Product Sales Data Analysis

Shakir Ullah Shakir

2024-06-21

Install and load necessary packages

The first step is to load the necessary packages.

```r
required_packages <- c("dplyr", "ggplot2", "tidyr", "reshape2", "readr")

install_if_missing <- function(pkg) {
  if (!require(pkg, character.only = TRUE)) {
    install.packages(pkg, dependencies = TRUE)
    library(pkg, character.only = TRUE)
  }
}

# Install and load packages
sapply(required_packages, install_if_missing)
```

```
## Loading required package: dplyr
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
## Loading required package: ggplot2
```

```
## Loading required package: tidyr
```

```
## Loading required package: reshape2
```

```
##
## Attaching package: 'reshape2'
```

```
## The following object is masked from 'package:tidyr':
##
##     smiths
```

```
## Loading required package: readr
## $dplyr
## NULL
##
## $ggplot2
## NULL
##
## $tidyr
## NULL
##
## $reshape2
## NULL
##
## $readr
## NULL
```

Load necessary libraries explicitly.

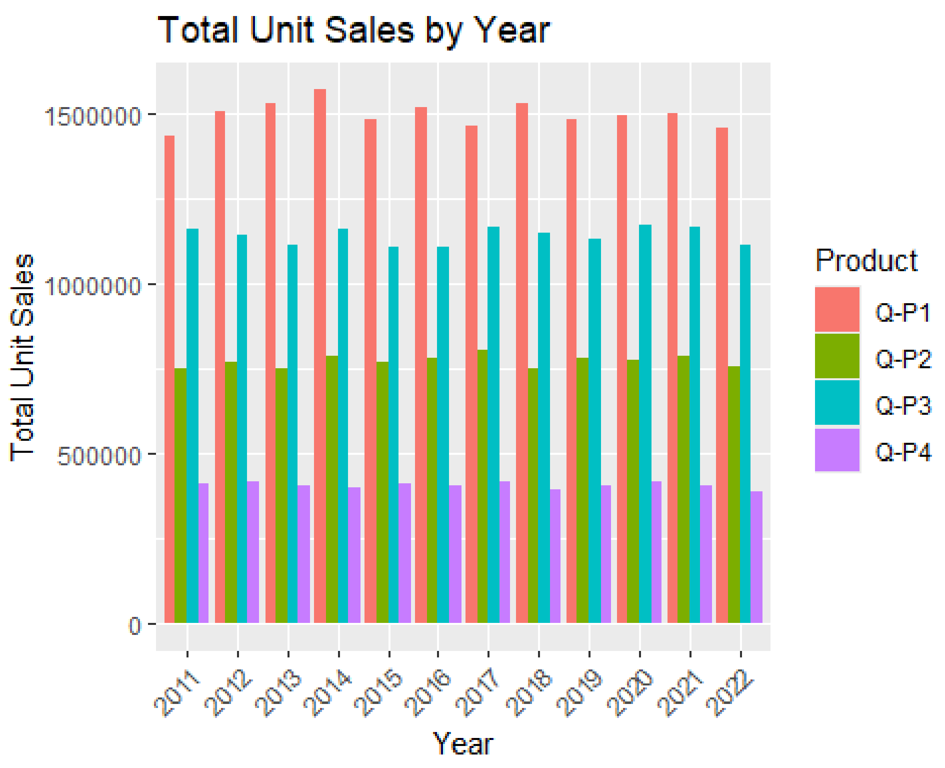
The first step is to load the necessary packages.

```
library(dplyr)
library(ggplot2)
library(tidyr)
library(reshape2)
library(readr)

# Import the data
data <- read_csv("C:/Users/Shakir Ullah Shakir/Downloads/Product Sales
Data.csv")

## New names:
## Rows: 4600 Columns: 10
## ── Column specification
## ──────────────────────────────────────────────────────── Delimiter: ","
chr
## (1): Date dbl (9): ...1, Q-P1, Q-P2, Q-P3, Q-P4, S-P1, S-P2, S-P3, S-P4
## ℹ Use `spec()` to retrieve the full column specification for this data. ℹ
## Specify the column types or set `show_col_types = FALSE` to quiet this
message.
## • `` -> `...1`

#  Drop the first column
data <- data %>% dplyr::select(-1)

# Display the structure of the data
str(data)

## tibble [4,600 × 9] (S3: tbl_df/tbl/data.frame)
##  $ Date: chr [1:4600] "13-06-2010" "14-06-2010" "15-06-2010" "16-06-2010"
...
##  $ Q-P1: num [1:4600] 5422 7047 1572 5657 3668 ...
##  $ Q-P2: num [1:4600] 3725 779 2082 2399 3207 ...
##  $ Q-P3: num [1:4600] 576 3578 595 3140 2184 ...
##  $ Q-P4: num [1:4600] 907 1574 1145 1672 708 ...
##  $ S-P1: num [1:4600] 17188 22339 4983 17933 11628 ...
##  $ S-P2: num [1:4600] 23617 4939 13200 15210 20332 ...
##  $ S-P3: num [1:4600] 3122 19393 3225 17019 11837 ...
##  $ S-P4: num [1:4600] 6467 11223 8164 11921 5048 ...
```

