Technical Report on The AdventureWorks Cycles Company

A Comprehensive Analysis of Revenue Drivers:
Regional Performance, Employee Behaviour,
Store Characteristics, and Sales Trends for Bikes

1. Introduction

The purpose of this report is to analysis the AdventureWorks data to simulate business operations for a global bicycle retailer. The dataset contains information about sales, customers, employees, products, and stores. In this analysis, we focus on sales performance based on different regions/ countries, sales revenue, store duration, square footage, and the number of employees and the employees' behaviours to determine how these factors influence overall performance.

2. Data Collection and Preparation

Data Source:
The primary data source for this analysis was retrieved from the AdventureWorks2022 database using SQL queries. Relevant tables include 'SalesOrderHeader', 'Customer', and 'StoreWithDemographics'.
WorldBank.org was used to analyse GDP performances in the countries of interest to develop insights into sales performances globally and domestically.

Data Preparation:
To understand the data structure and identify the relevant tables, relationships, and necessary joins between tables, we created a database diagram for the AdventureWorks2022 before planning the data extraction process.
The data for store sizes, annual revenue, and the number of employees is stored in XML data structures in Table Sales.Store. We need to extract the data from the XML using the CAST() function in SQL before proceeding with any further steps

Data Cleaning:
Data cleaning involved handling missing values in the 'TotalDue' column and removing duplicate records from the sales transactions. Outliers in store revenue were also identified and addressed.

To analyse store performance over time, we created a new variable, 'StoreDuration', by calculating the difference between the current year and the year the store was opened.

The data was cleaned by removing unnecessary information, and the required data was processed by joining relevant tables to facilitate the analysis.

3. Methodology and Analysis

For this analysis report, we primarily used descriptive and diagnostic analysis methods to explore and interpret the relationships between various factors and business performance. Our research adopted a quantitative approach, utilising numerical data and statistical techniques to draw meaningful conclusions.

- **Descriptive analysis:** It was used to summarize and describe the key characteristics of the data, such as identifying trends, patterns, and differences between variables. Visualizations like scatter plots, bar charts, and line charts helped us better understand the data and explore the relationships between different variables.

- **Diagnostic analysis:** It was used to understand the underlying causes of the observed patterns. For example, we explored the correlation between annual leave taken and bonus amounts to identify if there is a relationship that explains variations in bonuses based on time off. We also examined the relationship between store trading duration and revenue to understand how these factors may influence sales performance.

Validation and Verification

To ensure the accuracy and consistency of the data, the following validation steps were performed:

- **Currency Standardization:** All sales and revenue figures were standardized to USD to ensure consistency and eliminate discrepancies arising from currency differences across regions or countries. This step was critical for accurately comparing financial data across different markets.

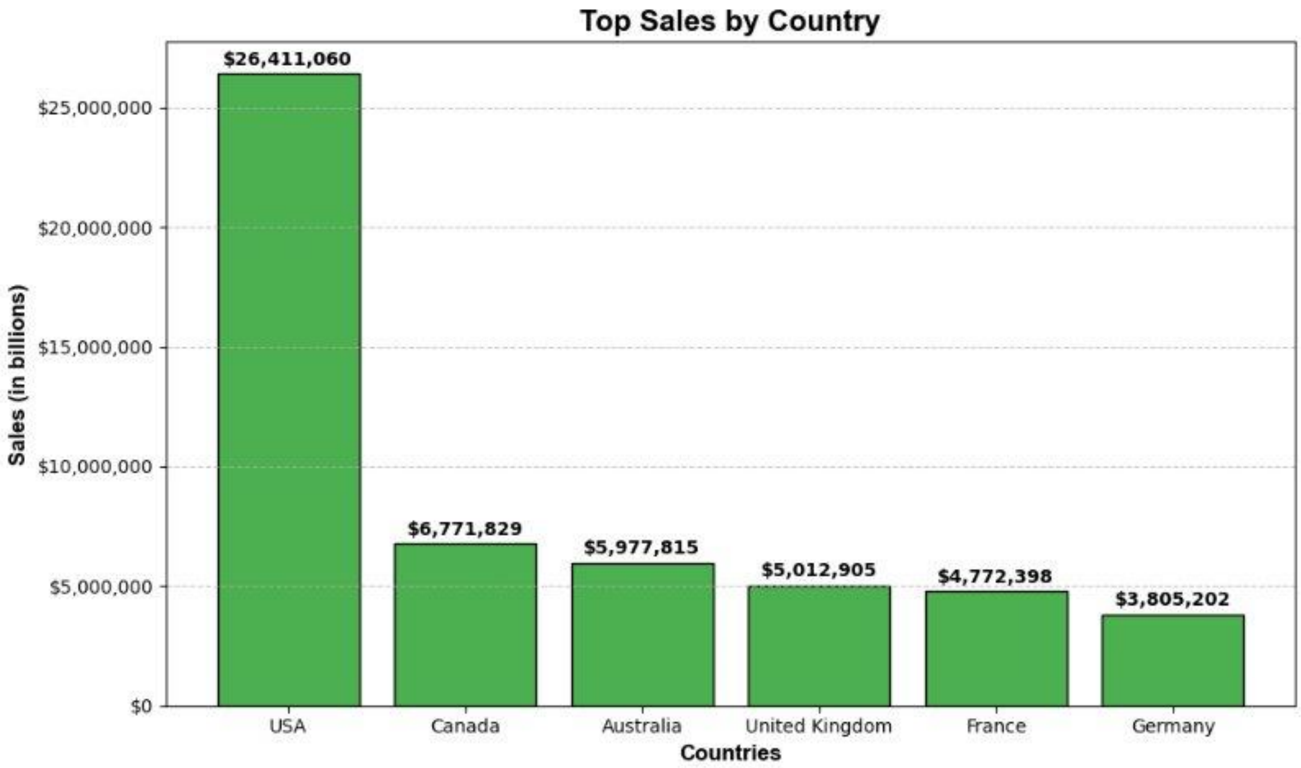
- **Data Cleaning:** We applied an "is not null" function to remove missing or irrelevant data points that could skew the analysis. Outliers were also identified and addressed to maintain the integrity of the dataset.

- **Consistency Checks:** To validate the results, correlation coefficients and regression analyses were cross verified with visualizations (e.g., scatter plots and bar charts). This ensured that the observed trends were consistent and supported by both the statistical analysis and the graphical representations.

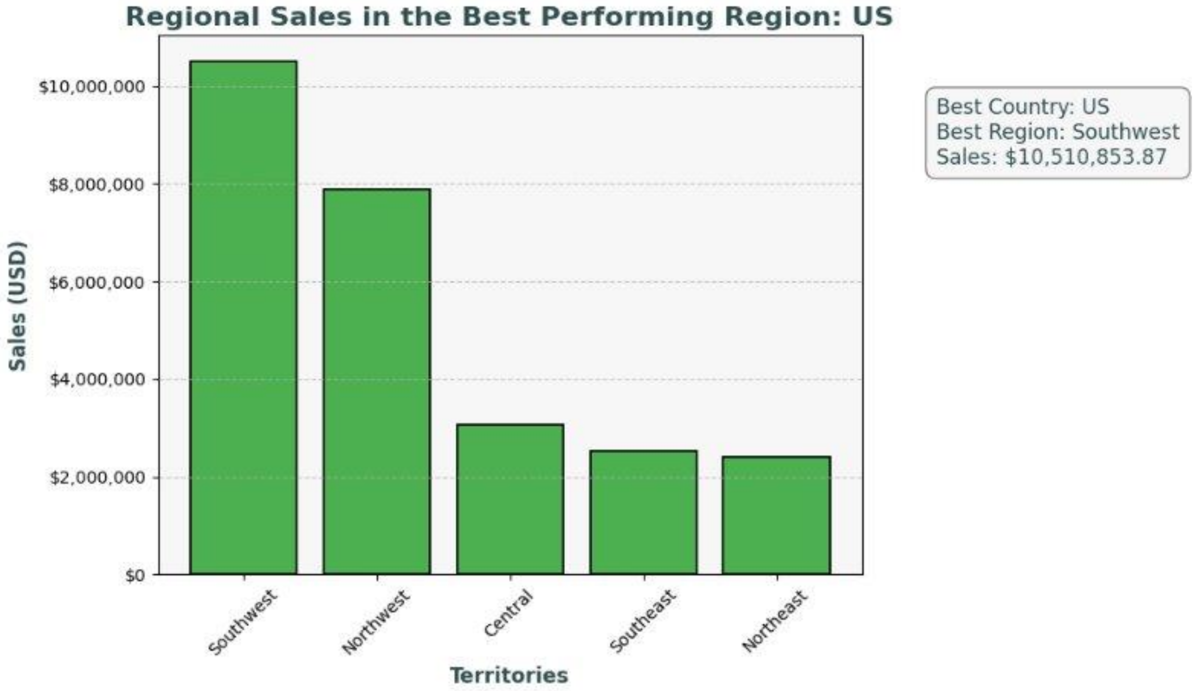
4. Results

4.1 Regional sales in the best performing country

This bar chart displays the top sales by country, showcasing the total sales figures for various countries. The USA leads with the highest sales of **$26,441,829** followed by Canada, Australia, the United Kingdom, France, and Germany with progressively lower sales. This visualization allows for a clear comparison of sales performance across these countries.



The United States of America is the country with the highest sales among the regions analysed. The chart displays regional sales performance across five key territories: Southwest, Northwest, Midwest, Southeast, and Northeast. The Southwest stands out with exceptional sales performance, recording the highest sales of **$10,510,853.87**. Meanwhile, the other regions show lower sales levels, indicating the need to reassess strategies in those areas to stimulate growth. This chart provides a comprehensive understanding of regional trends and helps guide resources more effectively toward areas that require additional support.

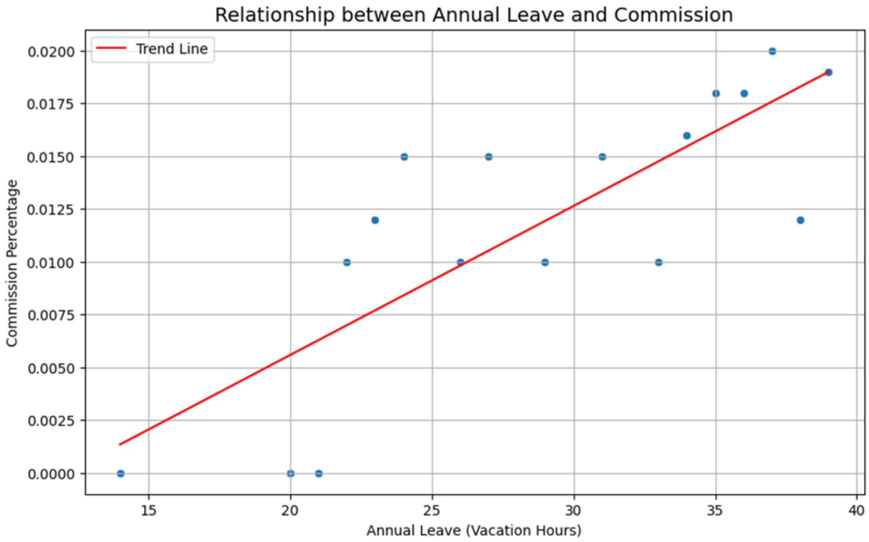


4.2 The relationship between annual leave taken and bonus

This report analyses employee data focusing on leave records and compensation. The dataset includes key information such as employee IDs, job titles, organisational structure, vacation and compensation details.

1. **Data Cleaning**:
   a. Rows with missing values in key fields (**Organisation Node**, **Organisation Level**) were removed to ensure accurate analysis.
2. **Key Findings**:
   a. **Average Vacation Hours**: Employees have an average of **55 hours** of vacation leave.
   b. **Average Bonus Percentage**: The average bonus percentage (commission) is **8.75%**.
3. **Correlation**:
   a. A **moderate positive correlation of 0.82** exists between vacation hours and bonus percentages, suggesting that employees with more vacation hours tend to receive higher bonuses.

This analysis further examines the relationship between vacation hours and commission percentage. Commission percentage reflects the earnings an employee receives based on sales. Vacation hours refer to the time off an employee takes.
The results show that 67.5% of the variation in commission can be explained by vacation hours, however, the relationship is not statistically significant.
The model suggests a slight positive effect, where each additional vacation hour may slightly increase commission. The effect is minimal and not strong enough to be conclusive.
Due to the limited sample size, we cannot conclusively establish that vacation hours directly affect commission. Further analysis with more data or additional factors may provide clearer insights.



The average was determined as part of the analysis using the following code.

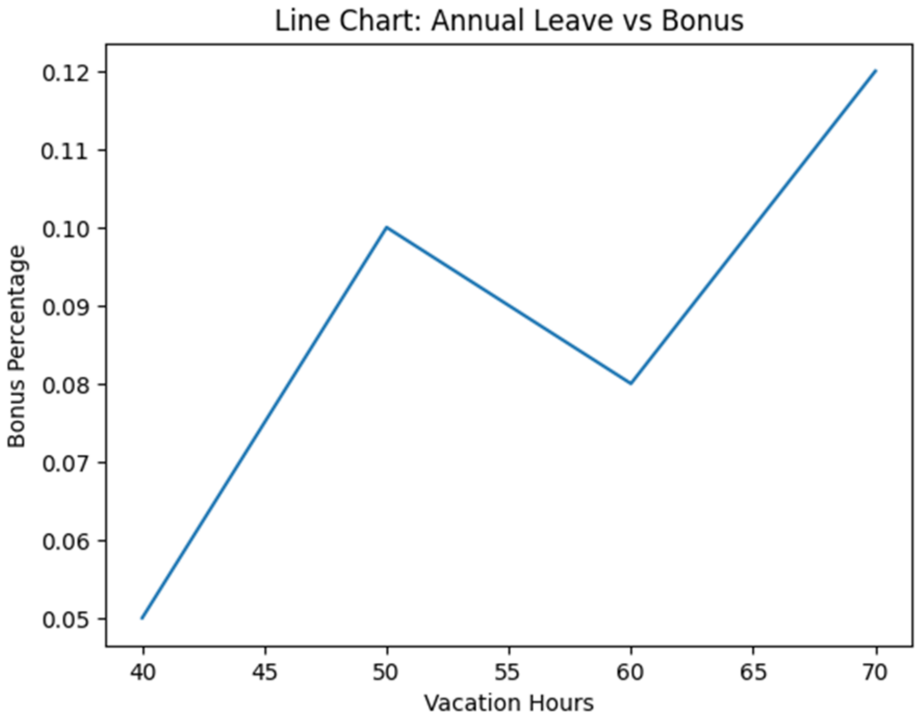
```
# Calculate average of VacationHours (Annual Leave) and CommissionPct (Bonus)
average_vacation_hours = df['VacationHours'].mean()
average_bonus = df['CommissionPct'].mean()
```

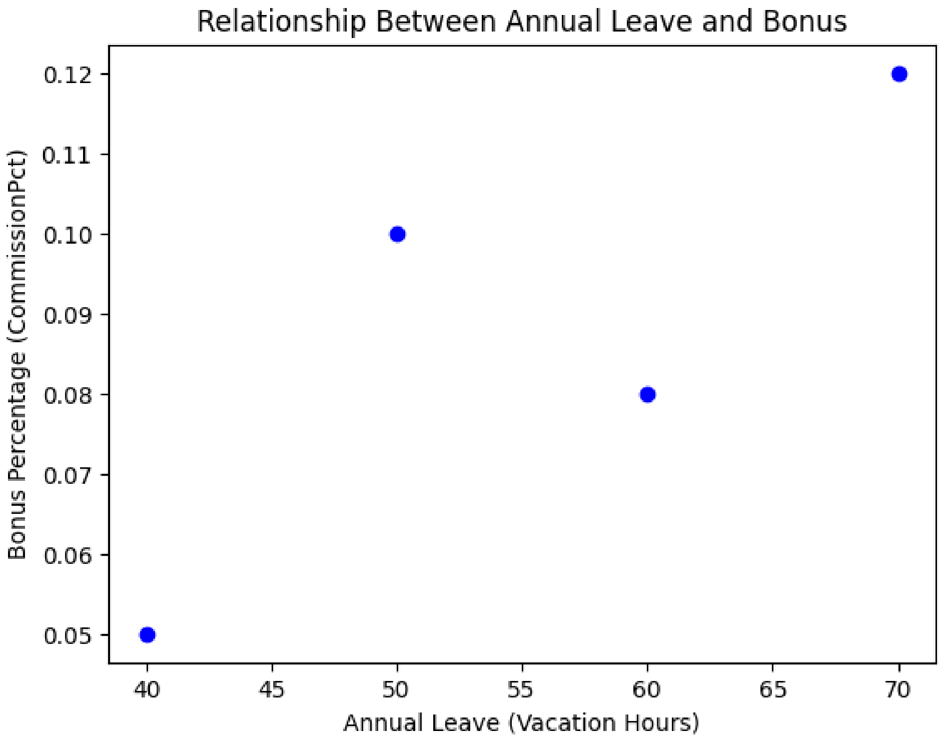
The correlation was determined using the following code.

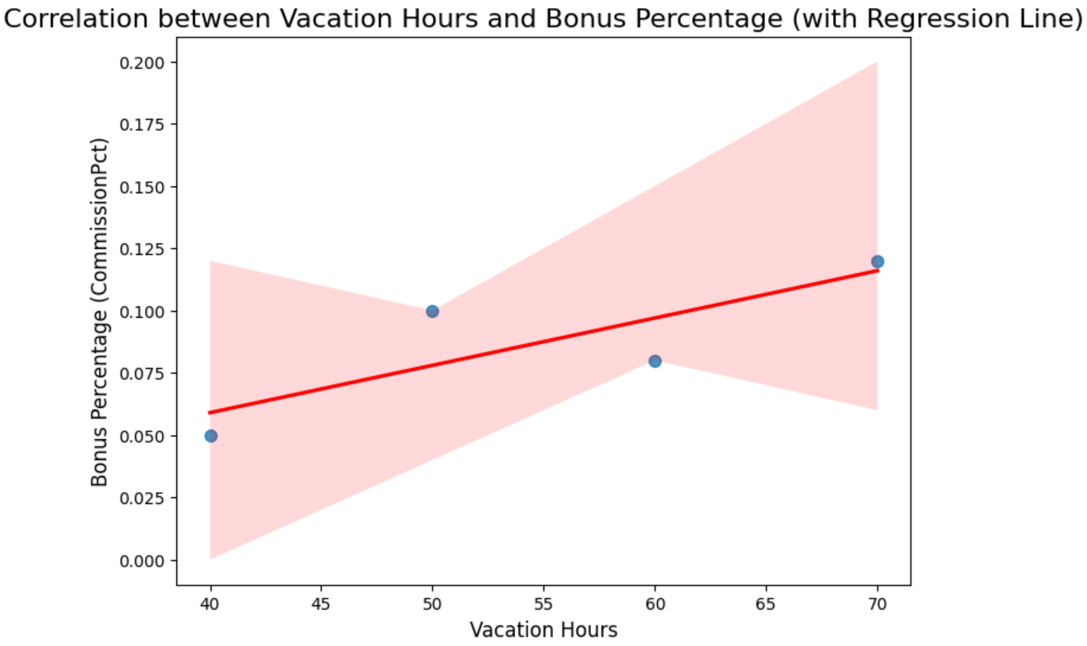
```
df = pd.DataFrame(data)

correlation = df['VacationHours'].corr(df['CommissionPct'])
print(f"Correlation between Annual Leave and Bonus: {correlation}")
```

- **Correlation Between Annual Leave (Vacation Hours) and Bonus**: There is a moderate positive correlation (0.82) between vacation hours and bonus percentages. This suggests that employees with more vacation hours tend to receive higher bonuses.
- **Averages**:
- **Average Annual Leave (Vacation Hours)**: 55 hours.
- **Average Bonus Percentage (CommissionPct)**: 0.0875 (about 8.75%).

A line plot and scatter plot were subsequently employed to visualise the relationship between annual leave and bonus.

