3813.74

Q-P2 2058.80

Q-P3 3183.88

Q-P4 1098.61

dtype: float64

In [21]:

*# Average for Revenue*

avg\_on\_31st(data\_reduced, ['S-P1', 'S-P2', 'S-P3', 'S-P4']).round(2)

Out[21]:

S-P1 12089.55

S-P2 13052.78

S-P3 17256.63

S-P4 7833.07

dtype: float64

Conclusion

Unit Sales 2011 - 2022

* P1 has the highest unit sales for each year. And it's highest is in year 2014.
* We can observe that P4 has the lowest unit sales of all the products.

Revenues 2011 - 2022

* We can observe that P3 brought in the most revenue. This could be as a result of multiple things:
  + P3 was sold for higher than the rest, as it had the second highest unit sales for each year.
* We can observe than P1 and P2 brought in similar revenues for each year. With P2 bringing in slightly more.
* P1 despite having the most unit sold, brought in the second lowest revenue each year.

Average Month Sales 2011 - 2022

* We can observe that all Products unit sales drop in Feb.
* We can observe that Feb and Dec have the lowest sales for each product
* For P1 We can observe Mar - Jul having the highest unit sales
* For P2 We can observe Jan, Mar - Aug having the highest unit sales
* For P3 We can observe May & Sep having the highest unit sales
* For P4 We can observe uniform sales from Jan - Dec

Estimated Unit Sales for 31st of Dec

This value can not be properly estimated with out Machine Learning. Currently we used the average for all the 31st days across all years for each product.

* Overall we can see that P1 has the highest unit sales on the 31st for each year, except for 2021 and 2022. (These could be as a result to Covid and other economy issues.)
* P3 has the second highest unit sales for all the 31st in each year.
* We can see that our previous observation correlate as Q-P1 has the higest estimate, followed by Q-P3
* We can approxiamte that the company will make:
  + Q-P1: 3813.74
  + Q-P2: 2058.80
  + Q-P3: 3183.88
  + Q-P4: 1098.61