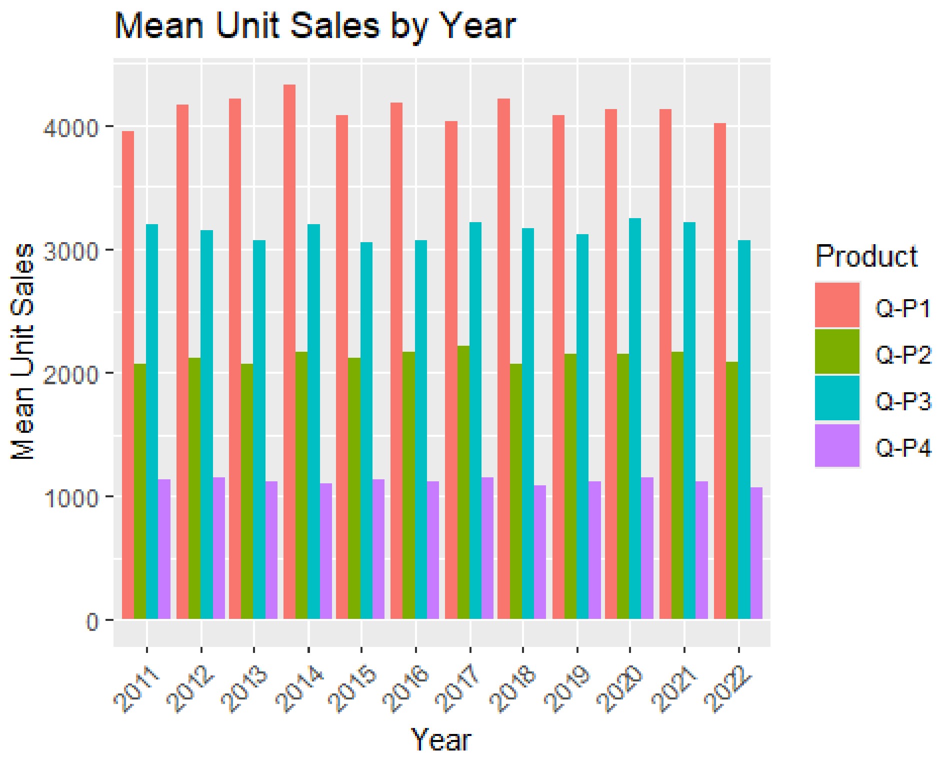
```
# Check for missing values
colSums(is.na(data))
```

```
## Date Q-P1 Q-P2 Q-P3 Q-P4 S-P1 S-P2 S-P3 S-P4
##    0    0    0    0    0    0    0    0    0
```

