Predicting Future Outcomes

Turtle Games

Data Analytics Report

Advanced Analytics for Organisational Impact

Cristina Alonso Robles

July 2023

Methodology

1. **Background/Context of the business**

2. **Analytical Approach, using Python and R**

3. **Visualisations and Insights - Patterns, Predictions and Recommendations**

1. Background/Context of the business

- **Organisation and Business Definition:** game manufacturer and retailer with a global customer base. They manufacture and sell their own products, along with sourcing and selling products manufactured by other companies. Their product range includes books, board games, video games, and toys. They collect data from sales as well as customer reviews.

- **Industry**: Entertainment.

- **Sector**: Retail and manufacturing.

- **Turtle Games' Business Problem:**

Even though Turtle Games collects data from sales as well as customer reviews, they do not have a clear understanding of their overall sales performance and markets or segments within their customer base.

- **Why should Turtle Games try to solve this problem?**

By properly gathering and analysing customers and sales data, Turtle Games can:

1. Improve their overall sales performance.
2. Understand their customers buying behaviours and reward their most loyal customers.
3. Invest more efficiently in new games and make predictions on what games would sell the most in the future.
4. Plan and use their marketing budget more efficiently by targeting specific markets and customers segments based on their customers buying behaviours within their current customer base.

- **Turtle Games' Business Objectives:**

1. Improving overall sales performance by utilising customer trends.
2. Understanding how users accumulate loyalty points.

To provide answers to the sales and marketing teams, we will analyse the current sales and social media data provided by Turtle Games, with the aim of giving answers to the following questions:

a. How customers accumulate loyalty points.
b. How useful are remuneration and spending scores data.
c. How groups within the customer base can be used to target specific market segments.
d. How social data (e.g. customer reviews) can be used to inform marketing campaigns.
e. The impact that each product has on sales.
f. How reliable the data is (e.g. normal distribution, skewness, or kurtosis).
g. What the relationship(s) is/are (if any) between North American, European, and global sales.

- **Stakeholders involved:** Senior stakeholders at Turtle Games.

2. Analytical Approach, using Python and R

A. Making predictions with Linear regression using Python

I'll apply linear regression techniques to determine how customers accumulate loyalty points. I decided to use simple linear regression because simple linear regression provides a straightforward interpretation, as the relationship is between two variables only.

In order to investigate the possible relationships between the loyalty points, age, remuneration, and spending scores, I followed the next steps:

Import the necessary libraries.

Cleaning the data as per the provided guidelines and saving a copy of the clean reviews for future use.

```
# Import all the necessary packages.
import numpy as np
import pandas as pd
import statsmodels.api as sm
import sklearn
import matplotlib.pyplot as plt
import seaborn as sns

from statsmodels.formula.api import ols
from sklearn import datasets
from sklearn import linear_model
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

```
# Load the CSV file(s) as reviews.
reviews = pd.read_csv('turtle_reviews.csv')

# View the DataFrame.
reviews.head()
```

	gender	age	remuneration (k£)	spending_score (1-100)	loyalty_points	education	language	platform	product	review	summary
0	Male	18	12.30	39	210	graduate	EN	Web	453	When it comes to a DM's screen, the space on t...	The fact that 50% of this space is wasted on a...
1	Male	23	12.30	81	524	graduate	EN	Web	466	An Open Letter to GaleForce9*:\n\nYour unpaint...	Another worthless Dungeon Master's screen from...
2	Female	22	13.12	6	40	graduate	EN	Web	254	Nice art, nice printing. Why two panels are f...	pretty, but also pretty useless
3	Female	25	13.12	77	562	graduate	EN	Web	263	Amazing buy! Bought it as a gift for our new d...	Five Stars
4	Female	33	13.94	40	366	graduate	EN	Web	291	As my review of GF9's previous screens these w...	Money trap

```
# Any missing values?
reviews_na = reviews[reviews.isna().any(axis=1)]

reviews_na.shape
```

```
(0, 11)
```

There are not missing values in the 'reviews' DataFrame.

```
# Drop unnecessary columns (language and platform).
reviews.drop(reviews.columns[[6, 7]], axis=1, inplace=True)

# View column names.
reviews.shape
list(reviews.columns)
```

```
['gender',
 'age',
 'remuneration (k£)',
 'spending_score (1-100)',
 'loyalty_points',
 'education',
 'product',
 'review',
 'summary']
```

3. Rename columns

```
# Rename the column headers.
reviews.rename(columns = {'remuneration (k£)': 'remuneration', 'spending_score (1-100)':'spending_score'},
               inplace = True)

# View column names.
list(reviews.columns)
```

```
['gender',
 'age',
 'remuneration',
 'spending_score',
 'loyalty_points',
 'education',
 'product',
 'review',
 'summary']
```

4. Save the DataFrame as a CSV file

```
# Create a CSV file as output.
reviews.to_csv('/Users/alonsoroblescristina/Desktop/COURSE 3/LSE_DA301_assignment_files/clean_reviews.csv')
reviews.to_csv('clean_reviews.csv', index=False)
```

```
# Load the clean_reviews CSV as reviews.
reviews = pd.read_csv('clean_reviews.csv')

# View the DataFrame.
reviews.head()
```

	gender	age	remuneration	spending_score	loyalty_points	education	product	review	summary
0	Male	18	12.30	39	210	graduate	453	When it comes to a DM's screen, the space on t...	The fact that 50% of this space is wasted on a...
1	Male	23	12.30	81	524	graduate	466	An Open Letter to GaleForce9*:\n\nYour unpaint...	Another worthless Dungeon Master's screen from...
2	Female	22	13.12	6	40	graduate	254	Nice art, nice printing. Why two panels are f...	pretty, but also pretty useless
3	Female	25	13.12	77	562	graduate	263	Amazing buy! Bought it as a gift for our new d...	Five Stars
4	Female	33	13.94	40	366	graduate	291	As my review of GF9's previous screens these w...	Money trap

Specify the independent and dependent variables.

5. Linear regression

5a) Spending score vs Loyalty points

```
# Independent variable.
x = reviews['spending_score']
```

Independent variable: what we use to predict or explain the outcome variable. Also called explanatory. I will use the following independent variables (spending score, age and remuneration) to predict the loyalty points.

```
# Dependent variable.
y = reviews['loyalty_points']
```

Dependent or response variable: what we are interested in understanding or predicting. In this case, since we are trying to predict the loyalty points, this will be our dependent variable.

Correlation between each of the variables of the DataFrame.

	age	remuneration	spending_score	loyalty_points	product
age	1.000000	-0.005708	-0.224334	-0.042445	0.003081
remuneration	-0.005708	1.000000	0.005612	0.616065	0.305309
spending_score	-0.224334	0.005612	1.000000	0.672310	-0.001649
loyalty_points	-0.042445	0.616065	0.672310	1.000000	0.183600
product	0.003081	0.305309	-0.001649	0.183600	1.000000

Spending score vs Loyalty points

```
# Determine the correlation between spending score and loyalty points.

# Access these columns as follows.
spending_score = reviews['spending_score']
loyalty_points = reviews['loyalty_points']

# Calculate the correlation coefficient between the two columns.
correlation = spending_score.corr(loyalty_points)

print("Correlation between spending score and loyalty points:", correlation)
```

```
Correlation between spending score and loyalty points: 0.6723101119155435
```

The correlation coefficient ranges from -1 to 1, where -1 indicates a perfect negative correlation, 1 indicates a perfect positive correlation, and 0 indicates no correlation. The closer the correlation coefficient is to -1 or 1, the stronger the correlation. If it is close to 0, there is little to no correlation.

The correlation between the loyalty points and spending score is 67%. This indicates a positive and moderately strong linear relationship between the two variables.

The positive sign of the correlation coefficient indicates that as the spending score increases, the loyalty points tend to increase as well. In other words, there is a direct relationship between spending score and loyalty points. Customers with higher spending scores tend to accumulate more loyalty points.

It's important to note that correlation only measures the strength and direction of the linear relationship. It does not imply causation. A high correlation between spending score and loyalty points does not necessarily mean that one variable causes the other to change.

While the correlation coefficient provides valuable information about the relationship between spending score and loyalty points, it is essential to perform additional analysis and consider other factors. Later, I will calculate the correlation between the loyalty points and the rest of the independent variables (remuneration and age).

Create the OLS model.

```
# Pass linear regression through OLS methods.
test = ols('y ~ x', data = reviews).fit()

# Print the regression table.
test.summary()
```

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.452
Model:	OLS	**Adj. R-squared:**	0.452
Method:	Least Squares	**F-statistic:**	1648.
Date:	Sat, 22 Jul 2023	**Prob (F-statistic):**	2.92e-263
Time:	16:47:40	**Log-Likelihood:**	-16550.
No. Observations:	2000	**AIC:**	3.310e+04
Df Residuals:	1998	**BIC:**	3.312e+04
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-75.0527	45.931	-1.634	0.102	-165.129	15.024
x	33.0617	0.814	40.595	0.000	31.464	34.659

Omnibus:	126.554	**Durbin-Watson:**	1.191
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	260.528
Skew:	0.422	**Prob(JB):**	2.67e-57
Kurtosis:	4.554	**Cond. No.**	122.

Explanation of the results of the OLS model:

How the Loyalty Points are affected by the Spending Score?

Variables:

Loyalty Points: y, response variable, dependent variable

Spending Score: x, explanatory variable, independent variable

R-squared: How much of the variability of the Y variable (Loyalty points) is explained by the explanatory variable (Spending Score). In this case, aroud 45% of the observed variation can be explained by the model's inputs. In other words, 45% of the variability of the Loyalty Points is explained by the Spending Score.

Estimate 1: B0, Intercept: the intercept of the regression line. It is the value where the regression line crosses the y-axis when the independent variable (spending score) is zero. In other words, it is the predicted value of the dependent variable (loyalt points) when the independent variable (spending score) is zero. In this case, when the spending score is zero, we would expect to see -75 loyalty points.

Estimate 2: B1, the slope coefficient of the independent variable (Spending Score). We are mainly interested in the estimate of the slope. We should not judge whether the estimate value is good or bad based just on the absolute value of it. In order to understand whether this is significant in our model, we need to look at the probabilities, (Pr). The smaller the probability is the most important and significant the estimate of that linear regression is. In this case is zero, hence we can say that the spending score is not a very significant estimate of the loyalty points.

The t-value tests the hypothesis that the slope is significant or not. If the corresponding probability is small (typically smaller than 0.05) the slope is significant. In this case, the probability of the t-value is zero, thus the estimated slope is not significant. In other words, a t-value of 40.595 suggests that the spending score do not have a strong effect on the loyalty points in this linear regression model.

The coefficient of 'x' describes the slope of the regression line, in other words, how much the response variable 'y' change when 'x' changes by 1 unit. In this case, if the spending score changes by 1 unit, the loyalty points will change by 33.0617 units.

The last two numbers describe the 95% confidence interval of the true coefficient, i.e. the true slope. For instance, if we take a different sample, the estimated slope will be slightly different. If we take 100 random samples each of 500 observations of spending scores and loyalty points, then 95 out of the 100 samples will derive a slope that is within the interval (31.464 , 34.659).

The "DF residuals" are the differences between the observed values of the dependent variable and the predicted values from the regression equation. They represent the unexplained variability or the errors of the model. The degrees of freedom for the residuals indicate the number of data points that are free to vary after estimating the model. In this case the model estimates that 1998 data points will not vary after estimating the model.

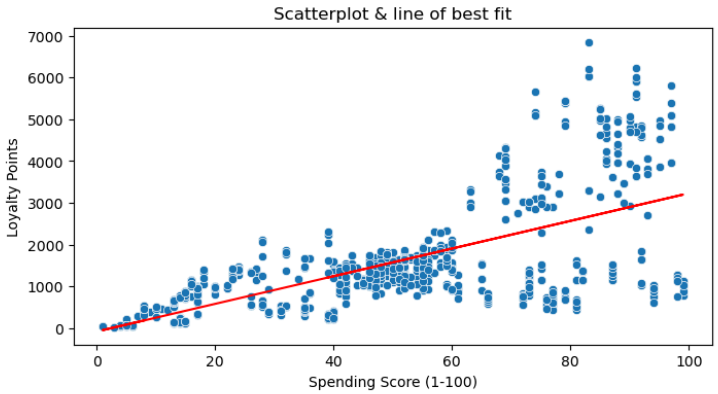
Extract the predicted values.

```
# Extract the predicted values.
print("Predicted values: ", test.predict())
```

```
Predicted values:  [1214.35337415 2602.94449102  123.31749662 ... 2933.56142361  453.93442921
  189.44088314]
```

Plot the linear regression, and add a regression line.

```
# Add a trendline to visualise the Linear regression.
# Use the NumPy polyval method, specify the regression and the independent variable.
trend = np.polyval(reg, reviews['spending_score'])

# View the previous scatterplot.
plt.figure(figsize=(8, 4), dpi=100)
plt.title("Scatterplot & line of best fit")
plt.xlabel("Spending Score (1-100)")
plt.ylabel("Loyalty Points")

sns.scatterplot (data=reviews,
x='spending_score' , y='loyalty_points')

# Add the trendline.
plt.plot(reviews['spending_score'],
         trend, color='red')
plt.show()
```



What we can see so far is that the higher the spending score (70 - 100 points) the more loyalty points the customers accumulate, however this does not imply causation between these two variables, since there are customers with high spending scores that do not accumulate many loyalty points, (all those blue dots below the red trend line.

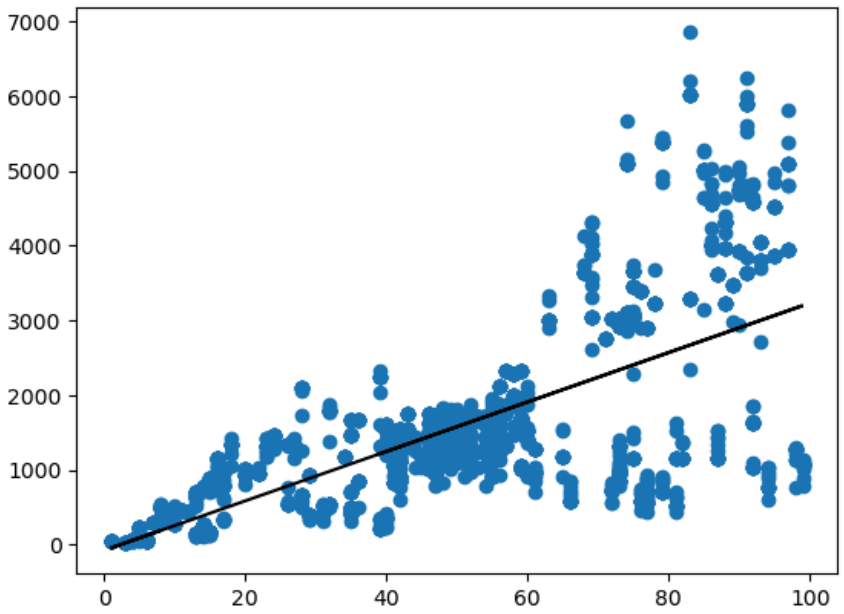
Creating, fitting, and improving the accuracy of a regression model in Python.

```
# Fit the linear regression model.
# x coef: 33.0617.
# Constant coef: -75.0527.
# Create the linear equation.
y_pred = (-75.0527) + 33.0617 * x

# View the output.
y_pred
```

```
0        1214.3536
1        2602.9450
2         123.3175
3        2470.6982
4        1247.4153
           ...
1995     2206.2046
1996      189.4409
1997     2933.5620
1998      453.9345
1999      189.4409
Name: spending_score, Length: 2000, dtype: float64
```

```
# Plot the data points.
plt.scatter(x, y)

# Plot the regresion line.
plt.plot(x, y_pred, color='black')
```

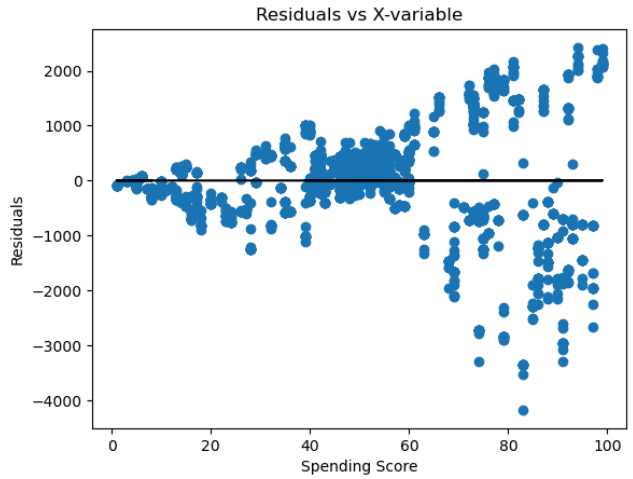
```
[<matplotlib.lines.Line2D at 0x15ac4c460>]
```



```
# Create the linear regression model.
New_value = 81
y_pred = (-75.0527) +33.0617 * New_value

# View the output
print("If Spendig Score is", New_value, "the predicted Loyalty Points are : ", y_pred)
```

```
If Spendig Score is 81 the predicted Loyalty Points are :  2602.945
```

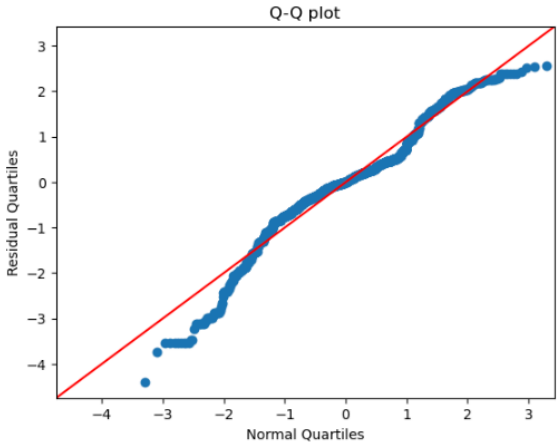
```
# Plot the residuals= (y-predict - y-observe) versus the x-values.
# Ideally, there should be no pattern in this plot.

plt.scatter(x, test.predict() - y)
plt.plot(x, y - y, color='black')
plt.title("Residuals vs X-variable")
plt.xlabel("Spending Score")
plt.ylabel("Residuals")
plt.show()
```



It's good to see a randomly scattered distribution of the residual variables because it indicates that the linear regression model is showing the underlying relationships between the spending score and the loyalty points effectively.

A randomly scattered distribution of residuals also indicates the absence of any systematic patterns or trends in the residuals and that the model is providing unbiased estimates of the coefficients.

```
# Import qqplot Library
from statsmodels.graphics.gofplots import qqplot

# Residuals are calcualted
residuals = test.predict() - y

# Plot the quartiles of the residuals versus the quartiles of the N(0,1)
# The fit=True argument first Standardises the observed data (Residuals) before plotting them
sm.qqplot (residuals, fit=True, line= '45')
plt.ylabel("Residual Quartiles")
plt.xlabel("Normal Quartiles")
plt.title("Q-Q plot")
plt.show()
```



It's also good to see that the residuals variables have a distribution that is very close to normal, since this indicates that the estimated coefficients have smaller standard errors, leading to more precise and reliable estimates, and hypothesis tests are more accurate.

This also means that, on average, the estimated coefficients are equal to the true population coefficients, making the model more accurate in making predictions.

When the residuals are normally distributed, the predictions tend to be more accurate because the model has effectively captured the underlying relationships in the data.

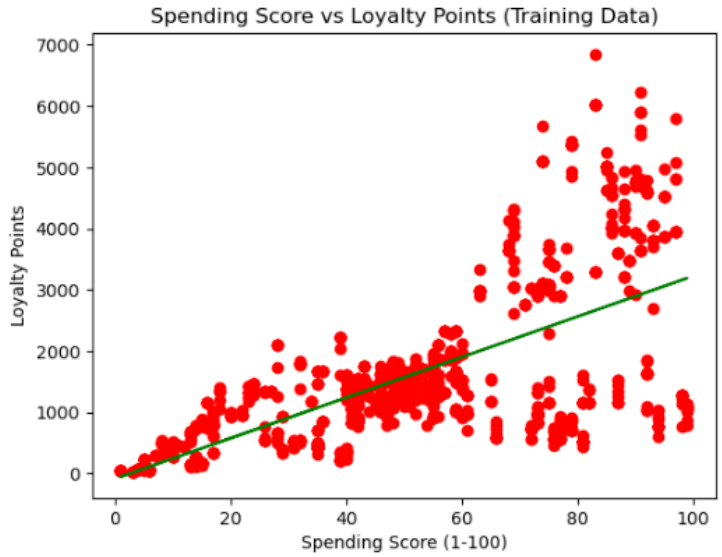
Split Data into Training and Testing Sets: Split the dataset into training and testing sets. The training set will be used to train the model, and the testing set will be used to evaluate its performance.

