AdventureWorks2019 Database Project

Interim Technical Report

Aminat Ademola
Kristian Ambruch
Hesham Fahiem
Marcos Ferreira
Adaobi Ikeasomba
Bayo Olabisi

Generation UK

23/10/2023

Contents

AdventureWorks2019 Database Overview

Database Schema

The AdventureWorks2019 database serves as a robust model for understanding the multi-faceted operations of a manufacturing company. It is compartmentalized into various schemas, each designed to manage different aspects of the business:

- **HumanResources:** Centralizes all employee-related data, including job titles, organizational hierarchy, and leave records.
- **Production:** Manages the intricacies of product details, inventory, and manufacturing workflows.
- **Sales:** Holds records related to customer interactions, sales territories, and revenue data.
- **Person:** Captures personal identifiers such as names and contact details.
- **Purchasing:** Focuses on vendors, procurement processes, and associated financial metrics. Table Structure

Table Structure

The AdventureWorks2019 database is structured to facilitate various business scenarios, but for the purpose of our analysis, we primarily focused on tables and views from the Sales and HumanResources schemas. Below are the specific tables and their relevant columns that were utilized:

Sales Schema

Sales.SalesTerritory

- CountryRegionCode: Unique code for each country or region.
- Name: Name of the sales territory.
- SalesYTD: Total sales for the current year-to-date.
- SalesLastYear: Total sales for the previous year.

Sales.Store

- BusinessEntityID: Unique identifier for each store.
- Name: Name of the store.
- SalesPersonID: Identifier for the linked salesperson.

Sales.SalesPerson

- BusinessEntityID: Unique identifier for each salesperson.
- Bonus: Bonus amount received by the salesperson.
- Sales.SalesOrderHeader
- TotalDue: Total amount due for a particular sales order.
- SalesPersonID: Identifier for the linked salesperson.

HumanResources Schema

HumanResources.Employee

- BusinessEntityID: Unique identifier for each employee.
- JobTitle: Designation of the employee.
- OrganizationLevel: Hierarchical level within the organization.

- VacationHours: Number of vacation hours taken.
- SickLeaveHours: Number of sick leave hours taken.

Views

Several views are also provided to facilitate complex queries, such as:

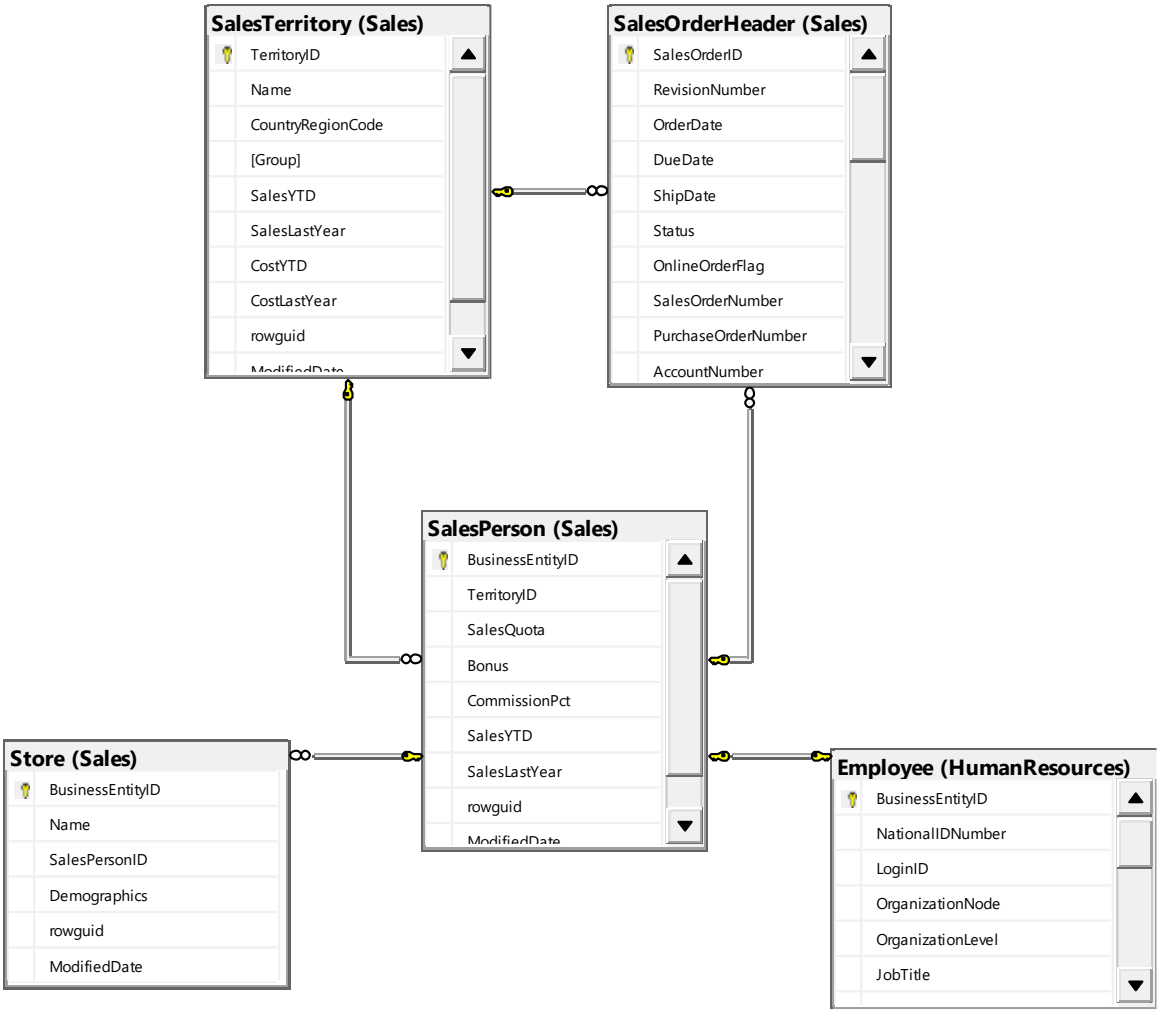
vEmployee: Combines data from multiple tables to provide comprehensive employee details.
vProductAndDescription: Provides product details along with their descriptions.

Custom Views

SizeEmployeeRevenue: This view was created from Sales.vStorewithDemographics and contains columns like Size (square footage of the store), Average_Revenue (annual revenue), and Avg_num_employees (average number of employees per store).

These tables and views were central to our investigation into relationships between store size, revenue, and employee metrics.

Entity-Relationship Model



Question 1 – Marcos Ferreira

What are the regional sales in the best performing country?

Initial Ideas/Thought Process

There are two necessary step processes to answer this question. First, find the best performing country and second, show the regional sales for that country. Tried to write the SQL code in one query only, as it's normally advisable, but struggled to generate two different charts at the same Python query. Therefore, it was decided to write two simple and easily understandable SQL queries, two Python queries which generated one chart each.

Methodology

SQL Queries

Table: Sales.SalesTerritory
Columns: CountryRegionCode (Country), Name (Region), salesYTD (This_Year_Sales), salesLastYear (Last_Year_Sales)

```
SELECT countryRegionCode AS Country,
SUM(salesYTD) AS This_Year_Sales,
SUM(salesLastYear) AS Last_Year_Sales
FROM Sales.SalesTerritory
GROUP BY countryRegionCode
ORDER BY SUM(salesYTD) DESC;

_____________________ // _______________

SELECT countryRegionCode AS Country,
       name AS Region,
       SalesYTD AS This_Year_Sales,
       SalesLastYear AS Last_Year_Sales
FROM sales.SalesTerritory
WHERE CountryRegionCode = 'US'
ORDER BY SalesYTD DESC;
```

On the first query, we identified the best performing country, United States, by extracting from the data all the countries in order by the highest sales for this year. To do this, we used the SUM function and GROUP BY statement to add all the sales altogether of each country. We also extract the total sales of each country on the last year. This way, we could later create a more insightful chart where we could compare the sales of each country on both last year and this year.

On the second query, we kept the same idea of extracting the sales of both last and current year, but this time we focused on the US regions: Northwest, Northeast, Southwest, Southeast and Central. The WHERE clause helped us to tell SQL we are only interested on the 'US' CountryRegionCode (renamed as 'Country').

Python Code

```
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.ticker import FuncFormatter
```

At the beginning of the Python query I installed (pip install) and imported the above libraries.

Pandas

- Used to read the CSV file and create a DataFrame: q1 = pd.read_csv(csv_file_path)
- Used for data manipulation: Calculating the percentage difference.

Matplotlib

- Used to create the bar chart: plt.bar(...), plt.xlabel(...), plt.ylabel(...), plt.title(...), plt.xticks(...), plt.legend(...), plt.tight_layout(), plt.show()

Matplotlib's FuncFormatter

- Used to format the y-axis ticks(display the sales figures in millions): plt.gca().yaxis.set_major_formatter(...)

In summary, Pandas was used for reading data and performing calculations, Matplotlib was used for creating the bar chart and customizing its appearance, and FuncFormatter from Matplotlib's ticker module was used to format the y-axis ticks.

Reading Data from a CSV file:

```
# Read the CSV file into a DataFrame
csv_file_path = r"C:\Users\Marcos\Desktop\Data Analytics\Interim Project\Question1\q1-sales comparison by country.csv"
q1 = pd.read_csv(csv_file_path)
```

csv_file_path = r"C:\Users\Marcos\Desktop\Data Analytics\Interim Project\Question1\q1- sales comparison by country.csv": This line specifies the file path where the CSV file is located on your computer. In this case, it's located at the specified path.

q1 = pd.read_csv(csv_file_path): This line uses the Pandas library (pd) to read the data from the CSV file. It creates a DataFrame named q1 to store this data. The function read_csv() is used to read the CSV file and convert it into a DataFrame.

The read_csv method is a simple and recommended approach to use when you have a relatively small dataset that doesn't require frequent updates. It's efficient and straightforward for reading and working with data stored in CSV format. The DataFrame is like a table in a spreadsheet or a database. It organises data into rows and columns, making it easy to perform various operations and analyses. In this case, you're likely using this DataFrame to analyse sales data by country.

Calculating Percentage Difference:

This part of the code is performing a calculation to find the percentage difference in sales between this year and last year for each entry in the DataFrame q1. Here's the explanation:

```
# Calculate the percentage difference
q1['Percentage_Difference'] = ((q1['This_Year_Sales'] - q1['Last_Year_Sales']) / q1['Last_Year_Sales']) * 100
```

q1['Percentage_Difference']: This creates a new column named Percentage_Difference in the DataFrame q1 where the results of the calculation will be stored.

q1['This_Year_Sales'] - q1['Last_Year_Sales']: This computes the difference between the sales figures for this year and last year.

/ q1['Last_Year_Sales']: This part of the formula divides the difference by the sales figures for last year. This is done to find the relative change in sales.

*** 100**: Finally, the result is multiplied by 100 to convert it into a percentage.

The entire calculation finds the percentage difference in sales between this year and last year, and stores the result in the new column Percentage_Difference. This allows for a clear comparison of the sales performance for each entry in the dataset.

Creating the Bar Chart:

This part of the code is responsible for creating the bar chart that will visualize the sales data. Here's the explanation:

```
# Create the bar chart
plt.figure(figsize=(10, 5))

# Define the width of each bar group
bar_width = 0.35
```

plt.figure(figsize=(10, 5)): This line creates a new figure for the plot with a specified size of 10 inches in width and 5 inches in height. This sets the dimensions of the plot canvas where the bar chart will be displayed.

bar_width = 0.35: This line defines the width of each bar group in the bar chart. The value 0.35 indicates the width of the bars. Adjusting this value would change the width of the bars in the chart.

Defining Bar Positions on the x-axis:

This part of the code is responsible for defining the positions of the bars on the x-axis of the bar chart. Here's the explanation:

```
# Define the positions of the bars on the x-axis
index = range(len(q1))
```

index = range(len(q1)): This line creates a sequence of numbers representing the positions where the bars will be placed along the x-axis. The length of q1 (the number of data points or entries) is determined using len(q1). The range() function then generates a sequence of numbers from 0 to len(q1)-1. This sequence represents the positions where the bars will be plotted on the x-axis.

Plotting Bars for This Year's Sales:

This part of the code is responsible for plotting the bars representing sales for the current year on the bar chart. Here's the explanation:

```
# Plot bars for this year sales
bars_this_year = plt.bar(index, q1['This_Year_Sales'], width=bar_width, color='blue',
alpha=0.7, label='This Year')
```

bars_this_year = plt.bar(...): This line creates the bars for this year's sales.

index: The index variable provides the positions on the x-axis where these bars will be placed.

q1['This_Year_Sales']: This specifies the heights of the bars. It retrieves the sales figures for this year from the DataFrame q1.

width=bar_width: This sets the width of the bars to the value stored in the bar_width variable, which was defined earlier.

color='blue': This sets the colour of the bars to blue.

alpha=0.7: This sets the transparency level of the bars. An alpha value of 0.7 means the bars will be 70% opaque.

label='This Year': This assigns a label to these bars, which can be used for the legend later. The label here is "This Year".

bars_this_year: This variable stores the information about the bars representing sales for this year. It can be used later for creating the legend or modifying the properties of these bars.

Plotting Bars for Last Year's Sales:

This section of the code is responsible for creating bars to represent sales figures for the previous year on the bar chart. Here's the explanation:

```
# Plot bars for last year sales (shifted by bar_width)
bars_last_year = plt.bar([p + bar_width for p in index], q1['Last_Year_Sales'],
width=bar_width, color='lightblue', alpha=0.5, label='Last Year')
```

bars_last_year = plt.bar(...): This line generates the bars that indicate sales for the previous year.

[p + bar_width for p in index]: This list comprehension is used to determine the positions on the x-axis where the bars will be located. It shifts each position from index by the bar_width, ensuring that the bars for last year are displayed alongside the bars for this year.

q1['Last_Year_Sales']: This retrieves the sales figures for the previous year from the DataFrame q1 and uses them as the heights of the bars.

width=bar_width: This sets the width of the bars to the value defined earlier in the bar_width variable.

color='lightblue': This sets the colour of the bars to light blue.

alpha=0.5: This adjusts the transparency level of the bars. A value of 0.5 means that the bars will be 50% opaque.

label='Last Year': This assigns a label to these bars, which can be used later for the legend. The label in this case is "Last Year".

bars_last_year: This variable stores information about the bars representing sales for the previous year. It can be used later for creating the legend or modifying the properties of these bars.

Adding Percentage Increase Labels:

This section of the code adds labels indicating the percentage increase on top of the bars representing sales for the current year. Here's the explanation:

```
# Add percentage increase labels on top of this year's bars
for bar, percentage_diff in zip(bars_this_year, q1['Percentage_Difference']):
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, "{:+.1f}%".format(percentage_diff), ha='center', va='bottom', color='black', fontsize=10)
```

for bar, percentage_diff in zip(bars_this_year, q1['Percentage_Difference']): This loop iterates through pairs of bars_this_year (representing bars for this year's sales) and corresponding percentage_diff values (calculated earlier). The zip() function pairs up these elements.

yval = bar.get_height(): This line retrieves the height (value) of the bar. This will be used as the y-coordinate for placing the text label.

plt.text(...): This function is used to add a text label to the plot.

bar.get_x() + bar.get_width()/2: This calculates the x-coordinate for placing the text label at the centre of the bar.

yval: This is the y-coordinate where the text label will be placed, which corresponds to the height of the bar.