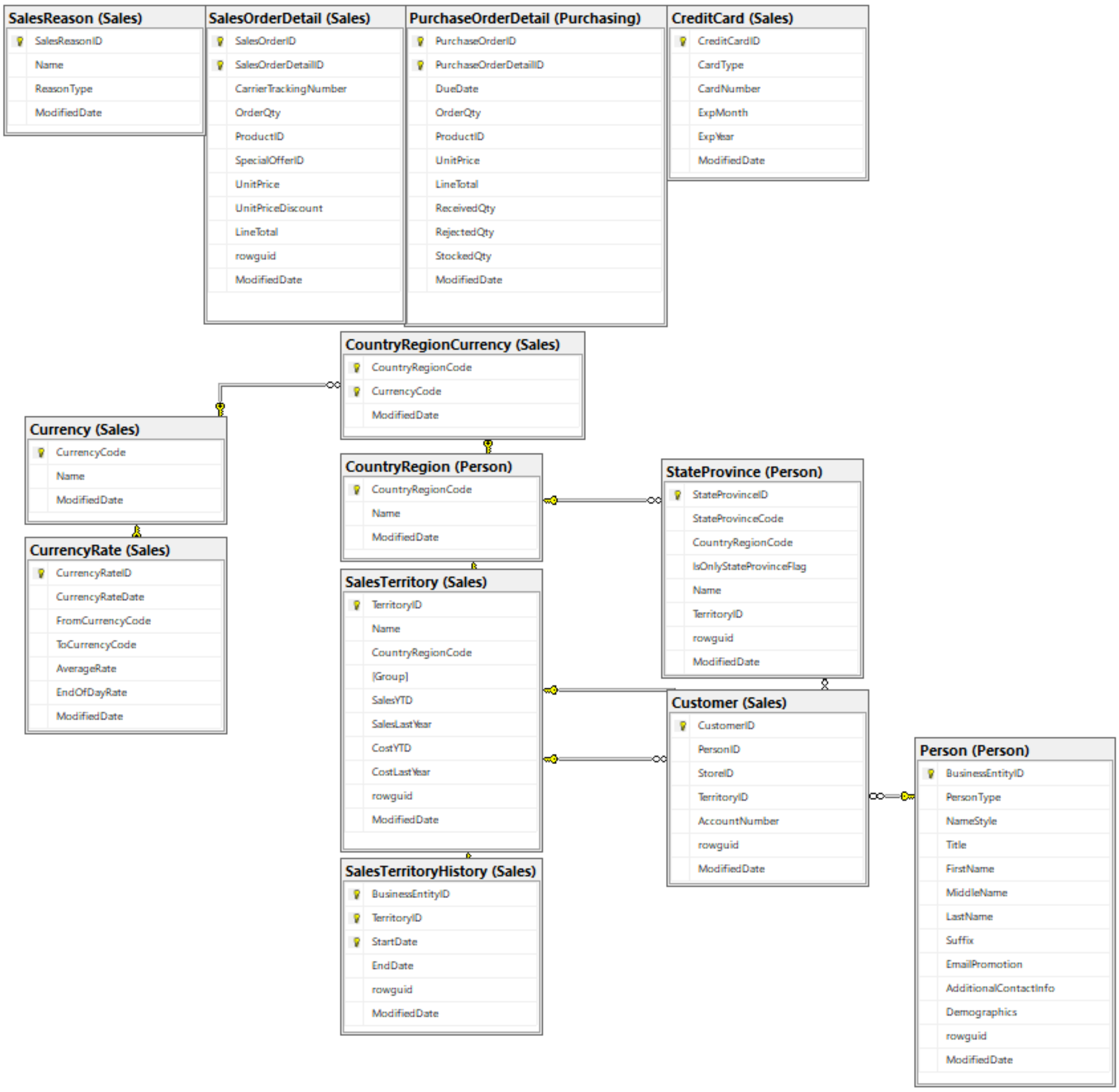

Line Chart: Annual Leave vs Bonus
Bonus Percentage
0.12
0.11
0.10
0.09
0.08
0.07
0.06
0.05
40
45
50
55
60
65
70
Vacation Hours


Relationship Between Annual Leave and Bonus
Bonus Percentage (CommissionPct)
0.12
0.11
0.10
0.09
0.08
0.07
0.06
0.05
40
45
50
55
60
65
70
Annual Leave (Vacation Hours)

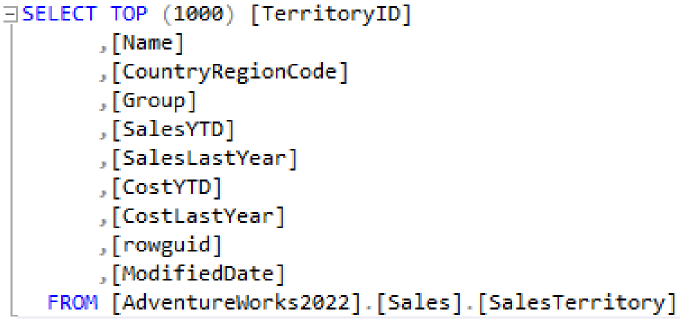


In conclusion, the analysis effectively examined the relationship between annual leave and bonus through the use of statistical methods, including the calculation of averages and the visualisation of data via line and scatter plots. The findings provide valuable insights into the correlation between these two variables, highlighting trends and potential patterns. These results can inform decision-making processes regarding employee benefits and compensation strategies. Further analysis could be conducted to explore additional factors influencing the relationship between annual leave and bonus.

4.3 The relationship between Country and Revenue



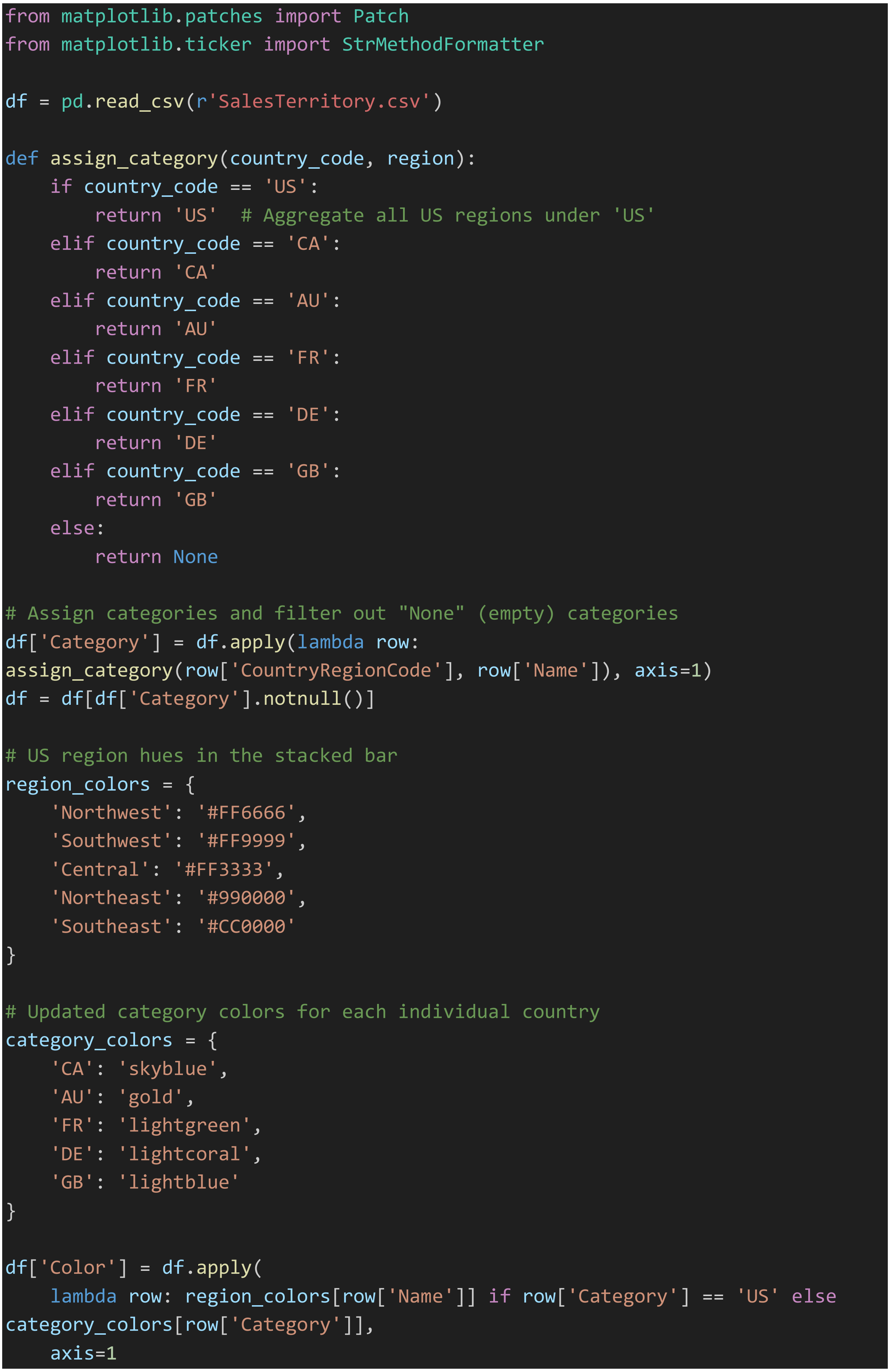
A modified schema was produced to more easily read through the data and determine what would be necessary for this report. Only the SalesTerritory sales data would be needed; the other tables would already be covered in other questions in the report. Instead, external data sources would be used instead to provide insights into the relationship between country and revenue performance for AdventureWorks.



```sql
SELECT TOP (1000) [TerritoryID]
      ,[Name]
      ,[CountryRegionCode]
      ,[Group]
      ,[SalesYTD]
      ,[SalesLastYear]
      ,[CostYTD]
      ,[CostLastYear]
      ,[rowguid]
      ,[ModifiedDate]
  FROM [AdventureWorks2022].[Sales].[SalesTerritory]
```

To begin the analysis, Sales data relating to regions was pulled from the AdvetureWorks2022 database using the following SQL query in SSMS (Sequel SQL Server Management Studio).

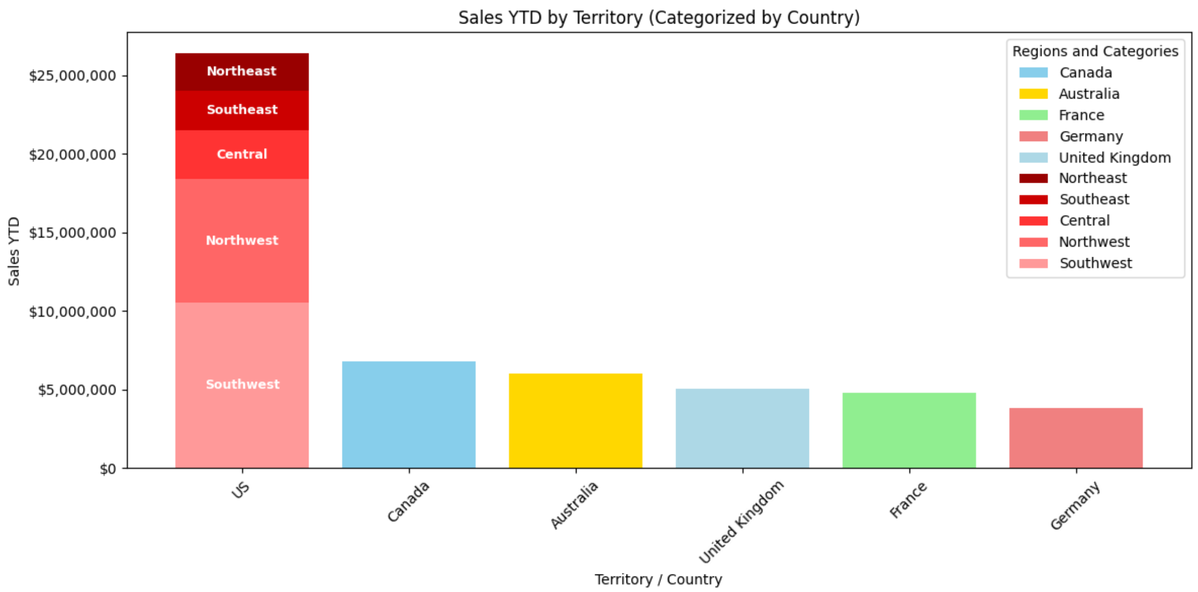
```python
import pandas as pd
import matplotlib.pyplot as plt
```



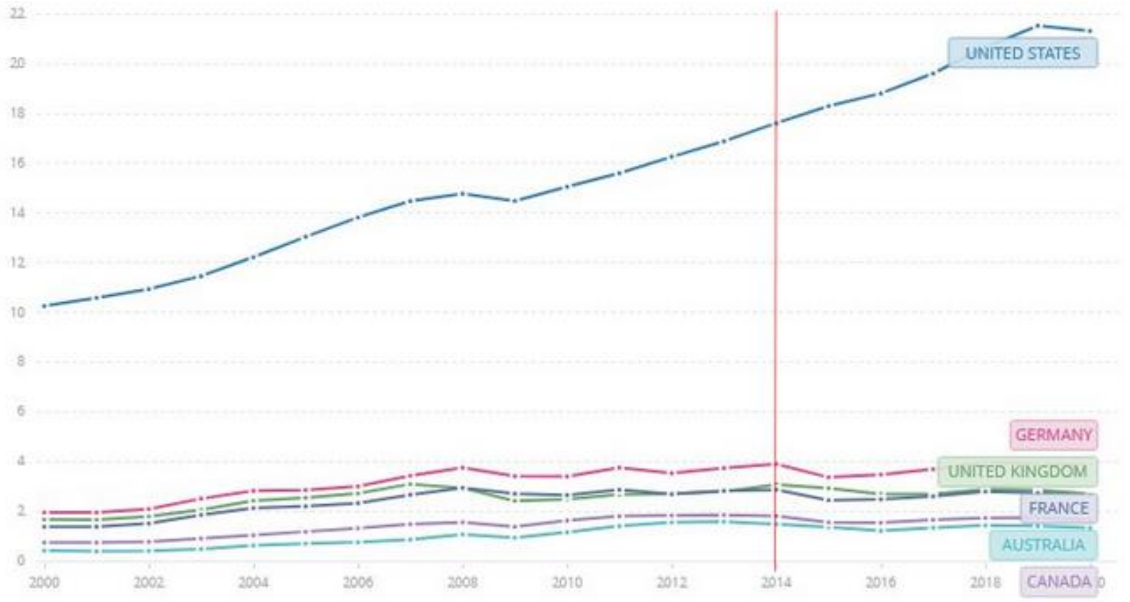
```python
from matplotlib.patches import Patch
from matplotlib.ticker import StrMethodFormatter

df = pd.read_csv(r'SalesTerritory.csv')

def assign_category(country_code, region):
    if country_code == 'US':
        return 'US'  # Aggregate all US regions under 'US'
    elif country_code == 'CA':
        return 'CA'
    elif country_code == 'AU':
        return 'AU'
    elif country_code == 'FR':
        return 'FR'
    elif country_code == 'DE':
        return 'DE'
    elif country_code == 'GB':
        return 'GB'
    else:
        return None

# Assign categories and filter out "None" (empty) categories
df['Category'] = df.apply(lambda row:
assign_category(row['CountryRegionCode'], row['Name']), axis=1)
df = df[df['Category'].notnull()]

# US region hues in the stacked bar
region_colors = {
    'Northwest': '#FF6666',
    'Southwest': '#FF9999',
    'Central': '#FF3333',
    'Northeast': '#990000',
    'Southeast': '#CC0000'
}

# Updated category colors for each individual country
category_colors = {
    'CA': 'skyblue',
    'AU': 'gold',
    'FR': 'lightgreen',
    'DE': 'lightcoral',
    'GB': 'lightblue'
}

df['Color'] = df.apply(
    lambda row: region_colors[row['Name']] if row['Category'] == 'US' else
category_colors[row['Category']],
    axis=1
)

# Aggregated US regions into one bar and sort regions by SalesYTD descending
in size (largest at the bottom)
us_data = df[df['Category'] ==
'US'].groupby('Name').sum(numeric_only=True).reset_index()
us_data = us_data.sort_values(by='SalesYTD', ascending=False)

# Data frame with the now aggregated US regions alongside the other nations
data_to_plot = pd.DataFrame({
    'Category': ['US', 'CA', 'AU', 'FR', 'DE', 'GB'],
    'SalesYTD': [
        us_data['SalesYTD'].sum(),
        df[df['Category'] == 'CA']['SalesYTD'].sum(),
        df[df['Category'] == 'AU']['SalesYTD'].sum(),
        df[df['Category'] == 'FR']['SalesYTD'].sum(),
        df[df['Category'] == 'DE']['SalesYTD'].sum(),
        df[df['Category'] == 'GB']['SalesYTD'].sum()
    ]
})

# Sort the data by SalesYTD in descending order for the x-axis
data_to_plot = data_to_plot.sort_values(by='SalesYTD', ascending=False)

# Plot stacked bar chart
fig, ax = plt.subplots(figsize=(12, 6))

# Stacked bar for US regions (sorted by size in descending order)
bottom_value = 0
for _, region_row in us_data.iterrows():
    region = region_row['Name']
    value = region_row['SalesYTD']
    color = region_colors[region]  # Use the slightly darker red hues
    ax.bar('US', value, bottom=bottom_value, color=color, label=region)

    # Text labels within each segment of the US bar
    plt.text(
        x='US',
        y=bottom_value + value / 2,
        s=region,
        ha='center',
        va='center',
        fontsize=9,
        color='white',
        weight='bold'
    )
    bottom_value += value
```

```
# Bars for non-US categories (CA, AU, FR, DE, GB)
for idx, row in data_to_plot.iloc[1:].iterrows():  # Exclude the aggregated US
row
    ax.bar(row['Category'], row['SalesYTD'],
color=category_colors[row['Category']], label=row['Category'])

# Labels and title
ax.set_xlabel('Territory / Country')
ax.set_ylabel('Sales YTD')
ax.set_title('Sales YTD by Territory')

# Renaming the x-axis labels with full country names instead of the
abbreviations in the database
category_labels = {
    'US': 'United States',
    'CA': 'Canada',
    'AU': 'Australia',
    'FR': 'France',
    'DE': 'Germany',
    'GB': 'United Kingdom'
}

# Matching labels and rotating them for better visualisation
ax.set_xticks(range(len(data_to_plot['Category'].unique())))
ax.set_xticklabels([category_labels[cat] for cat in
data_to_plot['Category'].unique()], rotation=45)
# Formatting y-axis to show raw numbers and in dollars
ax.yaxis.set_major_formatter(StrMethodFormatter('{x:,.0f}'))
ax.yaxis.set_major_formatter(StrMethodFormatter('${x:,.0f}'))

# Updating the legend to properly colour code the individual countries
legend_elements = [
    Patch(facecolor=color, label=category_labels[category]) for category,
color in category_colors.items()
] + [
    Patch(facecolor=color, label=region) for region, color in
sorted(region_colors.items(), key=lambda x: us_data.loc[us_data['Name'] ==
x[0], 'SalesYTD'].values[0], reverse=False)
]
ax.legend(handles=legend_elements, title="Regions and Categories")
plt.tight_layout()
plt.show()
```

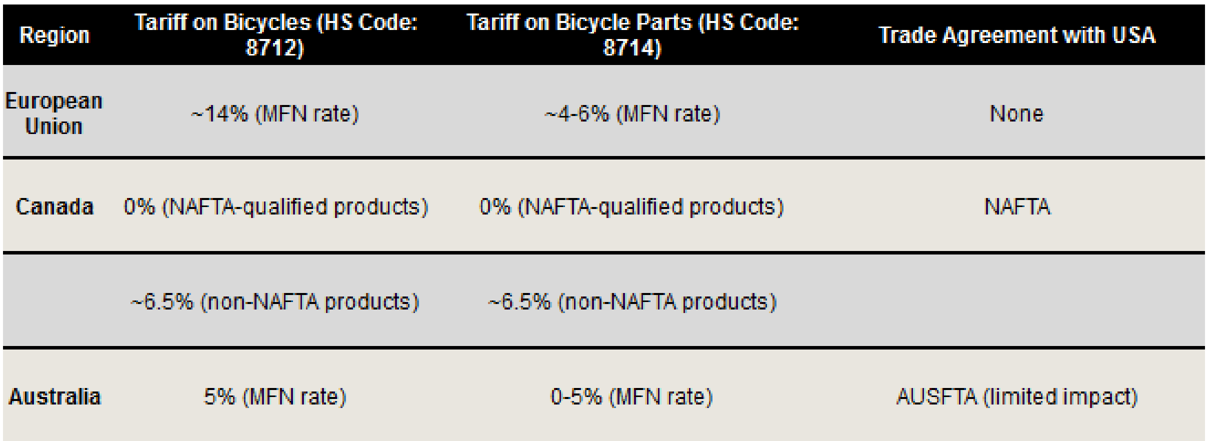
This is the code used to produce the primary graph for the analysis. I had chosen to break down the US into a stacked bar chart as I also wanted to display the performance of the regions within the country to provide progression from the opening question regarding regional sales in the best performing country.



Data shows that the US is the strongest performing market for AdventureWorks. This is to be expected as the company is based here, and it is considered a domestic market. AdventureWorks is a company based in Washington, in the northwest region. This would explain why the region is the second highest performer behind the Southwest which has wealthy states like California with strong purchasing power.



The US having the strongest GDP (and subsequent purchasing power) also supports this. However, it can also be assumed that sales performance should be the best in a company's base country as that is where they would have the most operational logistics, production, shops and complementary staff to begin with. Further analysis will be done on the performances of the other countries relative to each other. The following graph shows that Australia and Canada have the lowest GDP of the group, but the sales performance chart shows them outperforming the United Kingdom, France, and Germany. This means that there are other factors in play affecting sales, such as international tariffs.