```
# Finding the best fit line.
# Choose your variables.
x = reviews['spending_score'].values.reshape(-1, 1)
y = reviews['loyalty_points'].values.reshape(-1, 1)

# Split the data into training = 0.7 and testing = 0.3 subsets.
x_train, x_test, y_train, y_test = train_test_split(x, y,
                                                    test_size=0.3,
                                                    random_state=42)
```

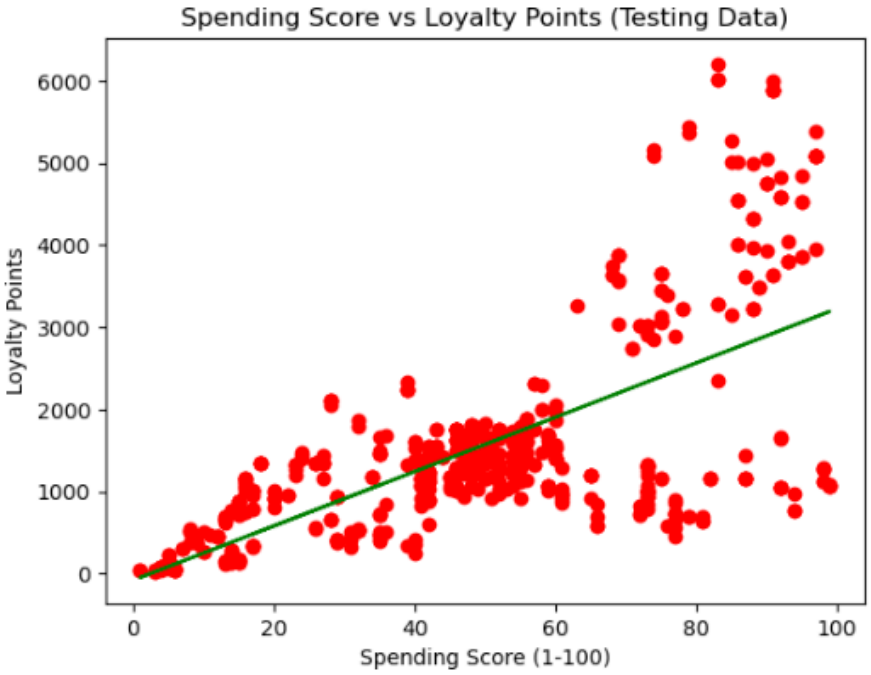
```
# Run linear regression model.
lr = LinearRegression()

# Fit the model on the training data.
lr.fit(x_train, y_train)

# Predict is used for predicting on the x_test.
y_pred = lr.predict(x_test)

# View the output.
y_pred
```

```
array([[  48.37264243],
       [2431.12031027],
       [2265.65172222],
       [1305.93391157],
       [1438.308782  ],
       [1769.24595809],
       [ 246.93494809],
       [2331.83915744],
       [1868.52711092],
       [2927.5260744 ],
       [1570.68365244],
       [1537.58993483],
       [1339.02762918],
       [1206.65275874],
       [ 346.21610091],
       [1703.05852287],
       [1107.37160591],
       [ 280.02866569],
       [ 875.71558265],
```

```
# Visualise the training set.
plt.scatter(x_train, y_train, color = 'red')
plt.plot(x_train, lr.predict(x_train), color = 'green')
plt.title("Spending Score vs Loyalty Points (Training Data)")
plt.xlabel("Spending Score (1-100)")
plt.ylabel("Loyalty Points")

plt.show()
```




```
# Visualise the test set.
plt.scatter(x_test, y_test, color = 'red')
plt.plot(x_train, lr.predict(x_train), color = 'green')
plt.title("Spending Score vs Loyalty Points (Testing Data)")
plt.xlabel("Spending Score (1-100)")
plt.ylabel("Loyalty Points")

plt.show()
```



```
: # Print the R-squared, intercept and coefficient values of the Testing Data.
print("R-squared value: ", lr.score(x_test, y_test))
print("Intercept value: ", lr.intercept_)
print("Coefficient value: ", lr.coef_)
```

```
R-squared value:  0.4608058962879792
Intercept value:  [-84.002228]
Coefficient value:  [[33.09371761]]
```

Notes:

- R-squared tells us how much of the variability of Loyalty points is explained by the Spending Score. In this case, around 45% of the observed variation in the Loyalty Points can be explained by the model's inputs. In other words, 46% of the variability of the Loyalty Points is explained by the Spending Score.
- Estimate 1: B0, Intercept: In this case, when the spending score is zero, we would expect to see -84 loyalty points.
- The coefficient of 'x' describes the slope of the regression line, in other words, how much the response variable 'loyalty points' change when the spending score changes by 1 unit. In this case, if the spending score changes by 1 unit, the loyalty points will change by 33 units.

The same process will be used to calculate the relationship between Remuneration, Age and loyalty points.

Remuneration vs Loyalty

```
|: # Determine the correlation between spending score and loyalty points.

# Access these columns as follows.
remuneration = reviews['remuneration']
loyalty_points = reviews['loyalty_points']

# Calculate the correlation coefficient between the two columns.
correlation = remuneration.corr(loyalty_points)

print("Correlation between remuneration and loyalty points:", correlation)
```

```
Correlation between remuneration and loyalty points: 0.6160647476356388
```

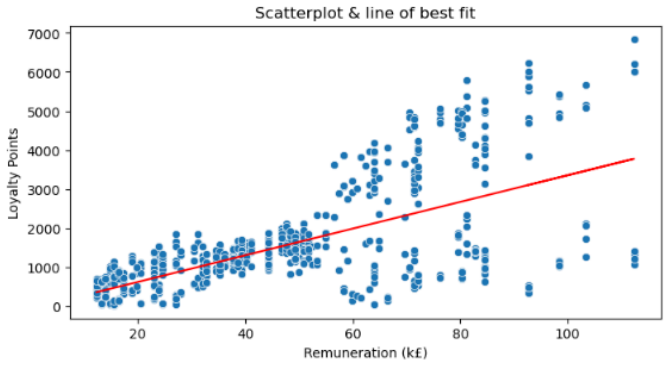
The correlation between the loyalty points and remuneration is 62%. This indicates a positive and moderately strong linear relationship between the two variables.

The positive sign of the correlation coefficient indicates that as remuneration increases, the loyalty points tends to increase as well.

It's important to note that correlation only measures the strength and direction of the linear relationship. It does not imply causation. A high correlation between remuneration and loyalty points does not necessarily mean that one variable causes the other to change.

While the correlation coefficient provides valuable information about the relationship between remuneration and loyalty points, it is essential to perform additional analysis and consider other factors. Later, I will calculate the correlation between the loyalty points and the rest of the independent variables in the DataFrame (age).

```
# Add a trendline to visualise the Linear regression.
# Use the NumPy polyval method, specify the regression and the independent variable.
trend = np.polyval(reg, reviews['remuneration'])

# View the previous scatterplot.
plt.figure(figsize=(8, 4), dpi=100)
plt.title("Scatterplot & line of best fit")
plt.xlabel("Remuneration (k£)")
plt.ylabel("Loyalty Points")

sns.scatterplot (data=reviews,
x='remuneration' , y='loyalty_points')

# Set axis values.
#plt.ylim(13, 17)
#plt.xlim(15, 55)

# Add the trendline.
plt.plot(reviews['remuneration'],
         trend, color='red')
plt.show()
```



What we can see so far is that the loyalty points tend to increase when the remuneration is higher (85K - 100K+), however this does not indicate causation between these two variables, since there are customers with hight remunerations who do not accumulate many loyalty points.

```
# Pass linear regression through OLS methods.
test = ols('y ~ x', data = reviews). fit()
# Print the regression table.
test.summary()
```

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.380
Model:	OLS	**Adj. R-squared:**	0.379
Method:	Least Squares	**F-statistic:**	1222.
Date:	Mon, 24 Jul 2023	**Prob (F-statistic):**	2.43e-209
Time:	05:25:18	**Log-Likelihood:**	-16674.
No. Observations:	2000	**AIC:**	3.335e+04
Df Residuals:	1998	**BIC:**	3.336e+04
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-65.6865	52.171	-1.259	0.208	-168.001	36.628
x	34.1878	0.978	34.960	0.000	32.270	36.106

Omnibus:	21.285	**Durbin-Watson:**	3.622
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	31.715
Skew:	0.089	**Prob(JB):**	1.30e-07
Kurtosis:	3.590	**Cond. No.**	123.

Explanation of the results of the model:

How the Loyalty Points are affected by the Remuneration?

R-squared: in this case, around 38% of the total variability of the Loyalty Points is explained by the Remuneration.

Estimate 1: B0, Intercept: in this case, when the remuneration is zero, we would expect to see -66 loyalty points.

Estimate 2: B1, the slope coefficient of the independent variable (remuneration). We are mainly interested in the estimate of the slope. In order to understand whether this is significant in our model, we need to look at the probabilities, (Pr). The smaller the probability is the most important and significant the estimate of that linear regression is. In this case is zero, hence we can say that remuneration is not a very significant estimate of the loyalty points.

The t-value tests the hypothesis that the slope is significant or not. If the corresponding probability is small (typically smaller than 0.05) the slope is significant. In this case, the probability of the t-value is zero, thus the estimated slope is not significant. In other words, a t-value of 34.960 suggests that the remuneration do not have a strong effect on the loyalty points in this linear regression model.

The coefficient of 'x' describes the slope of the regression line, in other words, how much the response variable (loyalty points) change when the remuneration changes by 1 unit. In this case, if the customer remuneration changes by 1 unit (k£) the loyalty points will change by 34.1878 units.

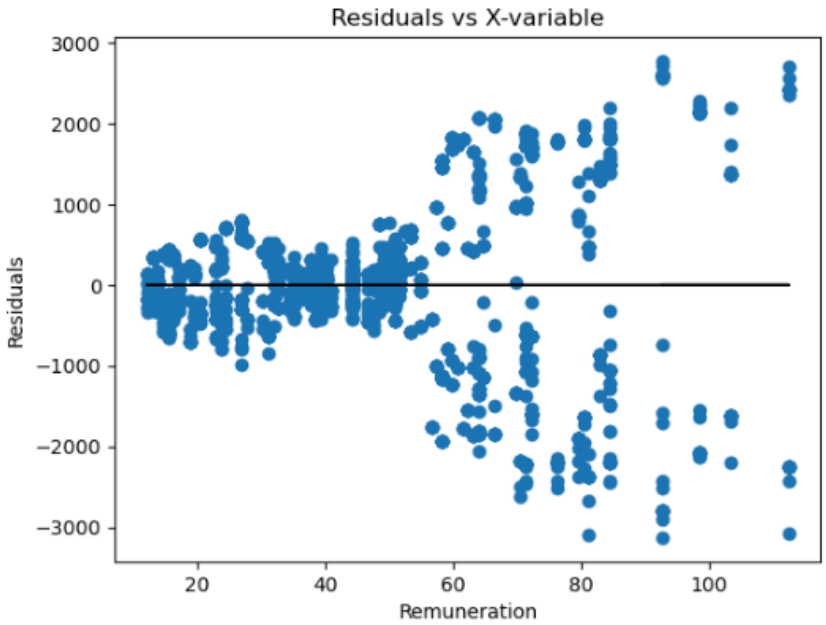
The last two numbers describe the 95% confidence interval of the true coefficient, i.e. the true slope. For instance, if we take a different sample, the estimated slope will be slightly different. If we take 100 random samples each of 500 observations of remunerations and loyalty points, then 95 out of the 100 samples will derive a slope that is within the interval (32.270 , 36.106).

The "DF residuals": in this case the model estimates that 1998 data points will not vary after estimating the model.

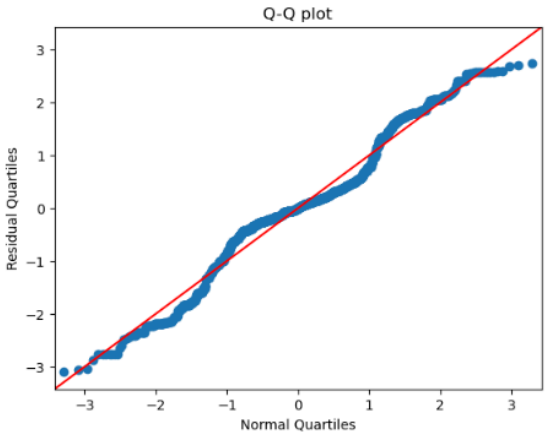
```
# Create the linear regression model.
New_value = 60
y_pred = (-65.6865) +34.1878 * New_value

# View the output
print("If Remuneration is", New_value, "the predicted Loyalty Points are : ", y_pred)
```

```
If Remuneration is 60 the predicted Loyalty Points are :  1985.5815
```

```
# Plot the residuals= (y-predict - y-observe) versus the x-values.
# Ideally, there should be no pattern in this plot

plt.scatter(x, test.predict() - y)
plt.plot(x, y - y, color='black')
plt.title("Residuals vs X-variable")
plt.xlabel("Remuneration")
plt.ylabel("Residuals")
plt.show()
```



```
# Import qqplot Library
from statsmodels.graphics.gofplots import qqplot

# Residuals are calcualted
residuals = test.predict() - y

# Plot the quartiles of the residuals versus the quartiles of the N(0,1)
# The fit=True argument first Standardises the observed data (Residuals) before plotting them
sm.qqplot (residuals, fit=True, line= '45')
plt.ylabel("Residual Quartiles")
plt.xlabel("Normal Quartiles")
plt.title("Q-Q plot")
plt.show()
```



It's good to see a randomly scattered distribution of the residual variables because it indicates that the linear regression model is showing the underlying relationships between the remuneration and the loyalty points effectively.

A randomly scattered distribution of residuals also indicates the absence of any systematic patterns or trends in the residuals and that the model is providing unbiased estimates of the coefficients.

It's also good to see that the residuals variables have a distribution that is very close to normal, since this indicates that the estimated coefficients have smaller standard errors, leading to more precise and reliable estimates, and hypothesis tests are more accurate.

This also means that, on average, the estimated coefficients are equal to the true population coefficients, making the model more accurate in making predictions.

When the residuals are normally distributed, the predictions tend to be more accurate because the model has effectively captured the underlying relationships in the data.

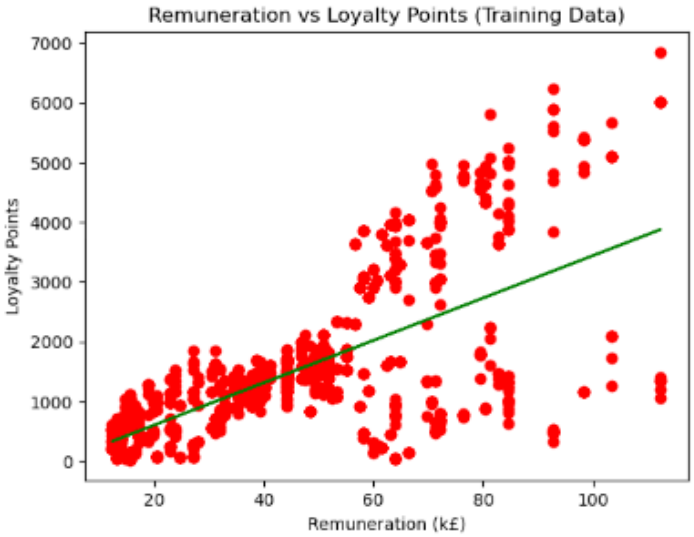
In [47]:
```
# Print the R-squared, intercept and coefficient values of the Testing Data.
print("R-squared value: ", lr.score(x_test, y_test))
print("Intercept value: ", lr.intercept_)
print("Coefficient value: ", lr.coef_)
```

```
R-squared value:  0.32173930708296905
Intercept value:  [-105.23352859]
Coefficient value:  [[35.45189173]]
```

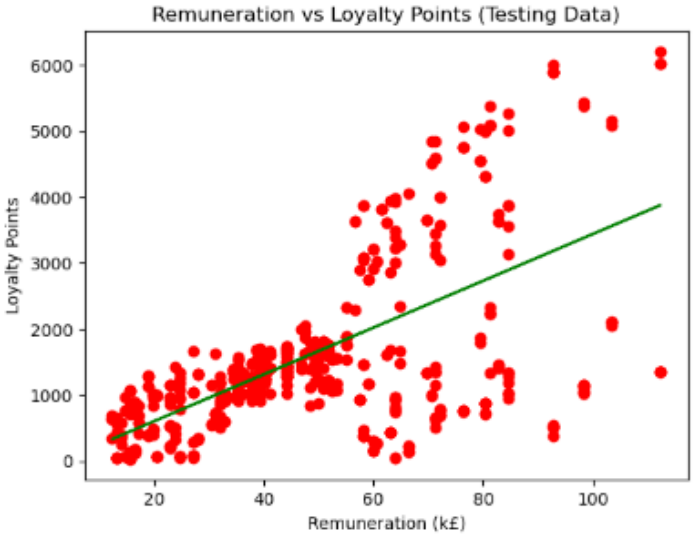
Notes:

- R-squared tells us how much of the variability of Loyalty points is explained by the Remuneration. In this case, around 32% of the observed variation in the Loyalty Points can be explained by the model's inputs. In other words, 32% of the variability of the Loyalty Points is explained by the Remuneration.
- Estimate 1: B0, Intercept: In this case, when the remuneration is zero, we would expect to see -105 loyalty points.
- The coefficient of 'x' describes the slope of the regression line, in other words, how much the response variable 'loyalty points' change when the remuneration changes by 1 unit. In this case, if the remuneration changes by 1 unit (1k£), the loyalty points will change by 35.45 units.

In [45]:
```
# Visualise the training set.
plt.scatter(x_train, y_train, color = 'red')
plt.plot(x_train, lr.predict(x_train), color = 'green')
plt.title("Remuneration vs Loyalty Points (Training Data)")
plt.xlabel("Remuneration (k£)")
plt.ylabel("Loyalty Points")

plt.show()
```



In [46]:
```
# Visualise the test set.
plt.scatter(x_test, y_test, color = 'red')
plt.plot(x_train, lr.predict(x_train), color = 'green')
plt.title("Remuneration vs Loyalty Points (Testing Data)")
plt.xlabel("Remuneration (k£)")
plt.ylabel("Loyalty Points")

plt.show()
```



Age vs Loyalty points

```
]: # Determine the correlation between age and loyalty points.

# Access these columns as follows.
age = reviews['age']
loyalty_points = reviews['loyalty_points']

# Calculate the correlation coefficient between the two columns.
correlation = age.corr(loyalty_points)

print("Correlation between age and loyalty points:", correlation)
```

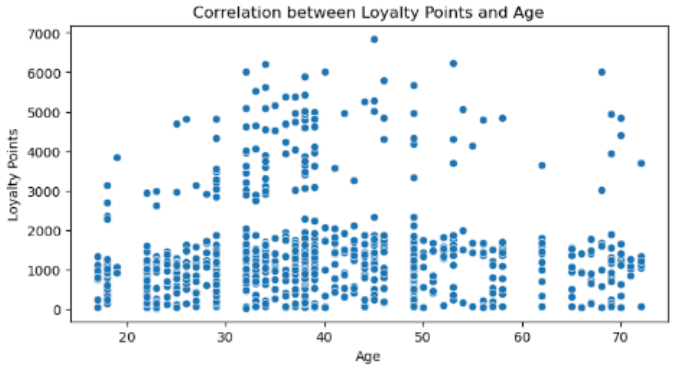
```
Correlation between age and loyalty points: -0.04244464682054187
```

The correlation between the loyalty points and age is -4.2%. This indicates that there is a weak negative linear relationship between age and loyalty points. As age increases, loyalty points tend to slightly decrease, and vice versa. However, since the correlation is close to zero, this linear relationship is very weak, and the points may be scattered around without following a clear pattern.

It's important to note that a weak correlation does not imply causation. The correlation coefficient only measures the strength and direction of the linear relationship between age and loyalty points. It does not imply that age is the cause of changes in loyalty points or vice versa.

The weak correlation may indicate that age alone is not a strong predictor of loyalty points. There may be other factors or interactions with age that better explain variations in loyalty points.

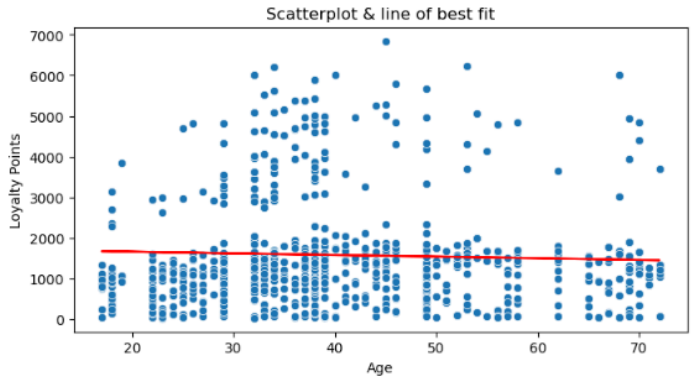
```
In [51]: # Starting with a visualisation BEFORE running linear regression.

plt.figure(figsize=(8, 4), dpi=100)
plt.title("Correlation between Loyalty Points and Age")
plt.xlabel("Age")
plt.ylabel("Loyalty Points")
sns.scatterplot(data=reviews,
                x="age",
                y="loyalty_points")

plt.show()
```




```
In [52]: # Fit the Linear model.
# Polyfit() good to use for simple Linear regression (only one variable).
# Degree = 1, degree of polynomium, for SLR always 1.
reg = np.polyfit(reviews['age'], reviews['loyalty_points'], deg = 1)

# View output.
reg
```

```
Out[52]: array([  -4.01280515, 1736.5177394 ])
```

```
# Add a trendline to visualise the Linear regression.
# Use the NumPy polyval method, specify the regression and the independent variable.
trend = np.polyval(reg, reviews['age'])

# View the previous scatterplot.
plt.figure(figsize=(8, 4), dpi=100)
plt.title("Scatterplot & line of best fit")
plt.xlabel("Age")
plt.ylabel("Loyalty Points")

sns.scatterplot (data=reviews,
x='age' , y='loyalty_points')

