**PRODUCT SALE ANALYSIS**

**PHASE 5:PROJECT DOCUMENTATION AND SUBMISSION**

**Done BY,**

**Muthuraj.M (TL),**

**Shakthi Priyan.T,**

**Jaiprakash.C,**

**InbarasuRavi.R.M,**

**Sujith.**

**Abstract:**

This comprehensive project centers around the thorough analysis of a substantial dataset comprising a decade of product sales data for REC corp LTD, a small-scale business established in India. The dataset encapsulates critical sales parameters for four distinct products, namely P1, P2, P3, and P4. The multifaceted analysis undertaken in this project encompasses data loading, meticulous data cleaning to handle any missing values, data transformation for refining data types and generating calculated columns, data filtering for precise data selection based on defined criteria, and data exploration employing a combination of Pandas for basic statistics and Matplotlib or Seaborn for visualization. Additionally, the project dives into the estimation of potential year-end sales, thereby offering a valuable projection for decision-making and future business planning for REC corp

This analysis unveils trends and patterns in product sales, enabling a comprehensive understanding of the company's sales performance over the years. It assists in pinpointing key insights, such as the products with the highest unit sales, revenue generation, and the impact of different months on sales. The detailed examination of year-end sales data, estimating potential unit sales for each product on the 31st of December, provides REC corp LTD with a strategic perspective for optimizing their retail center operations.

LTD.

The project's significance lies in its potential to empower REC corp LTD with actionable insights, aiding the company in making informed decisions, optimizing operations, and ultimately enhancing its business performance in the dynamic market landscape.

**Introduction:**

In product sales analysis project, it's crucial to effectively load and preprocess the dataset before diving into analysis. This process involves acquiring the data, cleaning and transforming it, and creating the foundation for meaningful insights. Below, we'll provide a step-by-step guide on how to achieve these tasks using Python, a versatile programming language often used for data manipulation and analysis.

Certainly, let's delve into creating visualizations and generating actionable insights using IBM Cognos. I'll provide a step-by-step guide to help you work through your project:

**1.Data Preparation:**

Ensure your data is properly cleaned and formatted. Load the dataset into IBM Cognos and verify that it's structured correctly for analysis.

**Create a Dashboard:**

Start by creating an interactive dashboard in IBM Cognos that serves as a central hub for your insights.

Include widgets and panels for different types of visualizations.

Identify Top-Selling Products:

Create a bar chart or table that lists the top-selling products based on total revenue.

Add interactive features like filters and drop-down menus to allow users to explore the data.

**2.Analyze Sales Trends:**

Build line charts showing monthly or yearly sales trends.

Highlight peak sales periods and seasonal variations.

**3.Visualize Customer Preferences**:

Use pie charts or bar charts to show customer preferences for specific product categories.

Explore which products are favored by different customer segments, if available in the dataset.

**Geospatial Analysis (if applicable):**

If you have location data, create geographic maps or heatmaps to visualize sales distribution across regions or stores.

Identify areas with high and low sales.

**4.Customer Segmentation (if applicable):**If you have customer data, create visualizations that segment customers based on demographics or purchase behavior.

Analyze the preferences of different customer segments.

**5.Interactive Filters:**

Include interactive filters that allow users to select specific time periods, product categories, or customer segments.

This empowers users to explore the data and discover insights tailored to their needs.

**6.Combining Visualizations:**

Use the dashboard layout to combine multiple visualizations on a single screen, making it easier for users to grasp insights.

**7.Forecasting:**

If historical data is available, consider creating predictive models for future sales trends.

Visualize actual sales alongside forecasted sales to identify discrepancies.

**8.Drill-Down Capabilities:**

Enable drill-down functionality so that users can click on specific data points and explore detailed information.

**9.Annotations and Explanations:**

Add text boxes or annotations to explain the significance of visualizations. Interpret what the data is showing.

**10.Collaboration and Sharing:**

Use IBM Cognos to share the dashboard and reports with relevant stakeholders within the organization.

**11.Actionable Insights:**

Derive actionable insights from your visualizations. For example, suggest increasing inventory for top-selling products, launching marketing campaigns during peak sales periods, or personalizing recommendations based on customer preferences.

**12.Scheduled Updates:**

If the data is regularly updated, set up scheduled updates for your dashboard so that it always reflects the latest data.