Region	Tariff on Bicycles (HS Code: 8712)	Tariff on Bicycle Parts (HS Code: 8714)	Trade Agreement with USA
European Union	~14% (MFN rate)	~4-6% (MFN rate)	None
Canada	0% (NAFTA-qualified products)	0% (NAFTA-qualified products)	NAFTA
	~6.5% (non-NAFTA products)	~6.5% (non-NAFTA products)	
Australia	5% (MFN rate)	0-5% (MFN rate)	AUSFTA (limited impact)

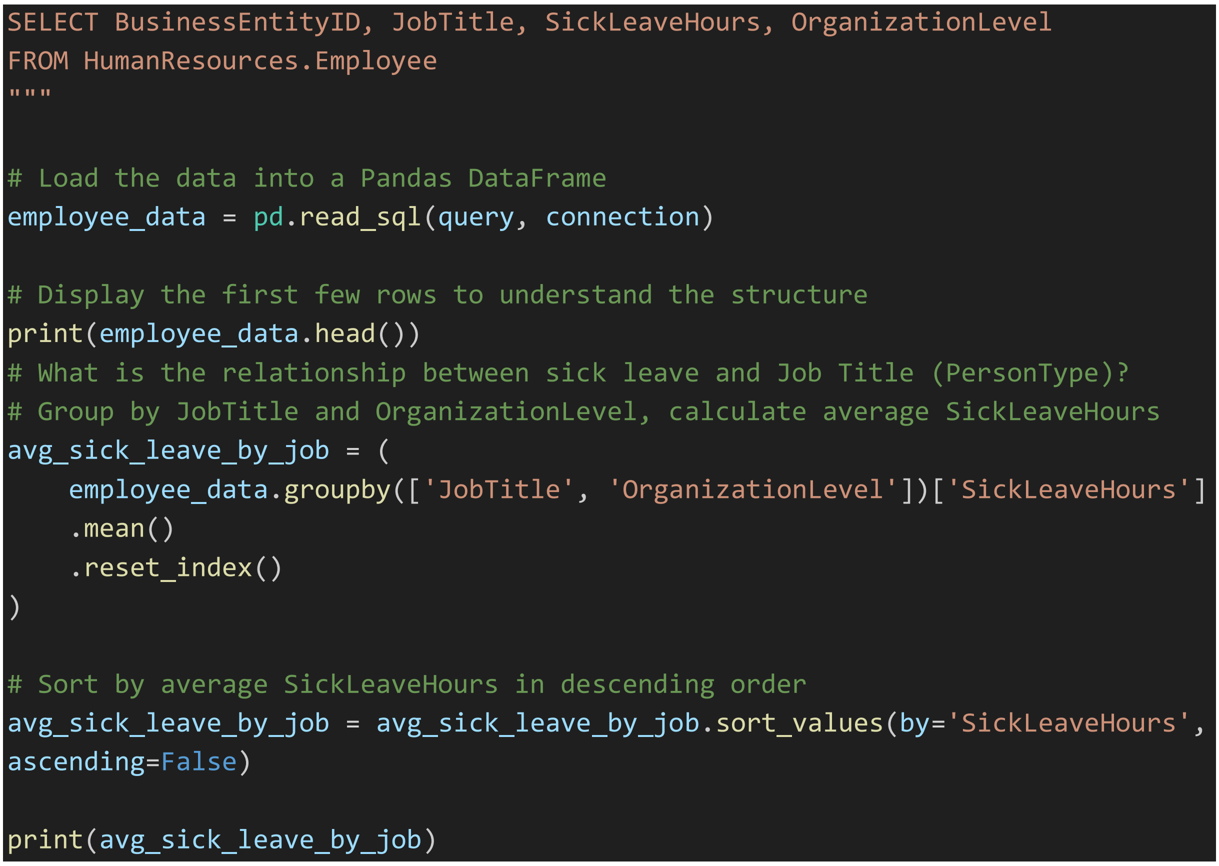
This is a table summarizing trade agreements on tariffs between the US and these countries. In 2014, the US and the EU did not have a special trade deal and so the Most Favoured Nation (MFN) rates were applied instead. They have a 14% tarriff on bicycles which can be considered a large factor in the low sales value when considering the GDPs of the European nations. Canada has a free trade agreement with the US that allows goods to be traded tariff free, and this could be the primary reason sales in Canada are higher than those in Europe; the addon costs of tariffs and transportation would make the bicycle products cheaper for Canadians, and therefore more desirable. Australia did have a free trade agreement, but it does not cover this industry, so MFN would still be applied. Whilst not as low as Canada's rates, Australia still had lower tariffs than Europe which may be the reason sales were higher there compared to Europe.

Further analysis would consider local tax rates (VAT) and the strength of foreign domestic competition to determine whether investing in overseas sales will be worth the return.

4.4 The relationship between sick leave and Job Title

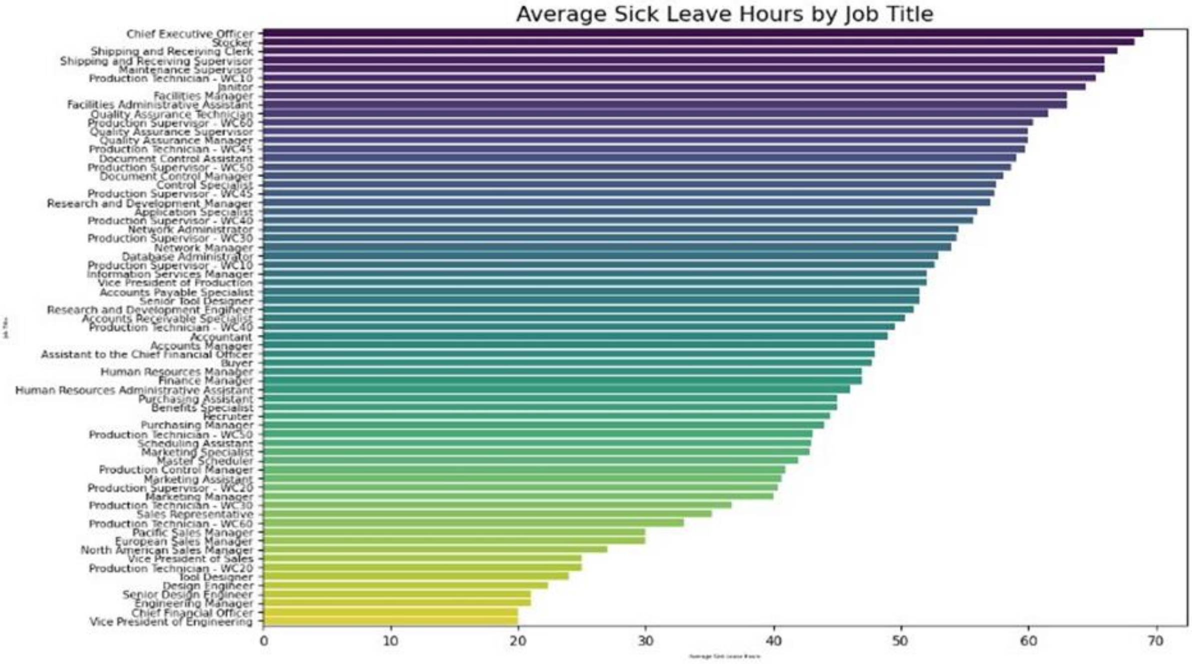
Methods and analysis: The first method we used was to analyse the average hours of sick leave to the job title. This was done on python, after importing the database from SQL.

```
# Retrieve relevant columns from SQL database
query = """
```

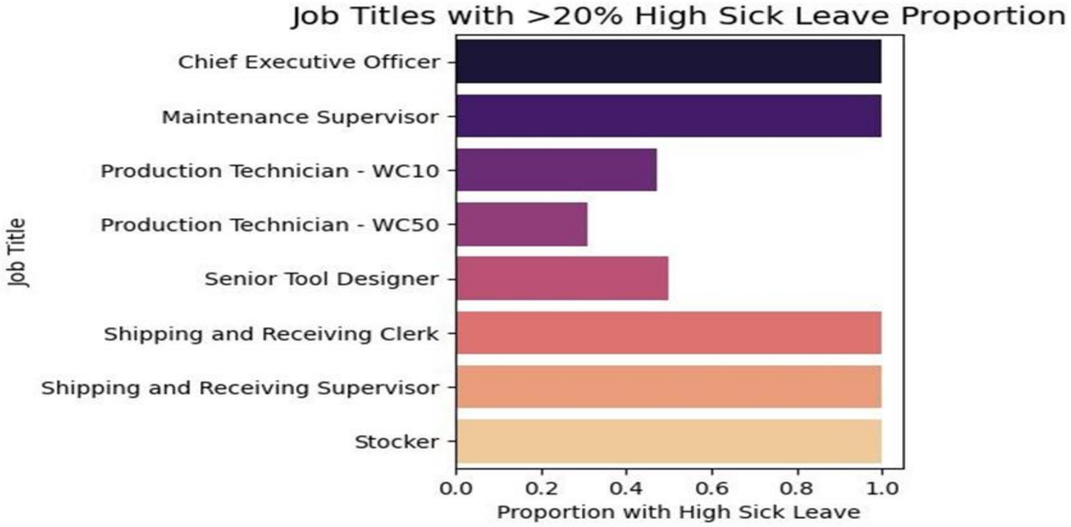


```
SELECT BusinessEntityID, JobTitle, SickLeaveHours, OrganizationLevel
FROM HumanResources.Employee
"""

# Load the data into a Pandas DataFrame
employee_data = pd.read_sql(query, connection)

# Display the first few rows to understand the structure
print(employee_data.head())
# What is the relationship between sick leave and Job Title (PersonType)?
# Group by JobTitle and OrganizationLevel, calculate average SickLeaveHours
avg_sick_leave_by_job = (
    employee_data.groupby(['JobTitle', 'OrganizationLevel'])['SickLeaveHours']
    .mean()
    .reset_index()
)

# Sort by average SickLeaveHours in descending order
avg_sick_leave_by_job = avg_sick_leave_by_job.sort_values(by='SickLeaveHours',
ascending=False)

print(avg_sick_leave_by_job)
```

Using the above code, we found the average sick leave hours by job title and organisation level, and we created a bar chart based on this data which showed us that the CEO took the highest number of average sick leave hours.



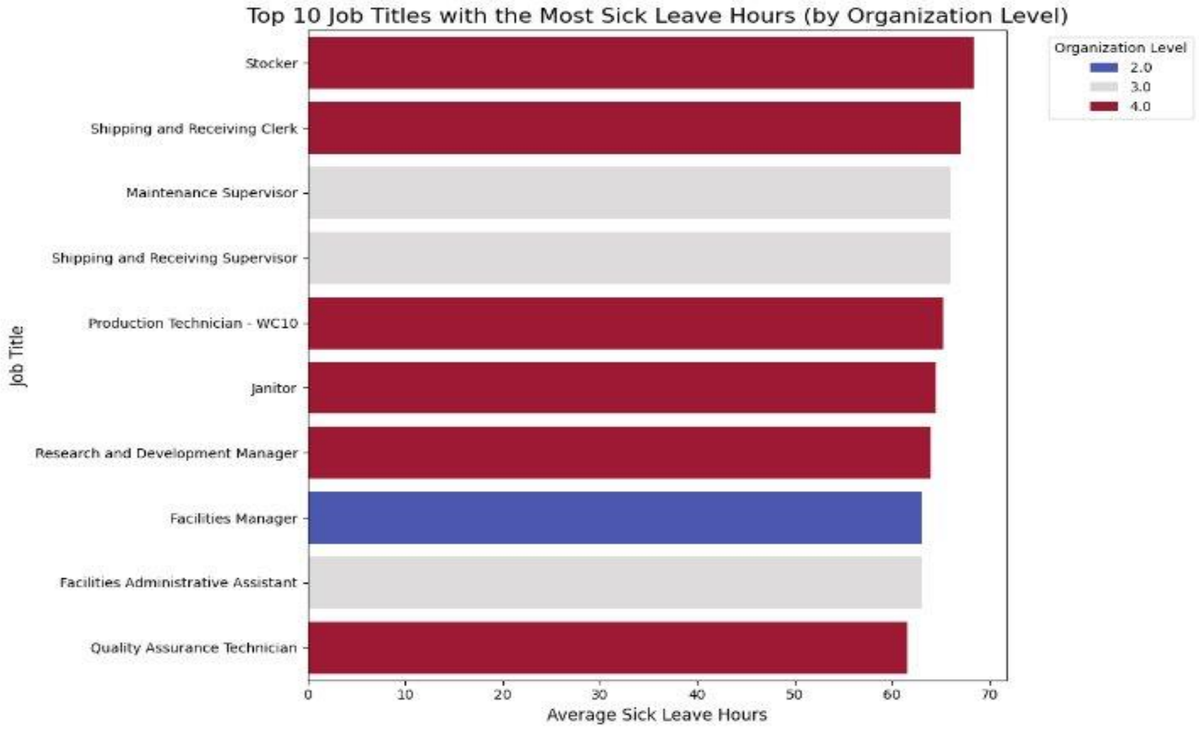
Due to the cluttered nature of the chart, we then limited this to just the top 10 job titles which had the highest sick leave, investigating with different variables. For example, one graph we attempted to make was a bar chart which depicted the job titles with more than 20% high sick leave proportion. Here, ‘high sick leave’ was defined as the top 10% of sick leave across the dataset. Essentially, we were looking at how many employees per job title had taken sick leave which were in the 90th percentile. Again, the CEO topped the list, with the proportion being 1. This is because there is only 1 CEO and they took 69 hours of sick leave, which was in the 90th percentile.



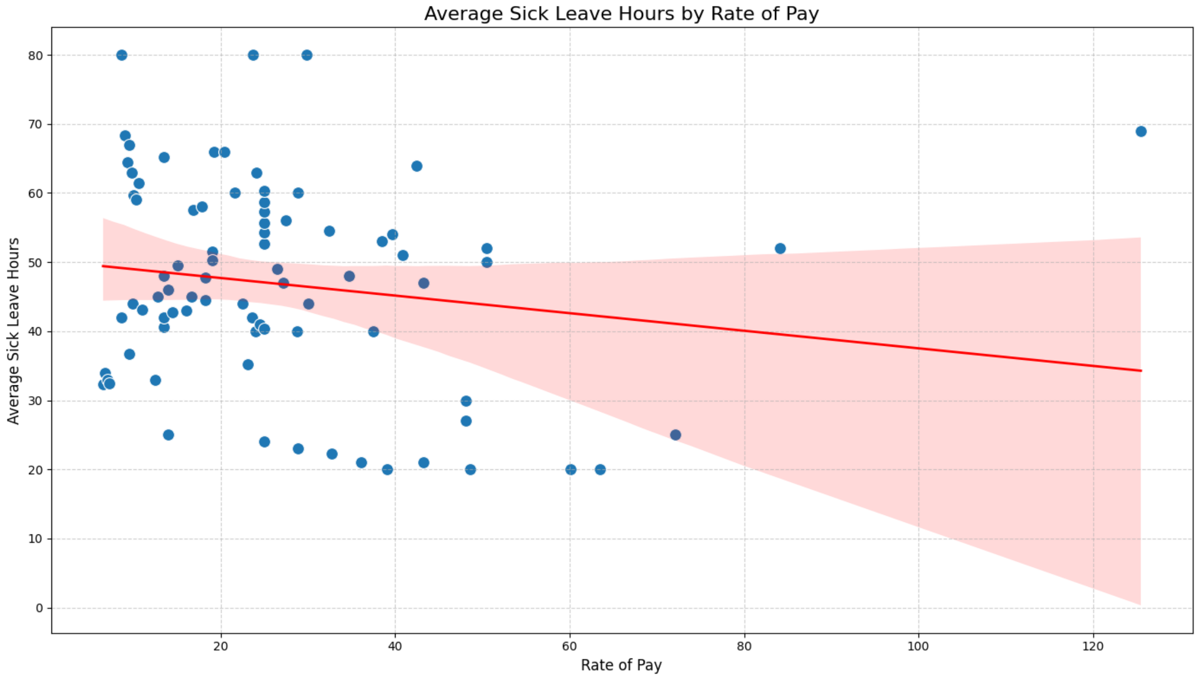
Note: High sick leave is defined as the top 10% (90th percentile) of sick leave across the dataset.

To improve on the second graph of the top 10 job titles with the highest sick leave, we then looked at the organisation level of each job title. Within the dataset, the CEO does not have a level in the organisation, as they are alone at the top. This meant, that the CEO was removed from this graph, but they can be assumed to be level 0, with level 1 directly below the CEO. From this improved graph showing the organisation levels, we could see that those who were at the bottom level (level 4) took the most hours of sick leave, while level 1 was not even found on the list of top 10.

One possible reason for this is that there are both more job roles and more employees per job role in level 4, then level 3, and so forth, which means there is more data for the lower levels and there is a higher probability of level 4 turning up in the top 10 job titles. Another reason that there may be more level 4 job titles in the top 10 is because the titles generally involve more practical, physically demanding tasks, which poses a greater risk to the health of workers, compared to desk-based jobs.



We also created a chart based on the rate of pay, and the number of sick leave hours. After checking the correlation, we can confirm that the correlation between rate of pay and organisation level is 0.74355062 which denotes a strong correlation, meaning the two may have a close relationship and we can assume that any result with the rate of pay can be extended to the organisation level.



For example, in the graph above, we can see that there isn't a strong correlation between the rate of pay and average sick leave hours. We've made this more precise by actually calculating the correlation coefficient:

```
# Merge the two tables on BusinessEntityID
```

```
merged_data = pd.merge(employee_data, pay_history, on='BusinessEntityID',
how='inner')

# Calculate the average sick leave hours per employee
avg_sick_leave = (
    merged_data.groupby('BusinessEntityID')['SickLeaveHours']
    .mean()
    .reset_index(name='AverageSickLeaveHours')
)

# Merge the average sick leave back with pay rates
sick_leave_pay_data = pd.merge(avg_sick_leave, pay_history,
on='BusinessEntityID', how='inner')

# Remove outliers using the IQR method
Q1_sick_leave = sick_leave_pay_data['AverageSickLeaveHours'].quantile(0.25)
Q3_sick_leave = sick_leave_pay_data['AverageSickLeaveHours'].quantile(0.75)
IQR_sick_leave = Q3_sick_leave - Q1_sick_leave

lower_bound_sick_leave = Q1_sick_leave - 1.5 * IQR_sick_leave
upper_bound_sick_leave = Q3_sick_leave + 1.5 * IQR_sick_leave

Q1_rate = sick_leave_pay_data['Rate'].quantile(0.25)
Q3_rate = sick_leave_pay_data['Rate'].quantile(0.75)
IQR_rate = Q3_rate - Q1_rate

lower_bound_rate = Q1_rate - 1.5 * IQR_rate
upper_bound_rate = Q3_rate + 1.5 * IQR_rate

# Filter the data
filtered_data = sick_leave_pay_data[
    (sick_leave_pay_data['AverageSickLeaveHours'] >= lower_bound_sick_leave) &
    (sick_leave_pay_data['AverageSickLeaveHours'] <= upper_bound_sick_leave) &
    (sick_leave_pay_data['Rate'] >= lower_bound_rate) &
    (sick_leave_pay_data['Rate'] <= upper_bound_rate)
]

# Calculate the correlation on filtered data
correlation_filtered = filtered_data[['AverageSickLeaveHours', 'Rate']].corr()

# Display the correlation matrix
print("Correlation Matrix (Filtered Data):")
print(correlation_filtered)

# Extract just the correlation coefficient
correlation_coefficient_filtered =
correlation_filtered.loc['AverageSickLeaveHours', 'Rate']
```

```
print(f"Correlation between Average Sick Leave Hours and Rate of Pay
(Filtered): {correlation_coefficient_filtered}")
```

This is the code to calculate the correlation coefficient between rate of pay and average sick leave hours, taking into account outliers:

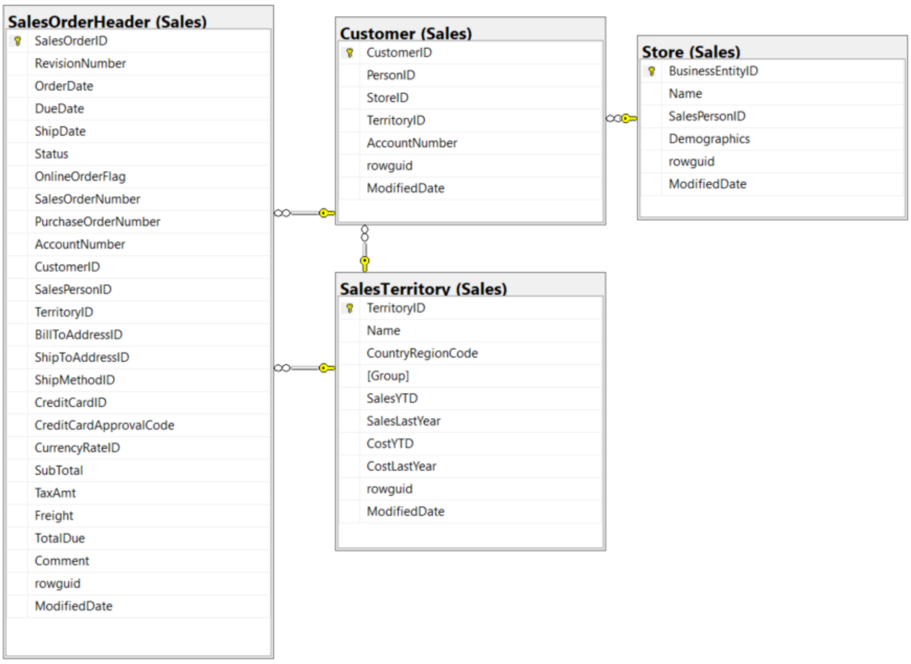
```
Correlation Matrix (Filtered Data):
                              AverageSickLeaveHours    Rate
AverageSickLeaveHours                1.00000          0.06231
Rate                                 0.06231          1.00000
Correlation between Average Sick Leave Hours and Rate of Pay (Filtered):
0.06230969997768214
```

From this result, we can see that the correlation between the two variables, considering outliers, results in 0.062. Without taking into account outliers, this coefficient is: -0.003413 which is even closer to 0 than the previous result. Regardless, both results are inconclusive, and we cannot determine any relationship between the two variables.