# Set axis values.
#plt.ylim(13, 17)
#plt.xlim(15, 55)

# Add the trendline.
plt.plot(reviews['age'],
         trend, color='red')
plt.show()
```



What we can see so far is that there is no a clear pattern or linear relationship between age and loyalty points.

The correlation between the loyalty points and age shown beore (-4.2%), indicates that there is a weak negative linear relationship between age and loyalty points, that is difficult to appreciate. As age increases, loyalty points tend to slightly decrease, and vice versa. However, since the correlation is negative, this linear relationship is very weak, and this why we see the points scattered around without following a clear pattern.

Next, we'll see how good is the fit line.

In [54]:
```
# Pass linear regression through OLS methods.
test = ols('y ~ x', data = reviews). fit()
# Print the regression table.
test.summary()
```

Out[54]:

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.002
Model:	OLS	**Adj. R-squared:**	0.001
Method:	Least Squares	**F-statistic:**	3.606
Date:	Mon, 24 Jul 2023	**Prob (F-statistic):**	0.0577
Time:	05:26:50	**Log-Likelihood:**	-17150.
No. Observations:	2000	**AIC:**	3.430e+04
Df Residuals:	1998	**BIC:**	3.431e+04
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1736.5177	88.249	19.678	0.000	1563.449	1909.587
x	-4.0128	2.113	-1.899	0.058	-8.157	0.131

Omnibus:	481.477	**Durbin-Watson:**	2.277
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	937.734
Skew:	1.449	**Prob(JB):**	2.36e-204
Kurtosis:	4.688	**Cond. No.**	129.

Explanation of the results of the model:

How the Loyalty Points are affected by the Age of the customers?

R-squared: in this case, around 0.2% of the variability of the Loyalty Points is explained by the Age.

Estimate 1: B0, Intercept: in this case, when the age is zero, we would expect to see 1736 loyalty points. This does not make sense, at it does not make sense either for all these independent values (age, remuneration and spending score)to be zero for a specific customer.

Estimate 2: B1, the slope coefficient of the independent variable (age). We are mainly interested in the estimate of the slope. In order to understand whether this is significant in our model, we need to look at the probabilities, (Pr). The smaller the probability is the most important and significant the estimate of that linear regression is. In this case is 0.058, hence we can say that age is not a significant estimate of the loyalty points.

The t-value tests the hypothesis that the slope is significant or not. If the corresponding probability is small (typically smaller than 0.05) the slope is significant. In this case, the probability of the t-value is 0.058, thus the estimated slope is not significant. In other words, a t-value of -1.899 suggests that the age do not have effect on the loyalty points in this linear regression model.

The coefficient of 'x' describes the slope of the regression line, in other words, how much the response variable (loyalty points) change when the age changes by 1. In this case, if the customer's age changes by 1 year, the loyalty points will change by -4.0128 units.

The last two numbers describe the 95% confidence interval of the true xcoefficient, i.e. the true slope. For instance, if we take a different sample, the estimated slope will be slightly different. If we take 100 random samples each of 500 observations of ages and loyalty points, then 95 out of the 100 samples will derive a slope that is within the interval (-8.157 , 0.131).

The "DF residuals": the degrees of freedom for the residuals indicate the number of data points that are free to vary after estimating the model. In this case the model estimates that 1998 data points will not vary after estimating the model.

In [55]:
```
#Extract the predicted values.
print("Predicted values: ", test.predict())
```

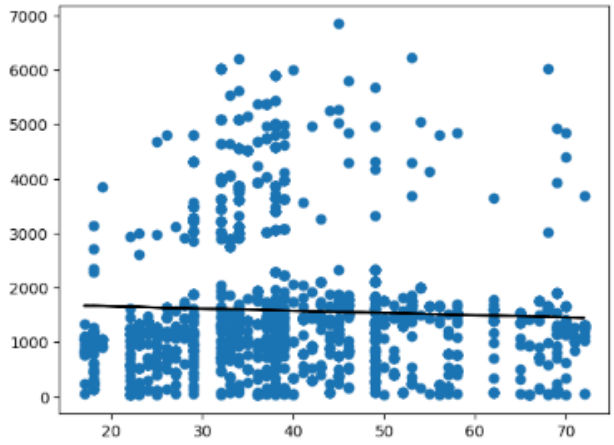
```
Predicted values:  [1664.2872467  1644.22322095 1648.2360261  ... 1600.0823643  1600.0823643
 1608.1079746 ]
```

In [56]:
```
# x coef: -4.0128.
# Constant coef: 1736.5177.
# Create the linear equation.
y_pred =(1736.5177) -4.0128 * x

# View the output.
y_pred
```

Out[56]:
```
0       1664.2873
1       1644.2233
2       1648.2361
3       1636.1977
4       1604.0953
          ...
1995    1588.0441
1996    1563.9673
1997    1600.0825
1998    1600.0825
1999    1608.1081
Name: age, Length: 2000, dtype: float64
```

In [57]:
```
# Plot the data points.
plt.scatter(x, y)

# Plot the line.
plt.plot(x, y_pred, color='black')
```

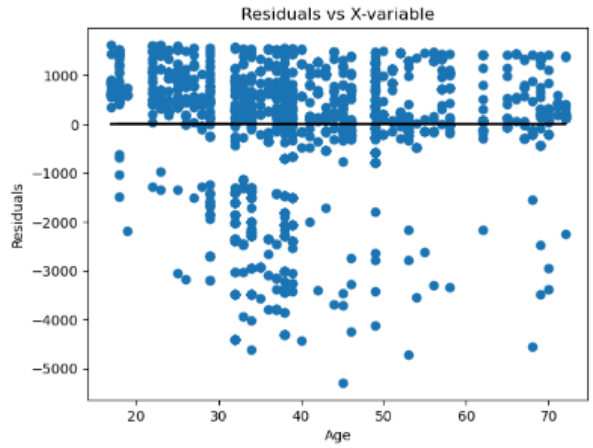
Out[57]:
```
[<matplotlib.lines.Line2D at 0x15b17fe50>]
```



In [58]:
```
# Create the linear regression model.
New_value = 30
y_pred = (1736.5177) -4.0128 * New_value

# View the output
print("If Age is", New_value, "the predicted Loyalty Points are : ", y_pred)
```

```
If Age is 30 the predicted Loyalty Points are :  1616.1337
```

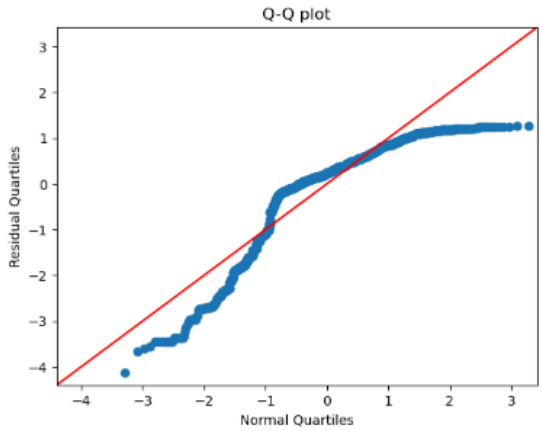
In [59]:
```
# Plot the residuals= (y-predict - y-observe) versus the x-values.
# Ideally, there should be no pattern in this plot

plt.scatter(x, test.predict() - y)
plt.plot(x, y - y, color='black')
plt.title("Residuals vs X-variable")
plt.xlabel("Age")
plt.ylabel("Residuals")
plt.show()
```



A randomly scattered distribution of the residuals is desirable because it indicates that the linear regression model is showing the underlying relationships between the predictor variable (age) and the dependent variable (loyalty points) effectively.

A randomly scattered distribution of residuals also indicates the absence of any systematic patterns or trends in the residuals and that the model is providing unbiased estimates of the coefficients.

In [60]:
```
# Import qqplot Library
from statsmodels.graphics.gofplots import qqplot

# Residuals are calcualted.
residuals = test.predict() - y

# Plot the quartiles of the residuals versus the quartiles of the N(0,1)
# The fit=True argument first Standardises the observed data (Residuals) before plotting them
sm.qqplot (residuals, fit=True, line= '45')
plt.ylabel("Residual Quartiles")
plt.xlabel("Normal Quartiles")
plt.title("Q-Q plot")
plt.show()
```

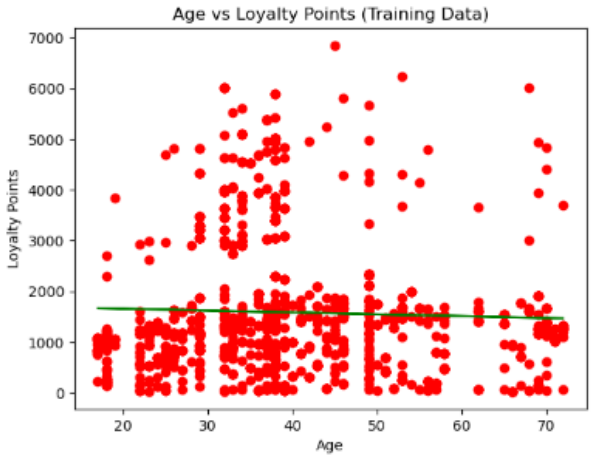


It's also good to see that the residuals variables have a distribution that is somehow close to normal, since the predictions tend to be more accurate because the model has effectively captured the underlying relationships in the data.

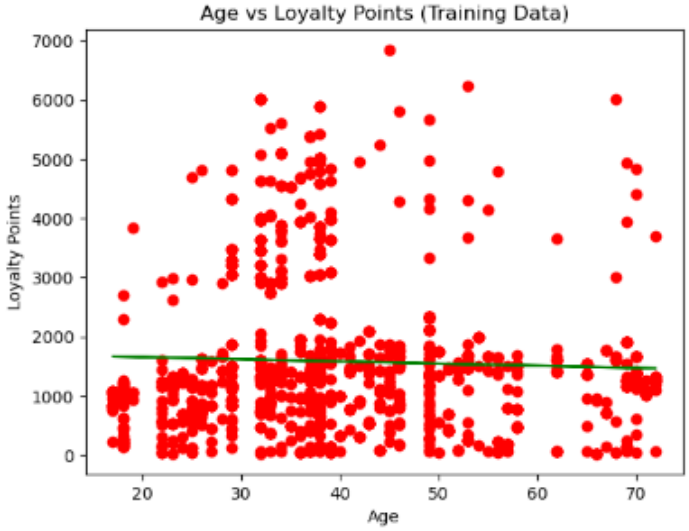
In [62]:
```
# Run linear regression model.
lr = LinearRegression()

# Fit the model on the training data.
lr.fit(x_train, y_train)

# Predict is used for predicting on the x_test.
y_pred = lr.predict(x_test)

# View the output.
y_pred
```

Out[62]:
```
array([[1465.83351109],
       [1590.78902174],
       [1609.16483213],
       [1645.91645291],
       [1550.36223888],
       [1502.58513187],
       [1664.2922633 ],
       [1587.11385966],
       [1660.61710122],
       [1587.11385966],
       [1546.68707681],
       [1561.38772512],
       [1546.68707681],
       [1587.11385966],
       [1543.01191473],
       [1627.54064252],
       [1561.38772512],
       [1517.28578018],
       [1550.36223888],
```

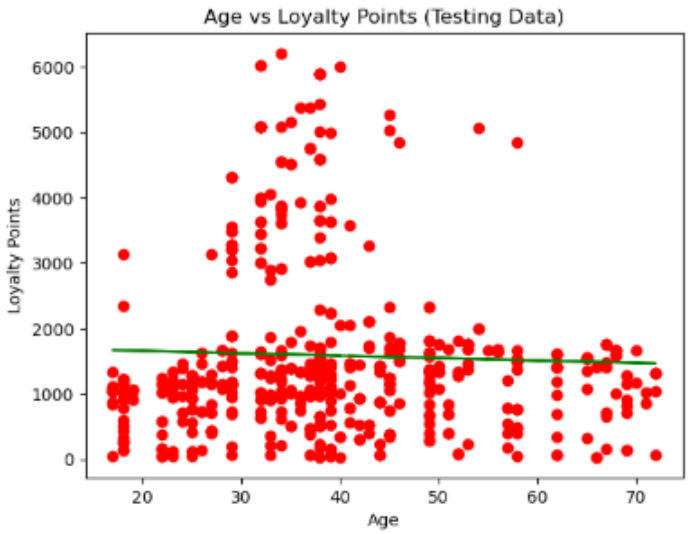
In [63]:
```
# Visualise the training set.
plt.scatter(x_train, y_train, color = 'red')
plt.plot(x_train, lr.predict(x_train), color = 'green')
plt.title("Age vs Loyalty Points (Training Data)")
plt.xlabel("Age")
plt.ylabel("Loyalty Points")

plt.show()
```



In [63]:
```
# Visualise the training set.
plt.scatter(x_train, y_train, color = 'red')
plt.plot(x_train, lr.predict(x_train), color = 'green')
plt.title("Age vs Loyalty Points (Training Data)")
plt.xlabel("Age")
plt.ylabel("Loyalty Points")

plt.show()
```



In [64]:
```
# Plot graph with regression line.
# Visualise the test set.
plt.scatter(x_test, y_test, color = 'red')
plt.plot(x_train, lr.predict(x_train), color = 'green')
plt.title("Age vs Loyalty Points (Testing Data)")
plt.xlabel("Age")
plt.ylabel("Loyalty Points")

plt.show()
```



In [65]:
```
# Print the R-squared, intercept and coefficient values of the Testing Data.
print("R-squared value: ", lr.score(x_test, y_test))
print("Intercept value: ", lr.intercept_)
print("Coefficient value: ", lr.coef_)
```

```
R-squared value:  0.002088348289358777
Intercept value:  [1730.4451807]
Coefficient value:  [[-3.67516208]]
```

6. Observations and insights

Your observations here...

- R-squared: in this case, around 0.2% of the observed variation in the Loyalty Points can be explained by the model's inputs. In other words, 0.2% of the variability of the Loyalty Points is explained by the customer's age.
- Estimate 1: B0, Intercept: In this case, when the age is zero, we would expect to see 1730 loyalty points. This does not make sense at all, at it does not make sense either for all these independent values (age, remuneration and spending score)to be zero for a specific customer.
- The coefficient of 'x': in this case, if the age increases by 1 year, the loyalty points willdecrease by 3.67 units.

B. Making predictions with clustering using Python

I'll apply k-means clustering to determine how useful remuneration and spending scores are in providing data for analysis.

Import the necessary libraries, and prepare the data for clustering.

Create a new DataFrame (e.g. df2) containing the remuneration and spending_score columns.

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

from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.metrics import accuracy_score
from scipy.spatial.distance import cdist

import warnings
warnings.filterwarnings('ignore')
```

```
# Load the CSV file(s) as df2.
df2 = pd.read_csv('clean_reviews.csv')

# View the DataFrame.
df2.head()
```

	gender	age	remuneration	spending_score	loyalty_points	education	product	review	summary
0	Male	18	12.30	39	210	graduate	453	When it comes to a DM's screen, the space on t...	The fact that 50% of this space is wasted on a...
1	Male	23	12.30	81	524	graduate	466	An Open Letter to GaleForce9*:\n\nYour unpaint...	Another worthless Dungeon Master's screen from...
2	Female	22	13.12	6	40	graduate	254	Nice art, nice printing. Why two panels are f...	pretty, but also pretty useless
3	Female	25	13.12	77	562	graduate	263	Amazing buy! Bought it as a gift for our new d...	Five Stars
4	Female	33	13.94	40	366	graduate	291	As my review of GF9's previous screens these w...	Money trap

```
# Drop unnecessary columns.
df2.drop(df2.columns[[0, 1, 4, 5, 6, 7, 8]], axis=1, inplace=True)

# View DataFrame.
df2.head()
```

	remuneration	spending_score
0	12.30	39
1	12.30	81
2	13.12	6
3	13.12	77
4	13.94	40

```
# Explore the data.
# Any missing values?
df2_na = df2[df2.isna().any(axis=1)]
df2_na.shape
```

```
(0, 2)
```

```
# Any null values?
df2.isnull().sum()
```

```
remuneration      0
spending_score    0
dtype: int64
```

Explore the new DataFrame.

```
# Explore the data.
# Determine the metadata of the 'reviews' DataFrame.
print(df2.columns)
print(df2.shape)
print(df2.dtypes)
df2.info()
```

```
Index(['remuneration', 'spending_score'], dtype='object')
(2000, 2)
remuneration      float64
spending_score      int64
dtype: object
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 2 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   remuneration    2000 non-null   float64
 1   spending_score  2000 non-null   int64
dtypes: float64(1), int64(1)
memory usage: 31.4 KB
```

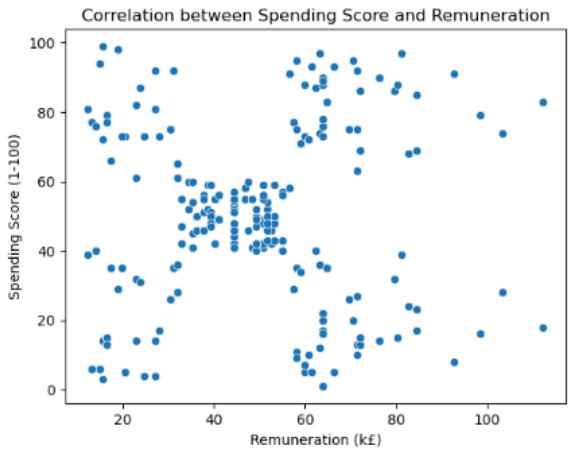
```
# Descriptive statistics.
df2.describe()
```

	remuneration	spending_score
count	2000.000000	2000.000000
mean	48.079060	50.000000
std	23.123984	26.094702
min	12.300000	1.000000
25%	30.340000	32.000000
50%	47.150000	50.000000
75%	63.960000	73.000000
max	112.340000	99.000000

Plot the remuneration versus spending score to determine any correlations and possible groups (clusters).

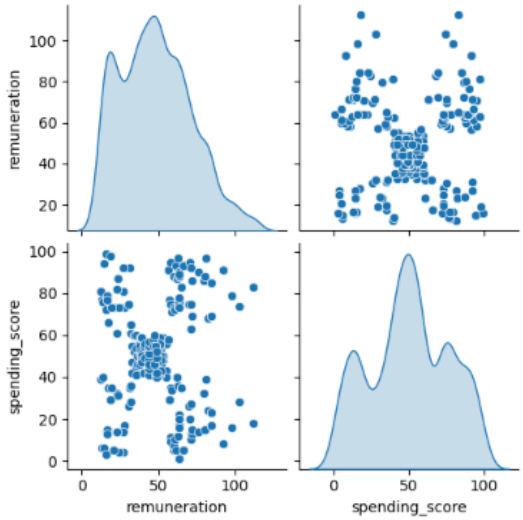
In [9]:
```
# Create a scatterplot with Seaborn.
plt.title("Correlation between Spending Score and Remuneration")
plt.xlabel("Remuneration (k£)")
plt.ylabel("Spending Score (1-100)")
sns.scatterplot(x='remuneration',
                y='spending_score',
                data=df2)
```

Out[9]:
```
<Axes: title={'center': 'Correlation between Spending Score and Remuneration'}, xlabel='Remuneration (k£)', ylabel='S
pending Score (1-100)'>
```



In [10]:
```
# Create a pairplot with Seaborn.
x = df2[['remuneration', 'spending_score']]

sns.pairplot(df2,
             vars=x,
             diag_kind= 'kde')
```

Out[10]: <seaborn.axisgrid.PairGrid at 0x1625539d0>

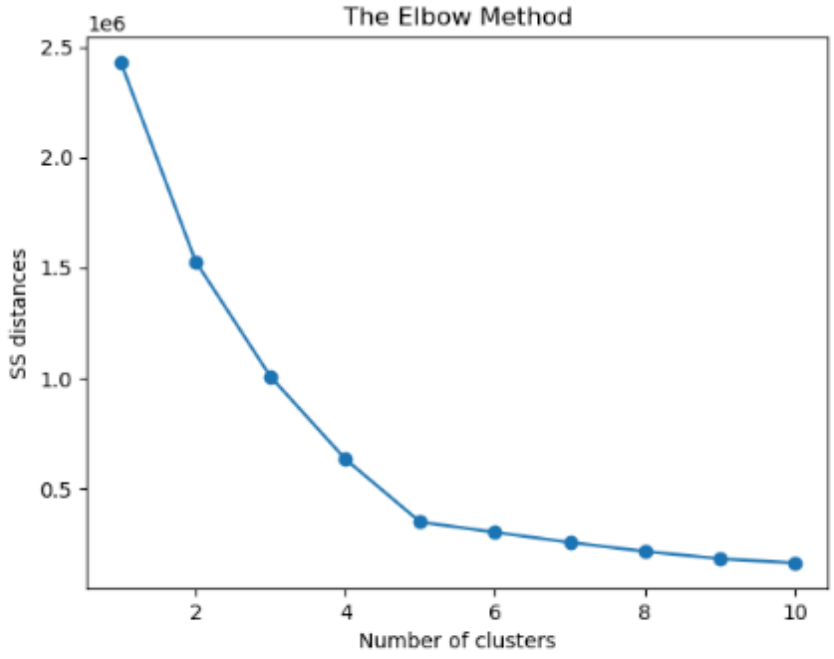


What we can see so far is that there are five clusters. Out of these five, there is one in the middle that seems to be the most prominent. Next, I'll employ the K-means clustering to determine the number of clusters using two methods: elbow and silhoutte methods.

Use the Silhouette and Elbow methods to determine the optimal number of clusters for *k*-means clustering.

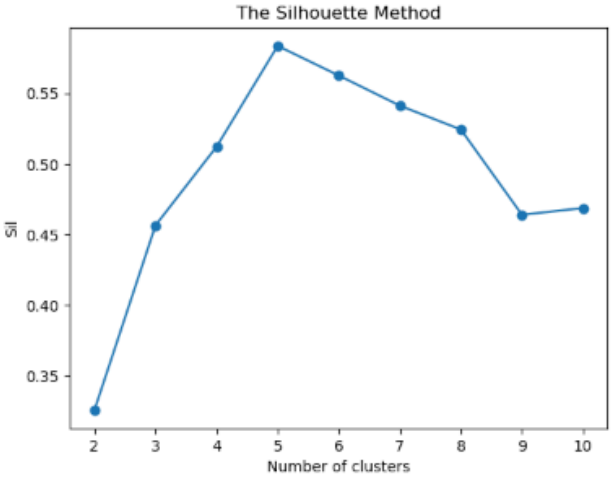
Plot both methods, and explained how you determine the number of clusters to use.