**13.Feedback and Iteration:**

Gather feedback from users and stakeholders, and be ready to make improvements or updates to your dashboard based on their input.

**Loading Data from Kaggle in Python:**

To load a dataset from Kaggle using Python, you can use the Kaggle API or manually download the dataset and load it. Here's how to load a CSV dataset manually:

import pandas as pd

# Assuming you've downloaded the CSV file from Kaggle

dataset\_path = "path\_to\_your\_dataset.csv"

# Load the dataset into a Pandas DataFrame

df = pd.read\_csv(dataset\_path)

Data Cleaning in Python:

Data cleaning involves handling missing values and removing duplicates. Here's how to do it in Python:

# Handling missing values (replace NaN with 0, for example)

df.fillna(0, inplace=True)

# Removing duplicates

df.drop\_duplicates(inplace=True)

Data Transformation in Python:

**Data transformation includes converting data types and creating calculated columns. For instance, if you have a date column, you can convert it to a datetime object:**

df['date\_column'] = pd.to\_datetime(df['date\_column'])

You can also create a new calculated column:

df['profit'] = df['revenue'] - df['cost']

**Data Filtering in Python:**

To filter data based on certain conditions, you can use boolean indexing. For example, to filter rows where sales are above a certain threshold:

threshold = 1000

filtered\_df = df[df['sales'] > threshold]

**Data Exploration in Python:**

Data exploration typically involves using Pandas for basic statistics and Matplotlib or Seaborn for visualization. For basic statistics:

# Summary statistics

print(df.describe())

# Visualization (example using Matplotlib)

import matplotlib.pyplot as plt

plt.scatter(df['sales'], df['profit'])

plt.xlabel('Sales')

plt.ylabel('Profit')

plt.show()

**Saving Data Source Configuration in Cognos:**

Saving data source configurations in Cognos is typically done within the Cognos platform. You can save the configuration after importing the data through the Cognos interface.

**Building Reports and Dashboards in Cognos:**

In Cognos, use the interface to create reports and dashboards. The code for this step is typically created in Cognos itself through the drag-and-drop interface.

**Analysis and Insights in Python:**

Analyzing the data and deriving insights can involve more complex Python code for specific analyses. For instance, to calculate correlations between variables:

python code

correlation\_matrix = df.corr()

Scheduling and Automation in Cognos (Optional):

Scheduling and automation are typically performed through Cognos' scheduling and automation features. No Python code is required for this step.

**Documentation and Sharing in Cognos:**

Cognos provides tools for documenting and sharing reports. The code for this step is typically generated within Cognos.

Please note that Python is primarily used for data preparation, analysis, and some visualization. While Cognos can interface with Python, many of the activities like report generation, automation, and sharing are typically managed within the Cognos platform itself.

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 |

**Observations:**

There is a column called 'Unnamed: 0' which we can drop as it is a repeat of our ID.

* The data contains date.
  + And for each date the total unit of sales for P1, P2, P3 & P4.
  + Also the total revenue from sales for P1, P2, P3 & P4.
* We can observe the first entry in the data, starts at 13-06-2010. This means the data for year 2010 is not complete.
* We can observe the last entry in the data, ends at 02-02-2023. This means the data for year 2023 is also not complete.it will be best to drop year 2010 and year 2023.

**Observations:**

* We can observe the last entry in the data, starts 13-06-2010. This means the data for year 2010 is not complete.

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

**Observations:**

* The train dataset has 4600 entries(rows) and 9 columns. (we dropped one column)
* Date is an object data type. the rest of numerical in nature.

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**

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 |

4600 rows × 12 columns

Lets drop rows for years 2010 and year 2023

data\_reduced = data.query("Year != '2010' and Year != '2023'") In [8]:

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

#Create a function that allows us to plot a bar chart for the 4 products In [9]:

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)