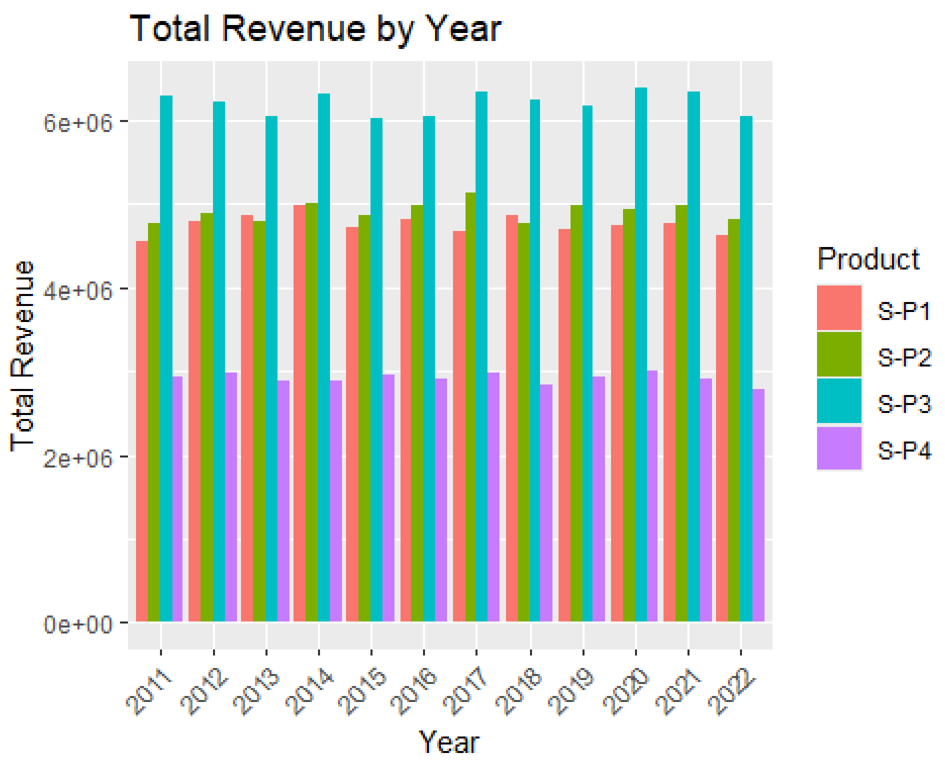
```
#  Extract 'Day', 'Month', and 'Year' from the 'Date' column
data <- data %>%
  mutate(Day = sapply(strsplit(Date, "-"), `[`, 1),
         Month = sapply(strsplit(Date, "-"), `[`, 2),
         Year = sapply(strsplit(Date, "-"), `[`, 3))

# Filter out data for years 2010 and 2023
data_reduced <- data %>% filter(Year != '2010' & Year != '2023')

#  Function to plot bar charts
plot_bar_chart <- function(df, columns, stri, str1, val) {
  if (val == 'sum') {
    sales_by_year <- df %>% group_by(Year) %>%
summarise(across(all_of(columns), sum))
  } else if (val == 'mean') {
    sales_by_year <- df %>% group_by(Year) %>%
summarise(across(all_of(columns), mean))
  }

  sales_by_year_melted <- melt(sales_by_year, id.vars = 'Year', variable.name
= 'Product', value.name = 'Sales')

  ggplot(sales_by_year_melted, aes(x = Year, y = Sales, fill = Product)) +
    geom_bar(stat = "identity", position = "dodge") +
    labs(x = "Year", y = stri, title = paste(stri, "by", str1)) +
    theme(axis.text.x = element_text(angle = 45, hjust = 1))
}

#  Use the plot_bar_chart function for unit sales
plot_bar_chart(data_reduced, c('Q-P1', 'Q-P2', 'Q-P3', 'Q-P4'), 'Total Unit
Sales', 'Year', 'sum')
```