Using a similar code, we found that the organisation level also has a similar inconclusive result of 0.030709259113 which makes sense, since organisation level and rate of pay have a strong correlation with each other. This brings into question the original findings which found that within the top 10 job titles who used the most sick hours, job titles that were level 4 within the organisation had the highest average sick leave hours. This may have been due to probability, rather than correlation, as there are many more roles in level 4, compared with the other levels, so there was a higher probability of level 4 being in the top 10. As a result, there seems to be no significant relationship between sick leave and job title.

4.5 The Relationship between Store Trading Duration and Revenue

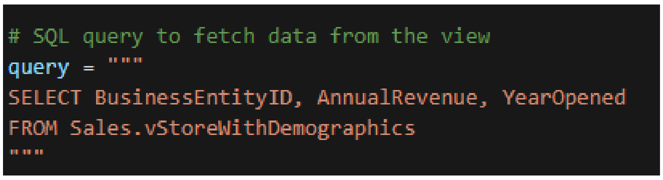
Data Collection Diagram



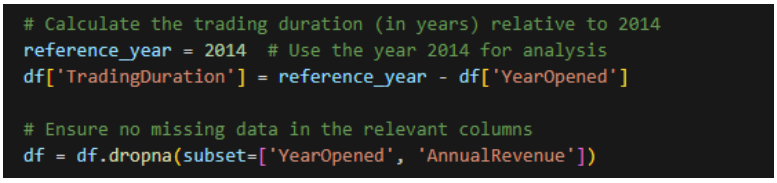
The data for this analysis was extracted from various sources as shown on the above. In addition, AdventureWorks Retail Stores Demographics Survey (‘Survey’) was also extracted from a view [Sales.vStoreWithDemographics] of the database. The following process was followed:

- **Data Extraction**: We extracted store-related data (such as trading duration, size, and location) from the Survey (xml data which was already extracted in a view, [Sales.vStoreWithDemographics]).
- **Sales Data**: The [Sales Order Header] table provided the total sales revenue per store (“TotalDue”) for the years 2011 to 2014.
- **Data Cleansing**: “is not null” function is applied for filtering out unrelated data. To ensure only relevant data were included for analysis.
- **Data Aggregation**: Data was grouping by store duration, total revenue per store duration.

Initially, it was considered analysing the relationship between store trading duration and sales revenue based on data from only one year (2014) in the [Sales.vStoreWithDemographics] view.

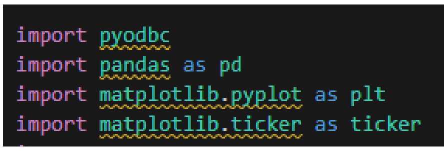


```
# SQL query to fetch data from the view
query = """
SELECT BusinessEntityID, AnnualRevenue, YearOpened
FROM Sales.vStoreWithDemographics
"""
```

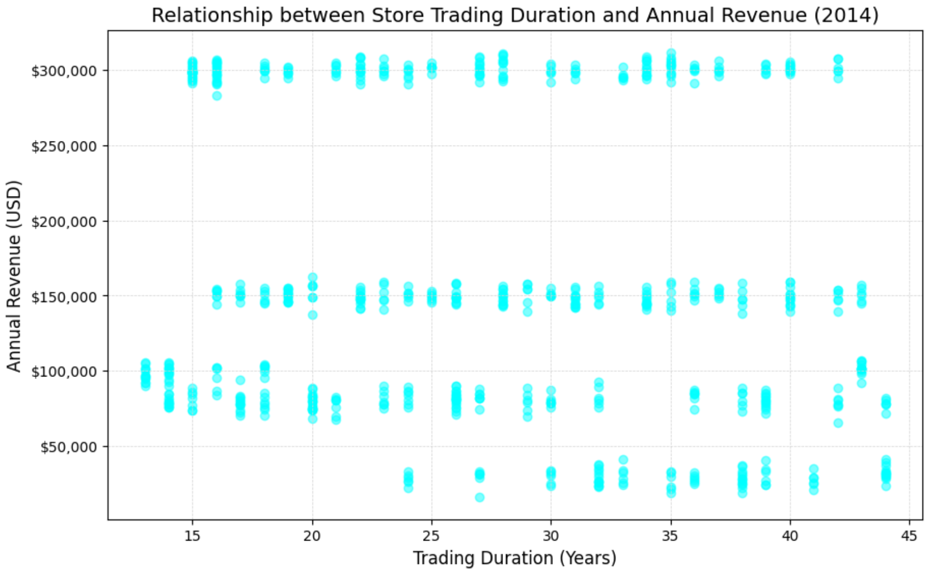


```
# Calculate the trading duration (in years) relative to 2014
reference_year = 2014  # Use the year 2014 for analysis
df['TradingDuration'] = reference_year - df['YearOpened']

# Ensure no missing data in the relevant columns
df = df.dropna(subset=['YearOpened', 'AnnualRevenue'])
```

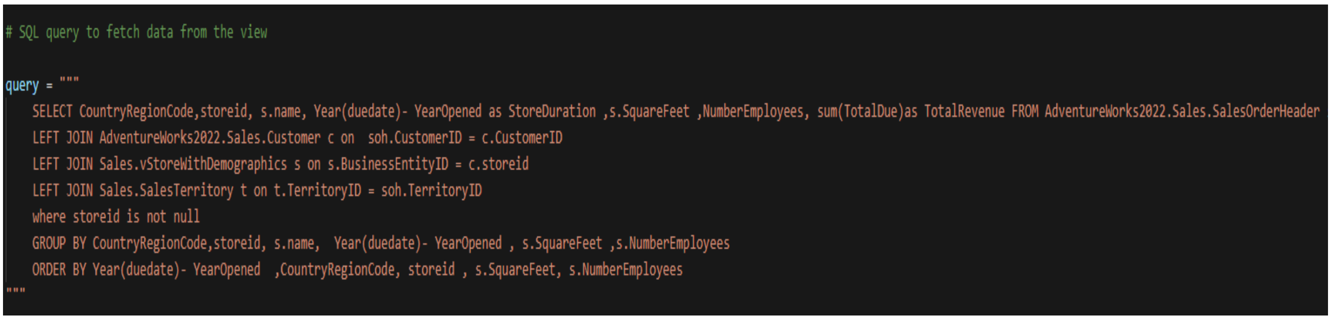
Visualizing the Data: Using Python in Visual Studio Code



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

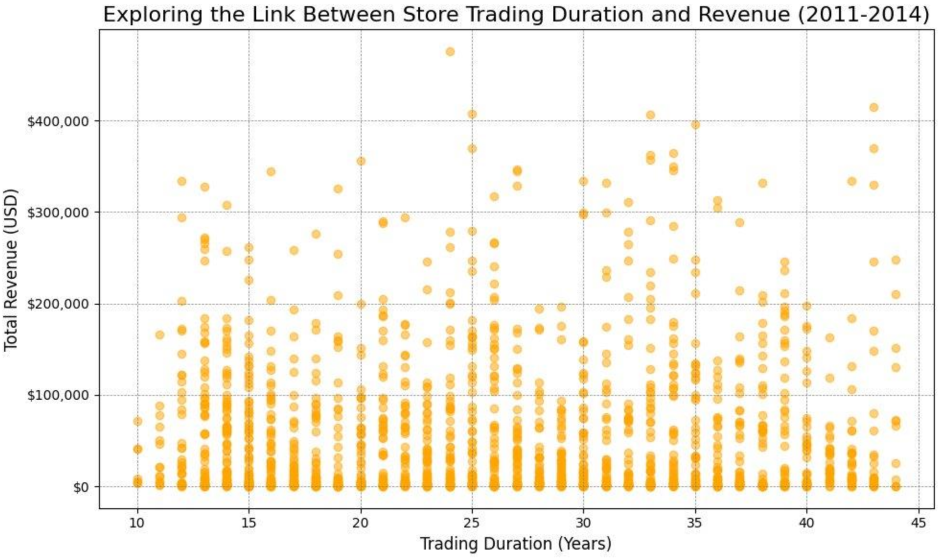


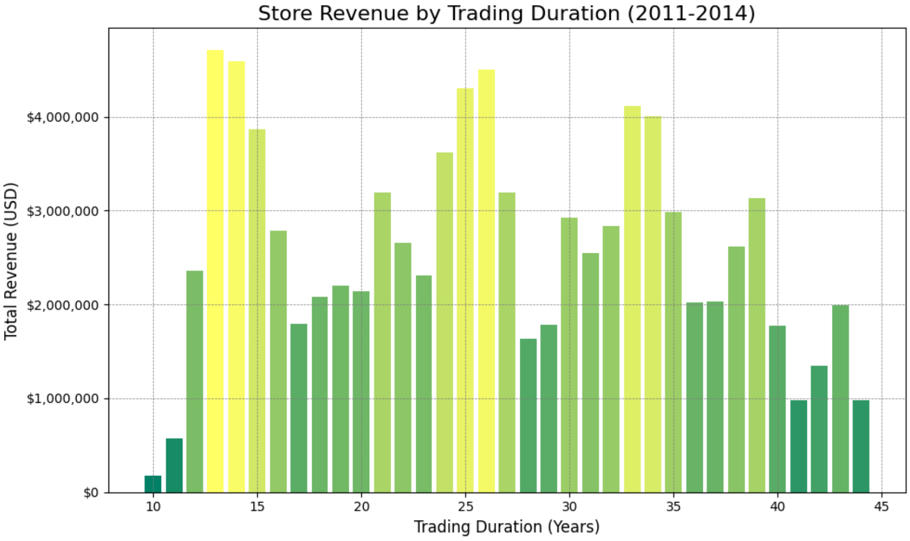
However, using just one year's data proved insufficient for a comprehensive analysis as it did not provide enough insight into long-term trends. To improve the analysis, we extended the dataset to include sales data from 2011 to 2014, extracted from the [Sales Order Header] table, offering a broader time frame for more reliable results.



```
# SQL query to fetch data from the view

query = """
    SELECT CountryRegionCode,storeid, s.name, Year(duedate)- YearOpened as StoreDuration ,s.SquareFeet ,NumberEmployees, sum(TotalDue)as TotalRevenue FROM AdventureWorks2022.Sales.SalesOrderHeader
    LEFT JOIN AdventureWorks2022.Sales.Customer c on  soh.CustomerID = c.CustomerID
    LEFT JOIN Sales.vStoreWithDemographics s on s.BusinessEntityID = c.storeid
    LEFT JOIN Sales.SalesTerritory t on t.TerritoryID = soh.TerritoryID
    where storeid is not null
    GROUP BY CountryRegionCode,storeid, s.name,  Year(duedate)- YearOpened , s.SquareFeet ,s.NumberEmployees
    ORDER BY Year(duedate)- YearOpened  ,CountryRegionCode, storeid , s.SquareFeet, s.NumberEmployees
"""
```

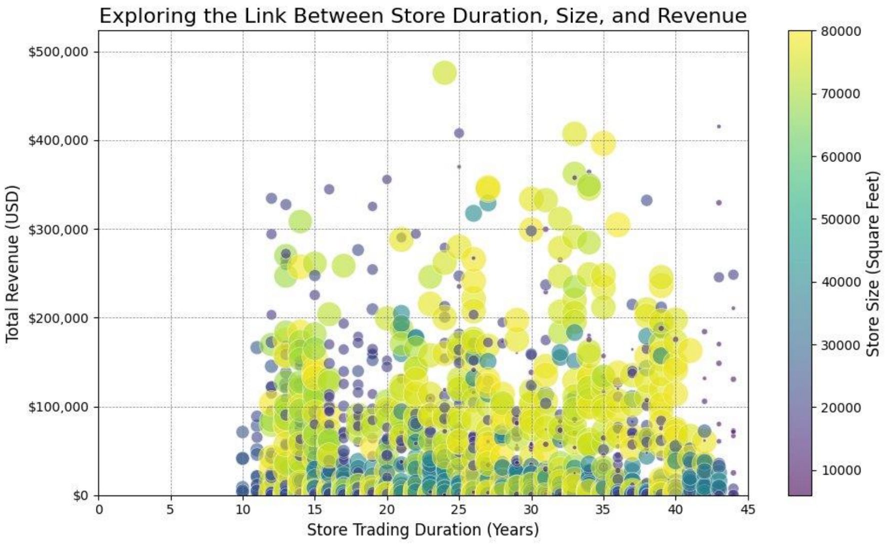
Justification for Using Both Bar Chart and Scatter Plot: Both bar chart and scatter plot were created. The bar chart offered a clear, aggregated view of total revenue for each store duration, allowing us to easily compare the overall trends. On the other hand, the scatter plot allows us to examine the data at a finer level of detail, revealing any potential correlations or outliers within the dataset. By using both visualizations, we aimed to gain a more complete understanding of how store duration impacts revenue, as each chart highlights various aspects of the data.





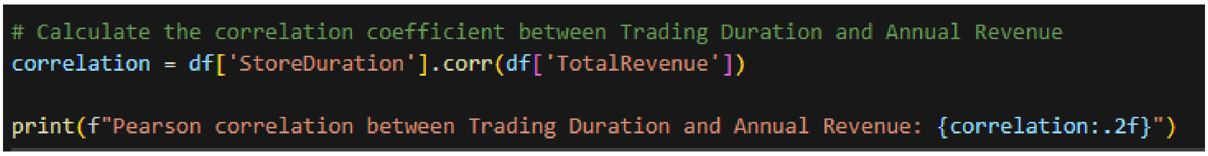
Incorporating Store Size as a Third Variable: To gain a deeper understanding of the factors influencing sales revenue, store size is added as a third variable in the analysis. This allowed us to explore whether store size had any impact on the relationship between trading duration and revenue. By including store size, we aimed to better understand if larger stores, for example, might show a stronger correlation with higher revenues, regardless of how long the store has been in operation.

Visual Representation of Store Size: To incorporate store size as a variable in our analysis, the “viridis” colormap was applied to represent store sizes, with the colour gradient ranging from darker to lighter shades. A legend is provided for easy interpretation, helping to clarify the relationship between store size and sales revenue. This colour mapping enables a clear visual representation of how store size varies and its potential influence on sales performance.



Weak Relationship Between Store Duration and Sales Revenue: The results from both the scatter and bubble charts show that the data points are widely spread out across the plot. This dispersion indicates that there is no trend or clear pattern in the relationship between store duration and sales revenue. The absence of a trend suggests that the length of time a store has been in operation does not meaningfully influence its total sales revenue.

Consistent Findings Across Bar, Scatter, and Bubble Charts: The bar chart like the scatter and bubble charts, also shows no clear trend between store duration and sales revenue. The bars do not exhibit a consistent upward or downward pattern, further supporting the conclusion that there is no significant correlation between the two variables. This consistent result across all three chart types reinforces the idea that store duration does not have a meaningful impact on total sales revenue.



```
# Calculate the correlation coefficient between Trading Duration and Annual Revenue
correlation = df['StoreDuration'].corr(df['TotalRevenue'])

print(f"Pearson correlation between Trading Duration and Annual Revenue: {correlation:.2f}")
```

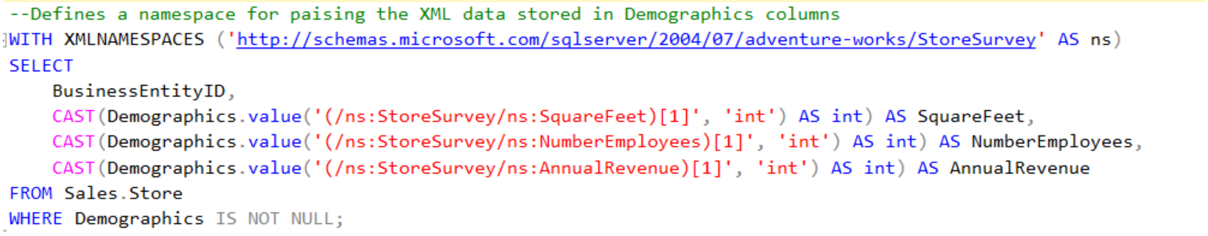
Correlation Analysis: By calculating the correlation coefficient, the analysis yielded a value of 0.02 between store trading duration and sales revenue. This indicates a very weak relationship. The near-zero correlation coefficient further confirms that the trading duration is not a strong predictor of revenue, reinforcing the observations from the scatter, bubble, and bar charts.

4.6 The relationship between the size of the stores, number of employees and revenue

In this task, our goal is to analyse the relationship among three variables. To achieve this, we first need to extract the relevant data from the Sales.Stores table in SQL. After extracting the data, we will clean it and create visualizations in Python. Specifically, we will use a pair plot and a correlation matrix to provide an overview of the relationships between the variables. Finally, we will examine the correlations between store sizes and the number of employees, store sizes and revenue, and the number of employees and revenue.

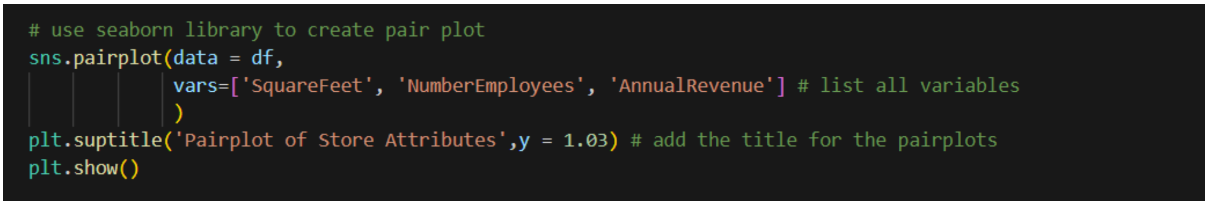
Data extraction and data cleaning

The required data, such as store size (*SquareFeet*), the number of employees, and annual revenue, is stored in an XML structure in the *Sales.Store* table. First, we need to extract this data from the XML and convert the extracted values into integers using the *CAST()* function in SQL to ensure they are stored in a numeric format suitable for analysis (see the code below). We extract only rows where the Demographics column contains non-null XML data. The results confirm that there are no missing values in the dataset and that all three variables are stored as integers. Subsequently, we connect SQL to Python and use libraries such as Pandas, Matplotlib, and Seaborn to create visualization plots.