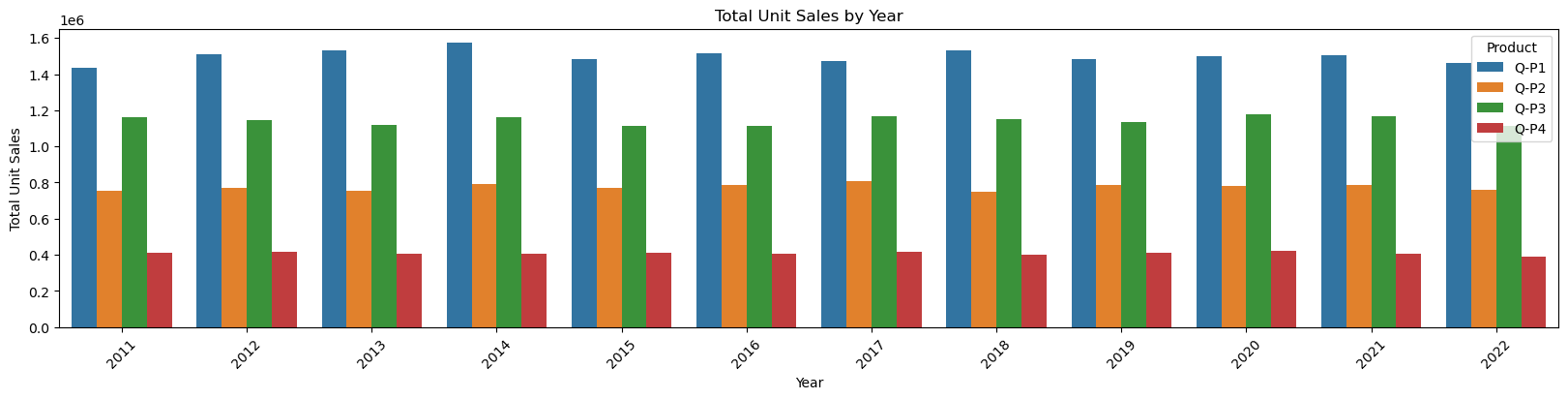
plt.show()

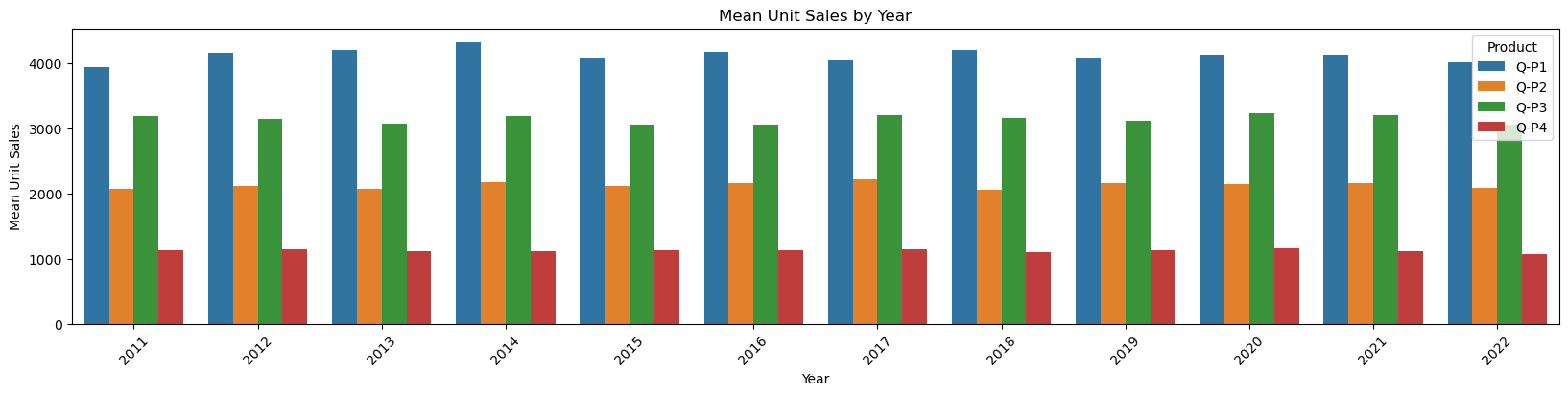
In [10]:

#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')





**Observation:**

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

**Note**

* Check out this link for more [palettes](https://r02b.github.io/seaborn_palettes/)
* The mean and the sum offer very similar data visualization and can both be used to understand the data.