"{:+.1f}%".format(percentage_diff): This formats the text label. It displays the percentage_diff with one decimal place, preceded by a sign (+ or -), and followed by a percentage symbol (%).

ha='center': This sets the horizontal alignment of the text label to be centred.

va='bottom': This sets the vertical alignment of the text label to be aligned with the bottom of the bar.

color='black': This sets the colour of the text label to black.

fontsize=10: This sets the font size of the text label to 10 points.

Adding Labels:

This part of the code is responsible for adding labels to the plot to provide context and information about the data being presented. Here's the explanation:

```
# Add labels
plt.ylabel('Sales (in Millions $)')
plt.xlabel('Country')
plt.title('Sales Comparison (This Year vs Last Year)')
```

---

plt.ylabel('Sales (in Millions $)'): This sets the label for the y-axis. It indicates that the values on the y-axis represent sales figures, and that the values are in millions of dollars.

plt.xlabel('Country'): This sets the label for the x-axis. It indicates that the values on the x-axis represent different countries.

plt.title('Sales Comparison (This Year vs Last Year)'): This sets the title of the plot. It provides an overall description of what the plot is showing, which is a comparison of sales figures between this year and last year.

Setting X-axis Labels and Ticks:

This part of the code is responsible for customizing the x-axis labels and ticks on the plot. Here's the explanation:

```
# Set x-axis labels and ticks
plt.xticks([p + bar_width/2 for p in index], q1['Country'], rotation=45)
```

---

plt.xticks([...], q1['Country'], rotation=45): This line configures the x-axis labels and their positions.

[p + bar_width/2 for p in index]: This list comprehension generates the positions where the x-axis labels will be placed. It takes each position in index, shifts it by half of the bar_width, and creates a list of these modified positions. This ensures that the labels are centred under each set of bars.

q1['Country']: This provides the actual labels. It uses the 'Country' column from the DataFrame q1.

rotation=45: This specifies that the labels should be rotated by 45 degrees for better visibility, especially if the country names are long.

Setting Y-axis Ticks in Millions:

This part of the code customizes the y-axis ticks to display sales figures in millions. Here's the explanation:

```
# Set y-axis ticks to display in millions
plt.gca().yaxis.set_major_formatter(FuncFormatter(lambda x, loc: "{:.0f}M".format(x / 1e6)))
```

---

plt.gca(): This retrieves the current Axes object, which represents the plot.

.yaxis.set_major_formatter(FuncFormatter(...)): This sets a custom tick formatter for the y-axis.

FuncFormatter(...): This specifies that a custom formatting function will be used to format the ticks.

lambda x, loc: "{:.0f}M".format(x / 1e6): This is a lambda function that takes a tick value x and a location loc (which is not used in this case). It divides x by 1e6 (which is one million) and formats it to display a whole number with an 'M' to indicate millions. For example, if x is 2000000, it will be formatted as '2M'.

This configuration ensures that the y-axis ticks will be displayed in millions for better readability and comprehension of the sales figures.

Adding a Legend:

This part of the code is responsible for adding a legend to the plot and displaying the graph. Here's the explanation:

```
# Add legend
plt.legend()

# Show the graph
plt.tight_layout()
plt.show()
```

---

plt.legend(): This command adds a legend to the plot. The legend helps to distinguish between different sets of data, in this case, "This Year" and "Last Year" sales.

Displaying the Graph:

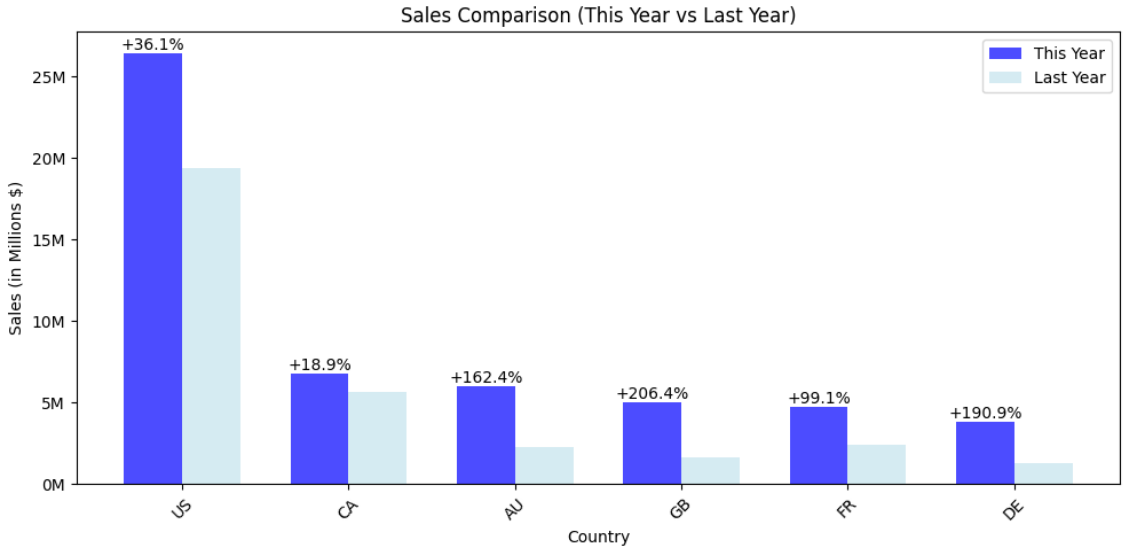
plt.tight_layout(): This adjusts the layout of the plot for better spacing and aesthetics.

plt.show(): This command displays the final plot on your screen. It's the final step to visualize the sales comparison chart.

```
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.ticker import FuncFormatter

# Read the CSV file into a DataFrame
csv_file_path = r"C:\Users\Marcos\Desktop\Data Analytics\Interim Project\Question1\q1-
sales comparison by country.csv"
q1 = pd.read_csv(csv_file_path)

# Calculate the percentage difference
q1['Percentage_Difference'] = ((q1['This_Year_Sales'] - q1['Last_Year_Sales']) /
q1['Last_Year_Sales']) * 100

# Create the bar chart
plt.figure(figsize=(10, 5))

# Define the width of each bar group
bar_width = 0.35

# Define the positions of the bars on the x-axis
index = range(len(q1))

# Plot bars for this year sales
bars_this_year = plt.bar(index, q1['This_Year_Sales'], width=bar_width, color='blue',
alpha=0.7, label='This Year')

# Plot bars for last year sales (shifted by bar_width)
bars_last_year = plt.bar([p + bar_width for p in index], q1['Last_Year_Sales'],
width=bar_width, color='lightblue', alpha=0.5, label='Last Year')

# Add percentage increase labels on top of this year's bars
for bar, percentage_diff in zip(bars_this_year, q1['Percentage_Difference']):
    yval = bar.get_height()
```

```
    plt.text(bar.get_x() + bar.get_width()/2, yval, "{:+.1f}%".format(percentage_diff),
ha='center', va='bottom', color='black', fontsize=10)

# Add labels
plt.ylabel('Sales (in Millions $)')
plt.xlabel('Country')
plt.title('Sales Comparison (This Year vs Last Year)')

# Set x-axis labels and ticks
plt.xticks([p + bar_width/2 for p in index], q1['Country'], rotation=45)

# Set y-axis ticks to display in millions
plt.gca().yaxis.set_major_formatter(FuncFormatter(lambda x, loc: "{:.0f}M".format(x /
1e6)))

# Add legend
plt.legend()

# Show the graph
plt.tight_layout()
plt.show()
```

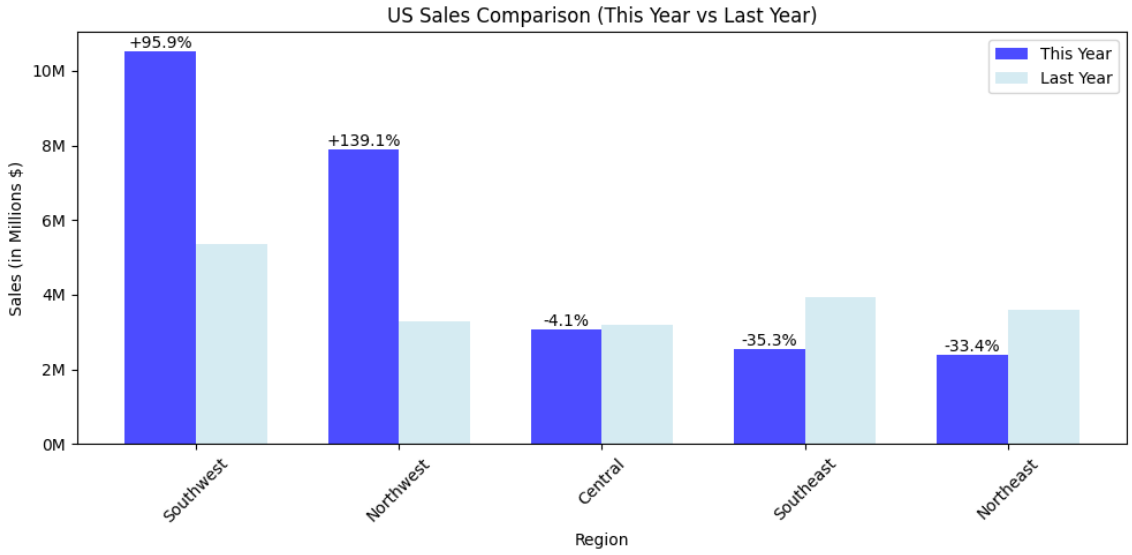
Charts



This bar chart was carefully chosen and designed to match the presentation's colour scheme. It's meant to show information in an easy-to-understand way and can provide some interesting insights.

Here, you can see the sales for this year (darker blue) compared to last year (lighter blue), along with the percentage difference. For instance, you can clearly see from the graph that while the US is the best-performing country in both last year and this year, its growth is up to 5 times lower compared to other countries.

Most graphs typically use the 'matplotlib.pyplot' library for plots. They show the y-axis with sales numbers like 0.5, 1.5, 2.5, etc., and at the top, they might show 1e7, which means the number plus 7 zeros after it. So, 2.5 on the chart would actually represent 25 million, and so on. We also added the $ sign at the “y label” so anyone can see the the values are in dollars for all countries.



On this second graph, we maintained the same color scheme, legend, labels, and other elements as the first graph. The only change is that it now displays the sales for different regions within the United States. This consistency between both graphs makes it easier for the audience to follow any spoken explanation and understand the intended message. For instance, we can observe that sales on the West Coast experienced significant growth, while sales on the East Coast and in the Central region declined.

Further Insights

Based on our findings we would suggest the following recommendations which could potentially help the company to build on its current successes, address challenges, and drive sustained growth in both the US and international markets:

Focus on Expanding in Europe and the Pacific: It is probably advisable to heavily invest on the business expansion on this countries and continents if in line with company’s vision, sales margins are high and other factors like Government Policies are favourable. SWOT and PESTLE analysis can be really helpful in evaluating the competition and external factors when considering the expansion.

Evaluate Regional Performance in the US: Despite the total sales for US increased 36% compared to last year, it’s crucial to search the answers for some questions: Why are the sales in the East Coast and Central declining? Is it related to customer behaviour change or economic factors? Can it influence the West Coast where the most sales are concentrated? What potential actions could revert this situation?

Consider Government Policies and Market Conditions: Monitor and adapt to any changes in government policies or market conditions that may impact sales performance, especially in regions with declining sales.

Review Marketing Efforts: Analyse the effectiveness of marketing campaigns in different regions and adjust strategies accordingly. Tailoring marketing initiatives to specific regional preferences and needs

can lead to improved sales performance. In case of external marketing agency instead of in house agency, it might be worth looking at local agencies specialised in the “Bicycle and Sports Equipment” industry.

Supply Chain Optimisation: Address any supply chain issues that may have contributed to the fluctuations in regional sales. Ensuring a smooth and efficient supply chain process is crucial for maintaining consistent sales growth.

Store Expansion/Closures Strategy: Evaluate the impact of opening or closing stores on regional sales. Consider whether adjustments in store presence are necessary based on performance trends.

Customer Feedback and Preferences: Collect and analyse customer feedback to gain insights into regional preferences and demands. This information can inform product offerings, marketing strategies, and customer engagement efforts.

Periodic Performance Reviews: Implement regular performance reviews for each region to identify areas of improvement and recognise exceptional performance. This allows for timely adjustments and ensures that strategies remain aligned with company goals.

Competitive Analysis: Conduct a thorough competitive analysis to understand the market landscape and identify opportunities for gaining a competitive edge in each region. The SWOT(Strengths, Weaknesses, Opportunities and Threats) analysis and also the 5P’s (Product, Price, Promotion, Place and People) are really useful for retail business.

Question 2 – Bayo Olabisi

What is the relationship between annual leave taken and bonus?

Initial Ideas/Thought Process

As part of standard employment package, employees are entitled to Annual leave which they can book and take subject to organisational policies. To promote commitment and productivity, employers also introduce Bonus policies. Employees may get bonus either as a result of their individual performance (e.g sales for salesmen) or team performance. To understand the correlation between **annual leave taken** and **bonuses received**, we examined data from the following columns in the respective tables:

Bonus data	-Sales.SalesPerson table
Annual Leave (VacationHours)	-HumanResources.Employee table
Job Title	-HumanResources.Employee table
Organisational Level	-HumanResources.Employee table.

We have included Organisational level and Job title to make our analysis more meaningful. This is reflected in the SQL query we ran.

Methodology

- **Data examination and outcome preview**: We examined the Views and Tables in the AdventureWorks Database to understand the schema and know which table has data relevant to the question.
- **Queries**: We ran our queries on SQL server using the ***AdventureWorks*** Database, and loaded the results to **Python** on **Visual Studio Code**. We used pandas to manipulate the result of the queries so we could get insights into the data. From this exercise, we got the Correlation Coefficient, Mean and Median (for the variables), regression line and the Chart.
- **Visualisation/Charts**: We used ***matplotlib, Seaborn*** to generate the Charts and **Scatter Plot** to visualise the results. Scatter Plots was selected to visualise the data because we are analysing two continuous variables.
- **Results**: The **Correlation Coefficient of 0.38** indicates a positive correlation, but not strong enough to suggest any definite pattern.

Tables involved:

We explored the contents of the tables and found two relevant tables (HumanResources.Employee and Sales.SalesPerson) each containing only one of the two key variables. We performed a JOIN operation on the two tables on common Column “BusinessEntityID”.

In addition to the data points for the two variables, we included Mean, Median and Regression line in the script. This is to give us better insights. We also used different colours for different Job Titles for clear visual effect.