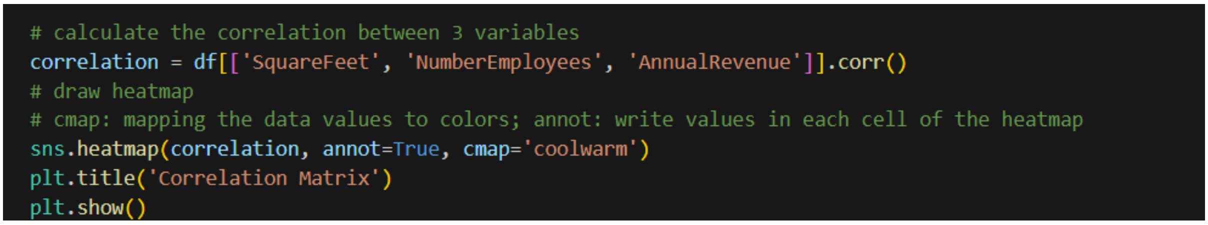


```sql
--Defines a namespace for paising the XML data stored in Demographics columns
WITH XMLNAMESPACES ('http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/StoreSurvey' AS ns)
SELECT
    BusinessEntityID,
    CAST(Demographics.value('(/ns:StoreSurvey/ns:SquareFeet)[1]', 'int') AS int) AS SquareFeet,
    CAST(Demographics.value('(/ns:StoreSurvey/ns:NumberEmployees)[1]', 'int') AS int) AS NumberEmployees,
    CAST(Demographics.value('(/ns:StoreSurvey/ns:AnnualRevenue)[1]', 'int') AS int) AS AnnualRevenue
FROM Sales.Store
WHERE Demographics IS NOT NULL;
```

To compare relationship among three variables, the simplest and most straightforward visualization is a pair plot from the Seaborn library. In addition to the plot, we can calculate the correlations between each pair of variables. Again, we can draw the correlation matrix by using the heatmap from Seaborn library. The heatmap's colours will indicate the strength of these relationships (see code below).



```python
# use seaborn library to create pair plot
sns.pairplot(data = df,
             vars=['SquareFeet', 'NumberEmployees', 'AnnualRevenue'] # list all variables
             )
plt.suptitle('Pairplot of Store Attributes',y = 1.03) # add the title for the pairplots
plt.show()
```



```python
# calculate the correlation between 3 variables
correlation = df[['SquareFeet', 'NumberEmployees', 'AnnualRevenue']].corr()
# draw heatmap
# cmap: mapping the data values to colors; annot: write values in each cell of the heatmap
sns.heatmap(correlation, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```

Overview of Pair plot and Correlation Matrix:

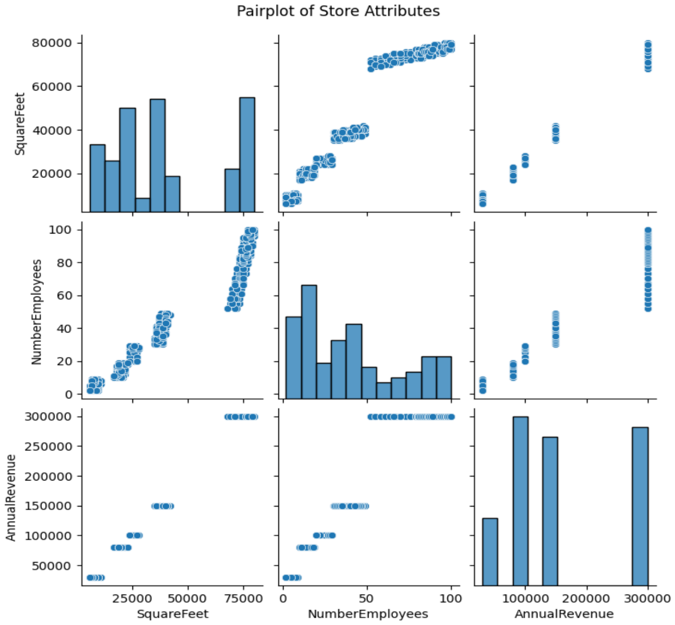
The diagonal plots illustrate the distributions of each variable using histograms:

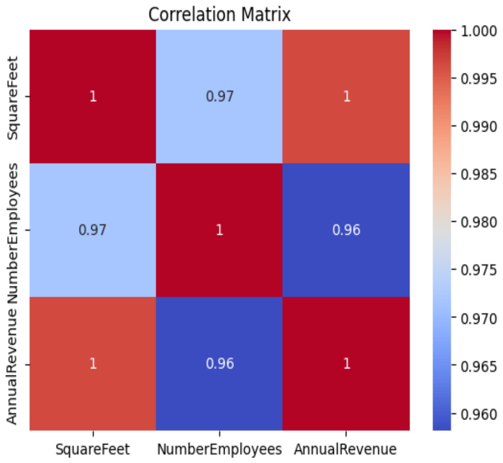
SquareFeet: The histogram (top-left) shows multiple peaks around 25,000, 30,000 and 75,000 square feet, indicating variability in store sizes.

Number of Employees: The histogram (middle) reveals that most stores have fewer than 50 employees, but there are still some stores employing up to 100 staff.

Annual Revenue: The histogram (bottom-right) shows most stores generate revenue around $100,000. However, some stores achieved revenue levels around $150,000 and $300,000.

The off-diagonal plots show the pairwise relationship among the three variables. These scatter plots highlight strong positive correlations, consistent with the correlation matrix.





Store Sizes vs. Number of Employees:

In this analysis, SquareFeet represents store sizes. The scatter plot(middle-left) reveals a strong linear relationship between store sizes and the number of employees. Larger stores tend to employ more staff, which is logical as bigger spaces require a larger workforce to manage operations effectively.

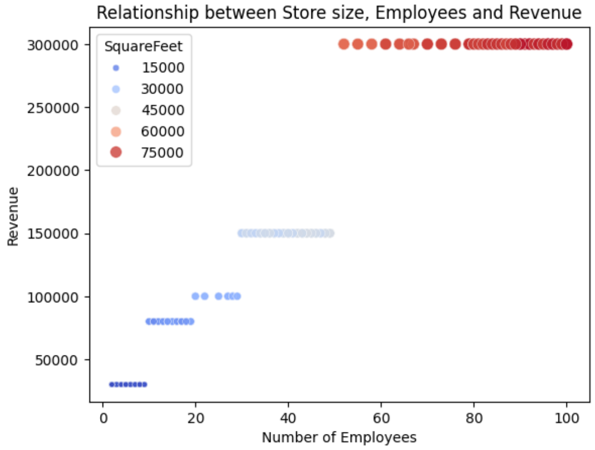
Number of Employees VS. Revenue:

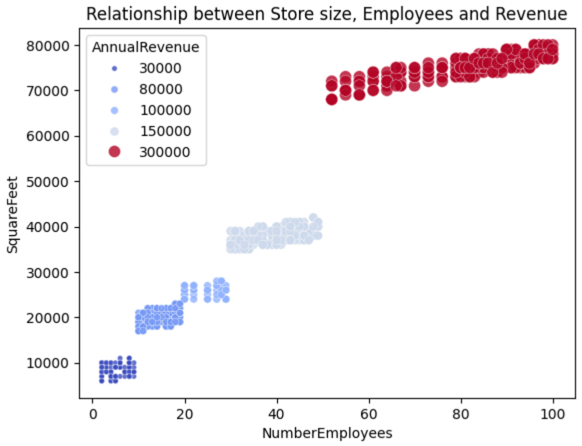
The scatter plot (bottom-middle) demonstrates that revenue is clustered into discrete levels, with only five distinct revenue levels in the dataset. Between these clusters, there is a strong positive relationship between the number of employees and revenue. Specifically, stores with higher revenue levels generally have more employees.

However, within individual revenue levels, the relationship is not as clear. For example, stores with 3–9 employees consistently generate $30,000 in revenue, indicating that within a specific revenue range, the number of employees does not significantly influence revenue.

Store Sizes vs. Revenue:

The scatter plot for store sizes and revenue (bottom-left) indicates a similar trend to the relationship between employees and revenue. Larger stores generally achieve higher revenue levels. For instance, stores with 18,000 square feet generate more revenue than those with 6,000 square feet. However, stores with 6,000 square feet generate the same revenue as stores with 11,000 square feet.

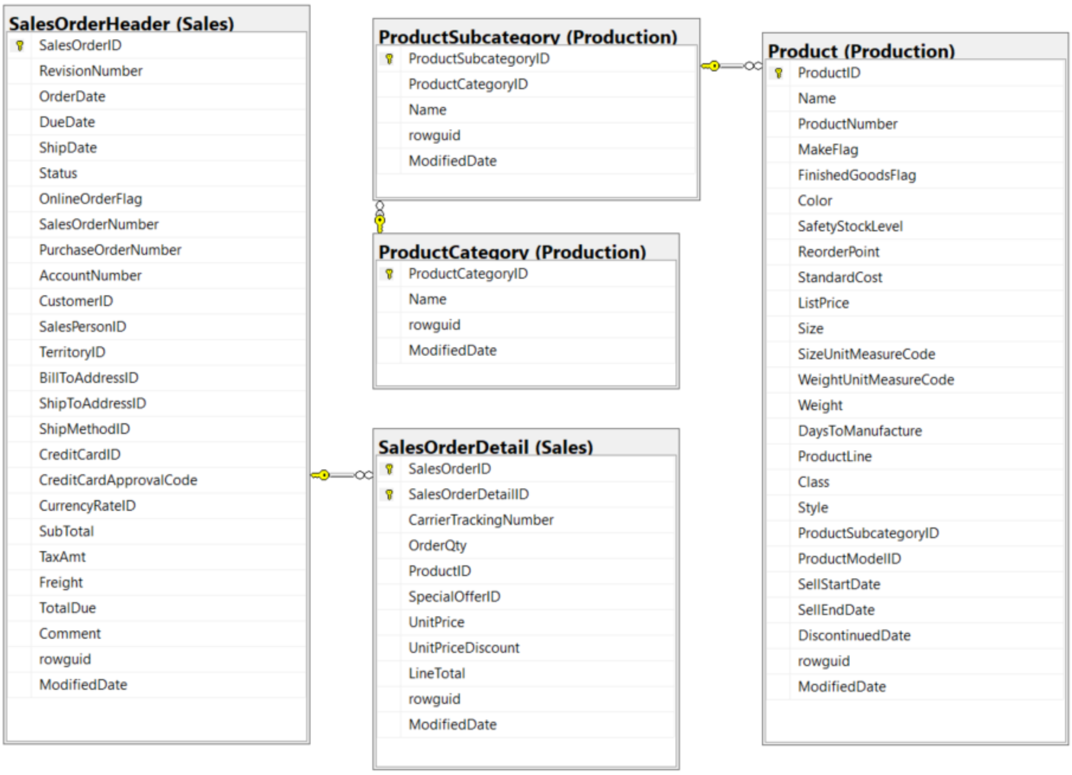




In summary, there are strong positive correlations among store sizes, number of employees and revenue. This means larger stores tend to employ more staff and generate higher revenue. However, within specific revenue clusters, increasing the number of employees does not necessarily result in higher revenue. The data suggests that the large stores with over 68,000 square feet and 52+ employees are associated with the highest revenue level. Thus, if this is the case, businesses could rent stores with 68,000 square feet and employ 52 employees to minimize labour and rental costs while maximizing profit. However, to validate this assumption, we need additional information and further analysis. Factors such as the store's location will also impact rental and labour costs, ultimately influencing the total cost and profit.

4.7 Sales Revenue Trend for Bikes Subcategories

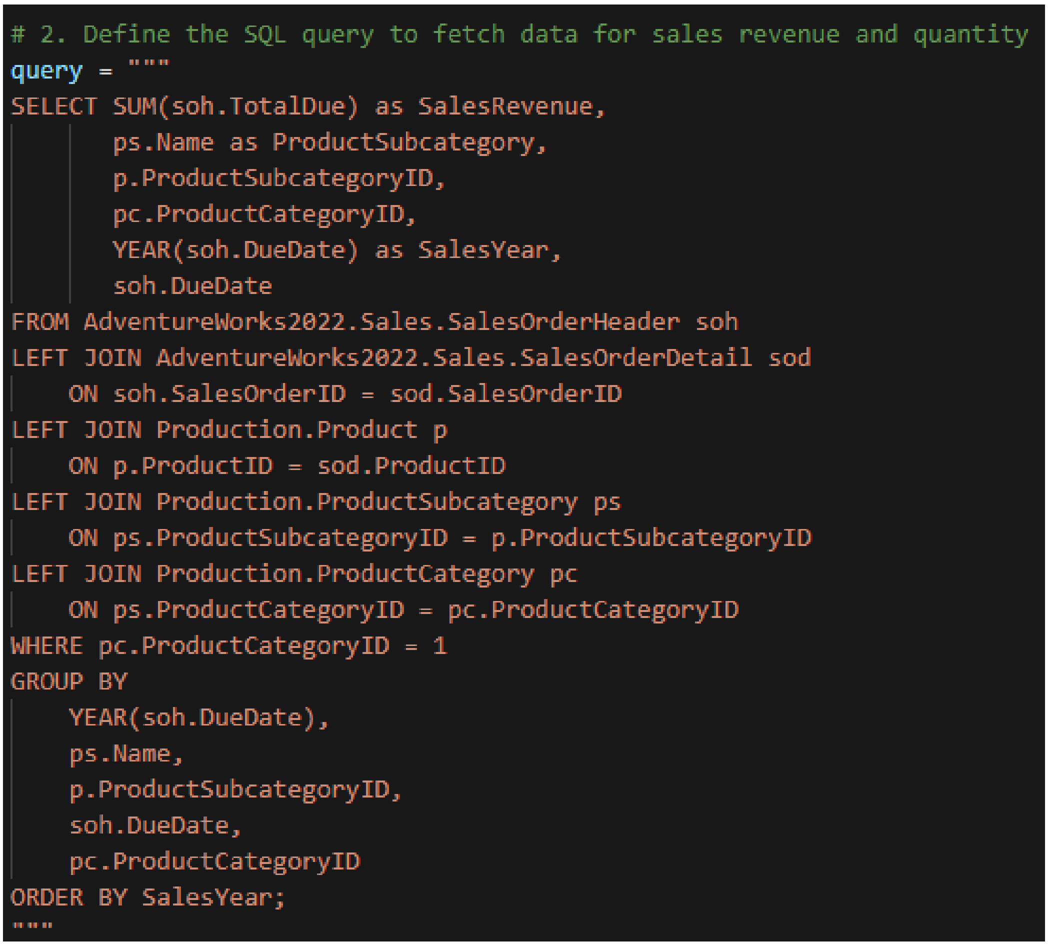
Data Collection Diagram



The data for this analysis was extracted from various sources as shown on the above. The following process was followed:

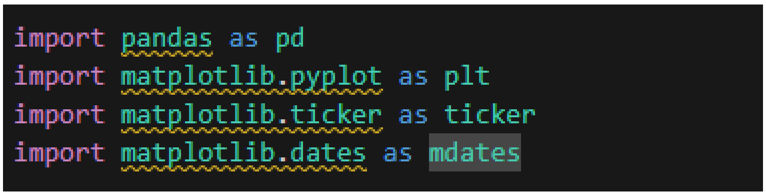
- **Data Extraction**: The primary source of sales revenue data came from the [SalesOrderHeader] table, which contains details about each sales order, especially the sales revenue (TotalDue) generated. It linked the [SalesOrderDetail], [Product], [ProductSubcategory], and [ProductCategory] tables to obtain data on bike product categories and subcategories (such as Mountain Bikes, Road Bikes, and Touring Bikes) to segment sales by product type.

  The [ProductSubcategory] table allowed us to differentiate between the various bike types, while the [ProductCategory] table helped filter the analysis to only include bikes, as the focus was on the bike product category (CategoryID = 1).

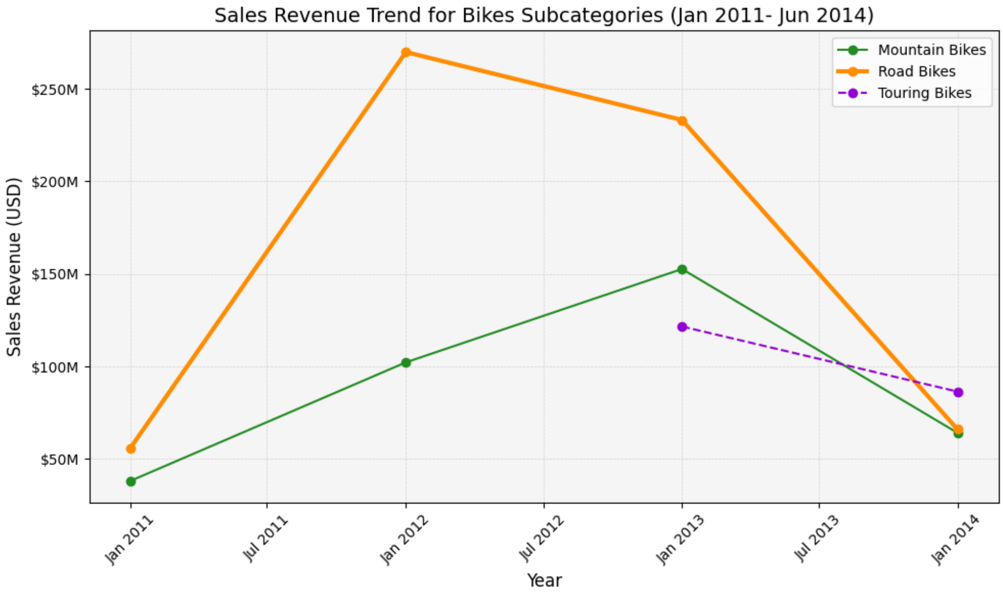


```
# 2. Define the SQL query to fetch data for sales revenue and quantity
query = """
SELECT SUM(soh.TotalDue) as SalesRevenue,
        ps.Name as ProductSubcategory,
        p.ProductSubcategoryID,
        pc.ProductCategoryID,
        YEAR(soh.DueDate) as SalesYear,
        soh.DueDate
FROM AdventureWorks2022.Sales.SalesOrderHeader soh
LEFT JOIN AdventureWorks2022.Sales.SalesOrderDetail sod
    ON soh.SalesOrderID = sod.SalesOrderID
LEFT JOIN Production.Product p
    ON p.ProductID = sod.ProductID
LEFT JOIN Production.ProductSubcategory ps
    ON ps.ProductSubcategoryID = p.ProductSubcategoryID
LEFT JOIN Production.ProductCategory pc
    ON ps.ProductCategoryID = pc.ProductCategoryID
WHERE pc.ProductCategoryID = 1
GROUP BY
    YEAR(soh.DueDate),
    ps.Name,
    p.ProductSubcategoryID,
    soh.DueDate,
    pc.ProductCategoryID
ORDER BY SalesYear;
"""
```

Visualizing the Data: Using Python in Visual Studio Code