In [10]:
```
# Determine the number of clusters: Elbow method.
# Import the KMeans class.
from sklearn.cluster import KMeans

# Elbow chart for us to decide on the number of optimal clusters.
ss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i,
                    init = 'k-means++',
                    max_iter = 300,
                    n_init = 10,
                    random_state = 0)
    kmeans.fit(x)
    ss.append(kmeans.inertia_)

plt.plot(range(1, 11),
         ss,
         marker='o')

plt.title("The Elbow Method")
plt.xlabel("Number of clusters")
plt.ylabel("SS distances")
plt.show()
```



Based on the Elbow method, five clusers have been identified. Next, I'll apply the Silhouette method.

In [12]:
```
# Determine the number of clusters: Silhouette method.
# Import silhouette_score class from sklearn.
from sklearn.metrics import silhouette_score

# Find the range of clusters to be used using silhouette method.
sil = []
kmax = 10

for k in range(2, kmax+1):
    kmeans_s = KMeans(n_clusters = k).fit(x)
    labels = kmeans_s.labels_
    sil.append(silhouette_score(x,
                                labels,
                                metric = 'euclidean'))

# Plot the silhouette method.
plt.plot(range(2, kmax+1),
         sil,
         marker='o')

plt.title("The Silhouette Method")
plt.xlabel("Number of clusters")
plt.ylabel("Sil")

plt.show()
```



Based on the Silhouette method, five main clusters have been identified, meaning that the 5 cluster solution has the highest average silhouette coefficient. Based on our data set, we will have to judge what number of clusters is the most appropriate to consider.

At the moment, we are only comparing and plotting two variables (remuneration and spending score), however if we wanted to introduce more variables (attributes) from our data set, let's say for instance, we wanted to introduce the gender of the customers, we would be able to exrtact more insights. For now, I'll keep looking at these two variables, and I'll evaluate later in my analysis, wheter or not it's worth introducing the gender variable as the hue of these plots.

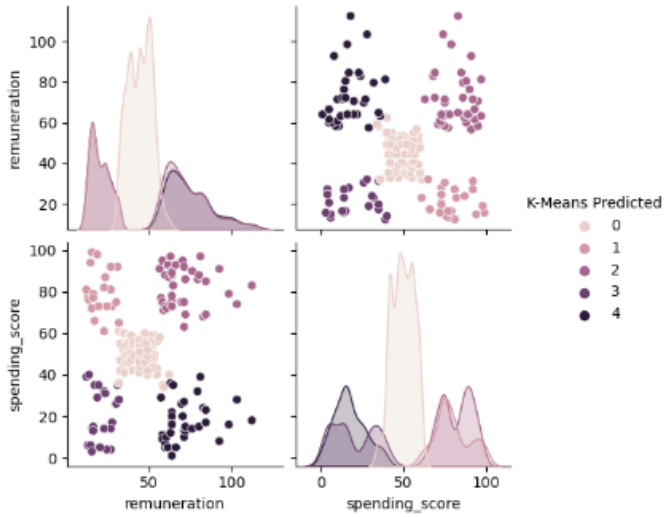
Based on the number of clusters suggested by the elbow and silhouette methods calculated earlier, I'll use 5 clusters first, since this number seems to be the most appropriate, and I will test the k means clustering with three and seven clusters to identify the behaviour of the data when using different number of clusters and see how they compare.

Evaluate the usefulness of at least three values for *k* based on insights from the Elbow and Silhouette methods.

Plot the predicted *k*-means.

In [13]:
```
# Use five clusters:
kmeans = KMeans(n_clusters = 5,
                max_iter = 15000,
                init='k-means++',
                random_state=42).fit(x)

clusters = kmeans.labels_
x['K-Means Predicted'] = clusters

# Plot the predicted.
sns.pairplot(x,
             hue='K-Means Predicted',
             diag_kind= 'kde')
```

Out[13]: <seaborn.axisgrid.PairGrid at 0x16c6ebaf0>



If we were to use five colusters solution, based on the classification on those 2000 observations to one of these five clusters solution, what the distribution of the remuneration and the spendig score are for these 5 distinct clusters.

If we look at the remuneration alone or the spending score alone, it is hard to identify these five clusters, however when we look at the remuneration and the spending score together, we can easily identify the five clusters with a different colour. What this is telling us is that if we wanted to target different customer segments within our data base, it might be a bit more difficult to identify those segments if we were to look at the remuneration only or the spending score only.

In this case, it might be better to consider these two variables together to identify different customer's segments to target in our marketing campaigns.

Cluster 0: customers with a remuneration between (£35K - (£55K) and a spending score between 35 - 60 points) Cluster 1: customers with a remuneration between (£15K - (£35K) and a spending score between 60 - 100 points) Cluster 2: customers with a remuneration between (£55K - (£100K+) and a spending score between 60 - 100 points) Cluster 3: customers with a remuneration between (£15K - (£35K) and a spending score between 5 - 40 points) Cluster 4: customers with a remuneration between (£55K - (£100K+) and a spending score between 5 - 40 points)

Based on the remuneration ranges and the spending score ranges for each of the clusters (customer segments), we can say that the spending score and the remuneration are widely distributed across the entire sample, meaning that neither the remuneration nor the spendig score are clear indicatives of the other, in other words none of these variables are causative of the other, so for us to better undertand and identify each of the segments, we have to consider these two variables togeter.

As mentioned earlier, in future analysis, it might be interesting to look at other variables along with these two, such as the customer gender, age and education. We would introduce these additional variables as the hue of these plots.

In [14]:
```
# Check the number of observations per predicted class.
x['K-Means Predicted'].value_counts()
```

Out[14]:
```
0    774
2    356
4    330
3    271
1    269
Name: K-Means Predicted, dtype: int64
```

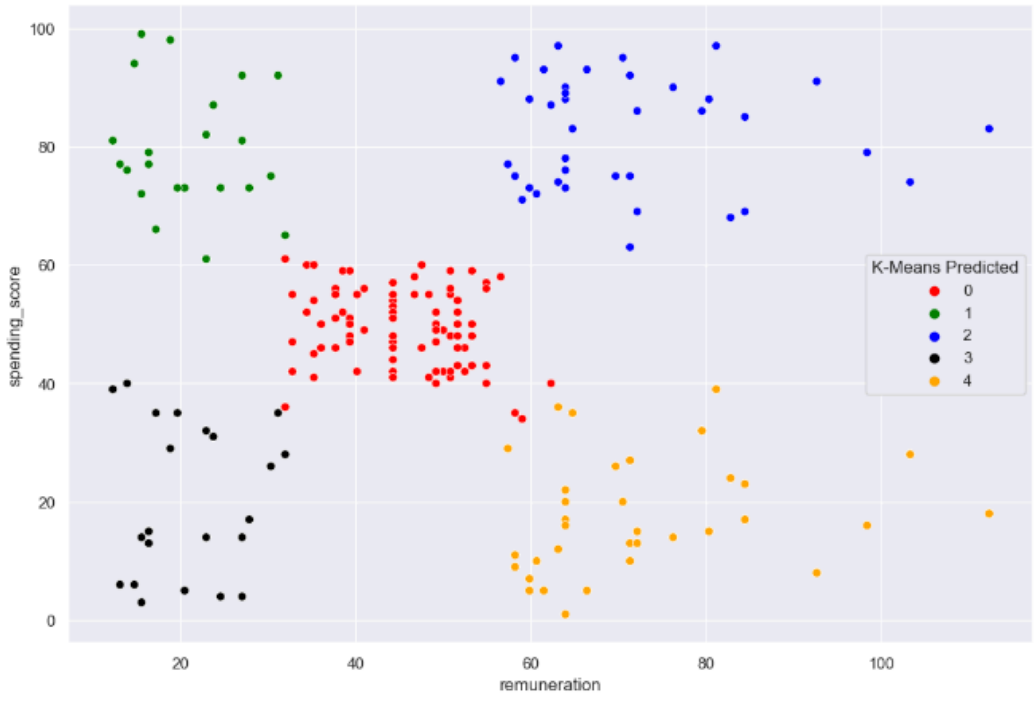
By looking at the number of observations by cluster, we can see that cluster zero has the highest number of observations (774). If we add up the number of obserevations of each of these five clusters, we get the total number of obseervations in our original reviews data set. This tells us that this number of clusters is accurately considering the correct number of observations in our data set and there is no misclassification.

In [15]:
```
# View the K-Means predicted.
print(x.head())
```

```
   remuneration  spending_score  K-Means Predicted
0         12.30              39                  3
1         12.30              81                  1
2         13.12               6                  3
3         13.12              77                  1
4         13.94              40                  3
```

In [16]:
```
# Visualising the clusters.
# Set plot size.
sns.set(rc = {'figure.figsize':(12, 8)})

sns.scatterplot(x='remuneration' ,
                y ='spending_score',
                data=x ,
                hue='K-Means Predicted',
                palette=['red', 'green', 'blue', 'black', 'orange'])
```

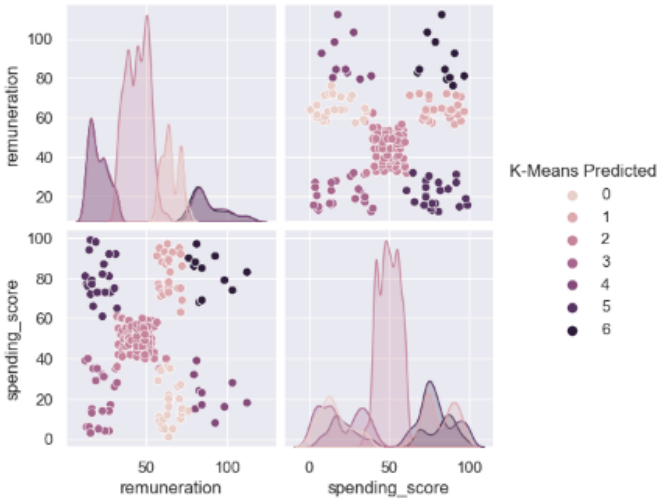
Out[16]:
```
<Axes: xlabel='remuneration', ylabel='spending_score'>
```



Next, I will use seven clusters.

In [17]:
```
# Use 7 clusters:
kmeans = KMeans(n_clusters = 7,
                max_iter = 15000,
                init='k-means++',
                random_state=0).fit(x)

clusters = kmeans.labels_
x['K-Means Predicted'] = clusters

# Plot the predicted.
sns.pairplot(x,
             hue='K-Means Predicted',
             diag_kind= 'kde')
```

Out[17]:
```
<seaborn.axisgrid.PairGrid at 0x1778ad840>
```



In [18]:
```
# Check the number of observations per predicted class.
x['K-Means Predicted'].value_counts()
```

Out[18]:
```
2    767
3    271
5    269
1    227
0    214
6    129
4    123
Name: K-Means Predicted, dtype: int64
```

By looking at the number of observations by cluster, we can see that cluster two has the highest number of observations (767). If we add up the number of obserevations of each of these seven clusters, we get the total number of obseervations in our original reviews data set. This tells us that this number of clusters is accurately considering the correct number of observations in our data set too and there is no misclassification.

We can also see that, cluster six and cluster four are the clusters with the lowest number of observations.

Considering that Turtle Games is looking to identify groups within the customer base that can be used to target specific market segments for their marketing campaigns, our job is to identify the optimal number of clusters for this purpose.

Looking for instane at those customers within cluster four, with a remuneration between 75K - 100K+, they behave very similar to those customers within cluster zero, with a remuneration between 50K - 75K, inthe sense that these two clusters share the same spending score being within the same range (5 - 40 points).

The same happens when we look at cluster six, and cluster one, even thought we can also identify two groups of customers within thse two clusters in terms of their remuneration, (cluster six being the customers with the higher remunertation between 75K - 100K, and cluster one with customers with a reunerration between 50K - 75K, both of these clusters share a similar spending score (60 - 100 points).

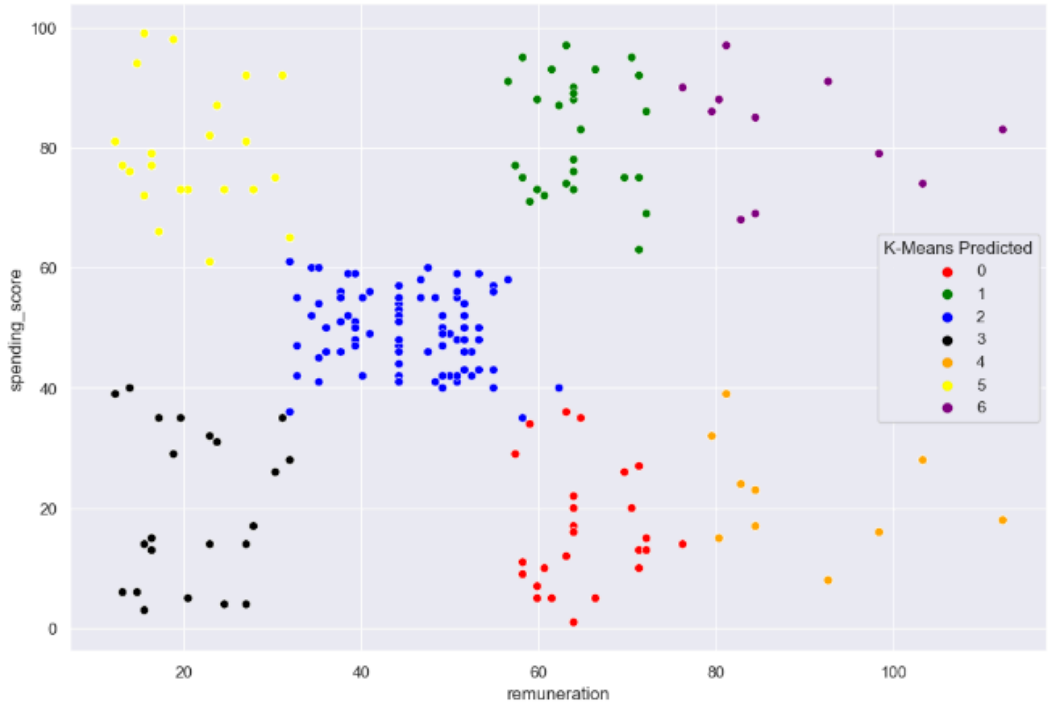
This is telling us that it might be worth grouping clusters six and one together as they were grouped when looking only five clusters, and grouping clusters four and zero together.

In [19]:
```
# View the K-Means predicted.
print(x)
```

	remuneration	spending_score	K-Means Predicted
0	12.30	39	3
1	12.30	81	5
2	13.12	6	3
3	13.12	77	5
4	13.94	40	3
...	...	...	...
1995	84.46	69	6
1996	92.66	8	4
1997	92.66	91	6
1998	98.40	16	4
1999	92.66	8	4

In [20]:
```
# Visualising the seven clusters.
# Set plot size.
sns.set(rc = {'figure.figsize':(12, 8)})

sns.scatterplot(x='remuneration' ,
                y ='spending_score',
                data=x , hue='K-Means Predicted',
                palette=['red', 'green', 'blue', 'black', 'orange', 'yellow', 'purple'])
```

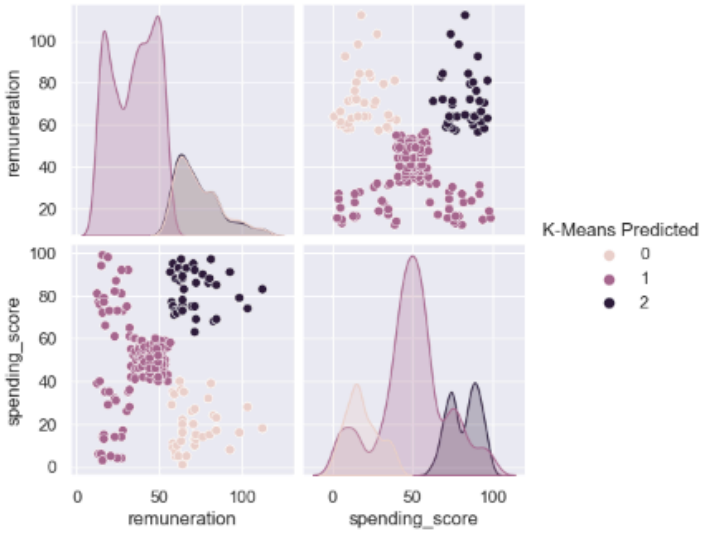
Out[20]: <Axes: xlabel='remuneration', ylabel='spending_score'>



Next, I will use 3 clusters.

In [21]:
```
# Use three clusters:
kmeans = KMeans(n_clusters = 3,
                max_iter = 15000,
                init='k-means++',
                random_state=0).fit(x)

clusters = kmeans.labels_
x['K-Means Predicted'] = clusters

# Plot the predicted.
sns.pairplot(x,
             hue='K-Means Predicted',
             diag_kind= 'kde')
```

Out[21]:
```
<seaborn.axisgrid.PairGrid at 0x16b49be50>
```



In [22]:
```
# Check the number of observations per predicted class.
x['K-Means Predicted'].value_counts()
```

Out[22]:
```
1    1293
2     356
0     351
Name: K-Means Predicted, dtype: int64
```

By looking at the number of observations by cluster, we can see that cluster one has the highest number of observations (1293). If we add up the number of obserevations of each of these three clusters, we get the total number of obseervations in our original reviews data set. This tells us that this number of clusters is accurately considering the correct number of observations in our data set too and there is no misclassification.

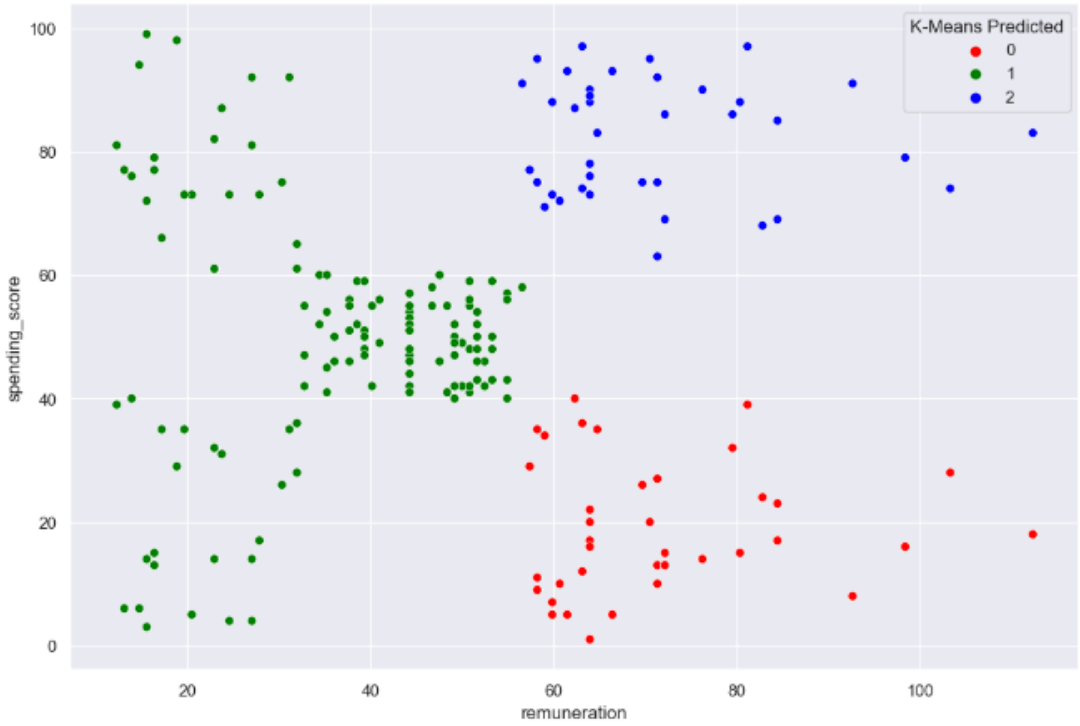
:
```
# Check the number of observations per predicted class.
x['K-Means Predicted'].value_counts()
```

:
```
1    1293
2     356
0     351
Name: K-Means Predicted, dtype: int64
```

By looking at the number of observations by cluster, we can see that cluster one has the highest number of observations (1293). If we add up the number of obserevations of each of these three clusters, we get the total number of obseervations in our original reviews data set. This tells us that this number of clusters is accurately considering the correct number of observations in our data set too and there is no misclassification.

```
# Visualising the three clusters.
# Set plot size.
sns.set(rc = {'figure.figsize':(12, 8)})

sns.scatterplot(x='remuneration' ,
                y ='spending_score',
                data=x , hue='K-Means Predicted',
                palette=['red', 'green', 'blue'])
```

```
<Axes: xlabel='remuneration', ylabel='spending_score'>
```

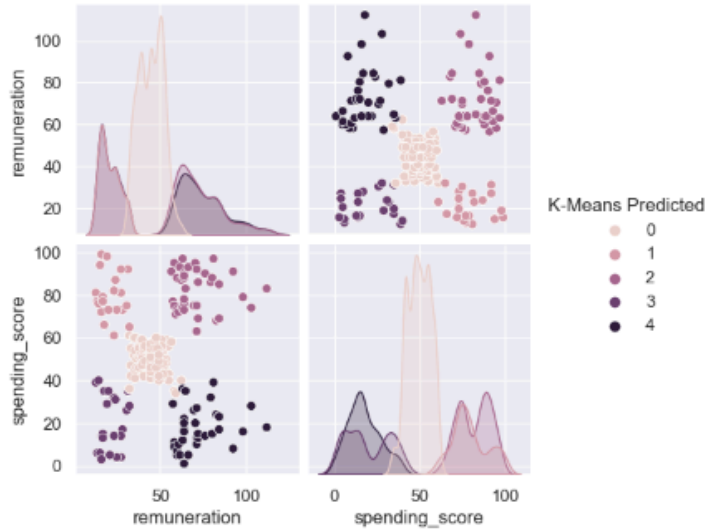


Fit the final model using your selected value for *k and j*ustify your selection.