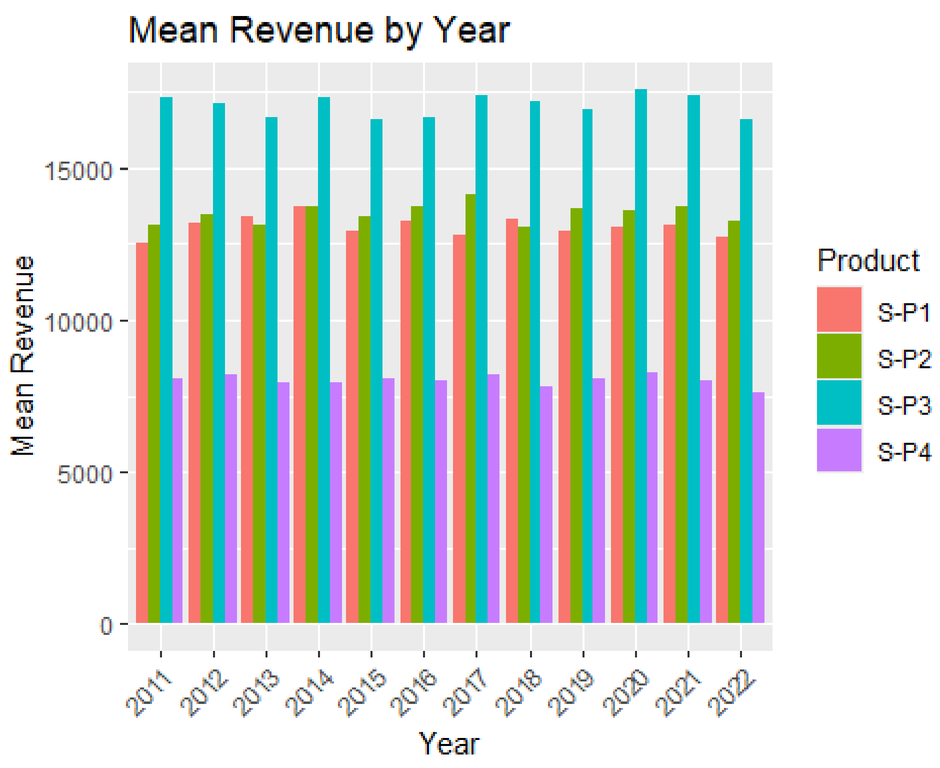
```r
plot_bar_chart(data_reduced, c('Q-P1', 'Q-P2', 'Q-P3', 'Q-P4'), 'Mean Unit
Sales', 'Year', 'mean')
```