SQL Queries

```
SELECT [VacationHours], [Bonus], [JobTitle], [OrganizationLevel]
FROM [HumanResources].[Employee] as e
INNER JOIN [Sales].[SalesPerson] as p
ON e.[BusinessEntityID] = p.[BusinessEntityID]
ORDER BY [VacationHours] DESC;
```

Python Code

We imported the following libraries:

- **Pandas**- to be able to carry out data manipulation, read the CSV file and create Dataframe.
- **Matplotlib**-To be able to create the Scatter Plot.
- **Matplotlib's FuncFormatter** - To be able to format the chart.

The SQL query result was stored as a CSV file and read into a Dataframe through the specified path: ***r"C:\Users\adeba\OneDrive\Documents\vacationleave result.csv"***

We colour coded the chart for ease of understanding:

- Regression line -Red
- Mean Vacation and Mean Bonus – Green
- Median Vacation – Purple
- Median Bonus -Purple

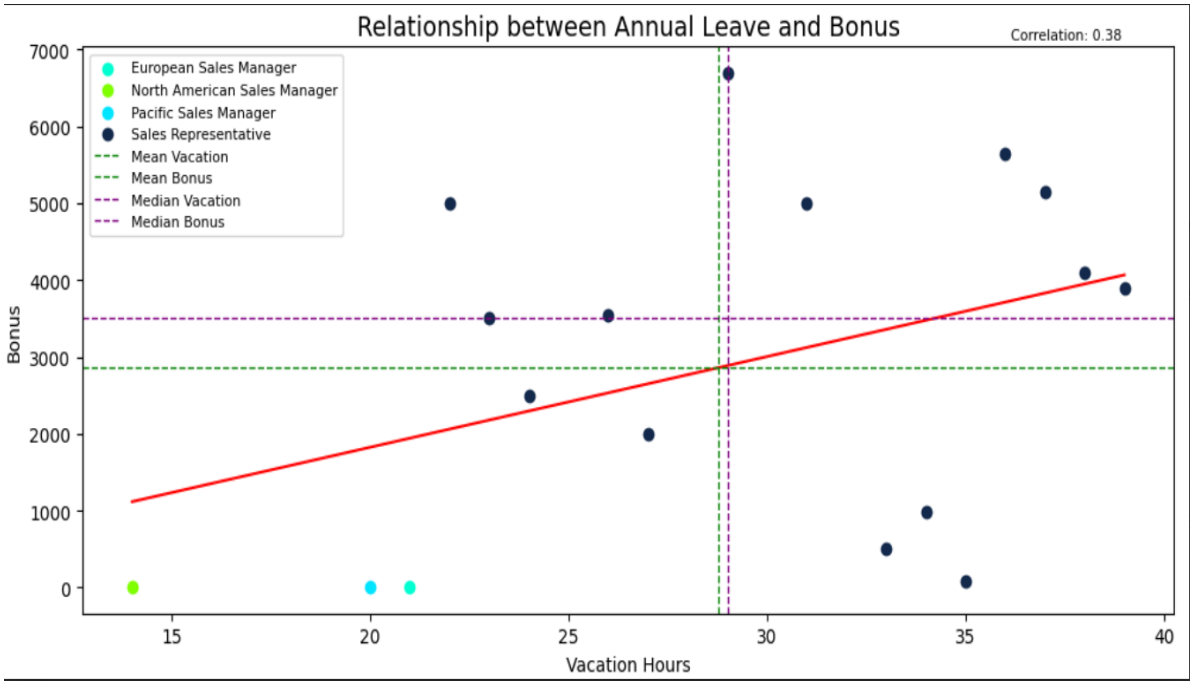
```
import pandas as pd
import matplotlib.pyplot as plt
from scipy import stats
import statsmodels.api as sm
import seaborn as sns

# Read the CSV file into a DataFrame
csv_file_path = r"C:\Users\Marcos\Desktop\Data Analytics\Interim Project\Question2\q2.csv"
q2 = pd.read_csv(csv_file_path)

# Map specific colors to Job Titles
color_mapping = {
    'Sales Representative': '#13294B',  # Dark Blue
    'European Sales Manager': '#00FFD1',  # Bright Green/Blue
    'Pacific Sales Manager': '#00E5FF',  # Bright Blue
    'North American Sales Manager': '#7FFF00',  # Bright Green
    }

# Grouping by Job Title
grouped_data = q2.groupby('JobTitle')

# Calculating Descriptive Statistics
summary_stats = grouped_data[['VacationHours', 'Bonus']].describe()

# Visualizing the Relationship
plt.figure(figsize=(12, 6))
```

```
for name, group in grouped_data:
    plt.scatter(group['VacationHours'], group['Bonus'], label=name,
color=color_mapping[name])

plt.xlabel('Vacation Hours')
plt.ylabel('Bonus')
plt.title('Relationship between Annual Leave and Bonus', fontsize=14)

# Add regression line
X = q2['VacationHours']
Y = q2['Bonus']
X = sm.add_constant(X)  # Adds a constant term to the predictor
model = sm.OLS(Y, X).fit()
predictions = model.predict(X)
plt.plot(X['VacationHours'], predictions, color='red')

# Add mean and median lines
plt.axvline(q2['VacationHours'].mean(), color='green', linestyle='dashed', linewidth=1,
label='Mean Vacation')
plt.axhline(q2['Bonus'].mean(), color='green', linestyle='dashed', linewidth=1, label='Mean
Bonus')
plt.axvline(q2['VacationHours'].median(), color='purple', linestyle='dashed', linewidth=1,
label='Median Vacation')
plt.axhline(q2['Bonus'].median(), color='purple', linestyle='dashed', linewidth=1,
label='Median Bonus')

# Add correlation coefficient near the title
correlation = q2['VacationHours'].corr(q2['Bonus'])
plt.text(0.85, 1.02, f'Correlation: {correlation:.2f}', fontsize=8, color='black',
verticalalignment='center', transform=plt.gca().transAxes)

# Keep legend in original position
plt.legend(fontsize=8)

plt.tight_layout()
plt.show()
```

Charts

The Script generated a Scatter Plot with Mean and Median for both Vacation and Annual Leave. Surprisingly, managers (level 3) took less leave and received smaller bonuses, while sales representatives (level 2) showed the opposite trend. These opposing trends could be due to role level, responsibilities, or

other factors; and may require further studies to conclude.



Further Insights:

More Leave, More Bonuses:

1. **Performance Reward:** People with more time off may perform exceptionally well, resulting in better sales and higher bonuses.
2. **Balance in Life:** Managing work-life balance effectively can increase motivation and productivity, leading to higher bonuses.
3. **Experience Matters:** Experienced employees may feel comfortable taking longer leave, leading to better sales and higher bonuses.
4. **Company Policies:** Company rules may encourage high-performing employees to use their entitled leave, contributing to higher bonuses.
5. **Job Satisfaction:** Encouraging vacation time can boost job satisfaction, leading to better performance and bonuses.

Managerial Roles:

1. **Less Leave:** European, Pacific, and North American Sales Managers took very little time off, suggesting they may be less inclined or able to take breaks.
2. **Lower Bonuses:** These managers received smaller bonuses, indicating a different pay structure compared to Sales Representatives.

Implications:

Role Demands:

Managerial positions may come with unique expectations, potentially limiting leave and performance-based bonuses.

Company Policies:

Policies around leave and bonus allocation may vary for managerial roles, affecting observed trends.

Summary

There is no clear relationship between Annual Leave taken and Bonuses **across the entire organisation.** Although there is a **positive correlation**, it is **not very strong (0.38**).A number of factors may be

responsible for the difference in indications for managers (level 3 who took less vacation and earned lower bonuses) and salesmen (level 2 who took more vacation and earned more bonuses). This may require further studies and analysis.

Question 3 – Aminat Ademola

What is the relationship between Country and Revenue?

Methodology

1. Loaded the Dataset, by exploring the Adventureworks2019 database using structure Queried Language
2. The Database was queried using SQL, to extract data related to country and revenue(sales)
3. Imported the necessary libraries (Pandas and matplotlib)
4. Established a database connection to SQL and PYTHON code using pyodbc
5. Defined and Executed the SQL Query as a CSV file and stored the result in a pandas Dataframe
6. Plotted the Data by using data visualization library i.e matplotlib to create visualization that shows the relationship between Country and Revenue (sales).

Comparing Sales - Last Year and This Year

SQL Statement

```
SELECT

    st.CountryRegionCode,
    SUM(st.SalesLastYear) AS TotalSalesLastYear,
    SUM(st.SalesYTD) AS TotalSalesYTD
FROM Sales.SalesTerritory AS st
GROUP BY st.CountryRegionCode
ORDER BY st.CountryRegionCode;
```

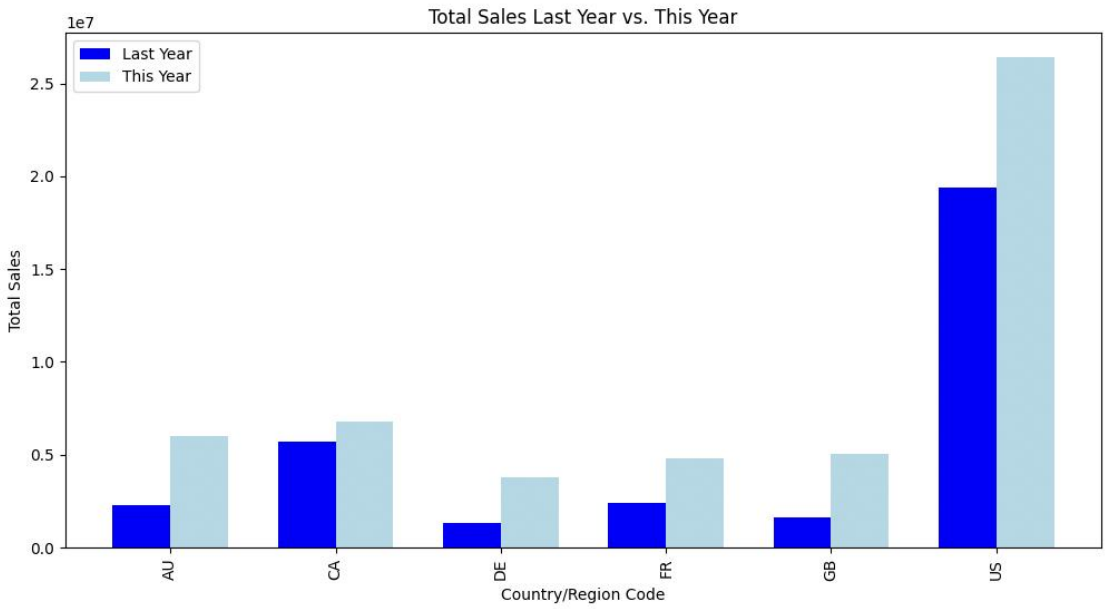
CountryRegionCode	TotalSalesLastYear	TotalSalesYTD
US	19402504.65	26411059.88
CA	5693988.86	6771829.138
FR	2396539.76	4772398.308
AU	2278548.978	5977814.915
GB	1635823.397	5012905.36
DE	1307949.792	3805202.348

Python Code

```
import pandas as pd
import matplotlib.pyplot as plt


# Read the CSV file and set column names
data = pd.read_csv('/Users/heshamfahiem/Downloads/Q3LastVsThis.csv', header=None,
names=["CountryRegionCode", "TotalSalesLastYear", "TotalSalesYTD"])

# Define the bar width and the x-axis locations for bars
bar_width = 0.35
x = range(len(data))
```

```
# Create a side-by-side bar chart
plt.figure(figsize=(12, 6))

plt.bar(x, data["TotalSalesLastYear"], bar_width, label='Last Year', color='blue')
plt.bar([i + bar_width for i in x], data["TotalSalesYTD"], bar_width, label='This Year',
color='lightblue')

plt.xlabel('Country/Region Code')
plt.ylabel('Total Sales')
plt.title('Total Sales Last Year vs. This Year')
plt.xticks([i + bar_width / 2 for i in x], data["CountryRegionCode"], rotation=90)
plt.legend()

plt.show()
```

Chart



This Year's sales

SQL Query

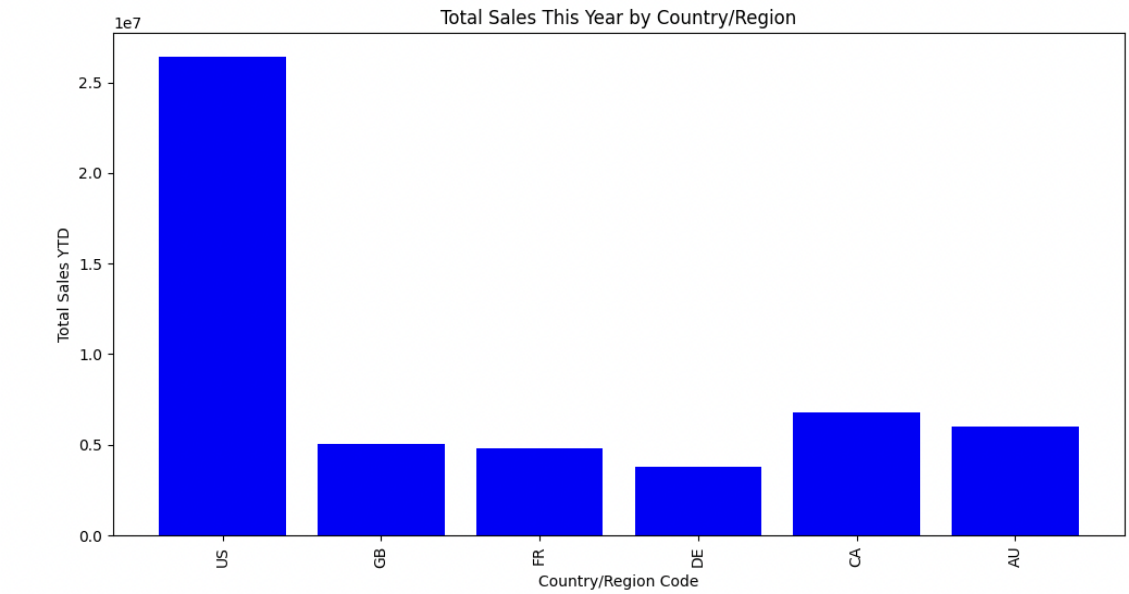
```
SELECT
    st.CountryRegionCode,
    SUM(st.SalesYTD) AS TotalSalesYTD
FROM Sales.SalesTerritory AS st
GROUP BY st.CountryRegionCode
ORDER BY st.CountryRegionCode DESC;
```

CountryRegionCode	TotalSalesYTD
US	26411059.88
CA	6771829.138
AU	5977814.915
GB	5012905.366
FR	4772398.308
DE	3805202.348