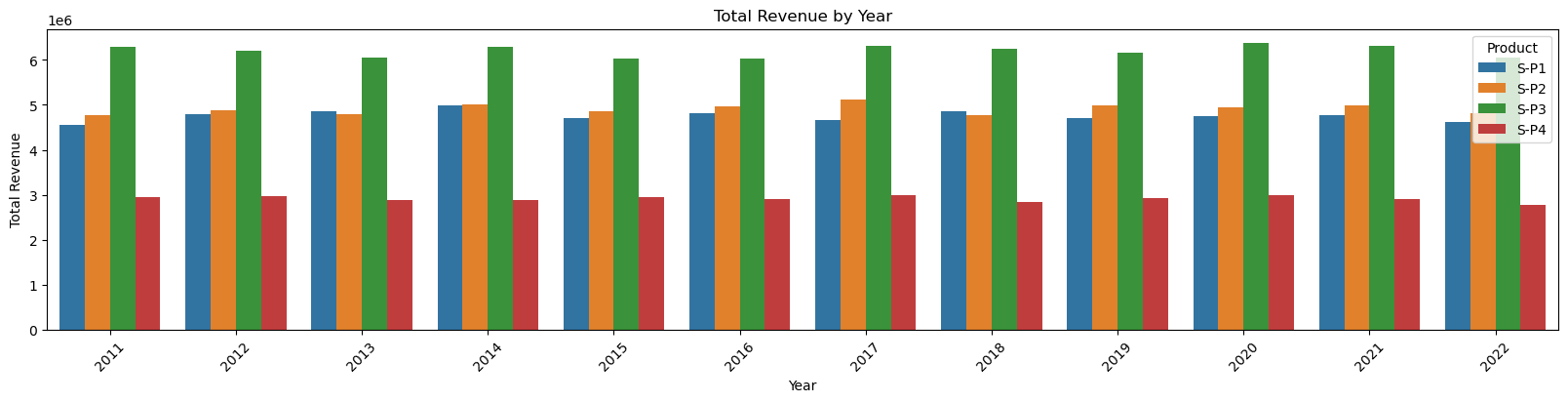
**Graph our TOTAL & MEAN revenue of sales for each product using a historgram.**

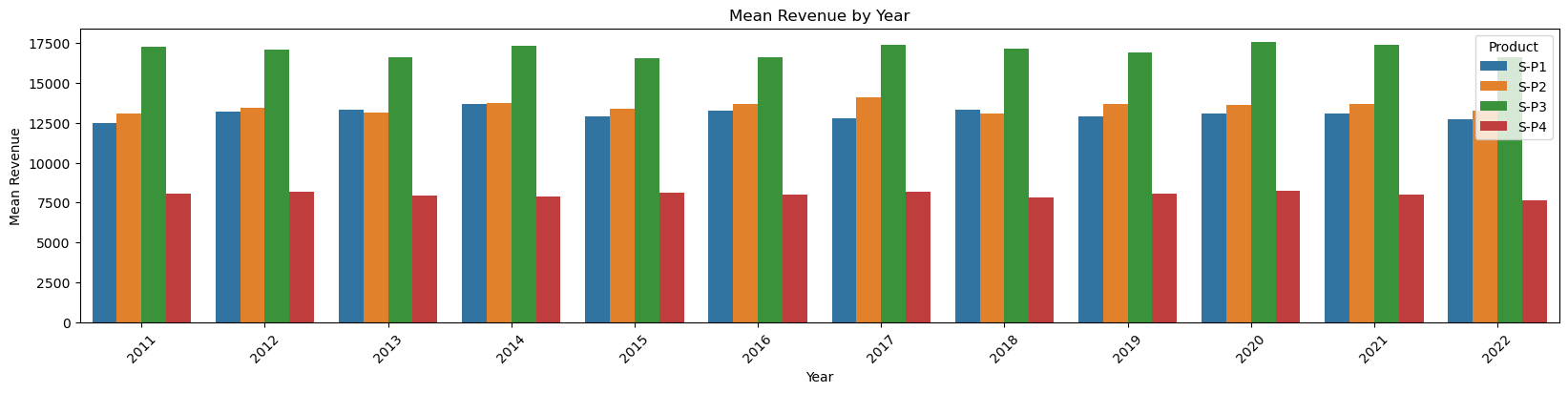
In [11]:

*#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 |

4600 rows × 12 columns

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')