```
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
import matplotlib.dates as mdates
```



The analysis of sales revenue for the bike subcategories—Road Bikes, Mountain Bikes, and Touring Bikes—reveals significant trends and patterns over the period from January 2011 to June 2014.

Road Bikes:

- **2011-2012:** The sales revenue saw a substantial increase of 382% (or 4.82 times) from $56 million in 2011 to $270 million in 2012. This dramatic growth could be attributed to numerous factors, including increased consumer demand, effective marketing campaigns, or the launch of new models that resonated with the target market. Road Bikes saw continued growth, indicating that this product line was particularly successful in this period.
- **2013-2014:** The 72% decrease in 2014 indicates a much more drastic downturn, which may have been caused by a combination of internal and external factors

Mountain Bikes:

- **2011-2012**: The early success of Mountain Bikes can be attributed to the rise in outdoor recreation, fitness trends, and effective product marketing. The market was expanding rapidly, with a clear surge in demand for these bikes.
- **2012-2013**: The momentum continued in 2013, although at a slower pace. The growth was driven by sustained interest in outdoor activities and improved product offerings, leading to a solid increase in revenue.
- **2013-2014**: The 58% drop in 2014 suggests that the market became saturated, consumer preferences shifted, and possibly economic or seasonal factors influenced the decline. Increased competition from other types of bikes (like Touring Bikes and Hybrid Bikes) and a slowdown in innovation may have also played a significant role in reducing demand for Mountain Bikes.

Touring Bikes:

- **2013**: Touring bikes was launched by AdventureWorks in 2013, and it quickly established a solid presence in the market, generating $122 million in sales in its first year. This strong start is noteworthy, as it mirrored the success of both the road bikes and mountain bikes during their initial sales years. Despite facing similar market

conditions, Touring Bikes displayed a unique pattern in sales performance.
- **2014:** The Touring Bikes category experienced a drop in sales, from $122 million to $86 million. While this represents a 29.5% decrease, the Touring Bikes segment outperformed both the Road Bikes and Mountain Bikes categories, which saw larger drops in revenue. The drop in Touring Bikes sales can likely be attributed to several factors, including the rise of bike-sharing programs, changing consumer preferences, and perhaps market saturation, which affected the entire bike industry.

Summary:

- **Olympic Games Influence**: The rise in bike sales during Olympic years, particularly in 2012, is consistent with trends identified by industry reports like those from the Association of Cycle Traders and Wikipedia. These reports highlight that Olympic events provide a significant catalyst for increased participation in cycling sports and related products.
- **Declining Sales Post-2012**: The decline in sales in 2013 and 2014 appears strongly linked to the growth of bike-sharing programs and the emergence of alternative transport options. According to source from Statista and Wikipedia, bike-sharing programs have become a key factor influencing the bike market, especially in urban areas.

5. Conclusion

The United States leads in sales, with the Southwest region recording the highest at $10,510,853.87. Other regions show lower sales, highlighting the need for strategy adjustments to boost growth. This chart helps identify regional trends and allocate resources effectively.

In conclusion for question 2, the analysis explored the relationship between annual leave and bonus using statistical methods and visualizations, revealing valuable insights. These findings can inform decisions on employee benefits and compensation, with further analysis possible to explore other influencing factors. Due to the limited sample size, we cannot conclusively establish that vacation hours directly affect commission. Further analysis with more data or additional factors may provide clearer insights.

In conclusion, overseas sales lag far behind compared to domestic sales due to strict international tariffs on the products. Data shows that neighbouring country Canada outperforms the wealthier overseas countries suggesting that going forwards, focusing on domestic and neighbouring markets will be easier than navigating global tariffs that can be volatile.

The analysis revealed no significant relationship between job title and sick leave hours, as evidenced by the correlation between related variables being close to zero. For instance, the correlation between organisation level and sick leave hours was 0.0307, indicating no conclusive evidence of a meaningful association between these factors.

Relationship between Store Duration and Sales Revenue: The analysis revealed a very weak correlation between store trading duration and sales revenue which indicated the length of time a store has been in operation is not a major factor in driving higher sales. In other words, other factors might have a more substantial impact on revenue generation.

From our analysis, we observe a very strong positive relationship between store size, the number of employees, and revenue. This result is logical, as larger stores typically require a bigger workforce and generate higher revenue. However, the dataset indicates that to optimize costs, businesses can achieve the highest revenue with a store size of 68,000 square feet and 52 employees.

In the additional insight, we focus on the analysis of the sales trends of the bikes as it is the primary source of generating the revenue for AdventureWorks. The initial years (2011-2012) saw growth in sales across all bike subcategories, 2013 and 2014 presented challenges in maintaining that growth. The Olympic Games in 2012 likely acted as a significant driver for the spike in bike sales, as large-scale events tend to boost consumer spending on sports and recreational activities. On the other hand, the rise of bike-share programs in the later years may have influenced the decline in sales, as these programs became more popular and reduced the need for individuals to purchase bikes.   In conclusion, the sales performance of road bikes, mountain bikes, and touring bikes was more influenced by external factors, such as global events and emerging trends. Future analyses could explore additional variables or external factors to better understand and predict bike sales trends in the market.

6. Limitations

One limitation of this analysis is that it is based solely on historical sales data from 2011 to 2014. The absence of more recent data may hinder the ability to capture current trends or seasonal fluctuations that could impact sales performance. Additionally, this analysis does not account for other influential factors such as marketing efforts, changes in consumer behaviour, or customer demographics, which could provide valuable context and insights into store performance and revenue patterns.

Furthermore, there is no data available on sales returns in the dataset. Sales returns can significantly affect net revenue, and the exclusion of this data may result in an overestimation of store performance and revenue. Future analyses incorporating returns data would provide a more accurate representation of store performance and profitability.

Another limitation from the dataset is that the data is not continuous, especially for store size and revenue levels. There are only a few discrete levels, and they are all rounded to thousands, which is unrealistic in real-world scenarios, particularly for revenue levels. Additionally, there are other issues, such as the dataset being too small, thus the result may be biased.

7. Recommendations

The data used in this analysis is last updated to year 2014, which is over 10 years old and does not account for significant events like the COVID-19 pandemic. To gain more accurate insights into current performance trends, it is recommended to update the analysis with more recent data, ideally from 2019 to 2023. This would reflect changes in consumer behaviour, business operations, and other post-pandemic factors that could impact store performance.

In future analysis, it would be beneficial to include variables that reflect more recent economic and social factors. For instance, incorporating data on online sales, changes in customer demographics, or the impact of marketing campaigns could provide a more comprehensive view of the factors influencing store performance.

One potential area for further analysis is to gather information on rental costs for various store sizes and employee salaries. This would allow us to calculate the total cost for each store and help determine the optimal store size and number of employees needed to maximize revenue while minimizing costs.

8. References

- “Bicycle-sharing system” - Wikipedia Bicycle-sharing system - Wikipedia

- “Bike-Sharing Clicks Into Higher Gear” - by Felix Richter, Statista, Jul 3 2018 https://www.statista.com/chart/14542/bike-sharing-programs-worldwide/

- “Cycling at the 2012 Summer Olympics” - Wikipedia Cycling at the 2012 Summer Olympics - Wikipedia

- “Demand for Bikes Soars Following Olympic Cycling Events, New Research Suggests.” posted on 9 Sep 2024 in Association of Cycle Trader Demand for bikes soars following Olympic cycling events, new research suggests. - News - The ACT

9. Appendices - SQL, Python queries & Additional Charts

A. Regional sales in the best performing country

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

file1 = pd.read_csv("c:/Users/AliM(DA-NAT4)/Desktop/SalesTerritory.csv")
file2 = pd.read_csv("c:/Users/AliM(DA-NAT4)/Desktop/SalesOrderHeaderSalesReason.csv")
file3 = pd.read_csv("c:/Users/AliM(DA-NAT4)/Desktop/CountryRegionCurrency.csv")

print("File 1:")
print(file1.head())
print("\nFile 2:")
print(file2.head())
print("\nFile 3:")
print(file3.head())
```

```
 mport matplotlib.pyplot as plt
import matplotlib.ticker as ticker

# Data for the US regions
territories = ["Southwest", "Northwest","Central", "Southeast", "Northeast"]
sales_ytd = [10510853.8739, 7887186.7882, 3072175.118, 2538667.2515,
2402176.8476]  # Sales YTD

# Determine the best-performing region
best_region = territories[sales_ytd.index(max(sales_ytd))]
best_sales = max(sales_ytd)

# Create the bar chart
plt.figure(figsize=(10, 6))
bars = plt.bar(territories, sales_ytd, color='#4CAF50', edgecolor='black',
linewidth=1.2)

# Format sales numbers with a dollar sign
plt.gca().yaxis.set_major_formatter(ticker.FuncFormatter(lambda x, _:
'${:,.0f}'.format(x)))

# Title and labels
plt.title('Regional Sales in the Best Performing Region: US', fontsize=16,
fontweight='bold', color='darkslategray')
plt.xlabel('Territories', fontsize=12, fontweight='bold',
color='darkslategray')
plt.ylabel('Sales (USD)', fontsize=12, fontweight='bold',
color='darkslategray')

# Add annotation for the best-performing country
plt.text(
    5.2,  # X-coordinate for the text box
    best_sales * 0.8,  # Y-coordinate for the text box (adjusted to align near
the chart)
    f"Best Country: US\nBest Region: {best_region}\nSales:
${best_sales:,.2f}",
    fontsize=12,
    color='darkslategray',
    bbox=dict(facecolor='#f7f7f7', edgecolor='gray', boxstyle='round,pad=0.5')
)

# Tweak tick parameters
```

```
plt.tick_params(axis='x', labelrotation=45, labelsize=10)
plt.tick_params(axis='y', labelsize=10)

# Add gridlines for better readability
plt.grid(axis='y', linestyle='--', alpha=0.7)

# Adjust the background
plt.gca().set_facecolor('#f7f7f7')

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

B. The relationship between annual leave taken and bonus

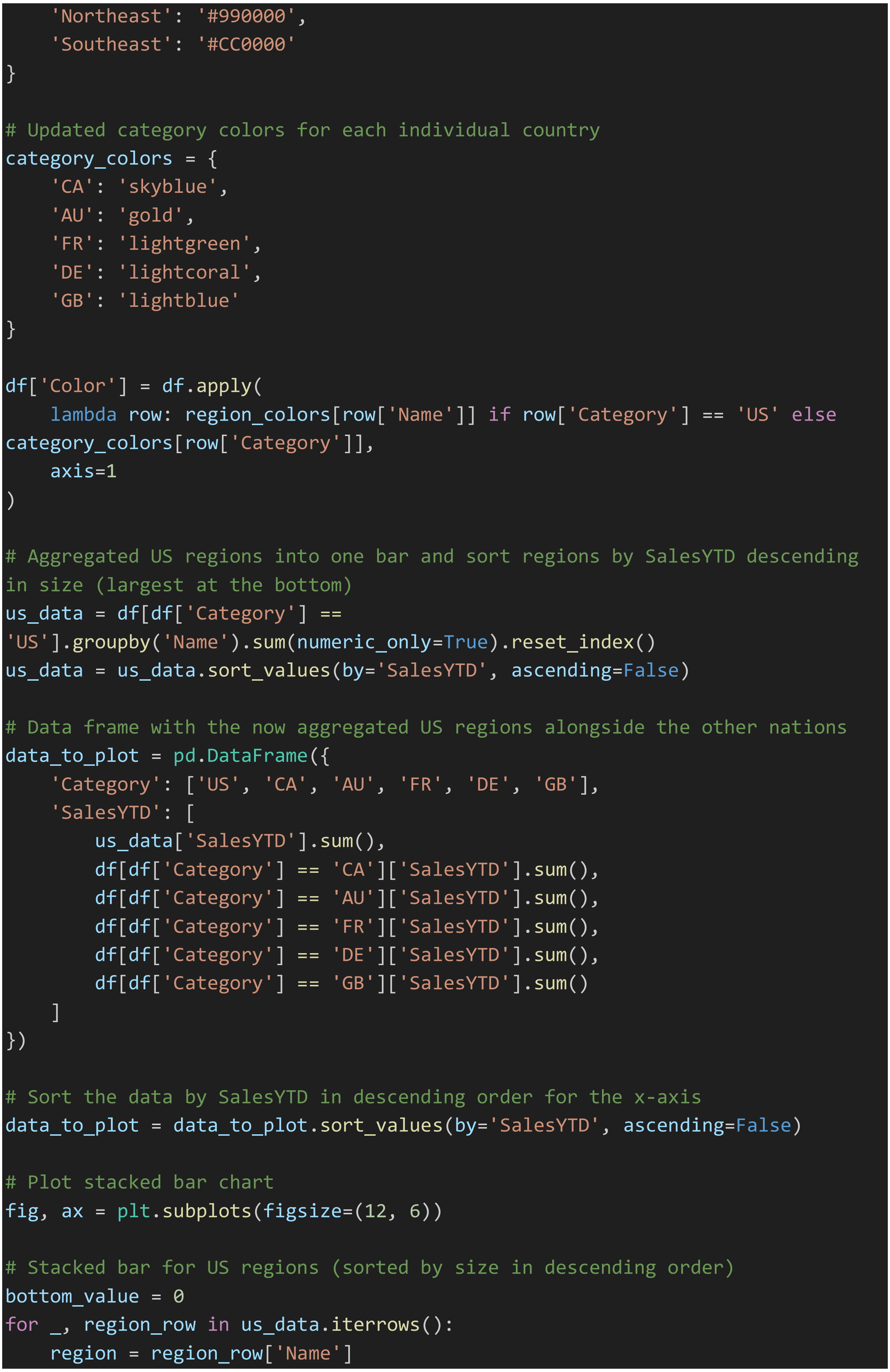
```
import pandas as pd

data_path = "C:\\Users\\AbdiS(DA-NAT4)\\Downloads\\HumanResources_Employee.csv"
data = pd.read_csv(data_path)
print(data.head())
data_path = "C:\\Users\\AbdiS(DA-NAT4)\\Downloads\\HumanResources_Employee.csv"
data = pd.read_csv(data_path)
print(data.head())
print("Original Data:")
print(data.head())
data_cleaned = data.dropna()
print("\nCleaned Data (Nulls Removed):")
print(data_cleaned.head())

data = {
    'BusinessEntityID': [1, 2, 3, 4],
    'VacationHours': [40, 50, 60, 70],
    'CommissionPct': [0.05, 0.10, 0.08, 0.12]
}

# Creating DataFrame
df = pd.DataFrame(data)

correlation = df['VacationHours'].corr(df['CommissionPct'])
print(f"Correlation between Annual Leave and Bonus: {correlation}")
```

```
import matplotlib.pyplot as plt

# Scatter plot to visualize the relationship between Annual Leave
(VacationHours) and Bonus (CommissionPct)
plt.scatter(df['VacationHours'], df['CommissionPct'], color='blue')

# Adding labels and title
plt.title("Relationship Between Annual Leave and Bonus")
plt.xlabel("Annual Leave (Vacation Hours)")
plt.ylabel("Bonus Percentage (CommissionPct)")

# Calculate average of VacationHours (Annual Leave) and CommissionPct (Bonus)
average_vacation_hours = df['VacationHours'].mean()
average_bonus = df['CommissionPct'].mean()

print(f"Average Annual Leave (Vacation Hours): {average_vacation_hours}")
print(f"Average Bonus Percentage (CommissionPct): {average_bonus}")

# Show plot
plt.show()

plt.scatter(df['VacationHours'], df['CommissionPct'], color='blue',
label='Data Points')

# Line plot to show the relationship between Annual Leave and Bonus
plt.plot(df['VacationHours'], df['CommissionPct'], color='blue', label='Line
Plot', marker='o')

# Adding labels and title
plt.title("Relationship Between Annual Leave and Bonus")
plt.xlabel("Annual Leave (Vacation Hours)")
plt.ylabel("Bonus Percentage (CommissionPct)")

# Calculate average of VacationHours (Annual Leave) and CommissionPct (Bonus)
average_vacation_hours = df['VacationHours'].mean()
average_bonus = df['CommissionPct'].mean()

print(f"Average Annual Leave (Vacation Hours): {average_vacation_hours}")
print(f"Average Bonus Percentage (CommissionPct): {average_bonus}")

# Show legend and plot
plt.legend()
plt.show()

import statsmodels.api as sm

# Defining the independent variable (X) and dependent variable (y)
```

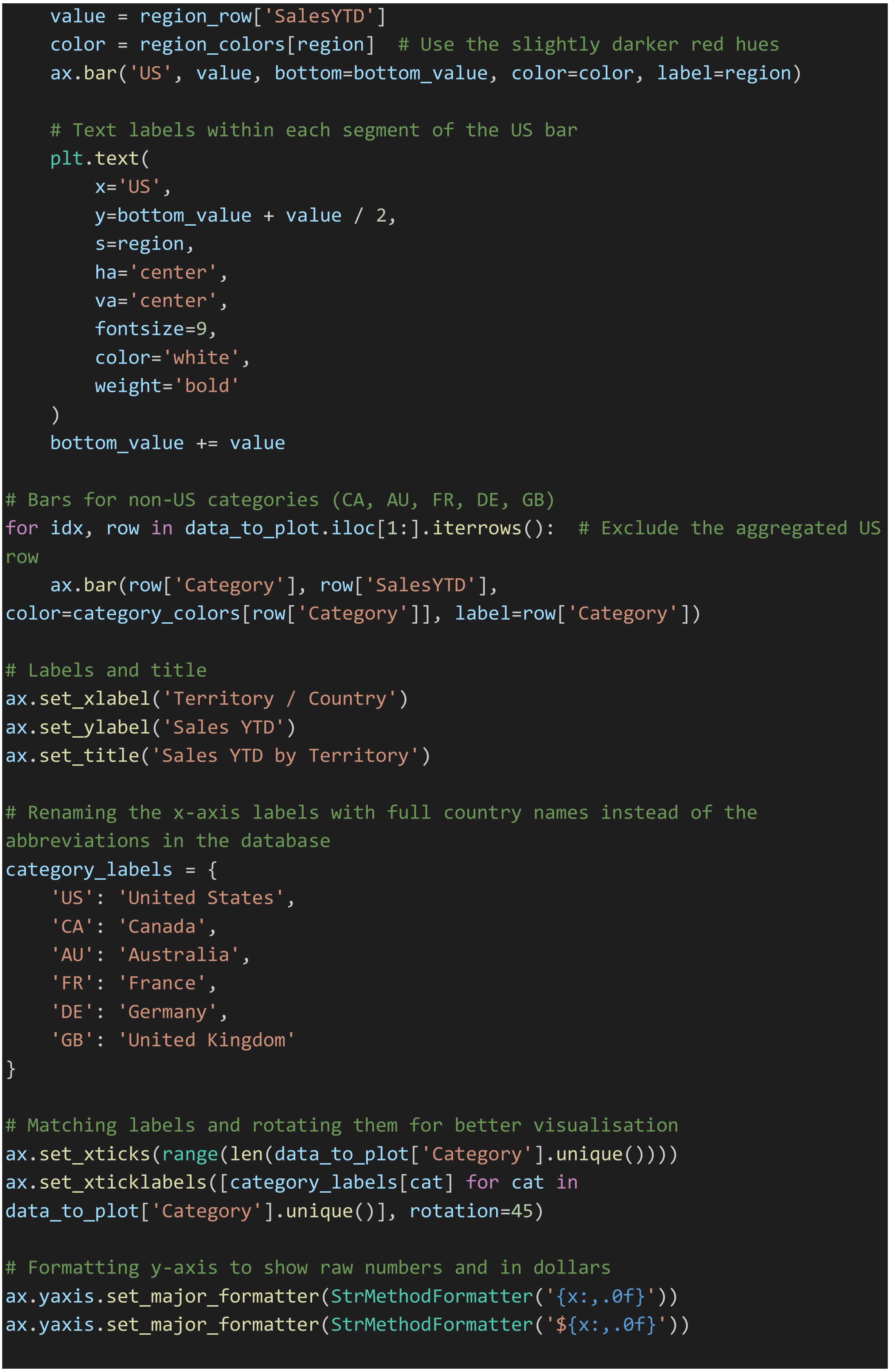
```python
X = df['VacationHours']
y = df['CommissionPct']

# Add constant to the model (for intercept)
X = sm.add_constant(X)

# Fit the linear regression model
model = sm.OLS(y, X).fit()

# Get the summary of the model
print(model.summary())
```

C. The relationship between Country and Revenue

```python
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.patches import Patch
from matplotlib.ticker import StrMethodFormatter

df = pd.read_csv(r'SalesTerritory.csv')

def assign_category(country_code, region):
    if country_code == 'US':
        return 'US'  # Aggregate all US regions under 'US'
    elif country_code == 'CA':
        return 'CA'
    elif country_code == 'AU':
        return 'AU'
    elif country_code == 'FR':
        return 'FR'
    elif country_code == 'DE':
        return 'DE'
    elif country_code == 'GB':
        return 'GB'
    else:
        return None

# Assign categories and filter out "None" (empty) categories
df['Category'] = df.apply(lambda row:
assign_category(row['CountryRegionCode'], row['Name']), axis=1)
df = df[df['Category'].notnull()]

# US region hues in the stacked bar
region_colors = {
    'Northwest': '#FF6666',
    'Southwest': '#FF9999',
    'Central': '#FF3333',
```



```
    'Northeast': '#990000',
    'Southeast': '#CC0000'
}

# Updated category colors for each individual country
category_colors = {
    'CA': 'skyblue',
    'AU': 'gold',
    'FR': 'lightgreen',
    'DE': 'lightcoral',
    'GB': 'lightblue'
}

df['Color'] = df.apply(
    lambda row: region_colors[row['Name']] if row['Category'] == 'US' else
category_colors[row['Category']],
    axis=1
)

# Aggregated US regions into one bar and sort regions by SalesYTD descending
in size (largest at the bottom)
us_data = df[df['Category'] ==
'US'].groupby('Name').sum(numeric_only=True).reset_index()
us_data = us_data.sort_values(by='SalesYTD', ascending=False)

# Data frame with the now aggregated US regions alongside the other nations
data_to_plot = pd.DataFrame({
    'Category': ['US', 'CA', 'AU', 'FR', 'DE', 'GB'],
    'SalesYTD': [
        us_data['SalesYTD'].sum(),
        df[df['Category'] == 'CA']['SalesYTD'].sum(),
        df[df['Category'] == 'AU']['SalesYTD'].sum(),
        df[df['Category'] == 'FR']['SalesYTD'].sum(),
        df[df['Category'] == 'DE']['SalesYTD'].sum(),
        df[df['Category'] == 'GB']['SalesYTD'].sum()
    ]
})

# Sort the data by SalesYTD in descending order for the x-axis
data_to_plot = data_to_plot.sort_values(by='SalesYTD', ascending=False)

# Plot stacked bar chart
fig, ax = plt.subplots(figsize=(12, 6))

# Stacked bar for US regions (sorted by size in descending order)
bottom_value = 0
for _, region_row in us_data.iterrows():
    region = region_row['Name']
```



```python
    value = region_row['SalesYTD']
    color = region_colors[region]  # Use the slightly darker red hues
    ax.bar('US', value, bottom=bottom_value, color=color, label=region)

    # Text labels within each segment of the US bar
    plt.text(
        x='US',
        y=bottom_value + value / 2,
        s=region,
        ha='center',
        va='center',
        fontsize=9,
        color='white',
        weight='bold'
    )
    bottom_value += value

# Bars for non-US categories (CA, AU, FR, DE, GB)
for idx, row in data_to_plot.iloc[1:].iterrows():  # Exclude the aggregated US row
    ax.bar(row['Category'], row['SalesYTD'], color=category_colors[row['Category']], label=row['Category'])

# Labels and title
ax.set_xlabel('Territory / Country')
ax.set_ylabel('Sales YTD')
ax.set_title('Sales YTD by Territory')

# Renaming the x-axis labels with full country names instead of the abbreviations in the database
category_labels = {
    'US': 'United States',
    'CA': 'Canada',
    'AU': 'Australia',
    'FR': 'France',
    'DE': 'Germany',
    'GB': 'United Kingdom'
}

# Matching labels and rotating them for better visualisation
ax.set_xticks(range(len(data_to_plot['Category'].unique())))
ax.set_xticklabels([category_labels[cat] for cat in data_to_plot['Category'].unique()], rotation=45)

# Formatting y-axis to show raw numbers and in dollars
ax.yaxis.set_major_formatter(StrMethodFormatter('{x:,.0f}'))
ax.yaxis.set_major_formatter(StrMethodFormatter('${x:,.0f}'))
```

```
# Updating the legend to properly colour code the individual countries
legend_elements = [
    Patch(facecolor=color, label=category_labels[category]) for category,
color in category_colors.items()
] + [
    Patch(facecolor=color, label=region) for region, color in
sorted(region_colors.items(), key=lambda x: us_data.loc[us_data['Name'] ==
x[0], 'SalesYTD'].values[0], reverse=False)
]

ax.legend(handles=legend_elements, title="Regions and Categories")