Python Code

```
import pandas as pd
import matplotlib.pyplot as plt

# Read the CSV file and set column names
data = pd.read_csv('/Users/heshamfahiem/Downloads/Q3Thisyear.csv', header=None,
names=["CountryRegionCode", "TotalSalesYTD"])

# Create a bar chart with blue bars
plt.figure(figsize=(12, 6))
plt.bar(data["CountryRegionCode"], data["TotalSalesYTD"], color='blue')
plt.xlabel('Country/Region Code')
plt.ylabel('Total Sales YTD')
plt.title('Total Sales This Year by Country/Region')
plt.xticks(rotation=90)
plt.show()
```

Chart



Percentage of sales – last year vs this year

SQL Statement

```
SELECT
    st.CountryRegionCode,
    SUM(st.SalesYTD) AS TotalSalesYTD,
    SUM(st.SalesLastYear) AS TotalSalesLastYear,
    SUM(st.SalesYTD) / grand_total.GrandTotalYTD * 100 AS SalesPercentageYTD,
    SUM(st.SalesLastYear) / grand_total.GrandTotalLastYear * 100 AS SalesPercentageLastYear
FROM Sales.SalesTerritory AS st
CROSS JOIN (
    SELECT SUM(SalesYTD) AS GrandTotalYTD, SUM(SalesLastYear) AS GrandTotalLastYear
    FROM Sales.SalesTerritory
) AS grand_total
GROUP BY st.CountryRegionCode, grand_total.GrandTotalYTD, grand_total.GrandTotalLastYear
ORDER BY SalesPercentageYTD DESC;
```

CountryRegionCode	TotalSalesYTD	TotalSalesLastYear	SalesPercentageYTD	SalesPercentageLastYear
US	26411059.88	19402504.65	50.06	59.3
CA	6771829.138	5693988.86	12.83	17.4
AU	5977814.915	2278548.978	11.33	6.96
GB	5012905.366	1635823.397	9.5	5
FR	4772398.308	2396539.76	9.04	7.32
DE	3805202.348	1307949.792	7.21	3.99

Python Code

```
import pandas as pd
import matplotlib.pyplot as plt

# Read the CSV file and set column names
data = pd.read_csv('/Users/heshamfahiem/Downloads/Q3lastYearPercentVsThisYear.csv',
header=None, names=["CountryRegionCode", "TotalSalesYTD", "TotalSalesLastYear",
"SalesPercentageYTD", "SalesPercentageLastYear"])
```

```
# Define the width of each bar
bar_width = 0.35

# Create a figure
plt.figure(figsize=(12, 6))

# Get the range of x values (country/region codes)
x = range(len(data))

# Create bars for this year's percentage
plt.bar(x, data["SalesPercentageYTD"], width=bar_width, color='blue', label='This Year')

# Create bars for last year's percentage next to the corresponding bars
plt.bar([i + bar_width for i in x], data["SalesPercentageLastYear"], width=bar_width,
color='lightblue', label='Last Year')

# Set the x-axis labels to be the country/region codes
plt.xticks([i + bar_width/2 for i in x], data["CountryRegionCode"], rotation=90)

plt.xlabel('Country/Region Code')
plt.ylabel('Sales Percentage')
plt.title('Sales Percentage Last Year vs. This Year by Country/Region')
plt.legend()
plt.show()
```

Chart

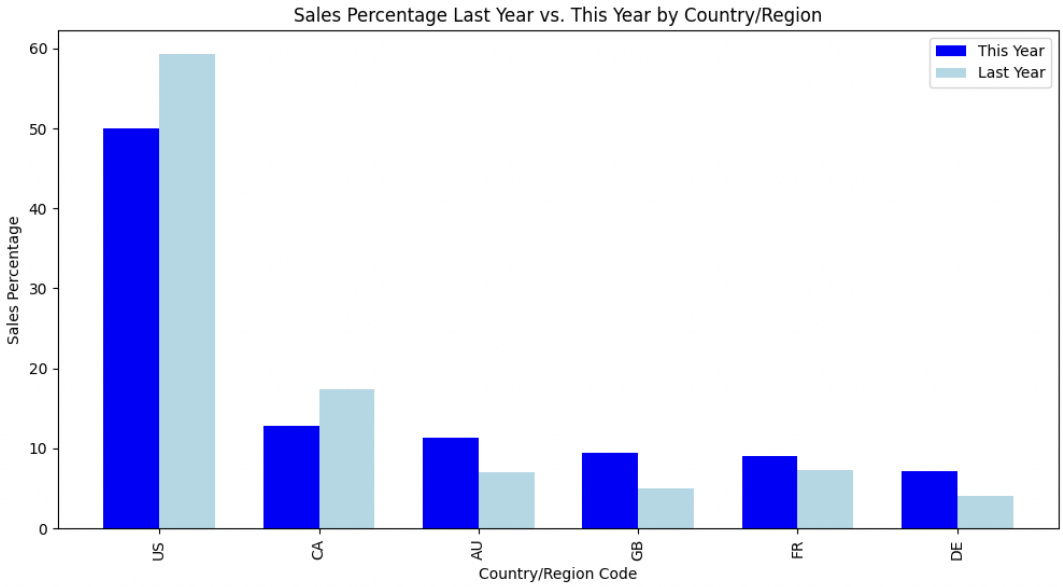


Chart Explanation:

Overall, the chart provides a visual representation of how different countries or regions are performing in terms of sales and their contribution to the grand total. The chart allows us to compare how each country/region's sales performance has changed from last year to this year. And we can easily identify which countries or regions have the highest and lowest percentages of sales for the current year (YTD) relative to the grand total YTD sales.

In this chart we can all easily identify countries or regions that has the highest and lowest percentage of sales last year compared to this year.

So, countries with bars that are taller on the right-hand side indicates significant growth in sales last year compared to this year while countries with bars that are taller on the left-hand side indicates significant growth in sales this year.

What this simply means is that every other country apart from the united state and Canada made a significant growth in sales percentage this year compared to last year. The United Kingdom for example went from 5% sales percentage last year to 9.5% sales percentage this year, the same goes to Germany that went from 3.99% sales percentage last year to 7.21% sales percentage this year. On the other hand, the united state and Canada made a significant growth in sales percentage last year compared to this year. This simple means that the united state sales percentage of 59.3% last year dropped to 50.06% this year and the same goes to Canada also which as a 17.4% sales percentage last year and 12.83% this year.

Countries with bars that are taller indicates significant growth in sales. According to the charts above, it can be seen that there is an increase in the total sales from the previous year to the current year for all the countries, and it can also be seen that the United States made the most sales in both years followed by Canada. However, the 3rd chart shows that the sales percentage in both the United States and Canada last year is greater than that of this year, while the sales percentage in Australia, United Kingdom, France, and Germany this year is greater than that of last year.

Comparing the chart, this simply means that even though there is an increase in the total sales for both US and Ca, there sales percentage is lower this year compared to last year, and the other countries sales percentage is higher this year compared to last year.

In conclusion, based on the above information, it is then safe to say that the United States generated the most Revenue in both years, followed by Canada. Even though the United States sales percentage reduces this year compared to last year, the fact still remains that the US generates the highest revenue. And this bring me to my question to us all, why do you think the United state generates the highest revenue, can it be attributed to their Market size, or could it be their Global Companies and diverse industries or their strong economy? However, it is essential to know that the reasons for high revenues in every country depends on its Unique economic and social context, and the relationship between a country and revenue is dynamic and influenced by a combination of these factors and more.

Question 4 – Hesham Fahiem

What is the relationship between sick leave and Job Title?

Initial Ideas/Thought Process

Job Titles with Health and Work-Life Balance Concerns:

By analysing the dataset, I want to pinpoint job titles or categories where employees tend to take more sick leave. This insight may suggest that certain roles or departments experience higher stress levels or workload, which could be addressed by adjusting workloads, offering additional support, or introducing health and wellness programs.

By identifying areas where sick leave is more common, management can take targeted actions to improve the overall well-being of employees, potentially leading to increased productivity and job satisfaction. Additionally, this analysis can help the organization optimize resource allocation and employee support programs to enhance workforce health and performance.

Methodology

Sick Leaves Grouped by Org Level

My SQL query is designed to retrieve and summarize sick leave data from a table called HumanResources.Employee. The goal is to group the data by the "OrganizationLevel" column and calculate the total sick leave hours for each organization level. The result is then ordered in descending order of the total sick leave hours.

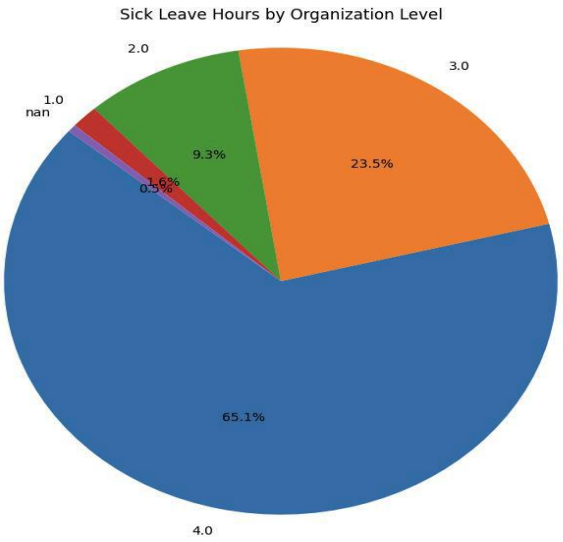
```
SELECT OrganizationLevel, SUM(SickLeaveHours) AS TotalSickLeaveHours
FROM HumanResources.Employee
GROUP BY OrganizationLevel
ORDER BY TotalSickLeaveHours desc;
```

Python Code

```
import pandas as pd

import matplotlib.pyplot as plt

# Load the data from the CSV file (assuming no column names)
data = pd.read_csv('/Users/heshamfahiem/Downloads/Q4GroupedByOrgLevel.csv', header=None,
names=['OrganizationLevel', 'TotalSickLeaveHours'])

# Create a pie chart
plt.figure(figsize=(8, 8))
plt.pie(data['TotalSickLeaveHours'], labels=data['OrganizationLevel'], autopct='%1.1f%%',
startangle=140)
plt.title('Sick Leave Hours by Organization Level')
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.

plt.show()
```

Chart Explanation

This chart below was generated using VSCODE with **pandas** and **matplotlib** packages.



Sick Leave Increases with Organizational Hierarchy:

The data shows that as you move up the organizational hierarchy from level 1 to level 4, the total sick leave hours tend to increase. Level 4, which typically represents higher-ranking job titles, has the highest sick leave hours.

Notable Impact at Higher Levels:

It's worth highlighting that the total sick leave hours at level 4 are significantly higher than at lower levels. This could be due to several factors, such as increased job responsibilities, stress, or a greater impact on work-life balance.

The top 10

The SQL query is designed to retrieve the top 10 job titles with the highest average sick leave hours for employees in the "HumanResources.Employee" table.

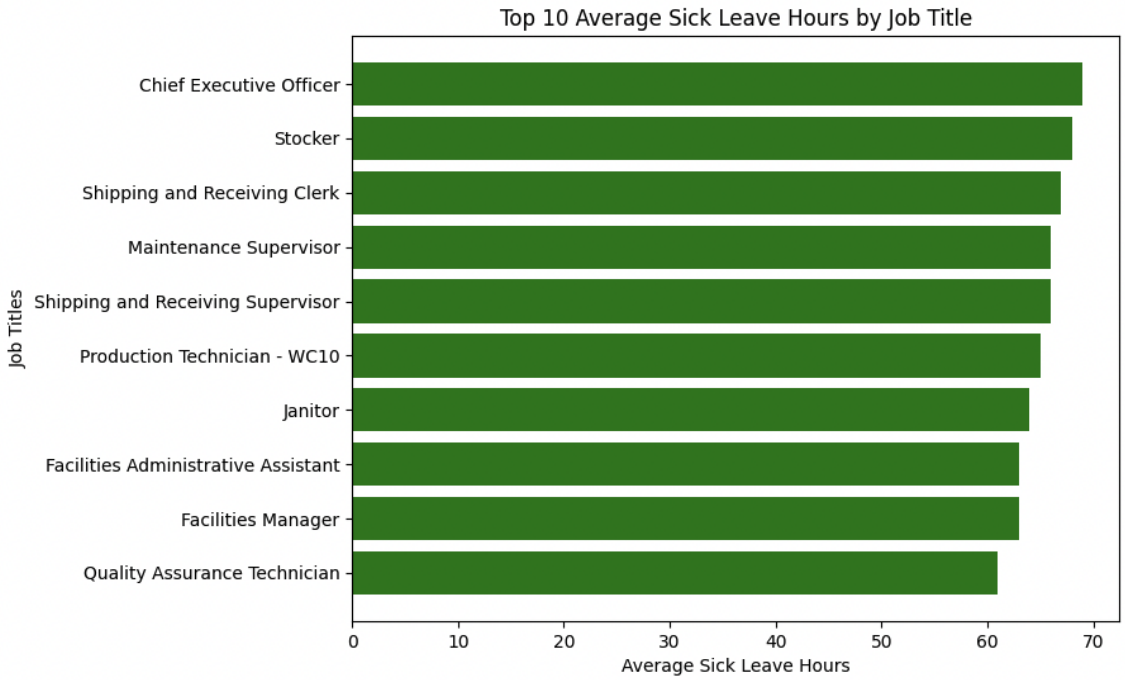
```
SELECT TOP 10 E.JobTitle, AVG(E.SickLeaveHours) AS AverageSickLeaveHours
FROM HumanResources.Employee AS E
GROUP BY E.JobTitle
ORDER BY AverageSickLeaveHours DESC;
```

Python Code

```
import pandas as pd
import matplotlib.pyplot as plt

# Load the data from the CSV file with custom column names
data = pd.read_csv('/Users/heshamfahiem/Downloads/avgAll.csv', header=None,
names=['JobTitle', 'AverageSickLeaveHours'])

# Select the top 30 rows
```

```
top_30_data = data.head(30)

# Set a single color (green)
color = 'green'

# Create a bar chart with the single color
plt.figure(figsize=(12, 6))
bars = plt.barh(top_30_data['JobTitle'], top_30_data['AverageSickLeaveHours'], color=color)
plt.xlabel('Average Sick Leave Hours')
plt.ylabel('Job Titles')
plt.title('Top 30 Average Sick Leave Hours by Job Title')
plt.gca().invert_yaxis()  # Invert the y-axis to show the highest at the top

plt.show()
```

Chart Explanation



Job Titles with High Sick Leave:

The Chief Executive Officer, Stocker, Shipping and Receiving Clerk, and Shipping and Receiving Supervisor have relatively high average sick leave hours. This suggests these roles may face more health-related challenges.

Management and Sick Leave:

Roles like Maintenance Supervisor and Production Technician - WC10 also have elevated average sick leave hours. This indicates that management positions and specific technical roles have notable sick leave patterns.