5. Fit final model and justify your choice

In [25]:
```
# Apply the final model.
# Use 5 clusters:
kmeans = KMeans(n_clusters = 5,
                max_iter = 15000,
                init='k-means++',
                random_state=42).fit(x)

clusters = kmeans.labels_
x['K-Means Predicted'] = clusters

# Plot the predicted.
sns.pairplot(x,
             hue='K-Means Predicted',
             diag_kind= 'kde')
```

Out[25]: <seaborn.axisgrid.PairGrid at 0x177873310>



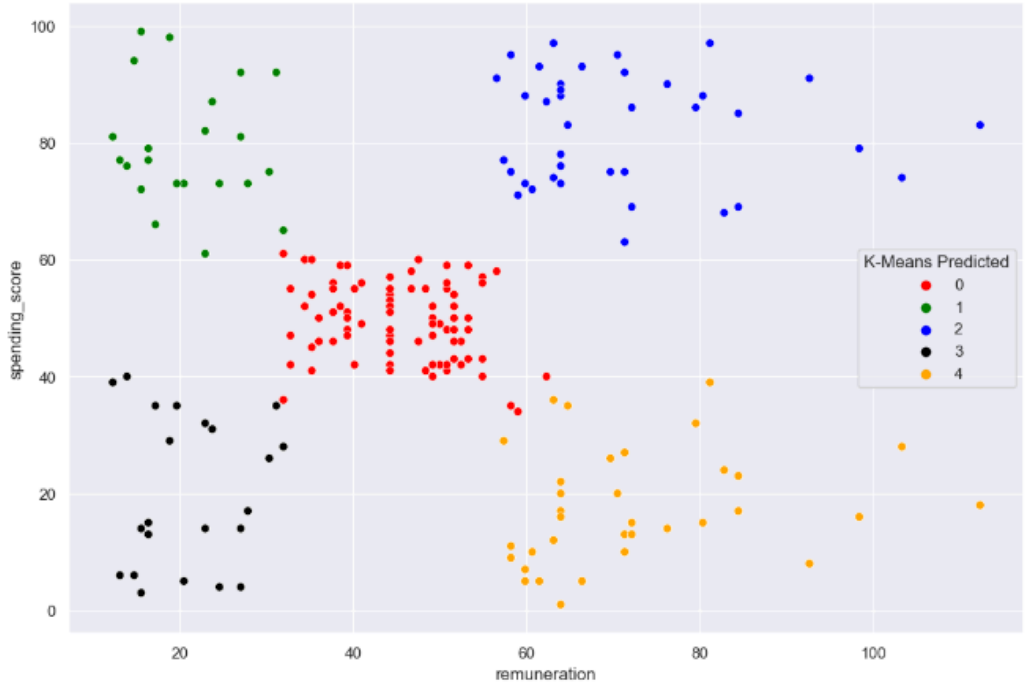
In [26]:
```
# Check the number of observations per predicted class.
x['K-Means Predicted'].value_counts()
```

Out[26]:
```
0    774
2    356
4    330
3    271
1    269
Name: K-Means Predicted, dtype: int64
```

6. Plot and interpret the clusters

In [27]:
```
# Visualising the clusters.
# Set plot size.
sns.set(rc = {'figure.figsize':(12, 8)})

sns.scatterplot(x='remuneration' ,
                y ='spending_score',
                data=x ,
                hue='K-Means Predicted',
                palette=['red', 'green', 'blue', 'black', 'orange'])

# View the DataFrame.
# View the K-Means predicted.
print(x)
```

```
      remuneration  spending_score  K-Means Predicted
0            12.30              39                  3
1            12.30              81                  1
2            13.12               6                  3
3            13.12              77                  1
4            13.94              40                  3
...            ...             ...                ...
1995         84.46              69                  2
1996         92.66               8                  4
1997         92.66              91                  2
1998         98.40              16                  4
1999         92.66               8                  4

[2000 rows x 3 columns]
```



The number of predicted values per class seems to indicate a better distribution for k=5 than k=7 and k=3, hence I selected five clusters as the optimal number of clusters based on all the analysis and insights mentioned above.

We can clearly see that there is a distinct demarcation in terms of remuneration and spending scores, however this is not an indication of the other variable, menaing that there's not a causative relationship between these two variables.

Clusters two and four are the groups with the highest reunertation, however this is not an indication of a high spending score.

And clusters three and one are the customers with the lowest remuneration, but again this is not an indication of a lower spending sore.

The fifth group is cluster zero, which might be the easier segment to taget for Turtle Games, since this group of customers seems to behave more homogenously and isolated in terms of all of the obserbations within our data set grouping customers with a remuneration between (£35K - (£55K) approx. and a spending score between 35 - 60 points), with no other clusters showing this combination of reemuneration and spendig score, unlike the rest of the clusters that share some similarities of behaviour in terms of sharing either the same remuneration level with another cluster or sharing the same spending score with another cluster.

So, my proposal of segments for Turtle Games to target in their marketing campaigns would be as follow:

Segment 1: customers with a remuneration between (£35K - (£55K) and a spending score between 35 - 60 points) Segment 2: customers with a remuneration between (£15K - (£35K) and a spending score between 60 - 100 points) Segment 3: customers with a remuneration between (£55K - (£100K+) and a spending score between 60 - 100 points) Segment 4: customers with a remuneration between (£15K - (£35K) and a spending score between 5 - 40 points) Segment 5: customers with a remuneration between (£55K - (£100K+) and a spending score between 5 - 40 points)

By grouping Turtle Games within the above segments, we could say that the spending score is an indication of the number of purchases or the amount of money spent by customers during a specific range of time, indicating that regardless of the customer remuneration, there customers with a low remuneration that tend to have a high spending score (we can assume that this is because they spend more), these customers belong to segment 2, and then to give another example of how customers behave, we can see that for instance, customers within segment 5, with a high remuneration above 60K, tend to have a lower spendig score, below 40 points (we can assume that this is because they spend less).

In future analysis, it might be worth looking at the behaviour of these segments by considering a specifc period of time, let's say one year, one month, or even quartley, so we can then compare the customers behaviour across different years, months or quarters.

It might also be interesting for future analysis, to introduce a third variable into our clustering analysis, such as the customer age, gender or education, based on the current data base that Turtle Games has.

C. Analysing customer sentiments through customer reviews

I'll apply natural language processing (NLP) to determine how customer reviews can be used to inform marketing campaigns.

Import the necessary libraries, and prepare the data for sentiment analysis: Sense-check the DataFrame.

```
# Import all the necessary packages.
import pandas as pd
import numpy as np
import nltk
import os
import matplotlib.pyplot as plt
import seaborn as sns


!pip3 install nltk
nltk.download ('punkt')
nltk.download ('stopwords')
from nltk.tokenize import sent_tokenize

from wordcloud import WordCloud
from nltk.tokenize import word_tokenize
from nltk.probability import FreqDist
from nltk.corpus import stopwords
from textblob import TextBlob
from scipy.stats import norm

# Import Counter.
from collections import Counter

import warnings
warnings.filterwarnings('ignore')
```

```
Requirement already satisfied: nltk in /Users/alonsoroblescristina/anaconda3/lib/python3.10/site-packages (3.7)
Requirement already satisfied: tqdm in /Users/alonsoroblescristina/anaconda3/lib/python3.10/site-packages (from nltk)
(4.64.1)
Requirement already satisfied: joblib in /Users/alonsoroblescristina/anaconda3/lib/python3.10/site-packages (from nlt
k) (1.1.1)
Requirement already satisfied: click in /Users/alonsoroblescristina/anaconda3/lib/python3.10/site-packages (from nlt
k) (8.0.4)
Requirement already satisfied: regex>=2021.8.3 in /Users/alonsoroblescristina/anaconda3/lib/python3.10/site-packages
(from nltk) (2022.7.9)
[nltk_data] Downloading package punkt to
[nltk_data]     /Users/alonsoroblescristina/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/alonsoroblescristina/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
# Load the data set as df3.
df3 = pd.read_csv('clean_reviews.csv')

# View the DataFrame.
df3.head()
```

	gender	age	remuneration	spending_score	loyalty_points	education	product	review	summary
0	Male	18	12.30	39	210	graduate	453	When it comes to a DM's screen, the space on t...	The fact that 50% of this space is wasted on a...
1	Male	23	12.30	81	524	graduate	466	An Open Letter to GaleForce9*:\n\nYour unpaint...	Another worthless Dungeon Master's screen from...
2	Female	22	13.12	6	40	graduate	254	Nice art, nice printing. Why two panels are f...	pretty, but also pretty useless
3	Female	25	13.12	77	562	graduate	263	Amazing buy! Bought it as a gift for our new d...	Five Stars
4	Female	33	13.94	40	366	graduate	291	As my review of GF9's previous screens these w...	Money trap

```
# Explore data set.
# Determine the metadata of the 'reviews' DataFrame.
print(df3.columns)
print(df3.shape)
print(df3.dtypes)
df3.info()
```

```
Index(['gender', 'age', 'remuneration', 'spending_score', 'loyalty_points',
       'education', 'product', 'review', 'summary'],
      dtype='object')
(2000, 9)
gender             object
age                 int64
remuneration      float64
spending_score      int64
loyalty_points      int64
education          object
product             int64
review             object
summary            object
dtype: object
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   gender          2000 non-null   object
 1   age             2000 non-null   int64
 2   remuneration    2000 non-null   float64
 3   spending_score  2000 non-null   int64
 4   loyalty_points  2000 non-null   int64
 5   education       2000 non-null   object
 6   product         2000 non-null   int64
 7   review          2000 non-null   object
 8   summary         2000 non-null   object
dtypes: float64(1), int64(4), object(4)
memory usage: 140.8+ KB
```

```
# Find duplicates based on all columns.
df3.duplicated()
```

```
0       False
1       False
2       False
3       False
4       False
        ...
1995    False
1996    False
1997    False
1998    False
1999    False
Length: 2000, dtype: bool
```

There are not duplicates in the DataFrame.

I will only retain the review and summary columns and determine whether there are any missing values.

```
# Keep necessary columns. Drop unnecessary columns.
df3.drop(df3.columns[[0, 1, 2, 3, 4, 5, 6]], axis=1, inplace=True)

# View DataFrame.
df3.head()
```

	review	summary
0	When it comes to a DM's screen, the space on t...	The fact that 50% of this space is wasted on a...
1	An Open Letter to GaleForce9*:\n\nYour unpaint...	Another worthless Dungeon Master's screen from...
2	Nice art, nice printing. Why two panels are f...	pretty, but also pretty useless
3	Amazing buy! Bought it as a gift for our new d...	Five Stars
4	As my review of GF9's previous screens these w...	Money trap

```
# Determine if there are any missing values.
df3_na = df3[df3.isna().any(axis=1)]
df3_na.shape
```

```
(0, 2)
```

```
# Any null values?
df3.isnull().sum()
```

```
review     0
summary    0
dtype: int64
```

Prepare the data for NLP:

Change the data to lower case, and join the elements in each column (review and summary).

Replace punctuation in each of the columns (review and summary).

2a) Change to lower case and join the elements in each of the columns respectively (review and summary)

In [9]:
```
# Review: Change all to lower case and join with a space.
df3['review'] = df3['review'].apply(lambda x: " ".join(x.lower() for x in x.split()))

# Preview the result.
df3['review'].head()
```

Out[9]:
```
0    when it comes to a dm's screen, the space on t...
1    an open letter to galeforce9*: your unpainted ...
2    nice art, nice printing. why two panels are fi...
3    amazing buy! bought it as a gift for our new d...
4    as my review of gf9's previous screens these w...
Name: review, dtype: object
```

In [10]:
```
# Summary: Change all to lower case and join with a space.
df3['summary'] = df3['summary'].apply(lambda x: " ".join(x.lower() for x in x.split()))

# Preview the result.
df3['summary'].head()
```

Out[10]:
```
0    the fact that 50% of this space is wasted on a...
1    another worthless dungeon master's screen from...
2                      pretty, but also pretty useless
3                                           five stars
4                                           money trap
Name: summary, dtype: object
```

2b) Replace punctuation in each of the columns respectively (review and summary)

In [11]:
```
# Replace all the punctuations in review column.
df3['review'] = df3['review'].str.replace('[^\w\s]','')

# View output.
df3['review'].head()
```

Out[11]:
```
0    when it comes to a dms screen the space on the...
1    an open letter to galeforce9 your unpainted mi...
2    nice art nice printing why two panels are fill...
3    amazing buy bought it as a gift for our new dm...
4    as my review of gf9s previous screens these we...
Name: review, dtype: object
```

In [12]:
```
# Replace all the puncuations in summary column.
df3['summary'] = df3['summary'].str.replace('[^\w\s]','')

# View output.
df3['summary'].head()
```

Out[12]:
```
0    the fact that 50 of this space is wasted on ar...
1    another worthless dungeon masters screen from ...
2                       pretty but also pretty useless
3                                           five stars
4                                           money trap
Name: summary, dtype: object
```

Drop duplicates in both columns (review and summary).

2c) Drop duplicates in both columns

In [13]:
```
# # Check the number of duplicate values in both columns.
df3.duplicated().sum()
```

Out[13]: 39

In [14]:
```
# Drop duplicates in both columns.
df3 = df3.drop_duplicates()

# View DataFrame.
df3.head()
```

Out[14]:

	review	summary
0	when it comes to a dms screen the space on the...	the fact that 50 of this space is wasted on ar...
1	an open letter to galeforce9 your unpainted mi...	another worthless dungeon masters screen from ...
2	nice art nice printing why two panels are fill...	pretty but also pretty useless
3	amazing buy bought it as a gift for our new dm...	five stars
4	as my review of gf9s previous screens these we...	money trap

In [15]:
```
# # Check the number of duplicate values in both columns.
df3.duplicated().sum()
```

Out[15]: 0

In [16]:
```
# View the shape of the data.
df3.shape
```

Out[16]: (1961, 2)

Tokenise and create word clouds for the respective columns (separately): Create a copy of the DataFrame.

3. Tokenise and create wordclouds

In [17]:
```
# Create new DataFrame (copy_df3) DataFrame.
# Create a CSV file as output.
df3.to_csv ('/Users/alonsoroblescristina/Desktop/COURSE 3/LSE_DA301_assignment_files/copy_df3.csv')
df3.to_csv('copy_df3.csv', index=False)
```

In [18]:
```
# Load the copy_df3.
copy_df3 = pd.read_csv('copy_df3.csv')

# View the DataFrame.
copy_df3.head()
```

Out[18]:

	review	summary
0	when it comes to a dms screen the space on the...	the fact that 50 of this space is wasted on ar...
1	an open letter to galeforce9 your unpainted mi...	another worthless dungeon masters screen from ...
2	nice art nice printing why two panels are fill...	pretty but also pretty useless
3	amazing buy bought it as a gift for our new dm...	five stars
4	as my review of gf9s previous screens these we...	money trap

In [19]:
```
# Apply tokenisation to 'review' column.
copy_df3['review'] = copy_df3['review'].apply(word_tokenize)

# View the review column.
copy_df3['review'].head()
```

Out[19]:
```
0    [when, it, comes, to, a, dms, screen, the, spa...
1    [an, open, letter, to, galeforce9, your, unpai...
2    [nice, art, nice, printing, why, two, panels, ...
3    [amazing, buy, bought, it, as, a, gift, for, o...
4    [as, my, review, of, gf9s, previous, screens, ...
Name: review, dtype: object
```

In [20]:
```
# Apply tokenisation to 'summary' column.
copy_df3['summary'] = copy_df3['summary'].apply(word_tokenize)

# View the summary column.
copy_df3['summary'].head()
```

Out[20]:
```
0    [the, fact, that, 50, of, this, space, is, was...
1    [another, worthless, dungeon, masters, screen,...
2                 [pretty, but, also, pretty, useless]
3                                       [five, stars]
4                                       [money, trap]
Name: summary, dtype: object
```

In [21]:
```
# View the DataFrame.
copy_df3.head()
```

Out[21]:

	review	summary
0	[when, it, comes, to, a, dms, screen, the, spa...	[the, fact, that, 50, of, this, space, is, was...
1	[an, open, letter, to, galeforce9, your, unpai...	[another, worthless, dungeon, masters, screen,...
2	[nice, art, nice, printing, why, two, panels, ...	[pretty, but, also, pretty, useless]
3	[amazing, buy, bought, it, as, a, gift, for, o...	[five, stars]
4	[as, my, review, of, gf9s, previous, screens, ...	[money, trap]

```
In [22]: # Review: Create a word cloud.
         # Set the colour palette.
         sns.set(color_codes=True)

         # Concatenate all the text from the 'text' column
         all_review = ' '.join(copy_df3['review'].astype(str))


         # Create a WordCloud object.
         word_cloud = WordCloud(width = 1600, height = 900,
                         background_color ='white',
                         colormap = 'plasma',
                         stopwords = 'none',
                         min_font_size = 10).generate(all_review)
```

```
In [23]: # Review: Plot the WordCloud image.
         plt.figure(figsize=(10, 6))
         plt.imshow(word_cloud, interpolation='bilinear')
         plt.axis('off')
         plt.show()
```

```python
# Summary: Create a word cloud.
# Set the colour palette.
sns.set(color_codes=True)

# Concatenate all the text from the 'text' column
all_summary = ' '.join(copy_df3['summary'].astype(str))

# Create a WordCloud object.
word_cloud = WordCloud(width = 1600, height = 900,
                       background_color ='white',
                       colormap = 'plasma',
                       stopwords = 'none',
                       min_font_size = 10).generate(all_summary)
```

```python
# Summary: Plot the WordCloud image.
plt.figure(figsize=(10, 6))
plt.imshow(word_cloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```

Determine the frequency distribution and polarity.

4a) Create frequency distribution

In [26]:
```
# Determine the frequency distribution.
# Concatenate all the text from the 'review' column.
all_reviews = ' '.join(copy_df3['review'].astype(str))

# Tokenize the text data.
tokens = nltk.word_tokenize(all_reviews)

# Create a frequency distribution using nltk FreqDist.
freq_dist = FreqDist(tokens)

# Display the frequency distribution.
print(freq_dist)

# Display the 15 most common words
print(freq_dist.most_common(15))
```

```
<FreqDist with 7524 samples and 344229 outcomes>
[("'", 118020), (',', 110163), ("'the", 5451), ("'and", 3233), ("'to", 3162), ('a', 3160), ("'of", 2488), ('i', 209
0), ("'it", 2083), ('[', 1961), (']', 1961), ("'is", 1782), ("'this", 1776), ("'game", 1671), ("'for", 1545)]
```

In [27]:
```
# Determine the frequency distribution.
# Concatenate all the text from the 'review' column.
all_summaries = ' '.join(copy_df3['summary'].astype(str))

# Tokenize the text data.
tokens = nltk.word_tokenize(all_summaries)

# Create a frequency distribution using nltk FreqDist.
freq_dist = FreqDist(tokens)

# Display the frequency distribution.
print(freq_dist)

# Display the 15 most common words
print(freq_dist.most_common(15))
```

```
<FreqDist with 1554 samples and 30167 outcomes>
[("'", 9644), (',', 7320), ('[', 1961), (']', 1961), ("'stars", 427), ("'five", 342), ("'game", 319), ("'great", 29
5), ("'the", 261), ('a', 240), ("'for", 232), ("'fun", 218), ("'to", 192), ("'and", 168), ("'it", 150)]
```

Remove alphanumeric characters and stop-words.

4b) Remove alphanumeric characters and stopwords

In [28]:
```
# Import the necessary libraries.
import pandas as pd
import nltk
from nltk.corpus import stopwords
import string
```

In [29]:
```
# Download the NLTK data.
nltk.download('punkt')
nltk.download('stopwords')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     /Users/alonsoroblescristina/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/alonsoroblescristina/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[29]: True

In [30]:
```
# Delete all the alphanumeric characteres and stopwords from the 'review' and 'summay' columns.
# Create a function to clean the text.

def clean_text(text):

    # Combine list elements into a single string
    text = ' '.join(text)

    # Tokenize the text
    tokens = nltk.word_tokenize(text.lower())

    # Remove alphanumeric characters and punctuation
    table = str.maketrans('', '', string.punctuation + string.digits)
    tokens = [token.translate(table) for token in tokens]

    # Remove stopwords
    stop_words = set(stopwords.words('english'))
    tokens = [token for token in tokens if token not in stop_words]

    # Join the cleaned tokens back into a string
    cleaned_text = ' '.join(tokens)
    return cleaned_text

# Apply the clean_text function to the 'review' and 'summary' columns.
copy_df3['cleaned_review'] = copy_df3['review'].apply(clean_text)
copy_df3['cleaned_summary'] = copy_df3['summary'].apply(clean_text)

# Print the output.
copy_df3.head()
```

Out[30]:

	review	summary	cleaned_review	cleaned_summary
0	[when, it, comes, to, a, dms, screen, the, spa...	[the, fact, that, 50, of, this, space, is, was...	comes dms screen space screen absolute premium...	fact space wasted art terribly informative ne...
1	[an, open, letter, to, galeforce9, your, unpai...	[another, worthless, dungeon, masters, screen,...	open letter galeforce unpainted miniatures bad...	another worthless dungeon masters screen galef...
2	[nice, art, nice, printing, why, two, panels, ...	[pretty, but, also, pretty, useless]	nice art nice printing two panels filled gener...	pretty also pretty useless
3	[amazing, buy, bought, it, as, a, gift, for, o...	[five, stars]	amazing buy bought gift new dm perfect	five stars
4	[as, my, review, of, gf9s, previous, screens, ...	[money, trap]	review gfs previous screens completely unneces...	money trap

Create a word cloud without stop-words.