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

D. The relationship between sick leave and Job Title

```
# Retrieve relevant columns from SQL database
query = """
SELECT BusinessEntityID, JobTitle, SickLeaveHours, OrganizationLevel
FROM HumanResources.Employee
"""

# Load the data into a Pandas DataFrame
employee_data = pd.read_sql(query, connection)

# Display the first few rows to understand the structure
print(employee_data.head())
# What is the relationship between sick leave and Job Title (PersonType)?
# Group by JobTitle and OrganizationLevel, calculate average SickLeaveHours
avg_sick_leave_by_job = (
    employee_data.groupby(['JobTitle', 'OrganizationLevel'])['SickLeaveHours']
    .mean()
    .reset_index()
)

# Sort by average SickLeaveHours in descending order
avg_sick_leave_by_job = avg_sick_leave_by_job.sort_values(by='SickLeaveHours',
ascending=False)

print(avg_sick_leave_by_job)
```

```
# Plot the results of average sick leave by job title
```

```
plt.figure(figsize=(12, 8))
sns.barplot(x='SickLeaveHours', y='JobTitle', data=avg_sick_leave_by_job,
palette='viridis')
plt.title('Average Sick Leave Hours by Job Title', fontsize=16)
plt.xlabel('Average Sick Leave Hours', fontsize=4)
plt.ylabel('Job Title', fontsize=4)
plt.gca().tick_params(axis='y', labelsize=8)
# plt.show()
```

```
# Extract the top 10 job titles with the highest average SickLeaveHours
top_10_sick_leave = avg_sick_leave_by_job.head(10)

# Display the top 10 job titles
print(top_10_sick_leave)
```

```
# Plot the results for the top 10 job titles, with OrganizationLevel
represented by colour
plt.figure(figsize=(12, 8))
sns.barplot(
    x='SickLeaveHours',
    y='JobTitle',
    hue='OrganizationLevel',  # Add OrganisationLevel as a colour
    data=top_10_sick_leave,
    palette='coolwarm'
)

# Add titles and labels
plt.title('Top 10 Job Titles with the Most Sick Leave Hours (by Organization
Level)', fontsize=16)
plt.xlabel('Average Sick Leave Hours', fontsize=12)
plt.ylabel('Job Title', fontsize=12)
plt.legend(title='Organization Level', bbox_to_anchor=(1.05, 1), loc='upper
left')
plt.tight_layout()
plt.show()
```

```
# Filter job titles with a high sick leave proportion greater than 20%
threshold = 0.2
filtered_high_sick_leave_jobs =
high_sick_leave_by_job[high_sick_leave_by_job['HighSickLeave'] > threshold]

# Plot
sns.barplot(x='HighSickLeave', y='JobTitle',
data=filtered_high_sick_leave_jobs, palette='magma')
plt.title('Job Titles with >20% High Sick Leave Proportion', fontsize=14)
plt.xlabel('Proportion with High Sick Leave')
plt.ylabel('Job Title')
plt.text(0.0, -0.2, "Note: High sick leave is defined as the top 10% (90th percentile) of sick leave across the dataset.",
         fontsize=10, color='black', ha='center', va='center', transform=plt.gca().transAxes)
plt.tight_layout()
plt.show()

# Employees are considered to have "high sick leave" if their sick leave hours are above the 90th percentile of sick leave hours across the entire dataset.
# This is designed to flag any outliers or employees who take much more sick leave than their peers.
```

```
# Create a table showing the relationship between rate of pay and sick leave hours
# Group by Rate, then calculate average SickLeaveHours
sick_leave_rate_job = (
    merged_data.groupby(['Rate'])['SickLeaveHours']
    .mean()
    .reset_index()
)

# Identify and remove outliers using the IQR method
Q1 = sick_leave_rate_job['SickLeaveHours'].quantile(0.25)
Q3 = sick_leave_rate_job['SickLeaveHours'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Filter the data to exclude outliers
cleaned_data = sick_leave_rate_job[(sick_leave_rate_job['SickLeaveHours'] >= lower_bound) &
                                   (sick_leave_rate_job['SickLeaveHours'] <= upper_bound)]

# Step 4: Plot the scatter plot with trendline using the cleaned data
plt.figure(figsize=(14, 8))
sns.scatterplot(
    data=cleaned_data,
    x='Rate',
    y='SickLeaveHours',
    palette='tab10',
    s=100
)

# Add a regression trendline
sns.regplot(
    data=cleaned_data,
    x='Rate',
    y='SickLeaveHours',
    scatter=False,
    color='red',
    line_kws={"linewidth": 2}
)

# Customize the plot
plt.title('Average Sick Leave Hours by Rate of Pay', fontsize=16)
plt.xlabel('Rate of Pay', fontsize=12)
plt.ylabel('Average Sick Leave Hours', fontsize=12)
plt.grid(True, linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()
```

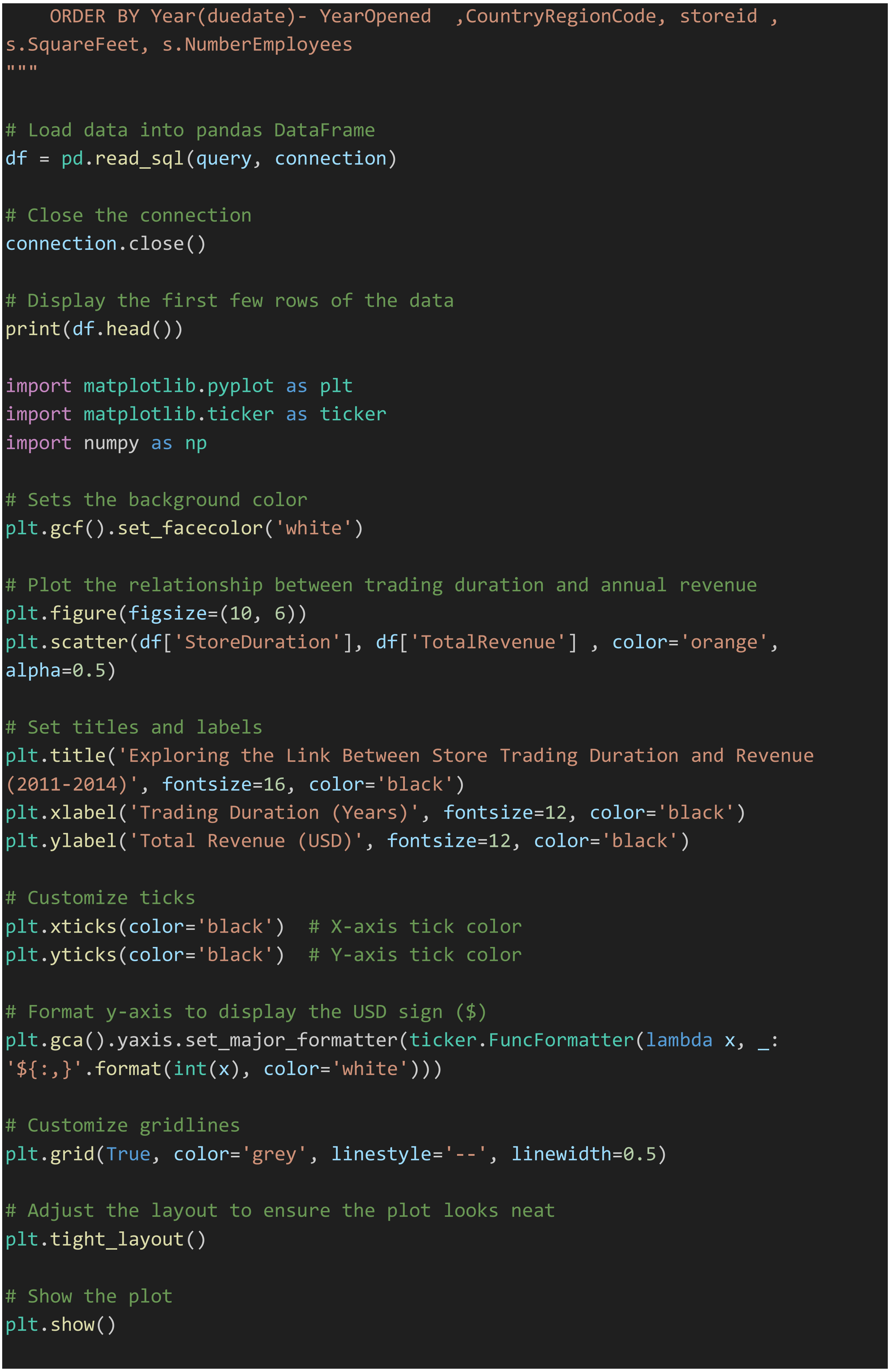
```
# To calculate the correlation between rate of pay and organisation level:
# Merge the two tables on BusinessEntityID
combined_data = pd.merge(employee_data, pay_history, on='BusinessEntityID',
how='inner')

# Calculate the correlation
correlation = combined_data[['OrganizationLevel', 'Rate']].corr()

# Display the correlation matrix
print("Correlation Matrix:")
print(correlation)

# Alternatively, extract just the correlation coefficient
correlation_coefficient = correlation.loc['OrganizationLevel', 'Rate']
print(f"Correlation between OrganizationLevel and Rate:
{correlation_coefficient}")
```

E. The relationship between store trading duration and revenue

```
# For plotting a Scatter Chart based on AdventureWorks Retail Stores
Demographics Survey Schema on April 2014

import pyodbc
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
import numpy as np

# Set up the connection string
connection_string = ("driver={odbc Driver 17 for SQL Server};"
                     "Server=DESKTOP-19LPOE8\SQLEXPRESS;"
                     "Database=AdventureWorks2022;"
                     "Trusted_Connection=yes")
connection = pyodbc.connect(connection_string)

# SQL query to fetch data from the view
query = """
SELECT BusinessEntityID, AnnualRevenue, YearOpened
FROM Sales.vStoreWithDemographics
"""

# Load data into pandas DataFrame
df = pd.read_sql(query, connection)

# Close the connection
connection.close()

# Calculate the trading duration (in years) relative to 2014
reference_year = 2014  # Use the year 2014 for analysis
df['TradingDuration'] = reference_year - df['YearOpened']

# Ensure no missing data in the relevant columns
df = df.dropna(subset=['YearOpened', 'AnnualRevenue'])

# Add random jitter to the 'AnnualRevenue' to avoid exact overlap
jitter = np.random.normal(0, 5000, size=df.shape[0])  # Small jitter

# Plot the relationship between trading duration and annual revenue
plt.figure(figsize=(10, 6))
plt.scatter(df['TradingDuration'], df['AnnualRevenue'] + jitter, color='aqua',
alpha=0.5)

# Set titles and labels
plt.title('Relationship between Store Trading Duration and Annual Revenue
(2014)', fontsize=14, color='black')
plt.xlabel('Trading Duration (Years)', fontsize=12, color='black')
plt.ylabel('Annual Revenue (USD)', fontsize=12, color='black')

# Customize x and y axis ticks visibility
plt.xticks(color='black')
plt.yticks(color='black')

# Customize the plot
plt.gca().set_facecolor('white')
plt.gcf().set_facecolor('white')

# Format y-axis to display the USD sign ($)
plt.gca().yaxis.set_major_formatter(ticker.FuncFormatter(lambda x, _:
'${:,}'.format(int(x), color='black')))
```

```
# Add grid lines for better readability
plt.grid(True, axis='both', linestyle='--', color='lightgray', linewidth=0.5)

# Adjust layout to ensure labels are not cut off
plt.tight_layout()

# Manually adjust the margins in case labels are cut off
plt.subplots_adjust(left=0.1, right=0.9, top=0.9, bottom=0.1)

# Show the plot
plt.show()


# Calculate the correlation coefficient between Trading Duration and Annual
Revenue
correlation = df['TradingDuration'].corr(df['AnnualRevenue'])

print(f"Pearson correlation between Trading Duration and Annual Revenue:
{correlation:.2f}")
```

```
# For plotting a Scatter Chart based on the data generated on 2010-2014

import pyodbc
import pandas as pd

# Set up the connection string
connection_string = ("driver={odbc Driver 17 for SQL Server};"
                     "Server=DESKTOP-19LPOE8\SQLEXPRESS;"
                     "Database=AdventureWorks2022;"
                     "Trusted_Connection=yes")
connection = pyodbc.connect(connection_string)


# SQL query to fetch data from the view

query = """
    SELECT CountryRegionCode,storeid, s.name, Year(duedate)- YearOpened as
StoreDuration ,s.SquareFeet ,NumberEmployees, sum(TotalDue)as TotalRevenue
FROM AdventureWorks2022.Sales.SalesOrderHeader soh
    LEFT JOIN AdventureWorks2022.Sales.Customer c on  soh.CustomerID =
c.CustomerID
    LEFT JOIN Sales.vStoreWithDemographics s on s.BusinessEntityID = c.storeid
    LEFT JOIN Sales.SalesTerritory t on t.TerritoryID = soh.TerritoryID
    where storeid is not null
    GROUP BY CountryRegionCode,storeid, s.name,  Year(duedate)- YearOpened ,
s.SquareFeet ,s.NumberEmployees
```



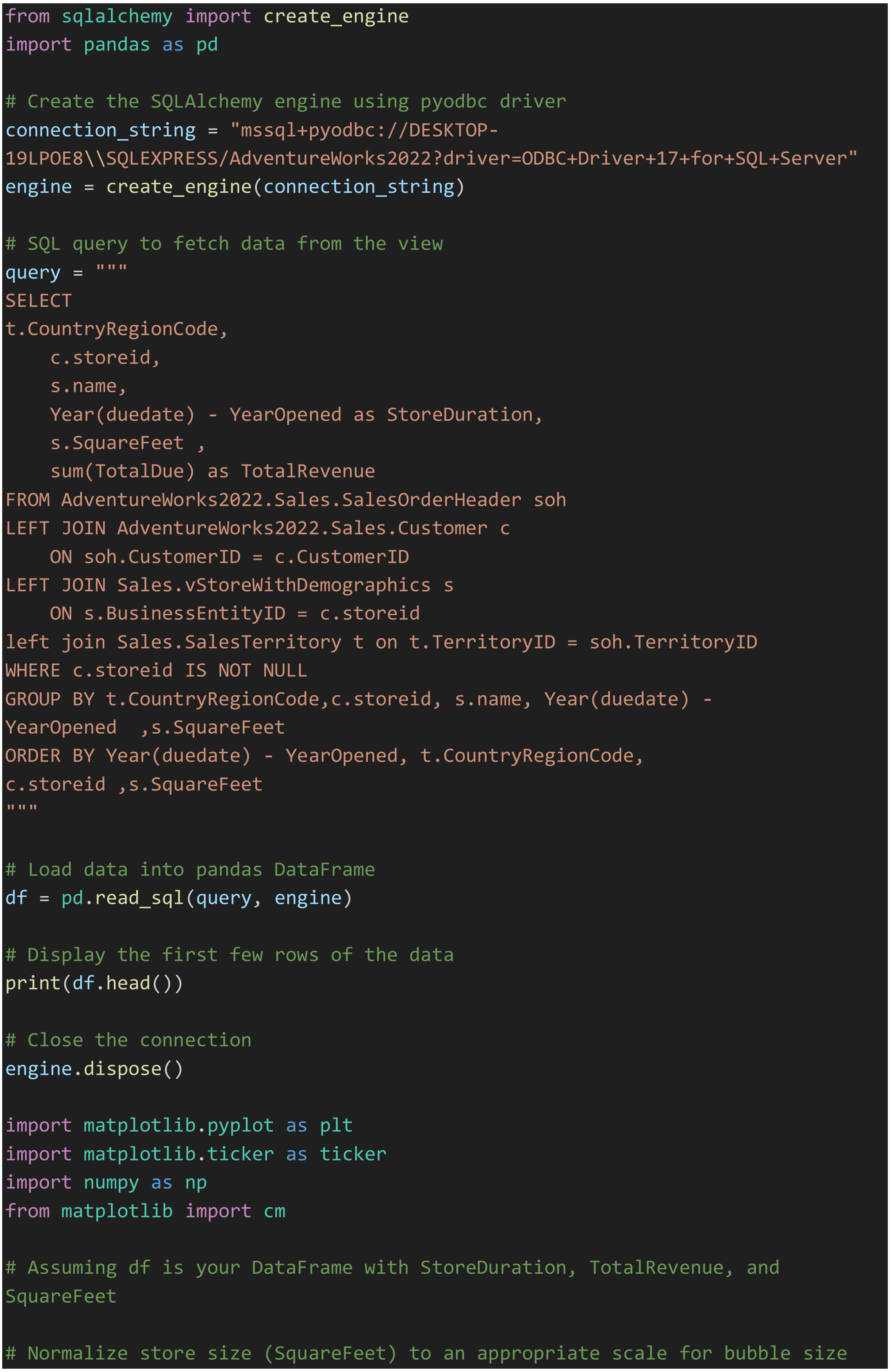
```
    ORDER BY Year(duedate)- YearOpened  ,CountryRegionCode, storeid ,
s.SquareFeet, s.NumberEmployees
"""

# Load data into pandas DataFrame
df = pd.read_sql(query, connection)

# Close the connection
connection.close()

# Display the first few rows of the data
print(df.head())

import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
import numpy as np

# Sets the background color
plt.gcf().set_facecolor('white')

# Plot the relationship between trading duration and annual revenue
plt.figure(figsize=(10, 6))
plt.scatter(df['StoreDuration'], df['TotalRevenue'] , color='orange',
alpha=0.5)

# Set titles and labels
plt.title('Exploring the Link Between Store Trading Duration and Revenue
(2011-2014)', fontsize=16, color='black')
plt.xlabel('Trading Duration (Years)', fontsize=12, color='black')
plt.ylabel('Total Revenue (USD)', fontsize=12, color='black')

# Customize ticks
plt.xticks(color='black')  # X-axis tick color
plt.yticks(color='black')  # Y-axis tick color

# Format y-axis to display the USD sign ($)
plt.gca().yaxis.set_major_formatter(ticker.FuncFormatter(lambda x, _:
'${:,}'.format(int(x), color='white')))

# Customize gridlines
plt.grid(True, color='grey', linestyle='--', linewidth=0.5)

# Adjust the layout to ensure the plot looks neat
plt.tight_layout()

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

```
# Calculate the correlation coefficient between Trading Duration and Annual Revenue
correlation = df['StoreDuration'].corr(df['TotalRevenue'])

print(f"Pearson correlation between Trading Duration and Annual Revenue: {correlation:.2f}")
```

```
from sqlalchemy import create_engine
import pandas as pd

# Create the SQLAlchemy engine using the pyodbc driver
connection_string = "mssql+pyodbc://DESKTOP-19LPOE8\\SQLEXPRESS/AdventureWorks2022?driver=ODBC+Driver+17+for+SQL+Server"
engine = create_engine(connection_string)

# SQL query to fetch data from the view
query = """
SELECT
    c.storeid,
    s.name,
    Year(duedate) - YearOpened as StoreDuration,
    sum(TotalDue) as TotalRevenue
FROM AdventureWorks2022.Sales.SalesOrderHeader soh
LEFT JOIN AdventureWorks2022.Sales.Customer c
    ON soh.CustomerID = c.CustomerID
LEFT JOIN Sales.vStoreWithDemographics s
    ON s.BusinessEntityID = c.storeid
WHERE c.storeid IS NOT NULL
GROUP BY c.storeid, s.name, Year(duedate) - YearOpened
ORDER BY Year(duedate) - YearOpened, c.storeid
"""

# Load data into pandas DataFrame
df = pd.read_sql(query, engine)

# Display the first few rows of the data
print(df.head())

# Close the connection
engine.dispose()
import matplotlib.pyplot as plt
import matplotlib.cm as cm
import matplotlib.colors as mcolors
import numpy as np

# Assuming df is your DataFrame
df_grouped = df.groupby('StoreDuration').agg({'TotalRevenue': 'sum'}).reset_index()
```

```
# Normalize the TotalRevenue values to the range [0, 1] for colormap
norm = mcolors.Normalize(vmin=df_grouped['TotalRevenue'].min(),
vmax=df_grouped['TotalRevenue'].max())

# Create a colormap which ranges from light to dark
cmap = cm.get_cmap('summer')  # You can try other colormaps like 'Reds',
'YlGnBu'

# Create a list of colors from the colormap for each bar based on TotalRevenue
colors = [cmap(norm(value)) for value in df_grouped['TotalRevenue']]

# Bar Chart
plt.figure(figsize=(10, 6))

# Plotting the bar chart with color mapping
bars = plt.bar(df_grouped['StoreDuration'], df_grouped['TotalRevenue'],
color=colors)

# Add title and labels
plt.title('Store Revenue by Trading Duration (2011-2014)', fontsize=16,
color='black')
plt.xlabel('Trading Duration (Years)', fontsize=12, color='black')
plt.ylabel('Total Revenue (USD)', fontsize=12, color='black')

# Format y-axis to display the USD sign ($) with commas
plt.gca().yaxis.set_major_formatter(plt.FuncFormatter(lambda x, _:
'${:,}'.format(int(x))))

# Customize the plot appearance
plt.gca().set_facecolor('white')
plt.gcf().set_facecolor('white')

# Set the color for ticks
plt.xticks(color='black')
plt.yticks(color='black')

# Add gridlines for better readability
plt.grid(True, color='grey', linestyle='--', linewidth=0.5)

# Display the plot
plt.tight_layout()
plt.show()
```

```
# For plotting a Bubble Chart based on the data generated on 2010-2014
# Add a third variable (Store Szie) to analysis any impact on the link between
store duration and total revenue
```



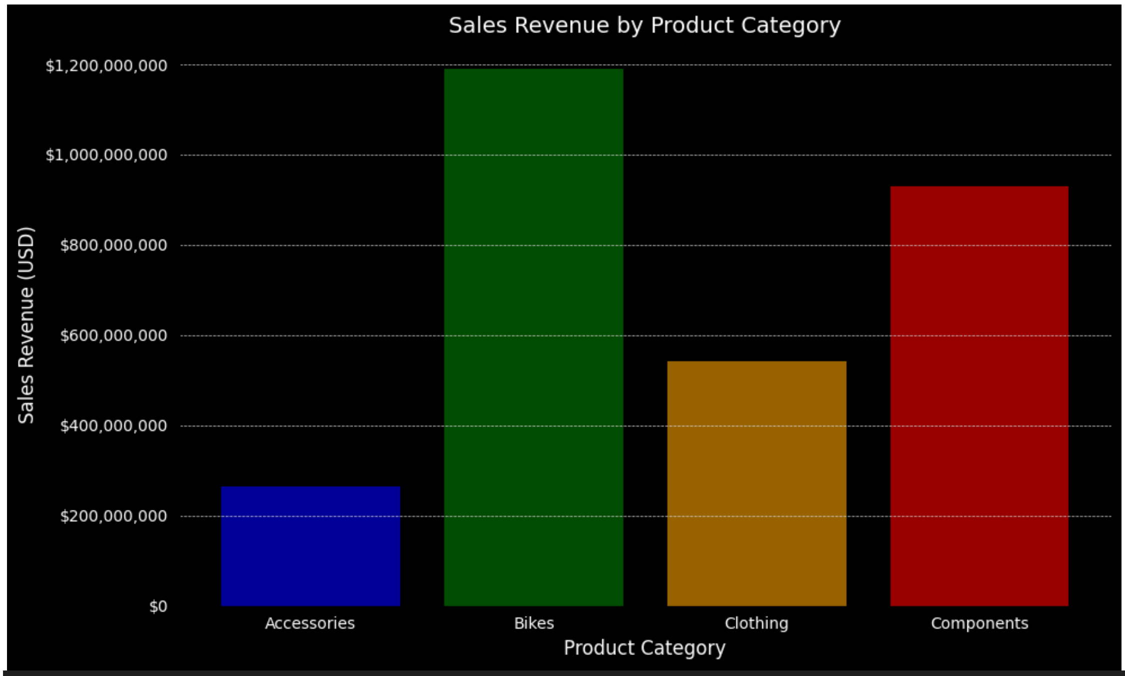
```
from sqlalchemy import create_engine
import pandas as pd