The Bottom 10

The query provides valuable insights into the average sick leave hours by job title, listed in ascending order:

```sql
SELECT TOP 10  E.JobTitle, AVG(E.SickLeaveHours) AS AverageSickLeaveHours
FROM HumanResources.Employee AS E
GROUP BY E.JobTitle
ORDER BY AverageSickLeaveHours ASC; -- Order in ascending order
```

Python Code

```python
import pandas as pd
import matplotlib.pyplot as plt

# Load the data from the CSV file with custom column names
data = pd.read_csv('/Users/heshamfahiem/Downloads/avgTop10.csv', header=None,
names=['JobTitle', 'AverageSickLeaveHours'])

# Create a bar chart with a single green color
plt.figure(figsize=(12, 6))
bars = plt.barh(data['JobTitle'], data['AverageSickLeaveHours'], color='green')
plt.xlabel('Average Sick Leave Hours')
plt.ylabel('Job Titles')
plt.title('Top 10 Average Sick Leave Hours by Job Title')
plt.gca().invert_yaxis()  # Invert the y-axis to show the highest at the top

plt.show()
```

Chart Explanation

Chief Financial Officer:

The Chief Financial Officer and Vice President of Engineering have the lowest average sick leave hours, indicating strong health and attendance.

Engineering Roles:

Roles such as Engineering Manager, Senior Design Engineer, and Design Engineer also have low average sick leave hours, suggesting a generally healthy workforce in engineering positions.

Technical and Design Roles:

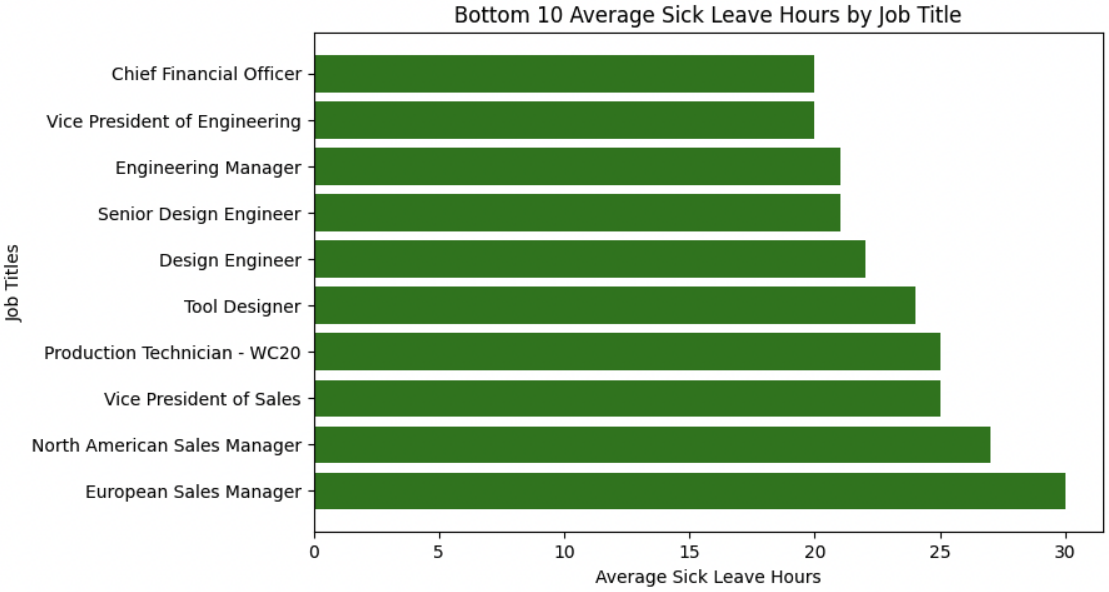
Tool Designer, Tool Designer, and Production Technician - WC20 are closely grouped, indicating they have similar average sick leave hours.

Sales and Management:

In the top 10, several roles in sales and management, such as Vice President of Sales, North American Sales Manager, and European Sales Manager, have relatively low average sick leave hours.

Consistent Trends:

The analysis suggests that technical and managerial positions, particularly in the engineering and sales departments, experience fewer sick leave hours, possibly due to lower physical demands or strong team leadership.


Bottom 10 Average Sick Leave Hours by Job Title
Chief Financial Officer
Vice President of Engineering
Engineering Manager
Senior Design Engineer
Design Engineer
Tool Designer
Production Technician - WC20
Vice President of Sales
North American Sales Manager
European Sales Manager
Job Titles
0
5
10
15
20
25
30
Average Sick Leave Hours

Question 5 – Adaobi Ikeasomba

What is the relationship between store trading duration and revenue?

The aim of this analysis is to understand how trading duration (the amount of time a store is open) and how it affects the revenue it generates, thereby determining the connection between these two variables.

Initial Ideas/Thought Process

At the initial consideration of the question, I assumed that there will be a direct correlation between a store's trading hours and the revenue. However, taking an analytical perspective, it became apparent that this assumption might not always hold true. To explore this further, I embarked on an extensive analysis that involved delving into the Adventure Works 2019 database to explore the dataset's potential for generating more comprehensive insights.

Methodology

Date Set Review

A dive in the Adventure works 19 -After reviewing the available datasets, I was drawn to investigate the **'Sales.vStoresWithDemographics'** table as it presented columns that held the information that would be helpful for my analysis.

Analysis Tools-

Adventure database, Azure data studio, Vscode

Graphical representation-

- Scattered plots
- Bart Chart,
- Heatmap

SQL Queries

```
SELECT TOP (1000) [BusinessEntityID]
      ,[Name]
      ,[AnnualSales]
      ,[AnnualRevenue]
      ,[BankName]
      ,[BusinessType]
      ,[YearOpened]
      ,[Specialty]
      ,[SquareFeet]
      ,[Brands]
      ,[Internet]
      ,[NumberEmployees]
  FROM [AdventureWorks2019].[Sales].[vStoreWithDemographics]
```

SQL QUERY EXPLANATION – the above query commands SQL to the extract and show the top 1000 rows from the table [Sales.vStoresWithDemographics]. This format is often used for reviewing a subset of data to understand its content and perform the initial analysis.
The query produced an extensive table comprising 408 rows, which I anticipate may lead to document clutter. Hence, I've included a link to access the tables in a separate tab for more convenient viewing.

https://d.docs.live.net/d090e6bfb75858be/Documentos/Results.xlsx

Python Code

```
# Import required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import pyodbc
from datetime import date
import seaborn as sns

# Database Connection
conn = pyodbc.connect('DRIVER={SQL Server};'
                      'SERVER=KA-HUAWEI;'
                      'DATABASE=AdventureWorks2019;')

# SQL Query for fetching data
# Assuming the SQL Views 'StoresRevenue' and 'SizeEmployeeRevenue' are already
created in the database
df = pd.read_sql_query('SELECT * FROM StoresRevenue', conn)

# Close the database connection
conn.close()

# Data Transformation
df['duration'] = date.today().year - df['YearOpened']

# Data Visualization
fig, ax = plt.subplots(2, 1, sharey=False, figsize=(17, 8))

# Plot 1: Relationship between Store trading duration and Average Revenue
ax[0].bar(df['duration'], df['Average_Revenue'], color='blue', alpha=0.6)
ax[0].set_xticks(df['duration'])
ax[0].invert_xaxis()
ax[0].set_yticklabels(['0K', '50K', '100K', '150K', '200K', '250K'])
ax[0].set_ylabel("Average Revenue ($)")
ax[0].set_xlabel('Time since Opening (Years)', fontsize=12)
```

```
ax[0].set_title('Relationship between Store Trading Duration and Average
Revenue', fontsize=15)

# Plot 2: Number of stores opened each year
ax[1].bar(df['YearOpened'], df['Num_of_Stores'], color='red', alpha=0.6)
ax[1].set_xticks(df['YearOpened'])
ax[1].set_ylabel("Number of Stores")
ax[1].set_xlabel('Opening Year', fontsize=12)

# Styling
plt.style.use('seaborn')

# Save plot
fig.savefig('Trading_Duration_and_Average_Revenue.png')

# Show plot
plt.show()
```

Charts

CHART ANALYSIS

AIM-To show the relationship between the number of stores, opening year and the average revenue generated, to identify any pattern or trends.

AXIS- The X axis shows the trading duration of the stores in years.

The Y axis shows the average revenue generates in (thousands and USD as currency).

Colours on the graph and markers- colour gradient and the sizes of the markets was used on this occasion to illustrate the number of stores that generated more revenue.

- The yellow marker shows the highest number of stores ranging from 30-33
- The green marker is the 2nd highest showing number of stores ranging from 25-29
- The Teal is the 3rd highest showing number of stores ranging 20-24
- The Blue is the 4th highest showing number of stores ranging 15-19
- The purple is the smallest showing number of stores ranging 10-14

INTERPRETATION- In this chart, fluctuations in outcomes are apparent, with the most prominent variances being;

1. Majority of the stores that have traded for shorter durations generated more revenue which has disputed my initial assumptions. This can be attributed to various factors such as;

- The new store hype -new stores often attracts and gains attentions of potential customers who are curious to explore.
- Location and market- stores in prime locations, growing neighbourhoods and underserved market can impact its revenue.
- Marketing and promotions- special offers, discounts, and marketing campaigns can drive higher sales initially.
- Product or service differentiation-If the new store offers unique products, services, or experiences not readily available in the area, it can attract a dedicated customer base.
- Local competition-new stores can enter markets with less competition and capture a larger share of the local customer base.
- Market saturation-In some markets, there may be a saturation of established stores, making it challenging for them to grow further. New entrants may have a better chance to thrive.
- Seasonal factors-: The time of year when a store open can impact its initial revenue
- Size and layout of store -smaller stores or stores with a well-planned layout can create a perception of exclusivity and may lead to higher spending per customer.

2. The oldest stores that had traded for 50 years generate the least revenue which logically will not make sense, but research shows that a lot if factors can contribute to this. These factors includes;

- **Market Saturation**: In mature markets or areas with a high density of stores, competition can be intense. Established stores may struggle to capture a larger share of the market due to market saturation.
- **Changing Consumer Preferences**: Over time, consumer preferences can evolve. Established stores may have difficulty adapting to changing trends and meeting evolving customer demands.
- **Lack of Innovation**: Failure to innovate and introduce new products, services, or retail experiences can lead to a decline in revenue. Established stores may become complacent and resist change.
- **Competition**: Increased competition, both from other brick-and-mortar stores and e-commerce businesses, can erode the customer base and revenue of established stores.
- **Economic Factors**: Economic downturns or recessions can impact consumer spending. Established stores may experience decreased sales during economic crises.
- **Brand Image and Reputation**: If an established store has faced negative publicity, customer trust and loyalty may be compromised, leading to lower revenue.
- **Operational Inefficiencies**: Poor inventory management, supply chain issues, or operational inefficiencies can lead to higher costs and lower profits.
- **Location**: Changes in the local area, such as shifts in population or traffic patterns, can affect store revenue. Established stores may struggle if their location becomes less favourable.
- **Aging Infrastructure**: The physical condition of the store and its facilities can influence customer experience. Neglected or outdated infrastructure may deter customers.
- **Declining Customer Base**: An established store may experience a decline in its customer base as demographics change or as younger consumers seek out newer, more innovative shopping experiences.
- **Failure to Adapt to Technology**: The failure to leverage technology for e-commerce, online marketing, or supply chain management can hinder the competitiveness of older stores.
- **Poor Customer Service**: A decline in the quality of customer service or customer support can drive customers away and negatively impact revenue.
- **Ineffective Marketing**: Established stores may struggle with marketing and advertising efforts, making it difficult to reach and attract new customers.
- **Internal Issues**: Internal conflicts, management problems, or a lack of employee motivation can affect store performance and customer satisfaction.

PYTHON CODE

```
# Import required libraries
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import pyodbc
from datetime import date
import seaborn as sns

# Database Connection
conn = pyodbc.connect('DRIVER={SQL Server};'
                      'SERVER=KA-HUAWEI;'
                      'DATABASE=AdventureWorks2019;')

# SQL Query for fetching data
# Assuming the SQL Views 'StoresRevenue' and 'SizeEmployeeRevenue' are already
created in the database
df = pd.read_sql_query('SELECT * FROM StoresRevenue', conn)

# Close the database connection
conn.close()

# Data Transformation
df['duration'] = date.today().year - df['YearOpened']

# Data Visualization
fig, ax = plt.subplots(2, 1, sharey=False, figsize=(17, 8))

# Plot 1: Relationship between Store trading duration and Average Revenue
ax[0].bar(df['duration'], df['Average_Revenue'], color='blue', alpha=0.6)
ax[0].set_xticks(df['duration'])
ax[0].invert_xaxis()
ax[0].set_yticklabels(['0K', '50K', '100K', '150K', '200K', '250K'])
ax[0].set_ylabel("Average Revenue ($)")
ax[0].set_xlabel('Time since Opening (Years)', fontsize=12)
ax[0].set_title('Relationship between Store Trading Duration and Average
Revenue', fontsize=15)

# Plot 2: Number of stores opened each year
ax[1].bar(df['YearOpened'], df['Num_of_Stores'], color='red', alpha=0.6)
ax[1].set_xticks(df['YearOpened'])
ax[1].set_ylabel("Number of Stores")
ax[1].set_xlabel('Opening Year', fontsize=12)

# Styling
plt.style.use('seaborn')

# Save plot
fig.savefig('Trading_Duration_and_Average_Revenue.png')
```

```
# Show plot
plt.show()
```

TO view tables
https://d.docs.live.net/d090e6bfb75858be/Documentos/

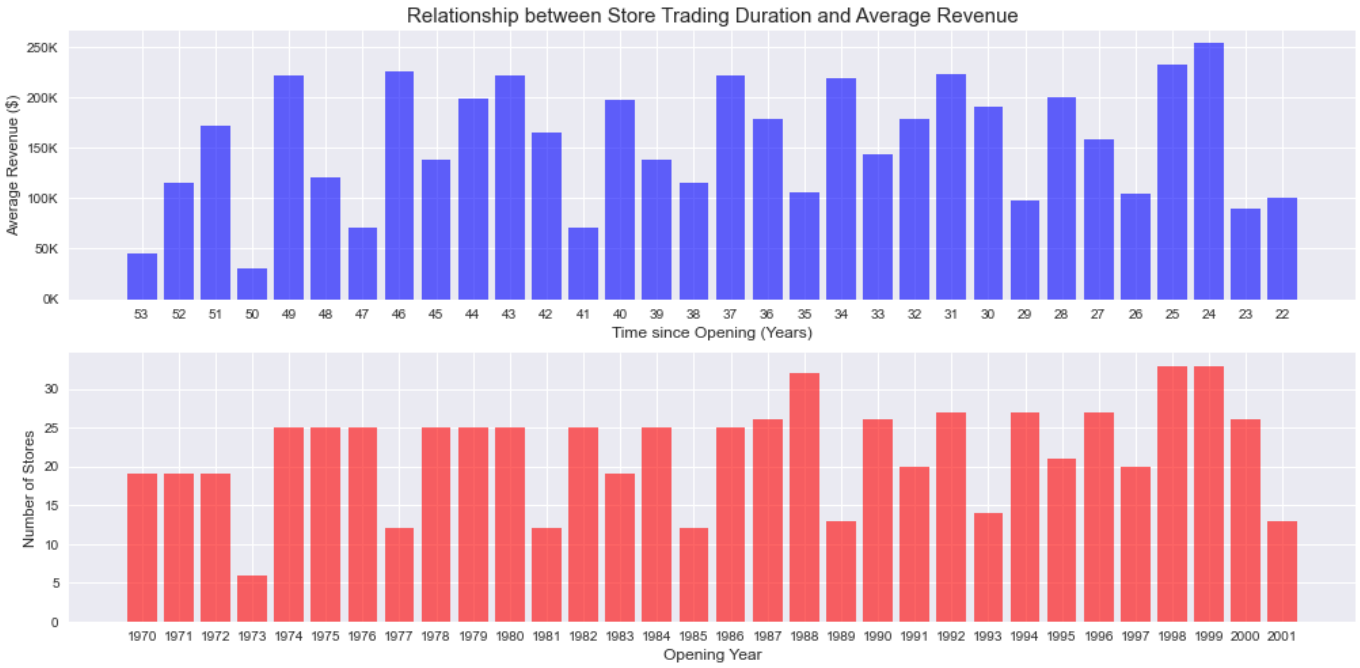


CHART ANALYSIS

I took it a step further to enhance My understanding and create a clear visual comparison between the average revenue throughout the trading duration and the number of stores over the opening years. For this analysis, I opted for a bar chart, employing the colours blue and red to represent the data.