4c) Create wordcloud without stopwords

In [31]:
```
# Create a wordcloud without stop words for 'cleaned_review' column.
# Set the colour palette.
sns.set(color_codes=True)

# Concatenate all the text from the 'text' column
all_review = ' '.join(copy_df3['cleaned_review'].astype(str))

# Create a WordCloud object.
word_cloud = WordCloud(width = 1600, height = 900,
                  background_color ='white',
                  colormap = 'plasma',
                  stopwords = 'none',
                  min_font_size = 10).generate(all_review)
```

In [32]:
```
# Plot the wordcloud image for 'cleaned_review' column.
plt.figure(figsize=(10, 6))
plt.imshow(word_cloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```

```
[ ]: # Create a wordcloud without stop words for 'cleaned_summary' column.
     # Set the colour palette.
     sns.set(color_codes=True)

     # Concatenate all the text from the 'text' column
     all_summary = ' '.join(copy_df3['cleaned_summary'].astype(str))

     # Create a WordCloud object.
     word_cloud = WordCloud(width = 1600, height = 900,
                     background_color ='white',
                     colormap = 'plasma',
                     stopwords = 'none',
                     min_font_size = 10).generate(all_summary)
```

```
[ ]: # Plot the wordcloud image for 'cleaned_summary' column.
     plt.figure(figsize=(10, 6))
     plt.imshow(word_cloud, interpolation='bilinear')
     plt.axis('off')
     plt.show()
```

Identify the 15 most common words used, and calculate their sentiment polarity.

4d) Identify 15 most common words and polarity

In [35]:
```
# Determine the 15 most common words for 'cleaned_review' column.
# Concatenate all the text from the 'cleaned_review' column.
all_reviews = ' '.join(copy_df3['cleaned_review'].astype(str))

# Tokenize the text data.
tokens = nltk.word_tokenize(all_reviews)

# Create a frequency distribution using nltk FreqDist.
freq_dist = FreqDist(tokens)

# Display the frequency distribution.
print(freq_dist)

# Display the 15 most common words
print(freq_dist.most_common(15))
```

```
<FreqDist with 7173 samples and 56556 outcomes>
[('game', 1671), ('great', 580), ('fun', 552), ('one', 530), ('play', 502), ('like', 414), ('love', 323), ('really',
319), ('get', 319), ('cards', 301), ('tiles', 297), ('time', 291), ('good', 289), ('would', 280), ('book', 273)]
```

In [36]:
```
# Generate a DataFrame from Counter.
counts = pd.DataFrame(Counter(tokens).most_common(15),
                      columns=['Review_Words', 'Frequency']).set_index('Review_Words')

# Preview data.
counts
```

Out[36]:

Review_Words	Frequency
game	1671
great	580
fun	552
one	530
play	502
like	414
love	323
really	319
get	319
cards	301
tiles	297
time	291
good	289
would	280
book	273

```
# Determine the 15 most common words for 'cleaned_summary' column.
# Concatenate all the text from the 'cleaned_summary' column.
all_summaries = ' '.join(copy_df3['cleaned_summary'].astype(str))

# Tokenize the text data.
tokens = nltk.word_tokenize(all_summaries)

# Create a frequency distribution using nltk FreqDist.
freq_dist = FreqDist(tokens)

# Display the frequency distribution.
print(freq_dist)

# Display the 15 most common words
print(freq_dist.most_common(15))
```

```
<FreqDist with 1414 samples and 6083 outcomes>
[('stars', 427), ('five', 342), ('game', 319), ('great', 295), ('fun', 218), ('love', 93), ('good', 92), ('four', 5
8), ('like', 54), ('expansion', 52), ('kids', 50), ('cute', 45), ('book', 43), ('one', 38), ('awesome', 36)]
```

```
# Generate a DataFrame from Counter.
counts = pd.DataFrame(Counter(tokens).most_common(15),
                      columns=['Summary_Words', 'Frequency']).set_index('Summary_Words')

# Preview data.
counts
```

Summary_Words	Frequency
stars	427
five	342
game	319
great	295
fun	218
love	93
good	92
four	58
like	54
expansion	52
kids	50
cute	45
book	43
one	38
awesome	36

Review the sentiment polarity.

a. Plot histograms of polarity (use 15 bins) for both columns.
b. Review the sentiment scores for the respective columns.

In [42]:
```
# Provided function. Define a function to calculate polarity.
def generate_polarity(comment):
    '''Extract polarity score (-1 to +1) for each comment'''
    blob = TextBlob(comment)
    return blob.sentiment.polarity
```

In [43]:
```
# Apply the get_polarity function to calculate polarity.
# Calculate polarity for each cleaned_review.
copy_df3['review_polarity'] = copy_df3['cleaned_review'].apply(generate_polarity)
copy_df3.head(15)
```

Out[43]:

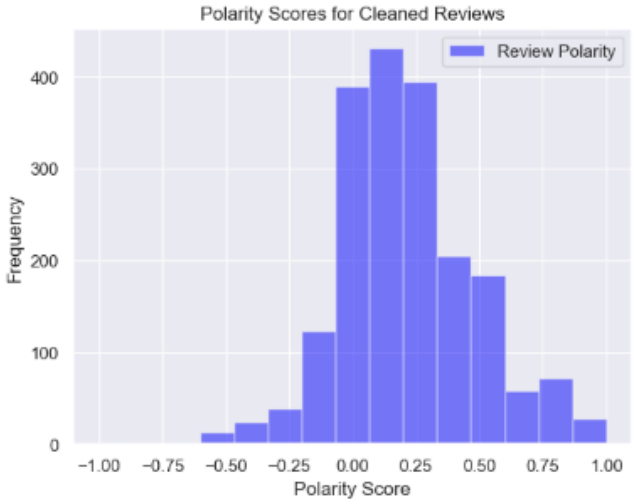
	review	summary	cleaned_review	cleaned_summary	polarity	review_polarity
0	[when, it, comes, to, a, dms, screen, the, spa...	[the, fact, that, 50, of, this, space, is, was...	comes dms screen space screen absolute premium...	fact space wasted art terribly informative ne...	-0.6000	-0.242857
1	[an, open, letter, to, galeforce9, your, unpai...	[another, worthless, dungeon, masters, screen,...	open letter galeforce unpainted miniatures bad...	another worthless dungeon masters screen galef...	-0.8000	-0.019468
2	[nice, art, nice, printing, why, two, panels, ...	[pretty, but, also, pretty, useless]	nice art nice printing two panels filled gener...	pretty also pretty useless	0.0000	0.118243
3	[amazing, buy, bought, it, as, a, gift, for, o...	[five, stars]	amazing buy bought gift new dm perfect	five stars	0.0000	0.578788
4	[as, my, review, of, gf9s, previous, screens, ...	[money, trap]	review gfs previous screens completely unneces...	money trap	0.0000	-0.316667
5	[grandson, loves]	[five, stars]	grandson loves	five stars	0.0000	0.000000
6	[i, have, bought, many, gm, screens, over, the...	[best, gm, screen, ever]	bought many gm screens years one best ever see...	best gm screen ever	1.0000	0.380000
7	[came, in, perfect, condition]	[five, stars]	came perfect condition	five stars	0.0000	1.000000
8	[could, be, better, but, its, still, great, i,...	[great, but, could, be, even, better]	could better still great love things dms side ...	great could even better	0.6500	0.372222
9	[my, review, will, mirror, others, in, that, t...	[another, missed, opportunity, not, a, value, ...	review mirror others kind misses mark lost opp...	another missed opportunity value add product line	0.0000	0.127761
10	[works, very, well]	[five, stars]	works well	five stars	0.0000	0.000000
11	[cant, wait, to, use, it]	[love, the, map]	cant wait use	love map	0.5000	0.000000
12	[this, is, a, campaign, specific, dm, screen, ...	[not, a, general, dm, screen]	campaign specific dm screen meant work storm k...	general dm screen	0.0500	0.137500
13	[this, is, one, of, the, worst, games, i, have...	[very, weak, game]	one worst games ever played basically come pun...	weak game	-0.3875	-0.060000
14	[it, sounded, like, a, really, amazing, concep...	[fell, completely, flat]	sounded like really amazing concept tried play...	fell completely flat	-0.0250	0.038500

In [44]:
```
# Calculate polarity for each cleaned_summary.
copy_df3['summary_polarity'] = copy_df3['cleaned_summary'].apply(generate_polarity)
copy_df3.head(15)
```

Out[44]:

	review	summary	cleaned_review	cleaned_summary	polarity	review_polarity	summary_polarity
0	[when, it, comes, to, a, dms, screen, the, spa...	[the, fact, that, 50, of, this, space, is, was...	comes dms screen space screen absolute premium...	fact space wasted art terribly informative ne...	-0.6000	-0.242857	-0.6000
1	[an, open, letter, to, galeforce9, your, unpai...	[another, worthless, dungeon, masters, screen,...	open letter galeforce unpainted miniatures bad...	another worthless dungeon masters screen galef...	-0.8000	-0.019468	-0.8000
2	[nice, art, nice, printing, why, two, panels, ...	[pretty, but, also, pretty, useless]	nice art nice printing two panels filled gener...	pretty also pretty useless	0.0000	0.118243	0.0000
3	[amazing, buy, bought, it, as, a, gift, for, o...	[five, stars]	amazing buy bought gift new dm perfect	five stars	0.0000	0.578788	0.0000
4	[as, my, review, of, gf9s, previous, screens, ...	[money, trap]	review gfs previous screens completely unneces...	money trap	0.0000	-0.316667	0.0000
5	[grandson, loves]	[five, stars]	grandson loves	five stars	0.0000	0.000000	0.0000
6	[i, have, bought, many, gm, screens, over, the...	[best, gm, screen, ever]	bought many gm screens years one best ever see...	best gm screen ever	1.0000	0.380000	1.0000

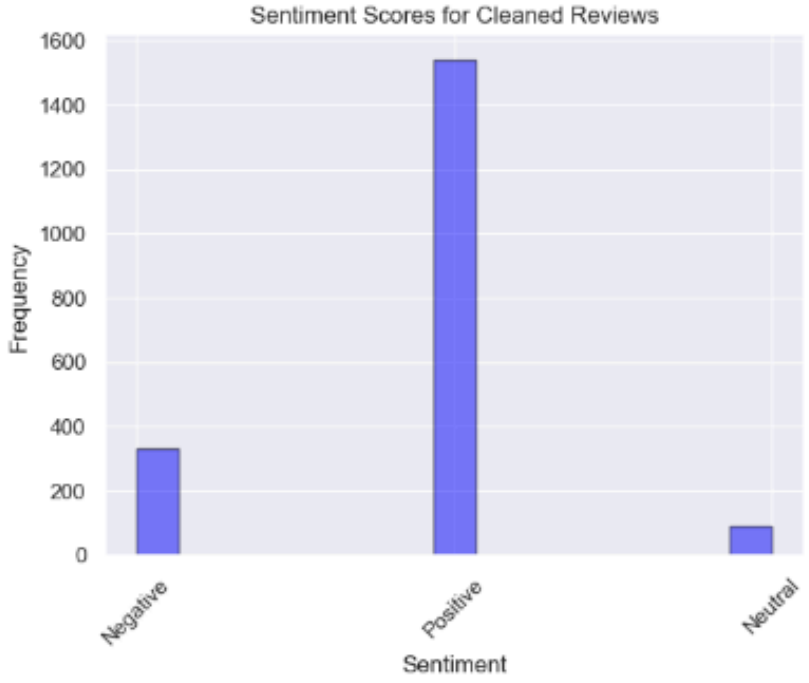
In [45]:
```
# Plot histograms of polarity scores for 'cleaned_review'
plt.hist(copy_df3['review_polarity'], bins=15, alpha=0.5, color='blue', label='Review Polarity')
plt.xlabel('Polarity Score')
plt.ylabel('Frequency')
plt.title('Polarity Scores for Cleaned Reviews')
plt.legend()
plt.show()
plt.figure(figsize=(8, 6))
```



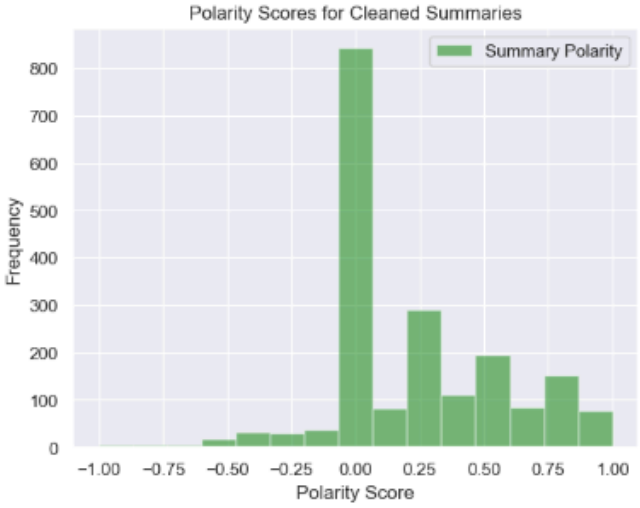

```
# Convert polarity scores to sentiment scores

# Convert polarity scores to sentiment scores
copy_df3['sentiment'] = copy_df3['review_polarity'].apply(lambda score:
                                                          'Positive' if score > 0 else 'Neutral'
                                                          if score == 0 else 'Negative')
copy_df3.head(15)
```

	review	summary	cleaned_review	cleaned_summary	polarity	review_polarity	summary_polarity	sentiment
0	[when, it, comes, to, a, dms, screen, the, spa...	[the, fact, that, 50, of, this, space, is, was...	comes dms screen space screen absolute premium...	fact space wasted art terribly informative ne...	-0.6000	-0.242857	-0.6000	Negative
1	[an, open, letter, to, galeforce9, your, unpai...	[another, worthless, dungeon, masters, screen,...	open letter galeforce unpainted miniatures bad...	another worthless dungeon masters screen galef...	-0.8000	-0.019468	-0.8000	Negative
2	[nice, art, nice, printing, why, two, panels, ...	[pretty, but, also, pretty, useless]	nice art nice printing two panels filled gener...	pretty also pretty useless	0.0000	0.118243	0.0000	Positive
3	[amazing, buy, bought, it, as, a, gift, for, o...	[five, stars]	amazing buy bought gift new dm perfect	five stars	0.0000	0.578788	0.0000	Positive
4	[as, my, review, of, gf9s, previous, screens, ...	[money, trap]	review gfs previous screens completely unneces...	money trap	0.0000	-0.316667	0.0000	Negative
5	[grandson, loves]	[five, stars]	grandson loves	five stars	0.0000	0.000000	0.0000	Neutral
6	[i, have, bought, many, gm, screens, over, the...	[best, gm, screen, ever]	bought many gm screens years one best ever see...	best gm screen ever	1.0000	0.380000	1.0000	Positive
7	[came, in, perfect, condition]	[five, stars]	came perfect condition	five stars	0.0000	1.000000	0.0000	Positive
8	[could, be, better, but, its, still, great, i,...	[great, but, could, be, even, better]	could better still great love things dms side ...	great could even better	0.6500	0.372222	0.6500	Positive
9	[my, review, will, mirror, others, in, that, t...	[another, missed, opportunity, not, a, value, ...	review mirror others kind misses mark lost opp...	another missed opportunity value add product line	0.0000	0.127761	0.0000	Positive
10	[works, very, well]	[five, stars]	works well	five stars	0.0000	0.000000	0.0000	Neutral
11	[cant, wait, to, use, it]	[love, the, map]	cant wait use	love map	0.5000	0.000000	0.5000	Neutral
12	[this, is, a, campaign, specific, dm, screen, ...	[not, a, general, dm, screen]	campaign specific dm screen meant work storm k...	general dm screen	0.0500	0.137500	0.0500	Positive
13	[this, is, one, of, the, worst, games, i, have...	[very, weak, game]	one worst games ever played basically come pun...	weak game	-0.3875	-0.060000	-0.3875	Negative
14	[it, sounded, like, a, really, amazing, concep...	[fell, completely, flat]	sounded like really amazing concept tried play...	fell completely flat	-0.0250	0.038500	-0.0250	Positive

```
# Plot a histogram of sentiment scores.
plt.hist(copy_df3['sentiment'], bins=15, alpha=0.5, color='blue', edgecolor='black')
plt.xlabel('Sentiment')
plt.ylabel('Frequency')
plt.title('Sentiment Scores for Cleaned Reviews')
plt.xticks(rotation=45)
plt.show()
```




```
# Summary: Create a histogram plot with bins = 15.
# Histogram of polarity.
# Plot histograms of polarity scores for 'cleaned_summary'
plt.hist(copy_df3['summary_polarity'], bins=15, alpha=0.5, color='green', label='Summary Polarity')
plt.xlabel('Polarity Score')
plt.ylabel('Frequency')
plt.title('Polarity Scores for Cleaned Summaries')
plt.legend()
plt.show()
```

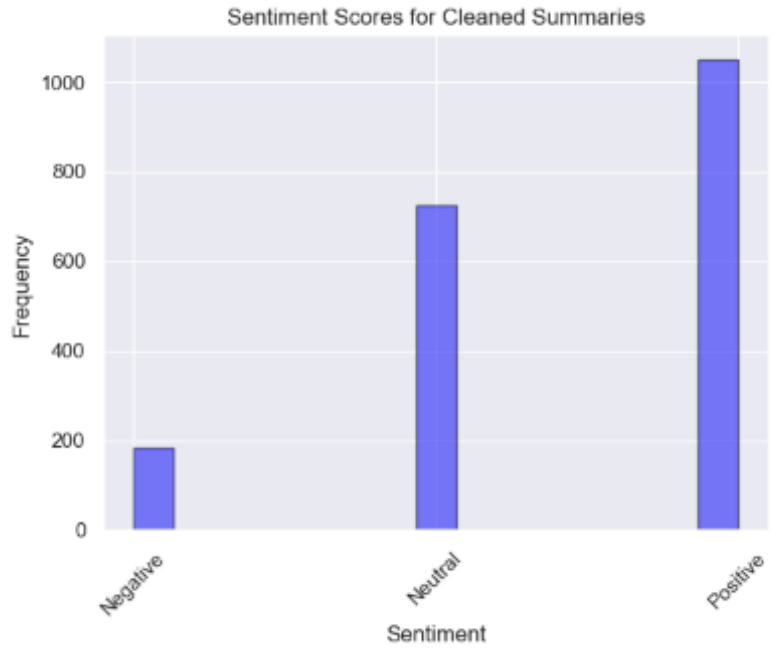


In [49]:
```
# Convert polarity scores to sentiment scores
copy_df3['sentiment'] = copy_df3['summary_polarity'].apply(lambda score:
                                                           'Positive' if score > 0 else 'Neutral'
                                                           if score == 0 else 'Negative')
copy_df3.head(15)
```

Out[49]:

	review	summary	cleaned_review	cleaned_summary	polarity	review_polarity	summary_polarity	sentiment
0	[when, it, comes, to, a, dms, screen, the, spa...	[the, fact, that, 50, of, this, space, is, was...	comes dms screen space screen absolute premium...	fact space wasted art terribly informative ne...	-0.6000	-0.242857	-0.6000	Negative
1	[an, open, letter, to, galeforce9, your, unpai...	[another, worthless, dungeon, masters, screen,...	open letter galeforce unpainted miniatures bad...	another worthless dungeon masters screen galef...	-0.8000	-0.019468	-0.8000	Negative
2	[nice, art, nice, printing, why, two, panels, ...	[pretty, but, also, pretty, useless]	nice art nice printing two panels filled gener...	pretty also pretty useless	0.0000	0.118243	0.0000	Neutral
3	[amazing, buy, bought, it, as, a, gift, for, o...	[five, stars]	amazing buy bought gift new dm perfect	five stars	0.0000	0.578788	0.0000	Neutral
4	[as, my, review, of, gf9s, previous, screens, ...	[money, trap]	review gfs previous screens completely unneces...	money trap	0.0000	-0.316667	0.0000	Neutral
5	[grandson, loves]	[five, stars]	grandson loves	five stars	0.0000	0.000000	0.0000	Neutral

:
```
# Plot a histogram of sentiment scores.
plt.hist(copy_df3['sentiment'], bins=15, alpha=0.5, color='blue', edgecolor='black')
plt.xlabel('Sentiment')
plt.ylabel('Frequency')
plt.title('Sentiment Scores for Cleaned Summaries')
plt.xticks(rotation=45)
plt.show()
```



Identify and print the top 20 positive reviews and summaries and the top 20 negative reviews and summaries, respectively.

6. Identify top 20 positive and negative reviews and summaries respectively

In [51]:
```
# Sort the DataFrame by polarity scores in descending order.
sorted_copy_df3 = copy_df3.sort_values(by='polarity', ascending=False)