# Create the SQLAlchemy engine using pyodbc driver
connection_string = "mssql+pyodbc://DESKTOP-
19LPOE8\\SQLEXPRESS/AdventureWorks2022?driver=ODBC+Driver+17+for+SQL+Server"
engine = create_engine(connection_string)

# SQL query to fetch data from the view
query = """
SELECT
t.CountryRegionCode,
    c.storeid,
    s.name,
    Year(duedate) - YearOpened as StoreDuration,
    s.SquareFeet ,
    sum(TotalDue) as TotalRevenue
FROM AdventureWorks2022.Sales.SalesOrderHeader soh
LEFT JOIN AdventureWorks2022.Sales.Customer c
    ON soh.CustomerID = c.CustomerID
LEFT JOIN Sales.vStoreWithDemographics s
    ON s.BusinessEntityID = c.storeid
left join Sales.SalesTerritory t on t.TerritoryID = soh.TerritoryID
WHERE c.storeid IS NOT NULL
GROUP BY t.CountryRegionCode,c.storeid, s.name, Year(duedate) -
YearOpened  ,s.SquareFeet
ORDER BY Year(duedate) - YearOpened, t.CountryRegionCode,
c.storeid ,s.SquareFeet
"""

# Load data into pandas DataFrame
df = pd.read_sql(query, engine)

# Display the first few rows of the data
print(df.head())

# Close the connection
engine.dispose()

import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
import numpy as np
from matplotlib import cm

# Assuming df is your DataFrame with StoreDuration, TotalRevenue, and
SquareFeet

# Normalize store size (SquareFeet) to an appropriate scale for bubble size
# Normalize so the smallest store (6,000) has a bubble of size ~50, and the
largest (80,000) ~1000
norm = plt.Normalize(df['SquareFeet'].min(), df['SquareFeet'].max())
bubble_size = norm(df['SquareFeet']) * 400  # Adjust this factor (1000) to
control bubble size

# Create the figure
plt.figure(figsize=(10, 6))

# Scatter plot for TotalRevenue and store size (SquareFeet)
scatter = plt.scatter(df['StoreDuration'], df['TotalRevenue'],
                      s=bubble_size,  # Bubble size proportional to SquareFeet
                      c=df['SquareFeet'],  # Color by Store Size
                      cmap='viridis', alpha=0.6, edgecolors="w",
linewidth=0.5)

# Title and labels
plt.title('Exploring the Link Between Store Duration, Size, and Revenue',
fontsize=16, color='black')
plt.xlabel('Store Trading Duration (Years)', fontsize=12, color='black')
plt.ylabel('Total Revenue (USD)', fontsize=12, color='black')

# Format y-axis to display the USD sign ($)
plt.gca().yaxis.set_major_formatter(ticker.FuncFormatter(lambda x, _:
'${:,}'.format(int(x))))

# Add color bar for the store size (SquareFeet)
cbar = plt.colorbar(scatter, orientation='vertical')
cbar.set_label('Store Size (Square Feet)', fontsize=12)

# Set background color
plt.gcf().set_facecolor('white')
plt.gca().set_facecolor('white')

# Set the grid
plt.grid(True, color='grey', linestyle='--', linewidth=0.5)

# Set x and y axis limits
plt.xlim(0, df['StoreDuration'].max() + 1)  # Padding for x-axis
plt.ylim(0, df['TotalRevenue'].max() * 1.1)  # Padding for y-axis

# Customize ticks color
plt.xticks(color='black')
plt.yticks(color='black')

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

F. The relationship between the size of the stores, number of employees and revenue

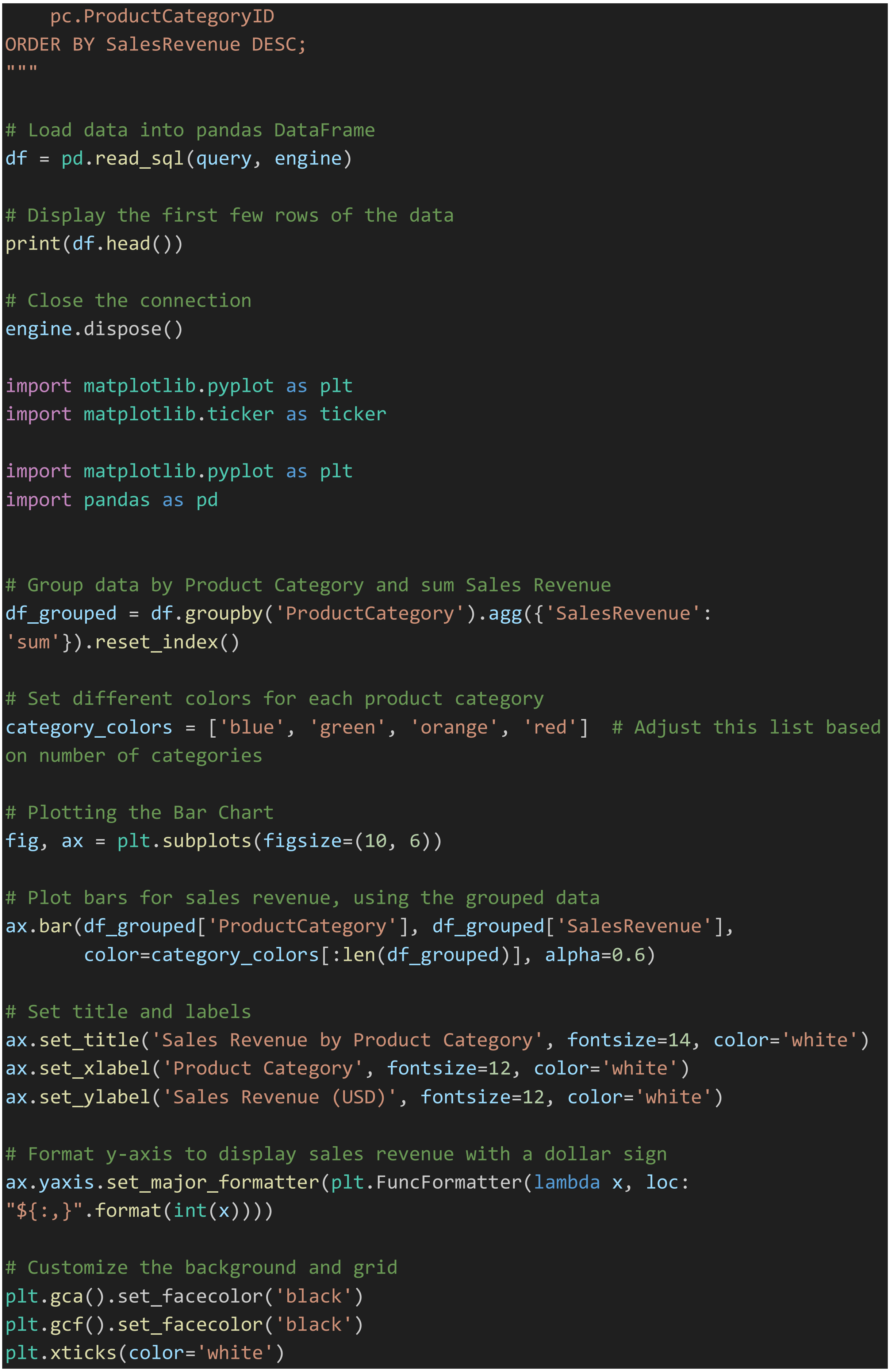
```
import pyodbc
import pandas as pd
import matplotlib.pyplot as plt
from tabulate import tabulate
import seaborn as sns

connection = pyodbc.connect(
    r'DRIVER={ODBC Driver 18 for SQL Server};'
    r'SERVER=LAPTOP-N2D8ADNR\SQLEXPRESS;'
    r'DATABASE=AdventureWorks2022;'
    r'Trusted_Connection=yes;'
    r'TrustServerCertificate=yes;'
)
query = """
-- Defines a namespace alia as ns
WITH XMLNAMESPACES ('http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/StoreSurvey' AS ns)
--CAST(..AS int): ensure the extracted values is explicitly cast to an integer data
-- .value() extracts a specific value from XML structure using an XPath query
-- /: strats at the root of the XML document
--ns: StoreSurvey: refers to the 'StoreSurvey' element in the namespace ns
-- [1]: extracts the first occurrence of the 'SquareFeet' element
-- int: specifies the desired return data type as integer
-- float: specifies the desired return data type as real number
SELECT
    BusinessEntityID,
    CAST(Demographics.value('(/ns:StoreSurvey/ns:SquareFeet)[1]', 'int') AS int) AS SquareFeet,
    CAST(Demographics.value('(/ns:StoreSurvey/ns:NumberEmployees)[1]', 'int') AS int) AS NumberEmployees,
    CAST(Demographics.value('(/ns:StoreSurvey/ns:AnnualRevenue)[1]', 'int') AS int) AS AnnualRevenue
FROM Sales.Store
-- only select rows with values in Demographics column
WHERE Demographics IS NOT NULL;
"""
df = pd.read_sql(query, connection)

# Close connection
connection.close()

# Check if the data has null values
print(df)
print(df.isnull().sum())
```

```
# histogram of 3 variables
plt.hist(df['AnnualRevenue'], color='blue',alpha = 0.7, edgecolor = 'black')
plt.title('Distribution of Annual Revenue')
plt.xlabel('Annual Revenue')
plt.ylabel('Frequency')
plt.show()

plt.hist(df['SquareFeet'],bins= 10,color='blue',alpha = 0.7, edgecolor = 'black')
plt.title('Distribution of Store Size')
plt.xlabel('Square Feet')
plt.ylabel('Frequency')
plt.show()

plt.hist(df['NumberEmployees'],bins= 10, color='blue',alpha = 0.7, edgecolor = 'black')
plt.title('Distribution of Number of Employees')
plt.xlabel('Number of Employees')
plt.ylabel('Frequency')
plt.show()
# use seaborn to draw scatter plots
sns.scatterplot(
    data = df,
    y = 'AnnualRevenue',
    x='NumberEmployees',
    hue='SquareFeet',
    size= 'SquareFeet',
    palette= 'coolwarm',
    sizes= (15,75),
    alpha = 0.8
)
plt.title('Relationship between Store size, Employees and Revenue')
plt.xlabel('Number of Employees')
plt.ylabel('Revenue')
plt.show()

sns.scatterplot(
    data = df,
    y = 'AnnualRevenue',
    x='SquareFeet',
    hue='NumberEmployees',
    size= 'NumberEmployees',
    palette= 'coolwarm',
    sizes= (15,75),
    alpha = 0.8
)
plt.title('Relationship between Store size, Employees and Revenue')
plt.xlabel('SquareFeet')
plt.ylabel('Revenue')
plt.show()


sns.scatterplot(
    data = df,
    x = 'NumberEmployees',
    y='SquareFeet',
    hue='AnnualRevenue',
    size= 'AnnualRevenue',
    palette= 'coolwarm',
    sizes= (15,75),
    alpha = 0.8
)
plt.title('Relationship between Store size, Employees and Revenue')
plt.xlabel('NumberEmployees')
plt.ylabel('SquareFeet')
plt.show()

# Use seaborn library to create pair plot
sns.pairplot(data = df,
             vars=['SquareFeet', 'NumberEmployees', 'AnnualRevenue'] # list
all variables
             )
plt.suptitle('Pairplot of Store Attributes',y = 1.03) # add the title for the
pairplots
plt.show()

# calculate the correlation between 3 variables
correlation = df[['SquareFeet', 'NumberEmployees', 'AnnualRevenue']].corr()
# draw heatmap
sns.heatmap(correlation, annot=True, cmap='coolwarm') # cmap: mapping the data
values to colors; annot:  write values in each cell of the heatmap
plt.title('Correlation Matrix')
plt.show()
```

G. Sales Revenue Trend for Bikes Subcategories



```
# Create the Bar Chart for ‘Sales Revenue by Product Category’

from sqlalchemy import create_engine
import pandas as pd

# Create the SQLAlchemy engine using the pyodbc driver
connection_string = "mssql+pyodbc://DESKTOP-
19LPOE8\\SQLEXPRESS/AdventureWorks2022?driver=ODBC+Driver+17+for+SQL+Server"
engine = create_engine(connection_string)

# 2. Define the SQL query to fetch data for sales revenue and quantity
query = """
SELECT SUM(soh.TotalDue) as SalesRevenue,
       pc.Name as ProductCategory,
       SUM(sod.OrderQty) as Qty,
       ps.Name as ProductSubcategory,
       p.ProductSubcategoryID,
       pc.ProductCategoryID
FROM AdventureWorks2022.Sales.SalesOrderHeader soh
LEFT JOIN AdventureWorks2022.Sales.SalesOrderDetail sod
    ON soh.SalesOrderID = sod.SalesOrderID
LEFT JOIN Production.Product p
    ON p.ProductID = sod.ProductID
LEFT JOIN Production.ProductSubcategory ps
    ON ps.ProductSubcategoryID = p.ProductSubcategoryID
LEFT JOIN Production.ProductCategory pc
    ON ps.ProductCategoryID = pc.ProductCategoryID
GROUP BY
    pc.Name,
    p.ProductSubcategoryID,
    ps.Name,
```



```
    pc.ProductCategoryID
ORDER BY SalesRevenue DESC;
"""

# Load data into pandas DataFrame
df = pd.read_sql(query, engine)

# Display the first few rows of the data
print(df.head())

# Close the connection
engine.dispose()

import matplotlib.pyplot as plt
import matplotlib.ticker as ticker

import matplotlib.pyplot as plt
import pandas as pd


# Group data by Product Category and sum Sales Revenue
df_grouped = df.groupby('ProductCategory').agg({'SalesRevenue':
'sum'}).reset_index()

# Set different colors for each product category
category_colors = ['blue', 'green', 'orange', 'red']  # Adjust this list based
on number of categories

# Plotting the Bar Chart
fig, ax = plt.subplots(figsize=(10, 6))

# Plot bars for sales revenue, using the grouped data
ax.bar(df_grouped['ProductCategory'], df_grouped['SalesRevenue'],
       color=category_colors[:len(df_grouped)], alpha=0.6)

# Set title and labels
ax.set_title('Sales Revenue by Product Category', fontsize=14, color='white')
ax.set_xlabel('Product Category', fontsize=12, color='white')
ax.set_ylabel('Sales Revenue (USD)', fontsize=12, color='white')

# Format y-axis to display sales revenue with a dollar sign
ax.yaxis.set_major_formatter(plt.FuncFormatter(lambda x, loc:
"${:,}".format(int(x))))

# Customize the background and grid
plt.gca().set_facecolor('black')
plt.gcf().set_facecolor('black')
plt.xticks(color='white')
```

```python
plt.yticks(color='white')
plt.grid(True, axis='y', linestyle='--', color='lightgray', linewidth=0.5)



# Adjust the layout to ensure it looks neat
plt.tight_layout()

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

```python
from sqlalchemy import create_engine
import pandas as pd

# Create the SQLAlchemy engine using the pyodbc driver
connection_string = "mssql+pyodbc://DESKTOP-
19LPOE8\\SQLEXPRESS/AdventureWorks2022?driver=ODBC+Driver+17+for+SQL+Server"
engine = create_engine(connection_string)

# 2. Define the SQL query to fetch data for sales revenue and quantity
query = """
SELECT SUM(soh.TotalDue) as SalesRevenue,
       ps.Name as ProductSubcategory,
       p.ProductSubcategoryID,
       pc.ProductCategoryID,
       YEAR(soh.DueDate) as SalesYear,
       soh.DueDate
FROM AdventureWorks2022.Sales.SalesOrderHeader soh
LEFT JOIN AdventureWorks2022.Sales.SalesOrderDetail sod
    ON soh.SalesOrderID = sod.SalesOrderID
LEFT JOIN Production.Product p
    ON p.ProductID = sod.ProductID
LEFT JOIN Production.ProductSubcategory ps
    ON ps.ProductSubcategoryID = p.ProductSubcategoryID
LEFT JOIN Production.ProductCategory pc
    ON ps.ProductCategoryID = pc.ProductCategoryID
WHERE pc.ProductCategoryID = 1
GROUP BY
    YEAR(soh.DueDate),
    ps.Name,
    p.ProductSubcategoryID,
    soh.DueDate,
    pc.ProductCategoryID
ORDER BY SalesYear;
"""

# Load data into pandas DataFrame
df = pd.read_sql(query, engine)
```

```python
# Display the first few rows of the data
print(df.head())

# Close the connection
engine.dispose()

import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
import matplotlib.dates as mdates

# Filter data for each subcategory
df_mountain_bikes = df[df['ProductSubcategory'] == 'Mountain Bikes']
df_road_bikes = df[df['ProductSubcategory'] == 'Road Bikes']
df_touring_bikes = df[df['ProductSubcategory'] == 'Touring Bikes']

# Aggregate sales revenue by year for each subcategory
df_mountain_bikes_yearly =
df_mountain_bikes.groupby('SalesYear').agg({'SalesRevenue':
'sum'}).reset_index()
df_road_bikes_yearly = df_road_bikes.groupby('SalesYear').agg({'SalesRevenue':
'sum'}).reset_index()
df_touring_bikes_yearly =
df_touring_bikes.groupby('SalesYear').agg({'SalesRevenue':
'sum'}).reset_index()

# Convert 'SalesYear' to datetime
df_mountain_bikes_yearly['SalesYear'] =
pd.to_datetime(df_mountain_bikes_yearly['SalesYear'], format='%Y')
df_road_bikes_yearly['SalesYear'] =
pd.to_datetime(df_road_bikes_yearly['SalesYear'], format='%Y')
df_touring_bikes_yearly['SalesYear'] =
pd.to_datetime(df_touring_bikes_yearly['SalesYear'], format='%Y')

# Scale sales revenue to millions for better readability
df_mountain_bikes_yearly['SalesRevenue'] /= 1e6  # Convert to millions
df_road_bikes_yearly['SalesRevenue'] /= 1e6
df_touring_bikes_yearly['SalesRevenue'] /= 1e6


# Plot the line chart
plt.figure(figsize=(10, 6))

# Plot the line chart for Mountain Bikes
plt.plot(df_mountain_bikes_yearly['SalesYear'],
df_mountain_bikes_yearly['SalesRevenue'], marker='o', color='forestgreen',
label='Mountain Bikes')
```

```
# Plot the line chart for Road Bikes
plt.plot(df_road_bikes_yearly['SalesYear'],
df_road_bikes_yearly['SalesRevenue'], marker='o', color='darkorange',
label='Road Bikes', linewidth='3')

# Plot the line chart for Touring Bikes
plt.plot(df_touring_bikes_yearly['SalesYear'],
df_touring_bikes_yearly['SalesRevenue'], marker='o', color='darkviolet',
label='Touring Bikes', linestyle="--")

# Add labels and title
plt.title('Sales Revenue Trend for Bikes Subcategories (Jan 2011- Jun 2014)',
fontsize=14, color='black')
plt.xlabel('Year', fontsize=12, color='black')
plt.ylabel('Sales Revenue (USD)', fontsize=12, color='black')

# Format x-axis labels as 'Jan 2011'
plt.gca().xaxis.set_major_locator(mdates.MonthLocator(bymonthday=1,
interval=6))
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%b %Y'))

# Rotate x-axis labels for better visibility
plt.xticks(rotation=45)

# Format y-axis to show dollar sign
plt.gca().yaxis.set_major_formatter(ticker.FuncFormatter(lambda x, _:
"${:,.0f}M".format(int(x))))

# Customize plot appearance
plt.gca().set_facecolor('whitesmoke')
plt.gcf().set_facecolor('white')

# Change tick colors to white
plt.xticks(color='black')
plt.yticks(color='black')

# Add gridlines for better readability
plt.grid(True, color='lightgray', linestyle='--', linewidth=0.5)

# Add legend to differentiate between the subcategories
plt.legend()

# Display the plot
plt.tight_layout()
plt.show()
```