- The first bar chart in blue which shows relationship between trading store duration and the average revenue hasn't shown any distinct pattern or trend as to the increase and decrease of the stores. To gain a more comprehensive understanding, it is necessary to review the annual reports of the stores to pinpoint and draw conclusive conclusions.

- The second bar chart in red shows Number of stores and the Opening years. This shows a pattern where an average no of stores (1970-1989) where opened after every 3 years following a decline in the fourth years as seen in 1973-6 stores,1977-12 stores,1981-12 stores,1985-12 stores, 1989-14 stores compared to the average of 19 stores over the other years. This pattern changed after 1989 we can now see a decline in the fourth years as in 1993,1997 and 2001.

PYTHON CODE

```
# Import required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import pyodbc
from datetime import date
import seaborn as sns

# Sample data for demonstration (Replace this with your actual SQL data)
df = pd.DataFrame({
    'YearOpened': [2000, 2001, 2002, 2003],
    'Average_Revenue': [50000, 60000, 55000, 70000],
    'Num_of_Stores': [5, 6, 4, 7]
})

EmployeeSizeRevenue = pd.DataFrame({
    'Size': [1000, 1500, 2000, 2500],
    'Average_Revenue': [60000, 55000, 70000, 65000],
    'Avg_num_employees': [10, 12, 11, 13]
})

# Data Transformation
df['duration'] = date.today().year - df['YearOpened']

# Calculate the correlation matrix
corr_matrix = df.corr()

# Generate a heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.title('Heatmap of Correlation Between Variables')
plt.show()
```

CHART

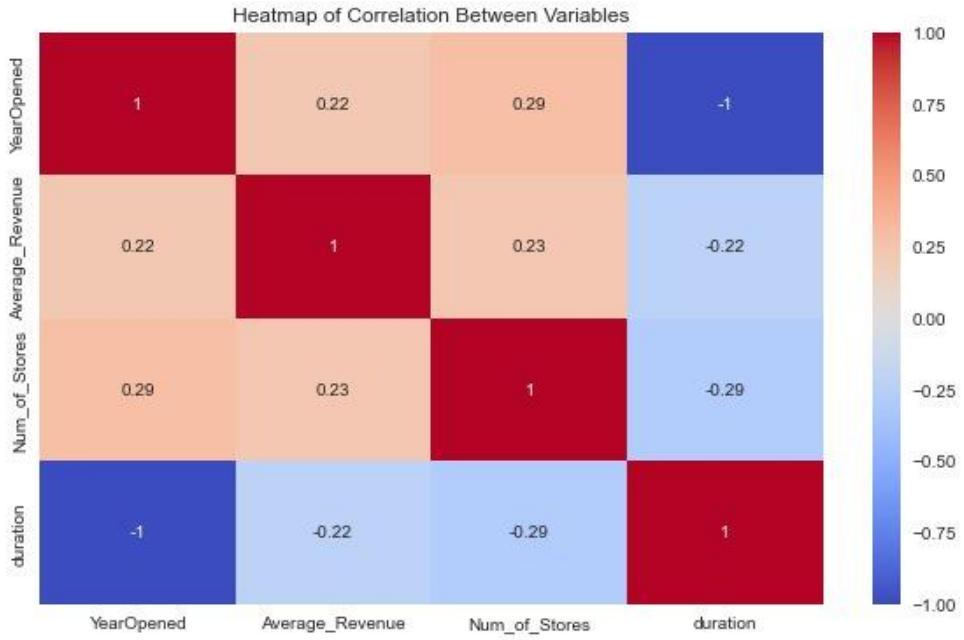
To conclude this analysis, I decided to generate a heatmap that shows the **Correlation matrix** to provide a comprehensive overview of the correlation coefficients between the multiple variables (Year opened, Average revenue, Number of stores, Trading duration), the coefficient result ranged from -0.22 to 1. It is a fundamental in this statistics and data analysis, particularly because we had a multivariate analysis, to understand the relationships between variables.

ANALYSIS-

Each variables correlation with others is depicted in a matrix using **cool warm** colours to indicate the strength and direction of these relationships.

- **Year Opened VS Duration** - This is represented by the deep blue colour, indicating a perfect negative correlation of (-1). This demonstrates that as the number of years of operation increased the trading duration also increased correspondingly.
- **Year Opened VS Number of stores**- represented with the neutral colour correlation of 0.29 indicates a positive, but relatively weak, linear relationship between these two variables.
- **Year Opened VS Average Revenue**- represented in lighter neutral shade, a correlation of (0.22) suggests a weak positive relationship between these two variables. As newer stores were opened, the average generated revenue also increased but at a slower rate.
- **Duration VS Number of stores**- represented in the lighter blue, (-0.29) indicates a weak negative linear relationship between these two variables.
- **Duration VS Average Revenue** indicated in light blue has a negative correlation of (-0.22) which suggests the opposite, where an increase in trading duration is associated with a decrease in average revenue but not proportionate.
- **Number of Stores VS Average Revenue**- indicated in a lighter neutral shade has positive correlation (0.23) which indicates that as the number of stores increases, the average revenue also tends to increase.
- **Variable VS Itself**- represented in dark red, the correlation of a variable with itself is always 1, as it represents a perfect positive linear relationship. This is because there is no variability in the relationship since the variable is compared to itself. In statistical terms, this is known as an auto-correlation or a variable's auto-correlation with itself.

HEATMAP



Question 6 – Kristian Ambruch

What is the relationship between the size of the stores, number of employees and revenue?

Initial Ideas/Thought Process

Problem Identification

Upon initial review of the AdventureWorks2019 database, I deduced that potentially valuable insights could be extracted about store sizes, the number of employees, and generated revenues. Using my previous limited experience with the database, I attempted to decide which tables and query types I could use to explore these metrics.

Data Exploration

My starting point involved identifying the relevant tables within the database that could serve as data sources for the chosen variables. The tables considered were:

- HumanResources.Employee: For obtaining employee data related to each store.
- Sales.Store: To gather details about the size of each store.
- Sales.SalesOrderHeader and Sales.SalesOrderDetail: For collating revenue and sales information.

SQL Query Design

I planned to employ SQL queries leveraging JOIN operations to merge data from these disparate tables, providing a unified dataset for python analysis. The SQL queries would use aggregation functions, 'GROUP BY' clauses, and calculated fields to transform the raw data into actionable insights. For example, using 'SUM' and 'COUNT' functions would allow for the calculation of total revenue and the number of employees for each store, respectively.

Data Visualization

My initial visualization idea was to use a scatterplot to showcase the relationship between the number of employees and store revenue. The plot would be enriched by encoding additional information—store size—in both the colour and size of each scatter point. This multi-dimensional representation would show a layered understanding of how these three key variables interact.

Data Analysis Tools

To execute this plan, I considered using several Python libraries tailored for data manipulation and visualization:

- Pandas: For data manipulation and preparation, given its robust capabilities in handling DataFrames.
- Matplotlib: To conduct preliminary statistical analyses and create basic visualizations.
- Seaborn: For advanced visualization techniques, including the scatterplot that would serve as the centrepiece of my analysis.

Correlation Analysis

An additional analytical step would involve calculating the correlation coefficients between these variables. This would provide a quantitative measure of how strongly these variables are related, offering a statistical basis for any observed trends.

Methodology

Analysis – Initial Attempt

The SQL script shown below constitutes our initial attempt at approximating store size through strategic computation. Additionally, it incorporates variables such as revenue and employee count to enable subsequent data transformation and visualization via Python.

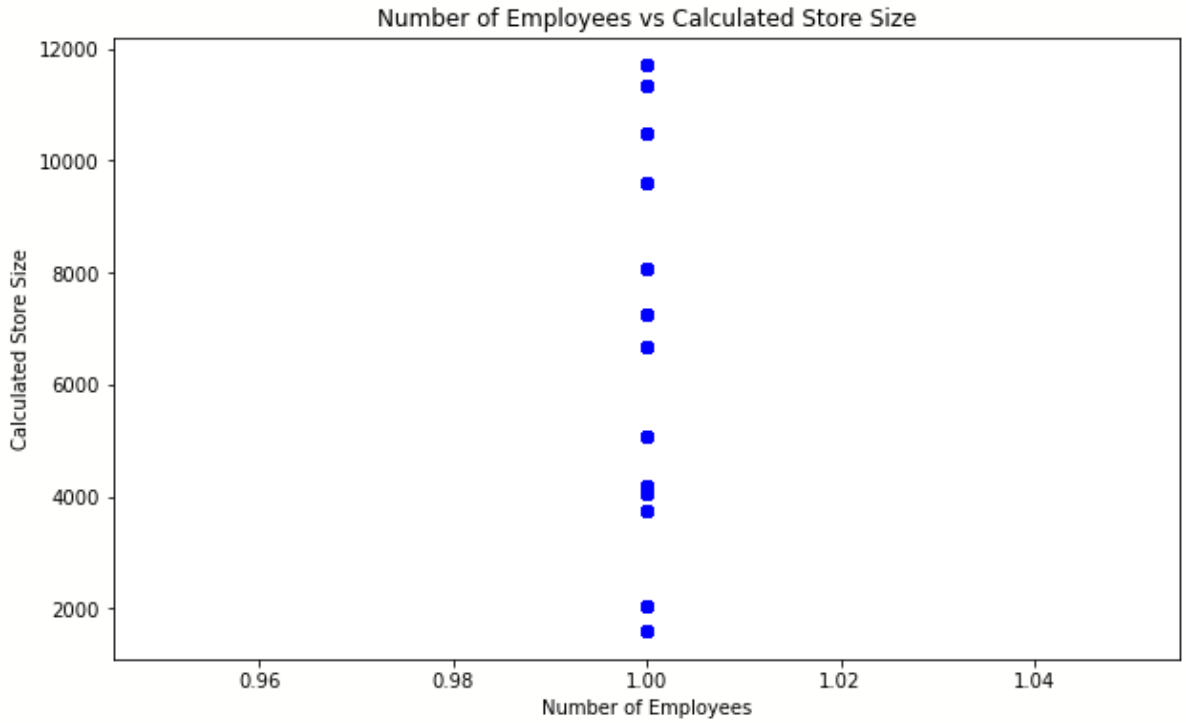
```
---------------------------------------------------------------------------------------------------------------------------------------
SELECT

    s.BusinessEntityID AS StoreID,
    s.Name AS StoreName,
    COUNT(DISTINCT e.BusinessEntityID) AS NumberOfEmployees,
    SUM(CAST(soh.TotalDue AS FLOAT)) AS TotalRevenue,
    COUNT(DISTINCT e.BusinessEntityID) + SUM(CAST(soh.TotalDue AS FLOAT)) / 1000 AS
CalculatedStoreSize
FROM
    Sales.Store AS s
JOIN
    Sales.SalesPerson AS sp ON s.SalesPersonID = sp.BusinessEntityID
JOIN
    HumanResources.Employee AS e ON sp.BusinessEntityID = e.BusinessEntityID  -- This join
assumes that each SalesPerson is also an Employee
JOIN
    Sales.SalesOrderHeader AS soh ON s.SalesPersonID = soh.SalesPersonID
GROUP BY
    s.BusinessEntityID, s.Name
ORDER BY
    CalculatedStoreSize DESC;
---------------------------------------------------------------------------------------------------------------------------------------
```

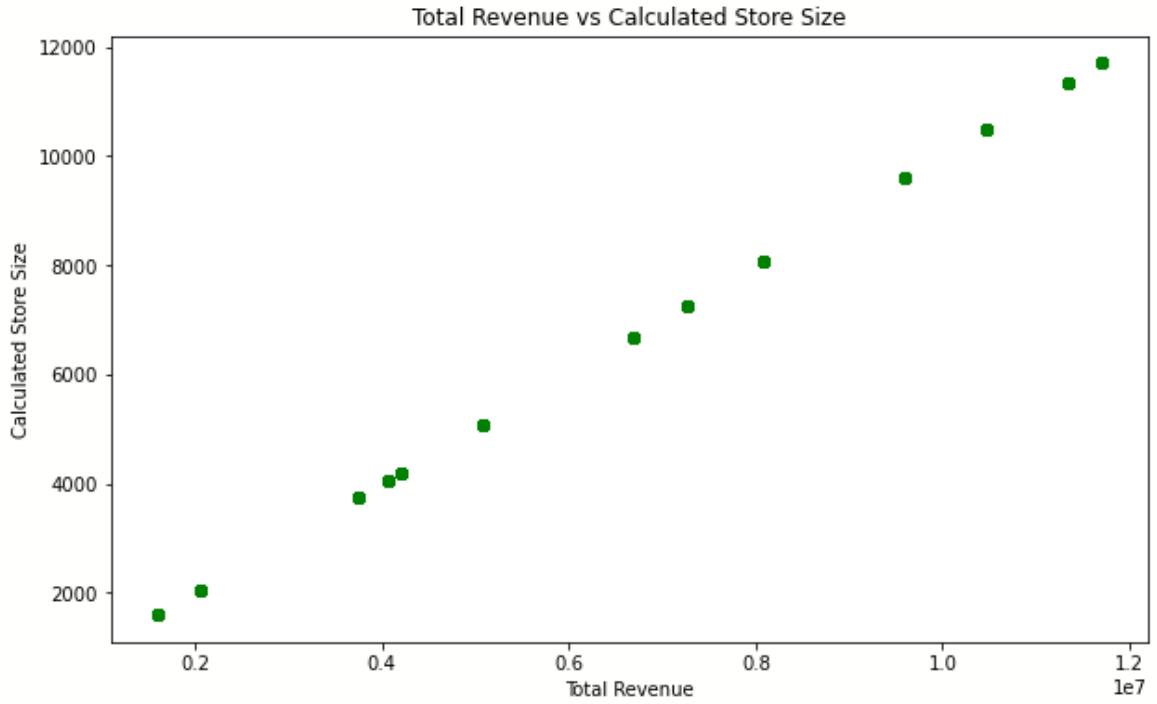
The corresponding Python code was engineered to generate initial scatter plots for exploratory analysis. The workflow was as follows: establish a database connection to the AdventureWorks server, execute the SQL query, populate a dataframe, and subsequently close the SQL connection. Scatter plots were then rendered, with visual elements meticulously designed for clarity and impact, including axis labels and titles.

```
---------------------------------------------------------------------------------------------------------------------------------------
import pandas as pd

import matplotlib.pyplot as plt
import pyodbc

# Establish a connection to the SQL Server
conn = pyodbc.connect('DRIVER={SQL Server};'
                      'SERVER= server_name;'
                      'DATABASE= database_name;'

# SQL query
sql_query = """INSERT ABOVE SQL QUERY HERE"""

# Execute the SQL query and load into a DataFrame
```

```
df = pd.read_sql_query(sql_query, conn)

# Close the SQL connection
conn.close()

# Scatter plot for Number of Employees vs Calculated Store Size
plt.figure(figsize=(10, 6))
plt.scatter(df['NumberOfEmployees'], df['CalculatedStoreSize'], c='blue')
plt.xlabel('Number of Employees')
plt.ylabel('Calculated Store Size')
plt.title('Number of Employees vs Calculated Store Size')
plt.show()

# Scatter plot for Total Revenue vs Calculated Store Size
plt.figure(figsize=(10, 6))
plt.scatter(df['TotalRevenue'], df['CalculatedStoreSize'], c='green')
plt.xlabel('Total Revenue')
plt.ylabel('Calculated Store Size')
plt.title('Total Revenue vs Calculated Store Size')
plt.show()
```

The following two scatter plots are the initial exploratory plots designed to enable me to understand the trends between the studied variables.