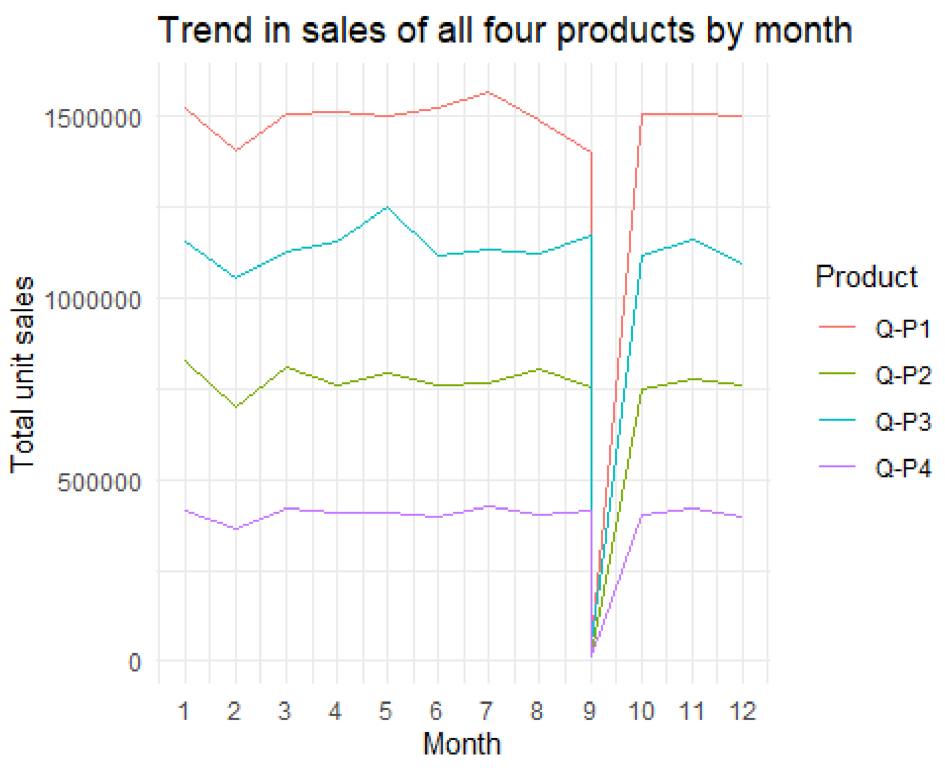
```r
# Use the plot_bar_chart function for revenue
plot_bar_chart(data_reduced, c('S-P1', 'S-P2', 'S-P3', 'S-P4'), 'Total
Revenue', 'Year', 'sum')
```



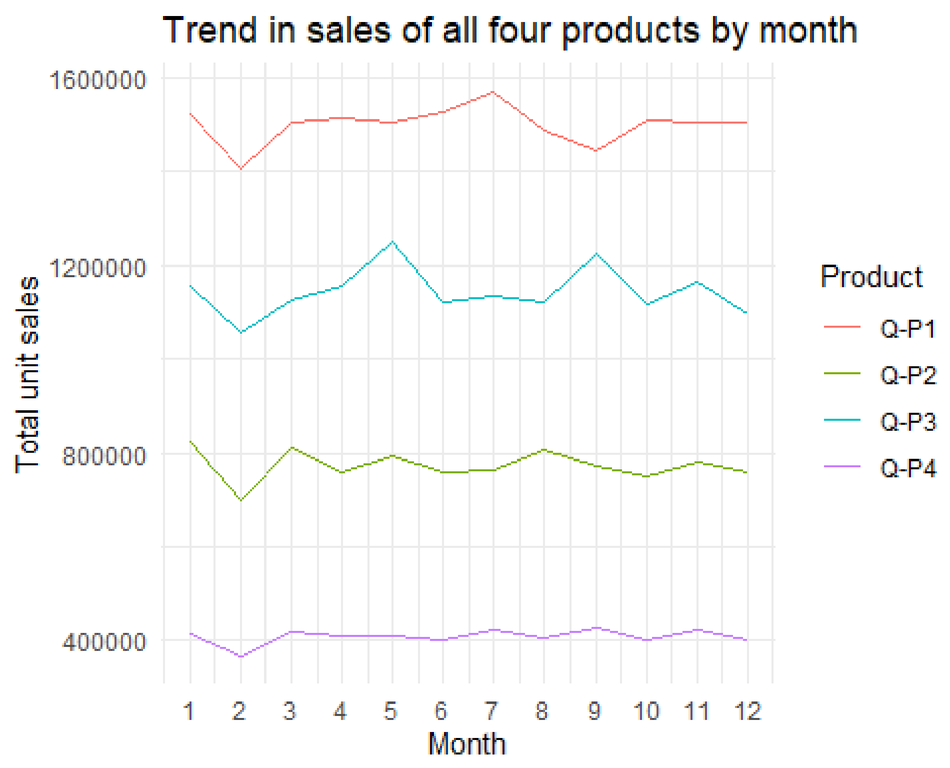
```r
plot_bar_chart(data_reduced, c('S-P1', 'S-P2', 'S-P3', 'S-P4'), 'Mean
Revenue', 'Year', 'mean')
```