# Top 20 negative reviews.
top_negative_reviews = sorted_copy_df3.tail(20)['cleaned_review'].tolist()

# View output.
top_negative_reviews
```

Out[51]:
```
['im sorry find product boring frank juvenile',
 'comes dms screen space screen absolute premium fact  space wasted art terribly informative needed art well makes co
mpletely useless reason gave  stars  technically speaking least still stand block notes dice rolls drops ball complet
ely',
 'really cheaply produced cardboard playing pieces cost',
 'high hopes game big fan fourth edition dd though like e enjoy im going tell one better ill tell one prefer also enj
oy games modular tiles really went game biased towards enjoying sadly couldnt rescue game exceedingly repetitive play
turns youll get entire experience game offer nothing new exciting game hopelessly shallow based much chance strategy
nonexistent despite modular tiles game offers virtually customization pieces seem wellmade fortunately might supply u
se actual dd since store bought doesnt returns however quality pieces matched game really wanted enjoy game played  w
hole games uncounted number aborted attempts accepting fact simply doesnt work extremely disappointing',
 'wish id watched gameplay videos ashardalon buying others mentioned rpg board game board game particularly bad board
game punishingly difficult little theme begs house rules clean terrible encounter card system tried playing aggressiv
ely quickly overwhelmed critters kill tried playing cautiously party wrecked encounter cards doesnt feel like proper
dungeon crawl youve killed five monsters lost half characters hp youve left sight staircase like combat game set wack
y arena bad stuff happens time seen another way like puzzle game balance risk generally much reward generally little
hopefully survive long enough reach objective way feel like dungeon crawl importantly poor introduction rpg elements
clearly minority wellreviewed game simply wish offer dissenting opinion ashardalon fun play im selling copy im going
try mice mystics instead looks like hoping ashardalon would balance teenfriendly story adventure combat need dm im al
so eager try arcadia quest grant two stars review instead one bits box fantastic excellent miniatures board pieces to
pnotch presentation looks fantastic take box',
 'book  pages size x note card much fun',
 'great game kids love play',
 'game tiles board tile stands made paper using times sustain paper board tiles move board making game messy inconven
ient manage shame done brilliant game',
 'guess look closely information game impression similar acquire used play quality one use looks cheap would paid get
better quality',
 'found card game opposite intended actually kids focusing ways get angry etc instead teaching calm act better really
tested sale better game would calm dragon tried game kids absolutely behavior anger problems began behaving badly get
ting angry second round dont recommend therapist work kids anger issues day long thought might good tool wrong',
 'kids grew peg bench hammer loved bought brand grandson disappointed pegs fit loosely bench even use hammer pound pu
sh hand sometimes fall automatically suggestion make pegs fit little tighter kids learn skills coordination etc pound
ing pegs nice thick little hands snug enough fitting really use toy intended',
 'mom already owned acquire game always commented poorly made thought would get new one christmas quality one much be
tter old one cards player see much hotel cost buy according many tiles one even expected better quality price paid di
dnt even come bag tiles think disappointed',
 'eggs split unusable',
 'thinking puppet doll still worked needed way get animals mouth little difficult little child',
 'open letter galeforce unpainted miniatures bad spell cards great board games meh dm screens however freaking terrib
le im still waiting single screen isnt polluted pointless artwork useful referenceable tables youve created single us
e screen useful running storm kings thunder adventure even despite fact geared adventure path usefulness negligible b
est massive swath inner panel wasted artwork bloated overland map could easily reduced single panel size table nighus
eless short stop making crap dm screens',
 'hated running rpg campaign dealing towns kills momentum becomes  hours haggling magic items helps open story ideas
plot hooks',
 'ive discovered im really new school comes board games except boggle couple chess boards novelty version yahtzee old
est game  years watered usual tastes gameplay consists playing tiles board start expand merge hotel chains buy stocks
trying predict ones grow buy low possible sell huge profits end game player money end wins think game certainly bette
r people accommodate  may many ive never played number  fewer easy run away game players offers competition gaps turn
s dilutes available stock three see tiles available buy huge amounts stock early players time react attempt thwart pl
an prevent big payout ive enjoyed power grid intended playing deeper economic game time certainly isnt much shallow e
xperience although play reasonably quickly may fill shorter time slot players looking quick buy sell stock game',
 'horrible nothing say would give zero stars possible',
 'booo unles patient know measure didnt patience neither daughter boring unless craft person',
 'cute tatoos love pirates however retail price  amazons price double never paid retail price item amazon shocked han
dful tatoos  wouldnt mind charge  item sells anywhere else  criminal find local drugstore save']
```

```
# Top 20 negative summaries.
top_negative_summaries = sorted_copy_df3.tail(20)['cleaned_summary'].tolist()

# View output.
top_negative_summaries
```

```
['disappointing',
 'fact  space wasted art terribly informative needed art',
 'disappointing',
 'disappointing',
 'disappointing',
 'small boring',
 'mad dragon',
 'bad qualityall made paper',
 'bad expecting',
 'promotes anger instead teaching calming methods',
 'disappointed',
 'disappointed',
 'disappointed',
 'disappointed',
 'another worthless dungeon masters screen galeforce',
 'hated running rpg campaign dealing towns',
 'boring',
 'horrible nothing say would give zero stars',
 'boring unless craft person',
 'worst value ive ever seen']
```

```
# Top 20 positive reviews.
# Get the top 20 positive reviews.
top_positive_reviews = sorted_copy_df3.head(20)['cleaned_review'].tolist()

# View output.
top_positive_reviews
```

```
['make game come alive battles go much quickly smoothly nicely designed great way give detail depth battles',
 'excellent expansion lords waterdeep importantly adds sixth player option game',
 'pigeon books elementary school library dont pigeon missing students love pigeon',
 'disclaimer one villain cards came factory defect machine burned rarity far rest best miniatures ive ever cave bears wifes favorite love otyugh insane number orcs duergar extra kobolds devils frickin awesome seriously well balanced even new game features stand even better previous castle ravenloft game',
 'bought doll  year old boy class preschool teacher perfect children love animals separately also book trio last years always favorite hands story preschoolers used one previous school belonged school time bought set',
 'impressed quality puzzle easy fun put together',
 'great resource bhis care coordinators works well kids teens says',
 'opinion best dungeon crawler setup played tore  hours makes fun ride fun kids mine  ',
 'wonderful ball manipulate consists  sided pieces fit together sided ball approximately inches diameter assemblies made',
 'love expand current game new additions completely change half expansion two different expansions',
 'grandson could put wants play',
 'daughter loves little books theyre perfect size keep car diaper bag purse keep hand times stuck waiting doctors office anywhere else',
 'far favorite dd board games fun easy learn even got family play',
 'quick fun easy learn wide age range fast play great family game',
 'great gift wedding party dr themed wedding everyone loved gift reasonably priced much fun',
 'yes quick wonderful accurate',
 'love game playing  years best played  people',
 'loads fun youve played dd boardgames set wrath ashardalon opinion wellrounded among includes gold tokens spend buying treasures adventures balanced set heroes memorable unique abilities without overpowered make game easy relatable setting havent played boardgames dont one standalone game rules board pieces cards characters interchangeable game dungeon crawl turn explore new room dungeon deal things find monster trap encounter anyall fully cooperative need one player dungeon master plays different rules others acts villain even play game solo really wanted ai rules monsters encounter game determined card dont worry making people feel like theyre singled ie game wont break friendships unlike risk monopoly approachable dont ever played game dungeons dragons understand even appreciate hand could actually great way introduce people dd youre inclined rules distilled simplified versions actual tabletop roleplaying game typical round play including setup packup could last anywhere  minutes  hours extreme end',
 'daughter loves stickers awesome seller thank',
 'somuchfun seriously addictive small card set family skeptical first prying away table almost midnight one wants quit playing best stocking stuffer gift ever good sense buy']
```

```
# Top 20 positive summaries.
top_positive_summaries = sorted_copy_df3.head(20)['cleaned_summary'].tolist()

# View output.
top_positive_summaries
```

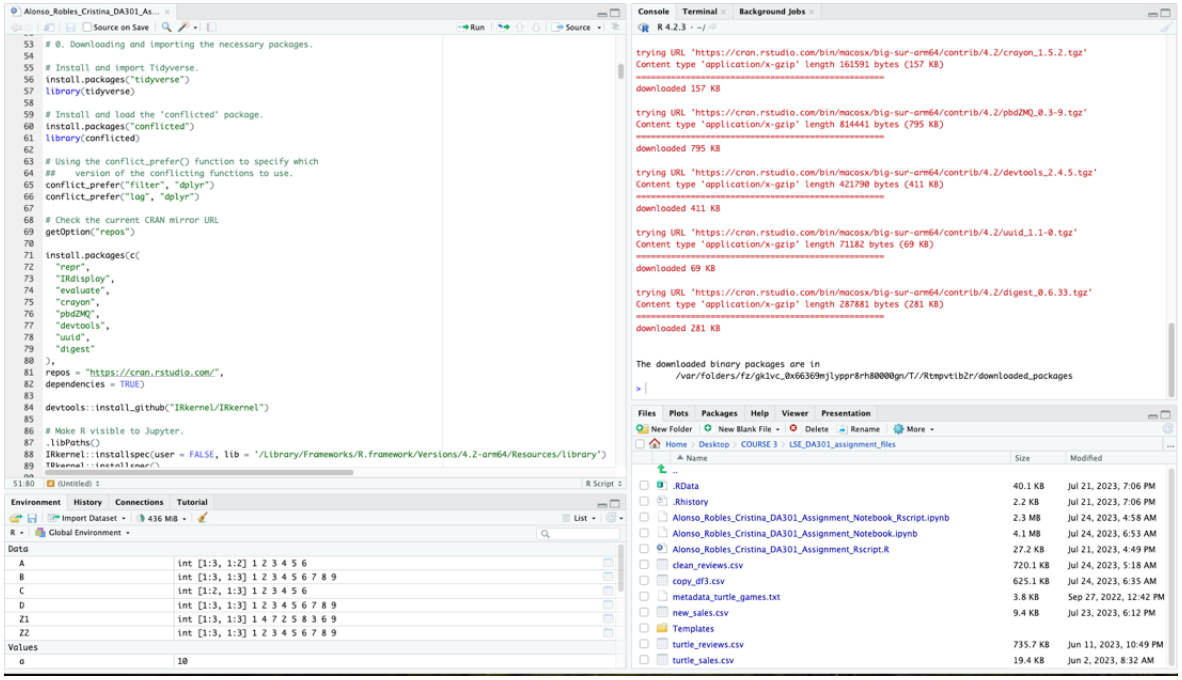
```
['awesome',
 'adds six player option excellent expansion',
 'pigeon perfect addition school library',
 'best boardgame ever',
 'perfect preschoolers',
 'excellent puzzle',
 'perfect',
 'best dungeon crawler',
 'wonderful ball manipulate',
 'awesome expansion',
 'awesome',
 'theyre perfect size keep car diaper',
 'best one series',
 'excellent',
 'perfect gift',
 'wonderful',
 'one best games ever',
 'best among dd boardgames',
 'awesome seller thank',
 'one best games ever']
```

D. Visualising sales data to gather insights using R

I'll explore and prepare the sales data set for analysis on the impact of sales per product.

I started my analysis in RStudio, and later I moved my R script to Jupyter Notebook since I find it to provide a cleaner view of the code and the outputs together.

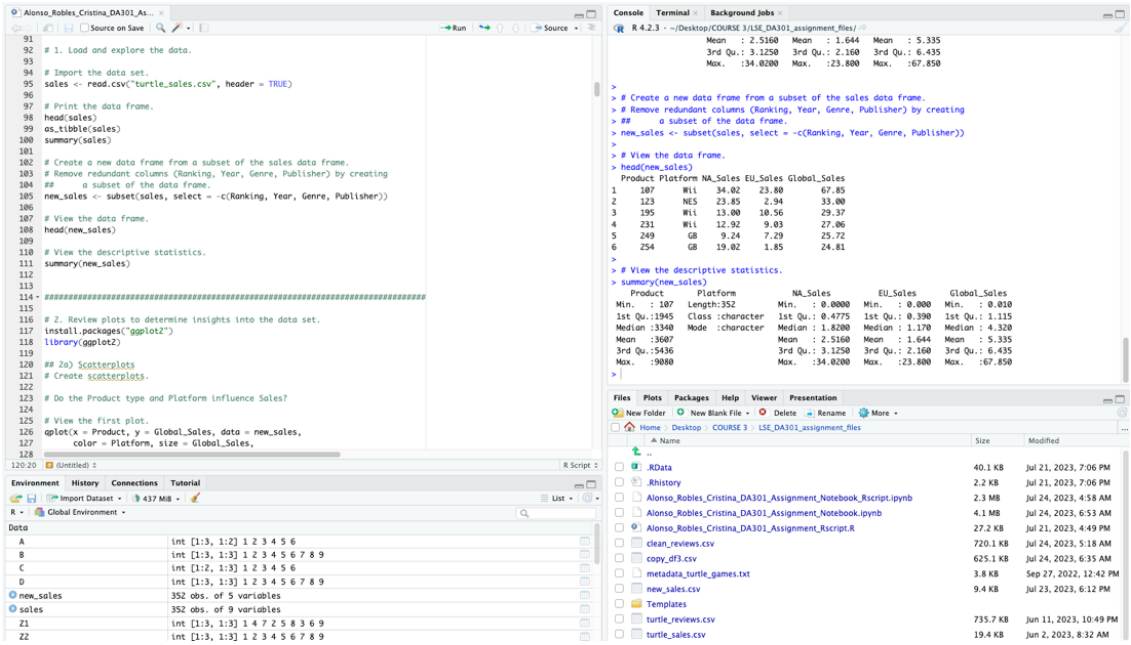
Import all the required libraries for the analysis and view the data.
Load and explore the data.



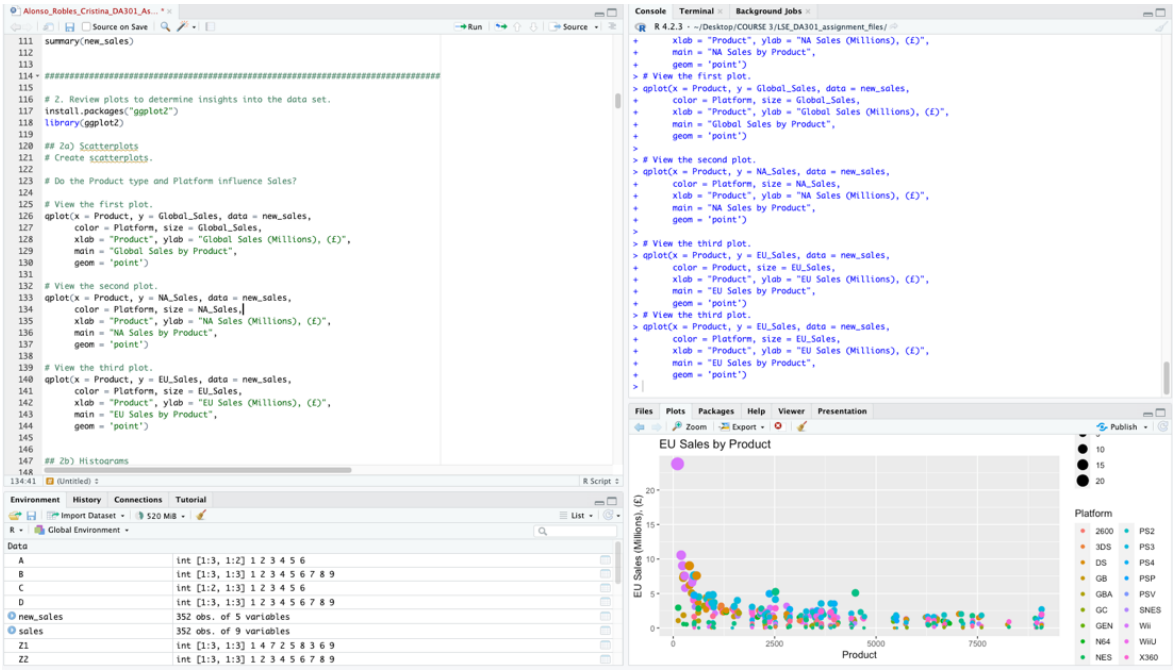
Open a new R script and import the turtle_sales.csv data file.

Remove redundant columns (Ranking, Year, Genre, Publisher) by creating a subset of the data frame.

Create a summary of the new data frame.

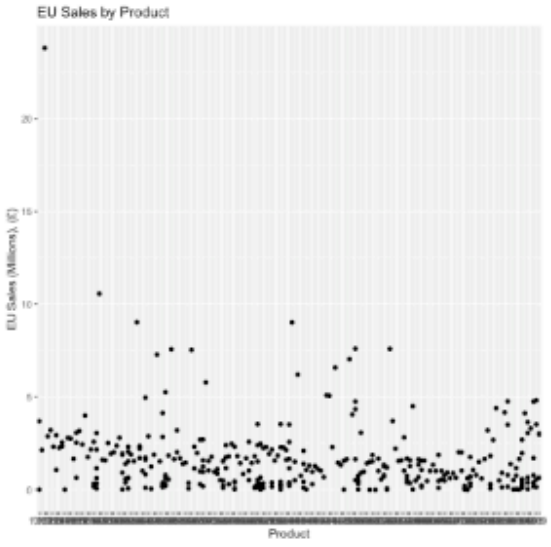


Create plots to review and determine insights into the sales data set.



I will continue with my analysis in R in Jupyter Notebook using the R Kernel.

```
# View the plot for EU Sales.
qplot(x = Product, y = EU_Sales, data = new_sales,
      xlab = "Product", ylab = "EU Sales (Millions), (£)",
      main = "EU Sales by Product",
      geom = 'point')
```



Due to the extense number of product types, it is a bit tricky to identify the outliers and the patterns in the sales by Product.

This is why I decided to print the first 10 rows of the data frame order by Product and Sales, so we can essily see the Sales by Product_id.

```r
# Group data based on Product and determine the sum of Global Sales per Product.
global_sales_summary <- new_sales %>%
  group_by(Product) %>%
  summarise(Global_Sales = sum(Global_Sales))

# View the first 10 rows of the data frame.
first_10_rows <- head(global_sales_summary, 10)
print(first_10_rows)
```

```
# A tibble: 10 × 2
   Product Global_Sales
   <chr>          <dbl>
 1 1012            10.8
 2 1031            10.7
 3 107             67.8
 4 1175            10.1
 5 1183            10.0
 6 1212             9.95
 7 123             37.2
 8 1241             9.76
 9 1307            18.3
10 1459             9.56
```

```r
# Order the data frame in descending order of Global_Sales
global_sales_summary_ordered <- global_sales_summary %>%
  arrange(desc(Global_Sales))

# View the first 10 rows of the ordered data frame.
first_10_rows <- head(global_sales_summary_ordered, 10)
print(first_10_rows)
```

```
# A tibble: 10 × 2
   Product Global_Sales
   <chr>          <dbl>
 1 107             67.8
 2 515             45.9
 3 123             37.2
 4 254             29.4
 5 195             29.4
 6 231             27.1
 7 249             25.7
 8 948             25.4
 9 876             25.3
10 263             24.6
```

```r
# Group data based on Product and determine the sum of NA Sales per Product.
NA_sales_summary <- new_sales %>%
  group_by(Product) %>%
  summarise(NA_Sales = sum(NA_Sales))

# View the first 10 rows of the data frame.
first_10_rows <- head(NA_sales_summary, 10)
print(first_10_rows)
```

```
# A tibble: 10 × 2
   Product NA_Sales
   <chr>      <dbl>
 1 1012        5.73
 2 1031        5.54
 3 107        34.0
 4 1175        2.09
 5 1183        3.89
 6 1212        6.54
 7 123        26.6
 8 1241        3.61
 9 1307        9.84
10 1459        2.47
```

```r
# Order the data frame in descending order of NA_Sales.
NA_sales_summary_ordered <- NA_sales_summary %>%
  arrange(desc(NA_Sales))

# View the first 10 rows of the ordered data frame.
first_10_rows <- head(NA_sales_summary_ordered, 10)
print(first_10_rows)
```

```
# A tibble: 10 × 2
   Product NA_Sales
   <chr>      <dbl>
 1 107         34.0
 2 123         26.6
 3 326         22.1
 4 254         21.5
 5 515         19.2
 6 948         14.4
 7 535         13.1
 8 195         13
 9 231         12.9
10 876         12.8
```

```r
# Group data based on Product and determine the sum of EU Sales per Product.
EU_sales_summary <- new_sales %>%
  group_by(Product) %>%
  summarise(EU_Sales = sum(EU_Sales))

# View the first 10 rows of the data frame.
first_10_rows <- head(EU_sales_summary, 10)
print(first_10_rows)
```

```
# A tibble: 10 × 2
   Product EU_Sales
   <chr>      <dbl>
 1 1012        3.71
 2 1031        2.14
 3 107        23.8
 4 1175        2.89
 5 1183        3.21
 6 1212        2.32
 7 123         4.01
 8 1241        2.27
 9 1307        4.89
10 1459        0.01
```

```r
# Order the data frame in descending order of NA_Sales.
EU_sales_summary_ordered <- EU_sales_summary %>%
  arrange(desc(EU_Sales))

# View the first 10 rows of the ordered data frame.
first_10_rows <- head(EU_sales_summary_ordered, 10)
print(first_10_rows)
```

```
# A tibble: 10 × 2
   Product EU_Sales
   <chr>      <dbl>
 1 107        23.8
 2 515        18.9
 3 195        10.6
 4 3967       10.2
 5 2371        9.26
 6 876         9.25
 7 3645        9.14
 8 979         9.07
 9 231         9.03
10 399         9.02
```

2b) Create histograms.

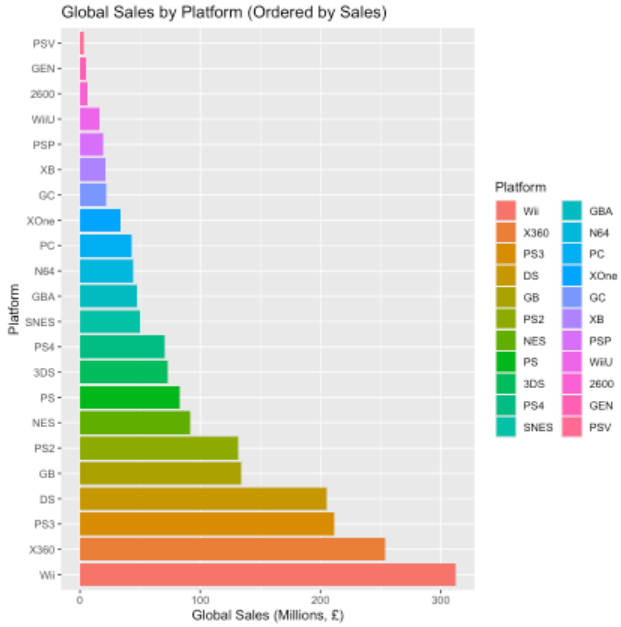
```
# Group data based on the Platform and determine the sum of Global Sales per Platform.
global_sales_summary <- new_sales %>%
  group_by(Platform) %>%
  summarise(Global_Sales = sum(Global_Sales))

# View the first 10 rows of the data frame.
first_10_rows <- head(global_sales_summary, 10)
print(first_10_rows)
```

```
# A tibble: 10 × 2
   Platform Global_Sales
   <chr>           <dbl>
 1 2600             6.4
 2 3DS             73.2
 3 DS             205.
 4 GB             134.
 5 GBA             47.1
 6 GC              21.7
 7 GEN              4.94
 8 N64             44.5
 9 NES             91.4
10 PC              43.1
```

```
# Order the data frame in descending order of Global_Sales
global_sales_summary_ordered <- global_sales_summary %>%
  arrange(desc(Global_Sales))

# View the first 10 rows of the ordered data frame.
first_10_rows <- head(global_sales_summary_ordered, 10)
print(first_10_rows)
```

```
# A tibble: 10 × 2
   Platform Global_Sales
   <chr>           <dbl>
 1 Wii             313.
 2 X360            254.
 3 PS3             212.
 4 DS              205.
 5 GB              134.
 6 PS2             132.
 7 NES              91.4
 8 PS               82.9
 9 3DS              73.2
10 PS4              70.5
```

```r
# Does the Platform type influence Sales?