Upon running the initial code and analysing the scatter plots, it became evident that the data wasn't aligning as expected. The variables showed a near-perfect correlation, an outcome that warranted further investigation.

This led me to scrutinize the use of the 'BusinessEntityID' columns from different tables. Initially, I had assumed that these IDs were specific to stores; however, they turned out to be linked to individual employees, which skewed the entire analysis.

Recognizing this error prompted me to seek additional guidance. Resources like Google and specialized platforms such as ChatGPT proved invaluable at this juncture. It was through this extended research that I discovered the 'Sales.vStoreWithDemographics' view in the AdventureWorks database, which turned out to be instrumental in resolving the analytical challenges at hand. The significance of this find will be elaborated upon in the subsequent section.

Analysis – Final Code and Charts

SQL Query

Having found the view I required in order to answer this question, I proceeded to create a new view containing the three studied variables: Size of store (SquareFeet), Store Revenue (AnnualRevenue), and Number of employees (NumberEmployees). The subsequent SQL query was then used in all python codes for the generation of various charts in order to help answer this question. The results of the query are sorted by store size for easier viewing.

```
--------------------------------------------------------------------------------------
        CREATE VIEW SizeEmployeeRevenue AS
```

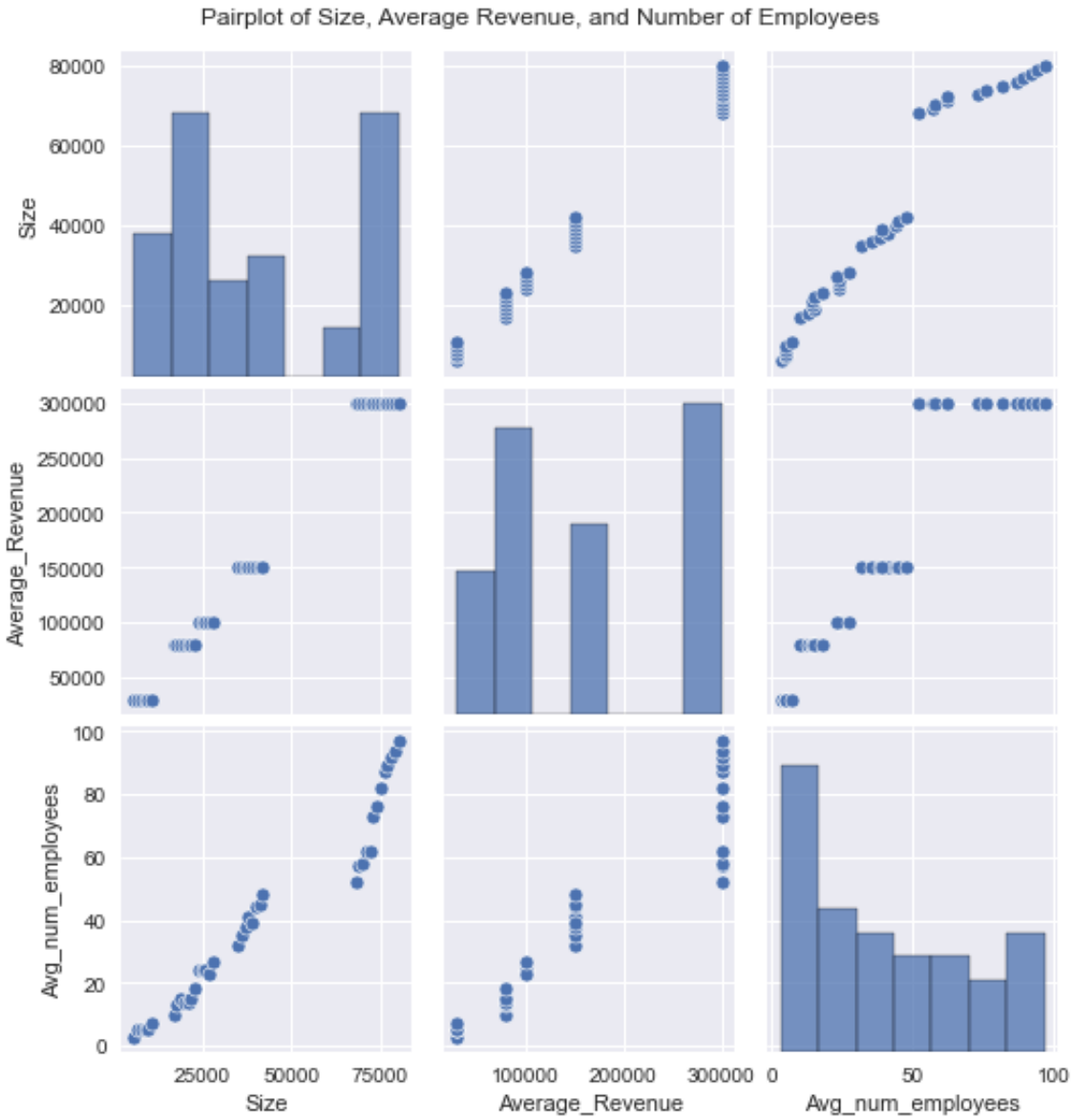
```
    SELECT  SquareFeet AS Size, AVG(AnnualRevenue) AS Average_Revenue, AVG(NumberEmployees)
AS Avg_num_employees
    FROM Sales.vStorewithDemographics
    GROUP BY SquareFeet
```

Pairplot

After successfully obtaining the required data through SQL, the next step was to delve into some exploratory analysis. A pairplot was an ideal starting point for this, allowing for the visualization of the relationships between all pairs of features in the data. The pairplot provides histograms on its diagonal to show the distribution of single variables and scatter plots on its off-diagonal to indicate the relationships between pairs of variables. This form of visualization can be incredibly informative, laying the groundwork for more advanced analyses.

```
# Import required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import pyodbc
import seaborn as sns

# Connect to the SQL Server
conn = pyodbc.connect('DRIVER={SQL Server};'
                      'SERVER=KA-HUAWEI;'
                      'DATABASE=AdventureWorks2019;')

# Fetch data from SQL into a DataFrame
EmployeeSizeRevenue = pd.read_sql_query('select * from SizeEmployeeRevenue', conn)

# Close the SQL connection
conn.close()

# Create pairplot using Seaborn
sns.pairplot(EmployeeSizeRevenue, kind='scatter', diag_kind='hist')
plt.suptitle('Pairplot of Size, Average Revenue, and Number of Employees', y=1.02)  # Add a
main title
plt.show()

# Calculate and print the correlation matrix
correlation_matrix = EmployeeSizeRevenue.corr()
print("Correlation Matrix:")
print(correlation_matrix)
```

The pairplot below was rendered using Seaborn's default settings, which offer a clean and straightforward look. The scatter plots in the off-diagonals were kept small enough to fit multiple plots in one view but large enough to clearly see the data points and any trends or clusters. This choice of simplicity and contrast made the pairplot efficient for quick interpretation during a presentation.



Line Charts

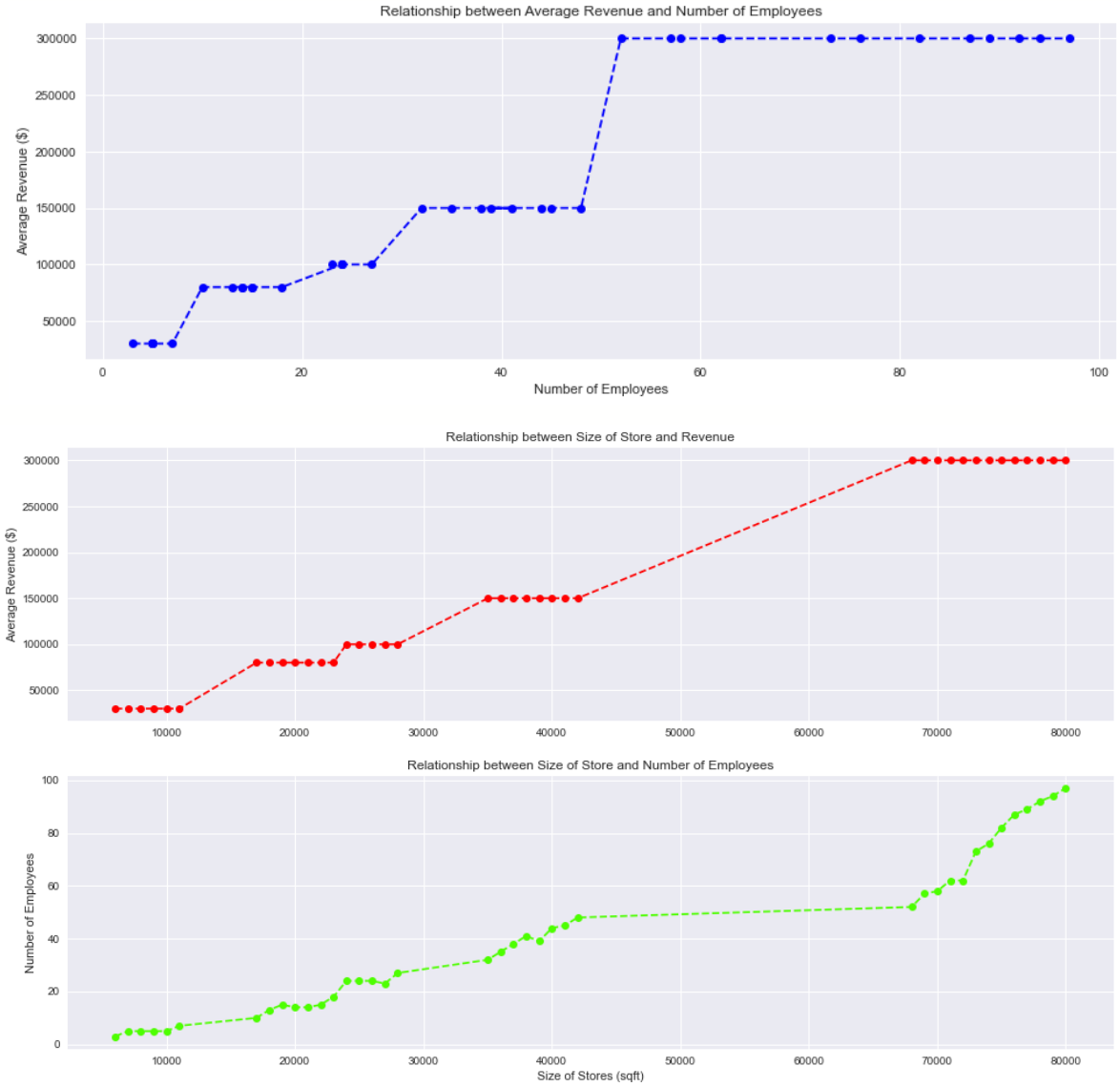
Line charts were then created to analyse the relationships between 'Average Revenue' and 'Number of Employees,' as well as 'Size of Store' and 'Revenue.' These charts help in identifying trends and patterns in the data. The use of dashed lines and markers aids in data point visibility and understanding the distribution of data. The colour choices were made to ensure maximum visibility while keeping the visual presentation clean and straightforward.

```
---------------------------------------------------------------------------------------------
# Import required libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import pyodbc
import seaborn as sns
```

```
# Establish a connection to the SQL Server
conn = pyodbc.connect('DRIVER={SQL Server};'
                      'SERVER=KA-HUAWEI;'
                      'DATABASE=AdventureWorks2019;')

# Fetch data from SQL into a DataFrame
EmployeeSizeRevenue = pd.read_sql_query('SELECT * FROM SizeEmployeeRevenue', conn)

# Close the SQL connection
conn.close()

# Plot 1: Relationship between Average Revenue and Number of Employees
fig, ax1 = plt.subplots(figsize=(15, 5))
ax1.plot(EmployeeSizeRevenue['Avg_num_employees'], EmployeeSizeRevenue['Average_Revenue'],
         color='blue', linestyle='--', marker='o')
ax1.set_ylabel('Average Revenue ($)')
ax1.set_xlabel('Number of Employees')
ax1.set_title('Relationship between Average Revenue and Number of Employees')
fig.savefig('Relationship_between_Average_Revenue_and_Number_of_Employees.png')

# Correlation between Avg_num_employees and Average_Revenue
correlation1 = np.corrcoef(EmployeeSizeRevenue['Avg_num_employees'],
EmployeeSizeRevenue['Average_Revenue'])
print(f"Correlation between Average Revenue and Number of Employees: {correlation1}")

# Plot 2: Size vs Revenue and Size vs Number of Employees
fig, (ax2, ax3) = plt.subplots(2, 1, figsize=(17, 10))

# Subplot 1: Size vs Revenue
ax2.plot(EmployeeSizeRevenue['Size'], EmployeeSizeRevenue['Average_Revenue'],
         color='red', linestyle='--', marker='o')
ax2.set_ylabel("Average Revenue ($)")
ax2.set_title('Relationship between Size of Store and Revenue')

# Subplot 2: Size vs Number of Employees
ax3.plot(EmployeeSizeRevenue['Size'], EmployeeSizeRevenue['Avg_num_employees'],
         color=(0.3, 1, 0), linestyle='--', marker='o')
ax3.set_ylabel("Number of Employees")
ax3.set_xlabel('Size of Stores (sqft)')
ax3.set_title('Relationship between Size of Store and Number of Employees')

# Styling and Saving the Plot
plt.style.use('seaborn')
fig.savefig('Store_size_vs_Revenue_and_Number_of_Employees.png')

# Correlations for Plot 2
correlation2 = np.corrcoef(EmployeeSizeRevenue['Avg_num_employees'],
EmployeeSizeRevenue['Size'])
correlation3 = np.corrcoef(EmployeeSizeRevenue['Average_Revenue'],
EmployeeSizeRevenue['Size'])
print(f"Correlation between Size and Number of Employees: {correlation2}")
print(f"Correlation between Size and Average Revenue: {correlation3}")
```

The line charts below were generated with a figsize of (15, 5) for clarity and visibility during the presentation. Dashed lines were used to make the lines easily distinguishable, and markers ('o'

shaped) were added at each data point for better visibility. The choice of blue for one plot, red for the next and green for the last was deliberate to ensure they could be easily told apart while remaining colourblind-friendly. These design choices aimed to make the line charts as informative as possible without overwhelming the audience.