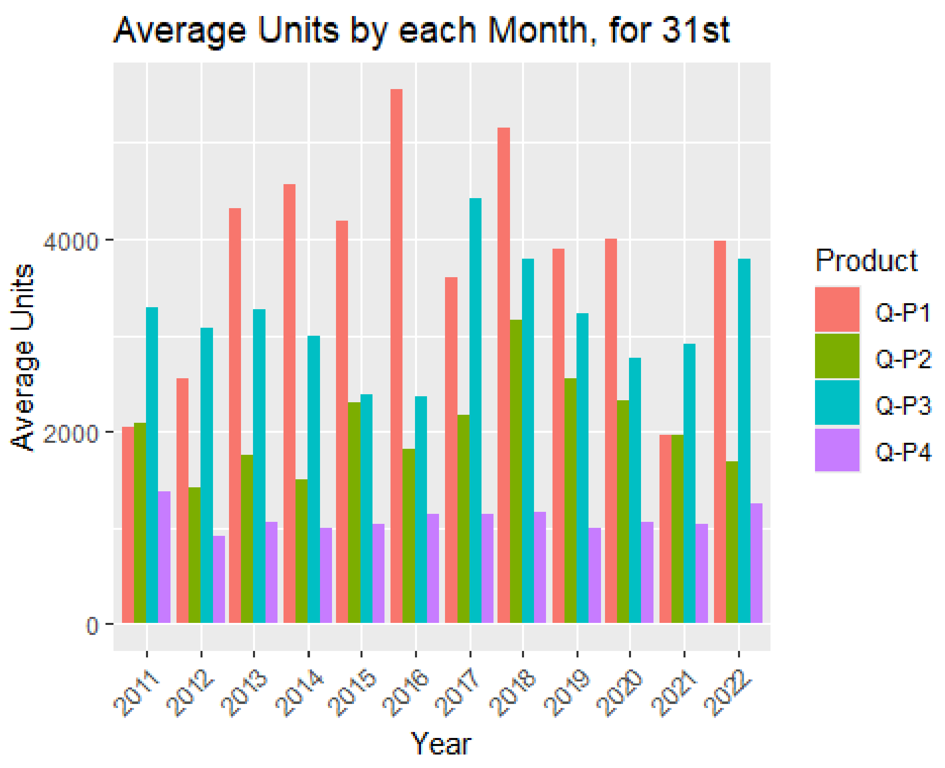
```
# Function to plot sales trend by month
month_plot <- function() {
  data_reduced %>%
    group_by(Month) %>%
    summarise(across(starts_with('Q-P'), sum)) %>%
    gather(key = "Product", value = "Sales", -Month) %>%
    ggplot(aes(x = as.integer(Month), y = Sales, color = Product)) +
    geom_line() +
    scale_x_continuous(breaks = 1:12) +
    labs(x = "Month", y = "Total unit sales", title = "Trend in sales of all
four products by month") +
    theme_minimal()
}

month_plot()
```