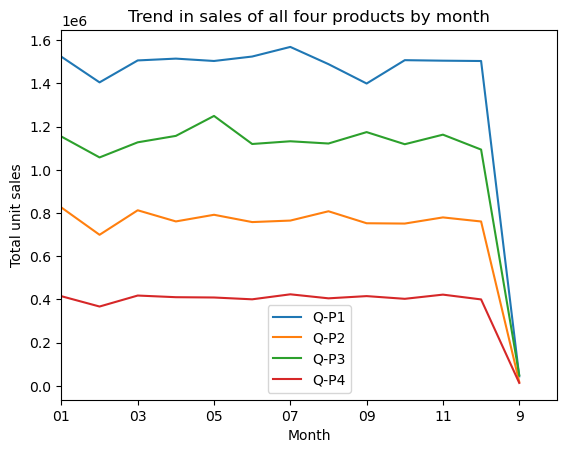
ax.set\_ylabel('Total unit sales')

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

# Show the plot

plt.show()

month\_plot()



**Observation:**

* We can observe that all products drop in Feb.
* There also appears a very drastic drop after 12th month. The value show 9, which must be part of month 09. We need to rename this column to match with the 09. Before doing further analysis.

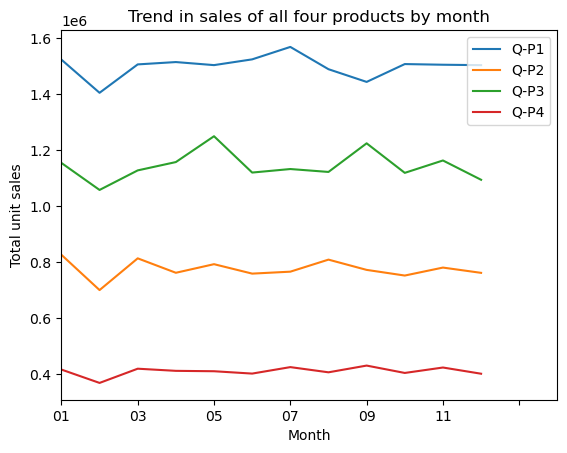
In [14]:

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

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

In [15]:

month\_plot()



**Observation:**

* We have merged the sales for months 9 and 09.
* 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

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

**Question**

* The company has all it's retail centers closed on the 31st of December every year. Mr:SHAKTHI PRIYAN , the CEO , would love to get an estimate on no: of units of each product that could be sold on 31st of Dec , every year , 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

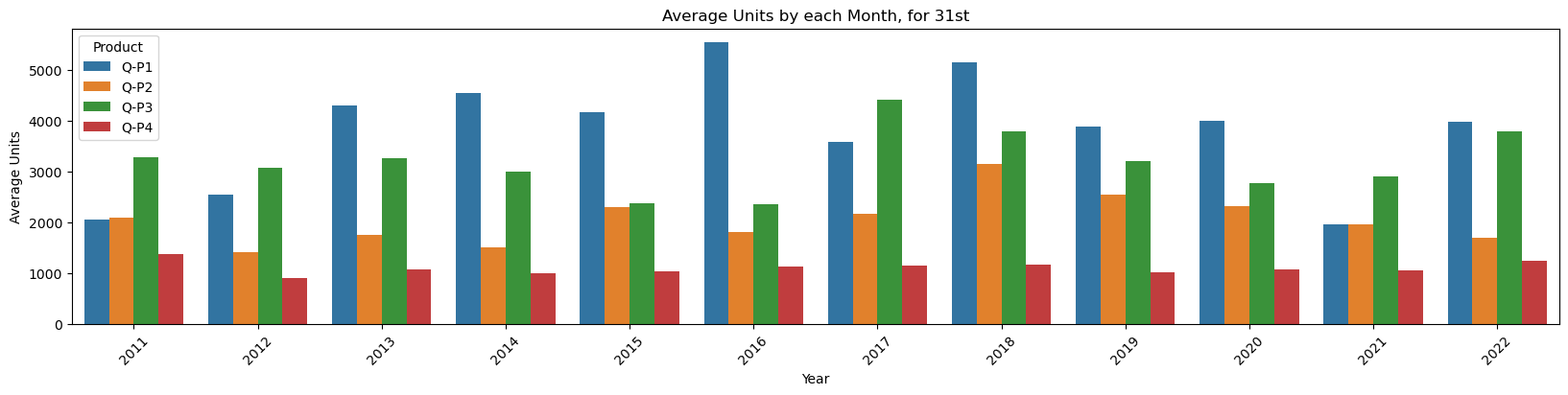
Out[16]:

|  | 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 |

84 rows × 12 columns

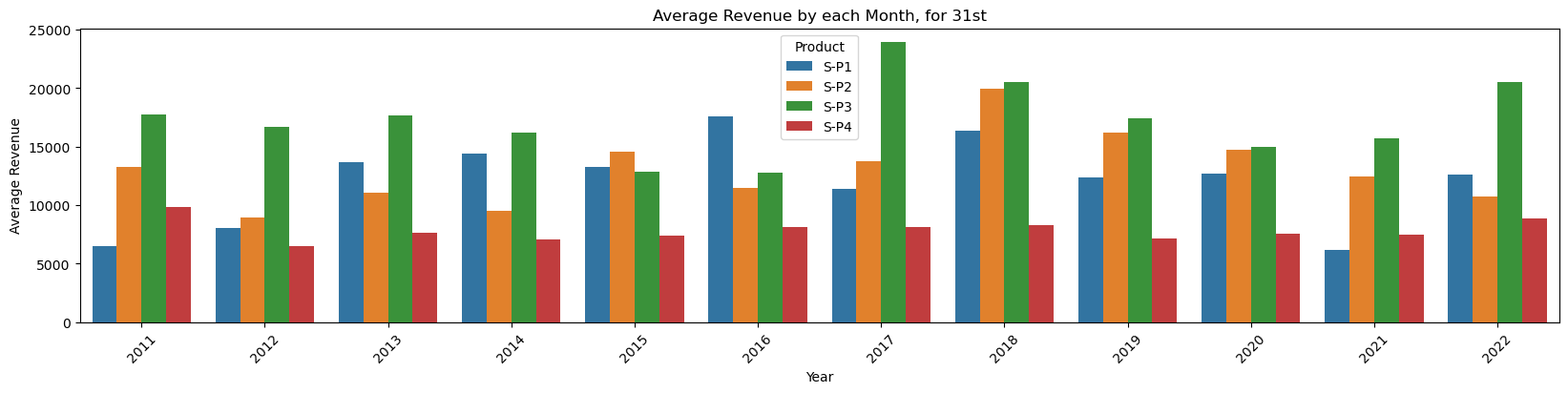
In [17]:

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')



**Observation:**

* 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.

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

**Observation:**

* We can see that our previous observation correlate as Q-P1 has the higest estimate, follwed 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

**Final Observations:**

**Product Sales Trends**: The project revealed that P1 consistently had the highest unit sales each year, with a notable peak in 2014. This trend suggests that REC corp LTD should consider strategies to capitalize on P1's popularity.

**Revenue Insights:** While P1 had the highest unit sales, P3 emerged as the highest revenue generator. This discrepancy can be attributed to P3's higher pricing, indicating the importance of both unit sales and pricing strategy in revenue generation.

**Seasonal Variations:** Analysis of monthly sales patterns unveiled distinct seasonality. Notably, February and December consistently exhibited lower sales, warranting a closer examination of the reasons behind these dips. In contrast, other months saw peaks in unit sales, suggesting that REC corp LTD could capitalize on these trends by adjusting their marketing and stocking strategies.

**Year-End Sales Estimations:** An estimate of year-end sales was provided, offering a ballpark figure for the number of units each product could sell on December 31st if all retail centers were open. These estimations provide REC corp LTD with insights into potential year-end performance and can guide their inventory and staffing planning.

**Data-Driven Decision-Making:** This project underscores the importance of data-driven decision-making. REC corp LTD can leverage the insights derived from this analysis to refine their product offering, marketing efforts, and inventory management, ultimately optimizing their business performance.

**CONCLUSION:**

In conclusion, this project represents a significant step toward REC corp LTD's data-driven future. It serves as a foundation for making informed decisions and adopting strategies that align with market trends, seasonality, and product performance. With these insights, REC corp LTD is better equipped to navigate the dynamic and competitive business landscape, maximizing their potential for growth and success.

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