Correlation Heatmap

The next logical step was to quantify the relationships we've been observing in our plots. A correlation heatmap serves this purpose well, offering a quick snapshot of how variables relate to each other. The heatmap was generated using Seaborn, a Python data visualization library based on Matplotlib. It provides an interface for drawing attractive and informative statistical graphics.

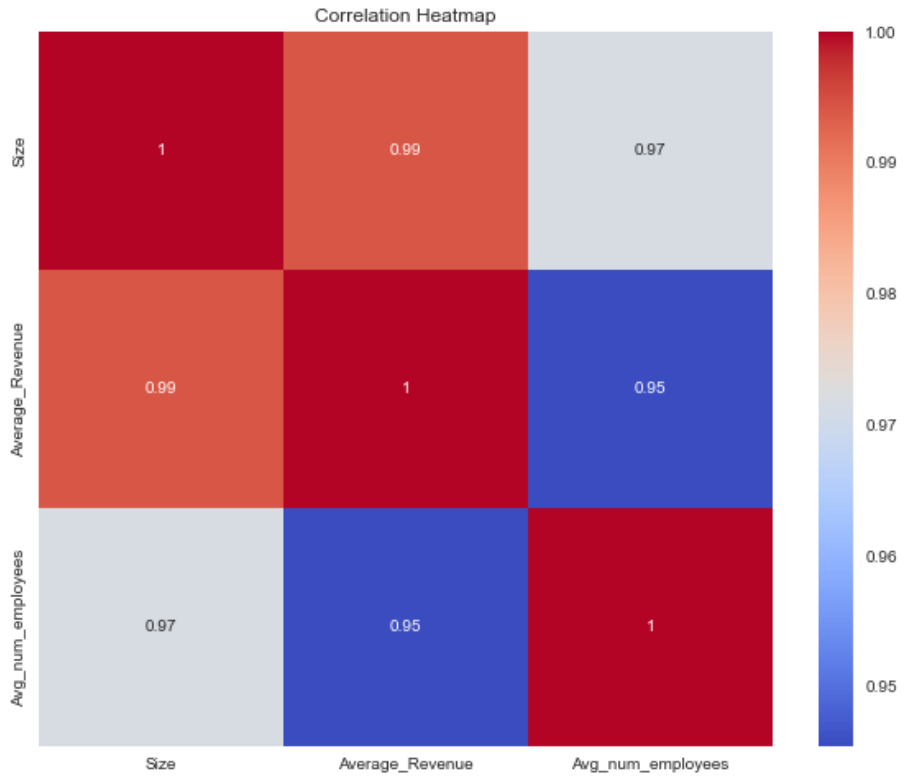
```
-----------------------------------------------------------------------------------------
# Import required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import pyodbc
```

```
import seaborn as sns

# Connect to the SQL Server
conn = pyodbc.connect('DRIVER={SQL Server};'
                      'SERVER=KA-HUAWEI;'
                      'DATABASE=AdventureWorks2019;')

# Fetch data from SQL into a DataFrame
EmployeeSizeRevenue = pd.read_sql_query('SELECT * FROM SizeEmployeeRevenue', conn)

# Close the SQL connection
conn.close()

# Create a heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(EmployeeSizeRevenue.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.savefig('Correlation_Heatmap.png')
plt.show()
```

The correlation matrix below provides a quantified overview of the relationships between store size, average revenue, and the average number of employees. All variables show strong positive correlations, with coefficients ranging from 0.945 to 1.000. Specifically, store size and average revenue share an almost perfect correlation of 0.994, suggesting that larger stores generally generate higher revenue. Similarly, the number of employees is highly correlated with both store size and average revenue, with coefficients of 0.972 and 0.945 respectively. These high correlation values indicate a strong linear relationship between the variables, supporting the need for further investigation into causality.

```
Correlation Matrix:
                        Size  Average_Revenue  Avg_num_employees
Size                1.000000         0.994207           0.971656
Average_Revenue     0.994207         1.000000           0.945457
Avg_num_employees   0.971656         0.945457           1.000000
```

The heatmap below was generated with a figsize of (10, 8), making it large enough to read clearly but not too large to distract. The 'coolwarm' colour palette was used to provide an easy-to-understand gradient from negative to positive correlations. Annotations were added to each cell to provide the exact correlation value, allowing for both quick visual interpretation and more detailed analysis if needed. This was especially useful for a presentation setting where time for interpretation is limited.



Unused Charts and Visualisations

While several visualizations were experimented with during this analysis, not all made it to the final presentation. The key factor in this selection was the informative value of the chart, weighed against its complexity and the time required to interpret it. Time restrictions also played an important role in determining which charts were used and which were not.

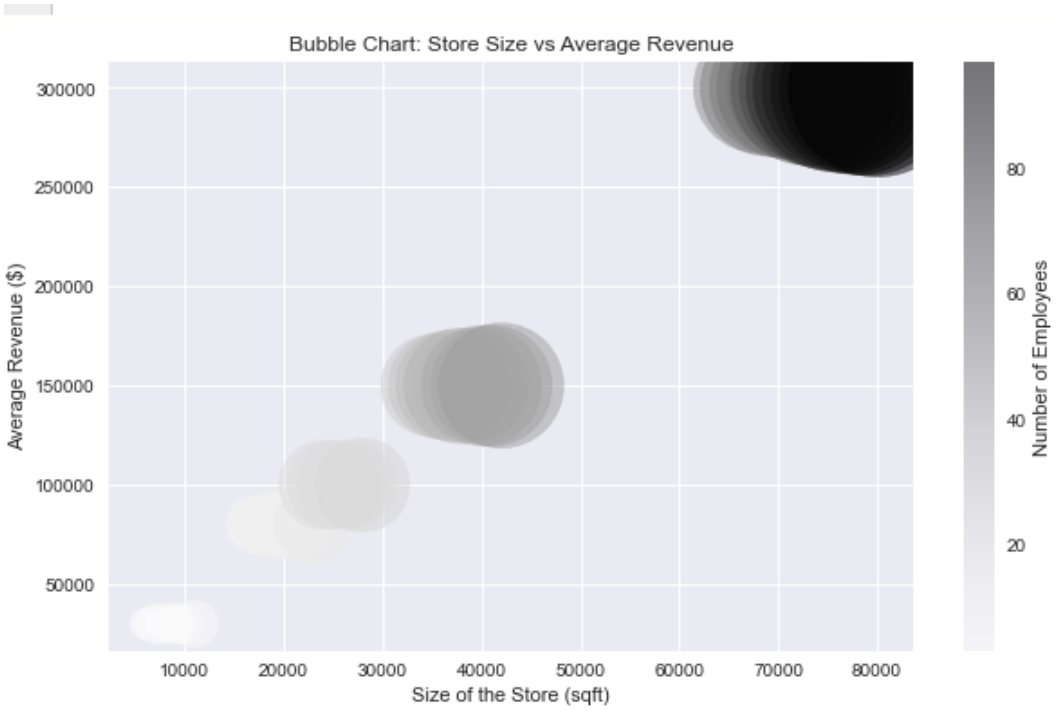
Bubble Chart

A Bubble Chart was considered as a possible visualization technique. The chart was designed so that the size of each bubble represented the number of employees, while its position indicated the store size and average revenue. Despite its aesthetic appeal and the additional layer of information provided by the size of the bubbles, it was ultimately not included in the final presentation. The primary reason for this was that the data could be more straightforwardly and effectively communicated through other types of visualizations, such as line charts and heatmaps, which were easier for the audience to quickly interpret.

```
-----------------------------------------------------------------------------------------
# Importing necessary libraries
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

# Establish a connection to the SQL Server
# Connect to the SQL Server
conn = pyodbc.connect('DRIVER={SQL Server};'
                      'SERVER=KA-HUAWEI;'
                      'DATABASE=AdventureWorks2019;')
```

```
# Fetch data from SQL into a DataFrame
EmployeeSizeRevenue = pd.read_sql_query('SELECT * FROM SizeEmployeeRevenue', conn)

# Create a Bubble chart
plt.figure(figsize=(10, 6))

# The size of the bubbles is proportional to the number of employees
bubble_size = EmployeeSizeRevenue['Avg_num_employees'] * 100

# The color of the bubbles is also mapped to the number of employees
bubble_color = EmployeeSizeRevenue['Avg_num_employees']

plt.scatter('Size', 'Average_Revenue',
            s=bubble_size,
            c=bubble_color,
            alpha=0.5,
            data=EmployeeSizeRevenue)

# Add color bar and labels
plt.colorbar(label='Number of Employees')
plt.xlabel('Size of the Store (sqft)')
plt.ylabel('Average Revenue ($)')
plt.title('Bubble Chart: Store Size vs Average Revenue')

plt.show()
```

The bubble chart was designed with a figsize of (10, 6) to emphasize bubbles. The size of each bubble was proportional to the number of employees, and the colour was mapped to the same variable but with a different scale, providing two layers of information in one chart. However, due to the complexity of interpreting this multi-layered information quickly, and the lack of clarity at each cluster of datapoints, this chart was not included in the final presentation. Instead, more comprehensive charts were chosen for their efficiency in conveying the message.



Presentation Insights/Recommendations

Insights

Bimodal Distributions: The bimodal distribution in store sizes and revenues suggests market segmentation, potentially between small-scale and large-scale operations. A gap around 50,000 sqft could indicate a threshold that businesses find challenging to cross, possibly due to logistics, management complexity, or market conditions.

Strong Correlations: The high correlation coefficients (0.95-0.99) among the variables indicate a strong linear relationship. This suggests that store size, revenue, and the number of employees are closely interconnected.

Revenue and Employee Count: While there is a general upward trend, the horizontal clustering of data points suggests that the number of employees is not the only factor affecting revenue. There could be external variables like location, consumer behaviour, or seasonal trends influencing revenue.

Recommendations

Market Segmentation Strategy: Given the bimodal distributions, it may be beneficial to tailor strategies specifically for small and large stores. For mid-sized stores hovering around the 50,000 sqft gap, a specialized strategy could help them scale up effectively.

Optimizing Staffing Levels: With a strong correlation between the number of employees and revenue, stores should analyse their staffing levels carefully. Too few employees might mean lost sales, while too many could lead to increased costs without proportional revenue gains.

Variable Cost Analysis: Given the horizontal clustering of revenue against the number of employees, it is recommended to conduct a variable cost analysis. Understanding other factors that influence revenue can lead to more informed decisions.

Location-based Analysis: Since store size and the number of employees are strong indicators of revenue, a location-based analysis could provide insights into optimal store sizes and staffing levels for different geographical areas.

Causality Investigation: Correlation does not imply causation. Further statistical tests like regression analysis should be carried out to understand the causality between these variables better.

Resource Allocation: Given the strong interrelationship between size, revenue, and employee count, resource allocation strategies (like staff training, inventory management, and marketing budgets) should be proportionate and aligned with these variables.

Conclusions

In the subsequent summaries, we distil key analysis techniques and findings from our multi-faceted data analyses, covering areas such as sales performance, employee benefits, and operational metrics. These insights serve as a basis for strategic decision-making and highlight opportunities for further research.

USA Sales by Region

United States was the best performing country last year and this year as well but apart from Canada all other countries have had a massive increase in sales which suggests the company is focusing on its expansion in Europe and the Pacific.

Despite the sales on the West Coast had grown exponentially, it decreased in the East Coast and Central. Government policies, marketing effort, supply chain issues and/or company strategies like opening/closing of stores might have influenced the regional sales results for this year.

Annual Leave and Bonus

In summary, the analysis revealed a weak but positive correlation of 0.38 between annual leave taken and bonuses received across the organization. This correlation was not strong enough to suggest a definitive pattern, and there were interesting discrepancies between different job titles, particularly between managers and sales representatives. The managers appeared to take less leave and also received smaller bonuses, suggesting that role demands, and company policies may play a significant part in these outcomes. Given these variances, further in-depth studies are advised to fully understand the underlying factors influencing this relationship.

Country and Revenue

In summary, the analysis highlights the United States and Canada as the top revenue-generating countries in both years, although their share of total sales decreased this year compared to last. Meanwhile, countries like Australia, the United Kingdom, France, and Germany saw an increase in their share of sales this year. The key question arising from this data is why the United States consistently tops the chart. While possible reasons could be its market size, global companies, diverse industries, or strong economy, it's crucial to recognize that revenue generation is influenced by a country's unique economic and social context. Therefore, understanding this relationship requires a comprehensive analysis of multiple dynamic factors.

Sick Leave and Job Title

Management should consider implementing measures to support employees at all levels, especially those in higher positions. Strategies for stress management, workload distribution, and work-life balance could be beneficial to reduce sick leave across the organization.

Store Trading Duration and Revenue

In conclusion, the examination of the connection between trading duration and average revenue has revealed an unexpected pattern in the scatter plot, bar charts, and heatmap. The observed correlation between these variables appears to be relatively weak, which contrasts with the anticipated linear progression. This discrepancy suggests the influence of multiple factors that may be affecting these variables. To gain a deeper understanding, further research is essential to pinpoint the exact underlying causes and contributing factors.

Store Size, Employee Number and Revenue

In conclusion, the comprehensive analysis of Question 6 revealed significant insights into the relationships between store size, average revenue, and the number of employees. Various data visualization techniques, including pairplots, line charts, and heatmaps, were employed to effectively illustrate these connections. While the initial approach encountered some hurdles due to the complexity of the database schema, the final methodology proved robust, as supported by high correlation coefficients.

This investigation underscores the importance of selecting the appropriate database views and employing meticulous data visualization design choices for effective communication in a presentation setting. The findings have potential implications for store optimization strategies, especially concerning staffing and revenue forecasting.