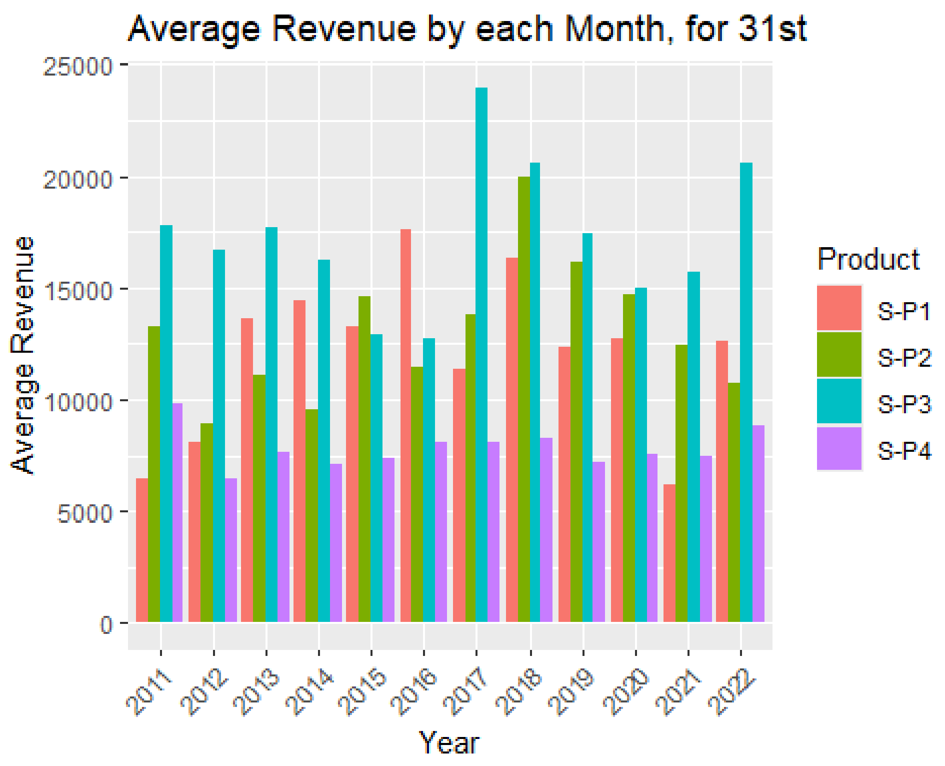
```r
# Replace all entries of '9' in the Month column with '09'
data_reduced$Month <- ifelse(data_reduced$Month == '9', '09',
data_reduced$Month)

month_plot()
```



```r
# Function to get data for months with 31st day
month_31_data <- function(df, months) {
  df %>% filter(Month %in% months & Day == '31')
}

months_31 <- month_31_data(data_reduced, c('01', '02', '03', '04', '05',
'06', '07', '08', '09', '10', '11', '12'))

# Use the plot_bar_chart function for average units and revenue on 31st day
of each month
plot_bar_chart(months_31, c('Q-P1', 'Q-P2', 'Q-P3', 'Q-P4'), 'Average Units',
'each Month, for 31st', 'mean')
```



```
plot_bar_chart(months_31, c('S-P1', 'S-P2', 'S-P3', 'S-P4'), 'Average
Revenue', 'each Month, for 31st', 'mean')
```



```
# Function to calculate average sales or revenue on 31st day
avg_on_31st <- function(df, product) {
  df %>% filter(Day == '31') %>% summarise(across(all_of(product), mean,
na.rm = TRUE)) %>% round(2)
}

#  Average for Unit Sales
avg_on_31st(data_reduced, c('Q-P1', 'Q-P2', 'Q-P3', 'Q-P4'))

## Warning: There was 1 warning in `summarise()`.
## ℹ In argument: `across(all_of(product), mean, na.rm = TRUE)`.
## Caused by warning:
## ! The `...` argument of `across()` is deprecated as of dplyr 1.1.0.
## Supply arguments directly to `.fns` through an anonymous function instead.
##
##   # Previously
##   across(a:b, mean, na.rm = TRUE)
##
##   # Now
##   across(a:b, \(x) mean(x, na.rm = TRUE))

## # A tibble: 1 × 4
##   `Q-P1` `Q-P2` `Q-P3` `Q-P4`
##    <dbl>  <dbl>  <dbl>  <dbl>
## 1  3814.  2059.  3184.  1099.

# Average for Revenue
avg_on_31st(data_reduced, c('S-P1', 'S-P2', 'S-P3', 'S-P4'))

## # A tibble: 1 × 4
##   `S-P1` `S-P2` `S-P3` `S-P4`
##    <dbl>  <dbl>  <dbl>  <dbl>
## 1 12090. 13053. 17257.  7833.
```