# Reorder the Platform variable based on Global Sales.
global_sales_summary_ordered$Platform <- reorder(global_sales_summary_ordered$Platform,
                                                 -global_sales_summary_ordered$Global_Sales)

# View the plot "Global Sales by Platform".
ggplot(global_sales_summary_ordered, aes(x = Global_Sales, y = Platform, fill = Platform)) +
  geom_col() +
  labs(x = "Global Sales (Millions, £)",
       y = "Platform",
       title = "Global Sales by Platform (Ordered by Sales)")
```



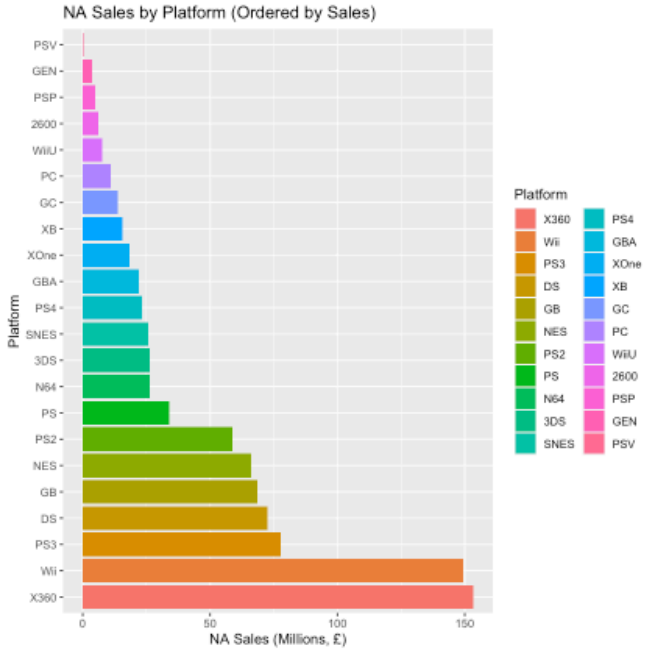
```r
# Group data based on the Platform and determine the sum of NA Sales per Platform.
NA_sales_summary <- new_sales %>%
  group_by(Platform) %>%
  summarise(NA_Sales = sum(NA_Sales))

# View the first 10 rows of the data frame.
first_10_rows <- head(NA_sales_summary, 10)
print(first_10_rows)
```

```
# A tibble: 10 × 2
   Platform NA_Sales
   <chr>       <dbl>
 1 2600         5.97
 2 3DS         26.4
 3 DS          72.6
 4 GB          68.7
 5 GBA         22.0
 6 GC          13.8
 7 GEN          3.67
 8 N64         26.4
 9 NES         66.0
10 PC          11.0
```

```r
# Order the data frame in descending order of NA_Sales.
NA_sales_summary_ordered <- NA_sales_summary %>%
  arrange(desc(NA_Sales))

# View the first 10 rows of the ordered data frame.
first_10_rows <- head(NA_sales_summary_ordered, 10)
print(first_10_rows)
```

```
# A tibble: 10 × 2
   Platform NA_Sales
   <chr>       <dbl>
 1 X360        153.
 2 Wii         150.
 3 PS3          77.8
 4 DS           72.6
 5 GB           68.7
 6 NES          66.0
 7 PS2          58.7
 8 PS           34.0
 9 N64          26.4
10 3DS          26.4
```

```r
# Reorder the Platform variable based on NA Sales.
NA_sales_summary_ordered$Platform <- reorder(NA_sales_summary_ordered$Platform,
                                             -NA_sales_summary_ordered$NA_Sales)

# View the plot "NA Sales by Platform".
ggplot(NA_sales_summary_ordered, aes(x = NA_Sales, y = Platform, fill = Platform)) +
  geom_col() +
  labs(x = "NA Sales (Millions, £)",
       y = "Platform",
       title = "NA Sales by Platform (Ordered by Sales)")
```



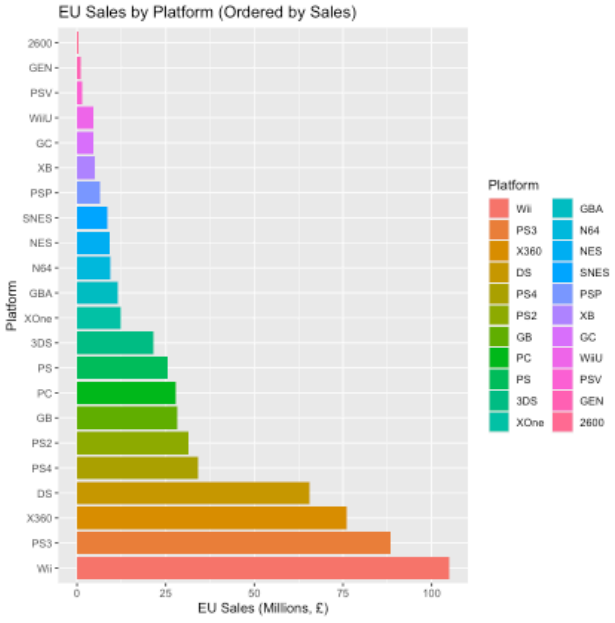
```r
# Group data based on the Platform and determine the sum of EU Sales per Platform.
EU_sales_summary <- new_sales %>%
  group_by(Platform) %>%
  summarise(EU_Sales = sum(EU_Sales))

# View the first 10 rows of the data frame.
first_10_rows <- head(EU_sales_summary, 10)
print(first_10_rows)
```

```
# A tibble: 10 × 2
   Platform EU_Sales
   <chr>       <dbl>
 1 2600         0.37
 2 3DS         21.6
 3 DS          65.6
 4 GB          28.2
 5 GBA         11.6
 6 GC           4.7
 7 GEN          0.98
 8 N64          9.36
 9 NES          9.14
10 PC          27.9
```

```r
# Order the data frame in descending order of EU Sales.
EU_sales_summary_ordered <- EU_sales_summary %>%
  arrange(desc(EU_Sales))

# View the first 10 rows of the ordered data frame.
first_10_rows <- head(EU_sales_summary_ordered, 10)
print(first_10_rows)
```

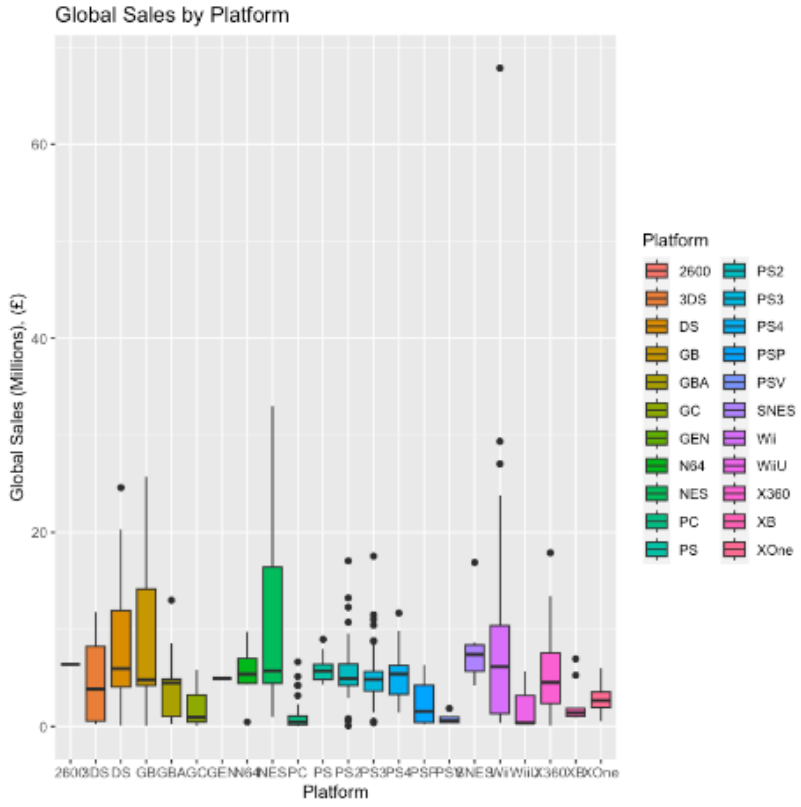
```
# A tibble: 10 × 2
   Platform EU_Sales
   <chr>       <dbl>
 1 Wii         105.
 2 PS3          88.5
 3 X360         76.0
 4 DS           65.6
 5 PS4          34.1
 6 PS2          31.5
 7 GB           28.2
 8 PC           27.9
 9 PS           25.6
10 3DS          21.6
```

```
# Reorder the Platform variable based on EU Sales.
EU_sales_summary_ordered$Platform <- reorder(EU_sales_summary_ordered$Platform,
                                             -EU_sales_summary_ordered$EU_Sales)

# View the plot "EU Sales by Platform".
ggplot(EU_sales_summary_ordered, aes(x = EU_Sales, y = Platform, fill = Platform)) +
  geom_col() +
  labs(x = "EU Sales (Millions, £)",
       y = "Platform",
       title = "EU Sales by Platform (Ordered by Sales)")
```

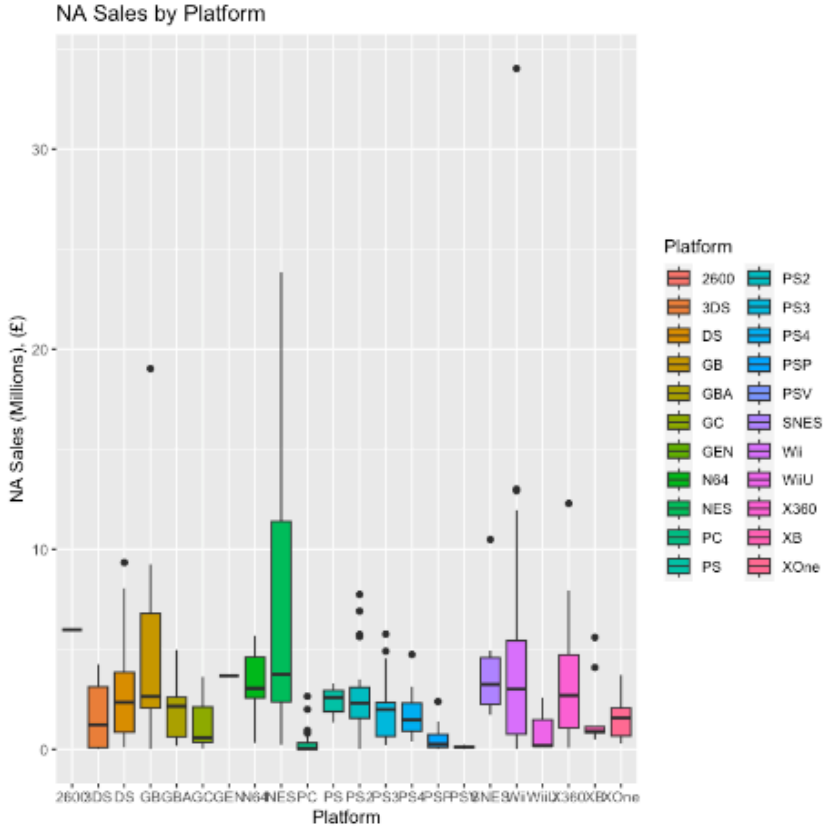


Since we already had a look at the Platforms that influence sales the most by market, it is worth having a look at how each platforms behave compared to the rest of the platforms, in terms of the sales that they produce. I will do this, by looking at the boxplots.

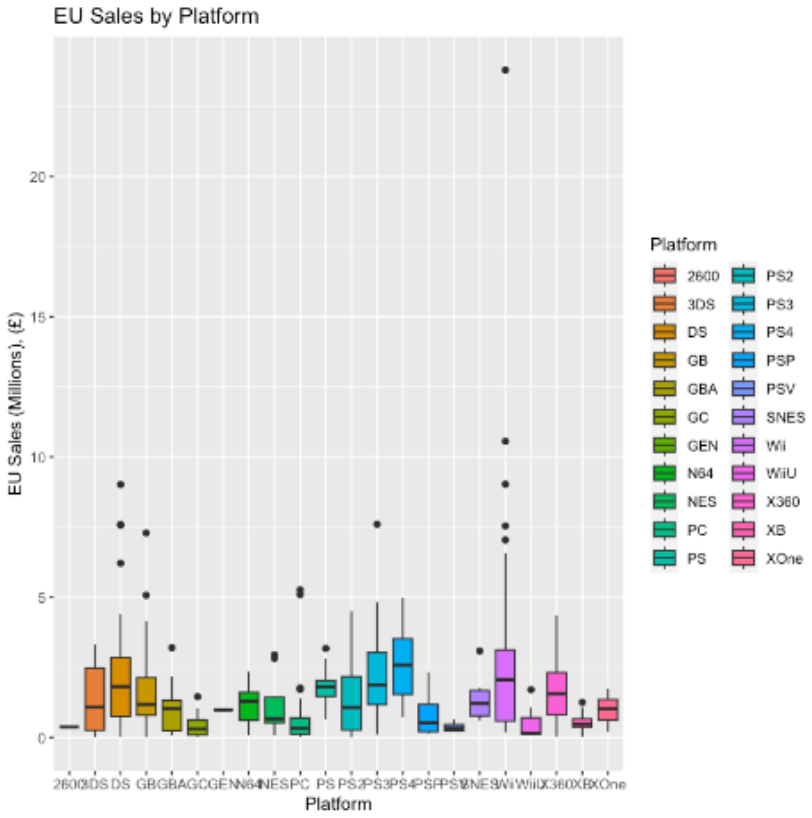
```
# 2c) Create boxplots.

# View the boxplot "Global Sales by Platform".
qplot(x = factor(Platform), y = Global_Sales, data = new_sales,
      xlab = "Platform", ylab = "Global Sales (Millions), (£)",
      main = "Global Sales by Platform",
      geom = "boxplot", fill = Platform)
```



```r
# View the boxplot "NA Sales by Platform".
qplot(x = factor(Platform), y = NA_Sales, data = new_sales,
      xlab = "Platform", ylab = "NA Sales (Millions), (£)",
      main = "NA Sales by Platform",
      geom = "boxplot", fill = Platform)
```



```
: # View the boxplot "EU Sales by Platform".
qplot(x = factor(Platform), y = EU_Sales, data = new_sales,
      xlab = "Platform", ylab = "EU Sales (Millions), (£)",
      main = "EU Sales by Platform",
      geom = "boxplot", fill = Platform)
```



After plotting the data for each of the sales my market, we can clearly see that there are some outliers, reflected on the black dots above each boxplot by platform.

The platforms with the most significant outliers are the ones that influence sales the most.

E. Cleaning, manipulating and visualising sales data using R

I'll perform exploratory data analysis (EDA) techniques to clean and manipulate the sales data so that we can determine how reliable the data is by explaining the normality of the data set based on plots, Skewness, Kurtosis, and a Shapiro-Wilk test.

1. Load and explore the data

]:
```
# 0.Prepare your workstation

# Import the necessary libraries.
# Create insightful summaries of data set.
install.packages("skimr")
library(skimr)

# Create insightful reports of data set.
install.packages("DataExplorer")
library(DataExplorer)
library(tibble)
```

```
The downloaded binary packages are in
        /var/folders/fz/gklvc_0x66369mjlyppr8rh80000gn/T//RtmpFG0plg/downloaded_packages

The downloaded binary packages are in
        /var/folders/fz/gklvc_0x66369mjlyppr8rh80000gn/T//RtmpFG0plg/downloaded_packages
```

]:
```
# Import the new_sales data set created in Week 4.
new_sales <- read.csv("new_sales.csv", header = TRUE)

# Print the data frame.
head(new_sales)
```

A data.frame: 6 × 5

	Product	Platform	NA_Sales	EU_Sales	Global_Sales
	<int>	<chr>	<dbl>	<dbl>	<dbl>
1	107	Wii	34.02	23.80	67.85
2	123	NES	23.85	2.94	33.00
3	195	Wii	13.00	10.56	29.37
4	231	Wii	12.92	9.03	27.06
5	249	GB	9.24	7.29	25.72
6	254	GB	19.02	1.85	24.81

]:
```
# Convert the 'product' column to a character type.
new_sales$Product <- as.character(new_sales$Product)

# Check the data types after conversion.
str(new_sales)
```

```
'data.frame':   352 obs. of  5 variables:
 $ Product     : chr  "107" "123" "195" "231" ...
 $ Platform    : chr  "Wii" "NES" "Wii" "Wii" ...
 $ NA_Sales    : num  34.02 23.85 13 12.92 9.24 ...
 $ EU_Sales    : num  23.8 2.94 10.56 9.03 7.29 ...
 $ Global_Sales: num  67.8 33 29.4 27.1 25.7 ...
```

Create a summary of the new data frame.

```
# Determine if there are missing values.
new_sales[is.na(new_sales)]
sum(is.na(new_sales))
```

```
0
```

```
# Check output: Determine the min, max, and mean values of the all the Sales columns.
# Using summary function.
summary(new_sales$Global_Sales)
summary(new_sales$NA_Sales)
summary(new_sales$EU_Sales)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  0.010   1.115   4.320   5.335   6.435  67.850 

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 0.0000  0.4775  1.8200  2.5160  3.1250 34.0200 

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  0.000   0.390   1.170   1.644   2.160  23.800 
```

```
# View the descriptive statistics.
summary(new_sales)
```

```
   Product            Platform           NA_Sales          EU_Sales      
 Length:352         Length:352         Min.   : 0.0000   Min.   : 0.000  
 Class :character   Class :character   1st Qu.: 0.4775   1st Qu.: 0.390  
 Mode  :character   Mode  :character   Median : 1.8200   Median : 1.170  
                                       Mean   : 2.5160   Mean   : 1.644  
                                       3rd Qu.: 3.1250   3rd Qu.: 2.160  
                                       Max.   :34.0200   Max.   :23.800  

  Global_Sales    
 Min.   : 0.010  
 1st Qu.: 1.115  
 Median : 4.320  
 Mean   : 5.335  
 3rd Qu.: 6.435  
 Max.   :67.850  
```

By looking at the above values, we can see that NA Sales are higher than EU Sales, being the highest sales observed in NA (34M) versus 23.8M of EU Sales. The Global Sales are 67.8M.

2. Determine the impact on sales per product_id

]:
```r
## 2a) Use the group_by and aggregate functions.
# Group data based on Product and determine the sum of Global Sales per Product.
global_sales_summary <- new_sales %>%
  group_by(Product) %>%
  summarise(Total_Global_Sales = sum(Global_Sales))
```

]:
```r
# View the data frame for Total Global Sales.
head(global_sales_summary)
```

A tibble: 6 × 2

Product	Total_Global_Sales
<chr>	<dbl>
1012	10.75
1031	10.69
107	67.85
1175	10.06
1183	10.01
1212	9.95

]:
```r
# Order the data frame in descending order of Total_Global_Sales.
total_global_sales_summary_ordered <- global_sales_summary %>%
  arrange(desc(Total_Global_Sales))

# View the first 10 rows of the ordered data frame.
first_10_rows <- head(total_global_sales_summary_ordered, 10)
print(first_10_rows)
```

```
# A tibble: 10 × 2
   Product Total_Global_Sales
   <chr>                <dbl>
 1 107                   67.8
 2 515                   45.9
 3 123                   37.2
 4 254                   29.4
 5 195                   29.4
 6 231                   27.1
 7 249                   25.7
 8 948                   25.4
 9 876                   25.3
10 263                   24.6
```

```r
# Group data based on Product and determine the sum of NA Sales per Product.
NA_sales_summary <- new_sales %>%
  group_by(Product) %>%
  summarise(Total_NA_Sales = sum(NA_Sales))
```

```r
# View the data frame for Total NA Sales.
head(NA_sales_summary)
```

A tibble: 6 × 2

Product	Total_NA_Sales
<chr>	<dbl>
1012	5.73
1031	5.54
107	34.02
1175	2.09
1183	3.89
1212	6.54

```r
# Order the data frame in descending order of Total_NA_Sales
total_NA_sales_summary_ordered <- NA_sales_summary %>%
  arrange(desc(Total_NA_Sales))

# View the first 10 rows of the ordered data frame.
first_10_rows <- head(total_NA_sales_summary_ordered, 10)
print(first_10_rows)
```

```
# A tibble: 10 × 2
   Product Total_NA_Sales
   <chr>            <dbl>
 1 107               34.0
 2 123               26.6
 3 326               22.1
 4 254               21.5
 5 515               19.2
 6 948               14.4
 7 535               13.1
 8 195               13
 9 231               12.9
10 876               12.8
```

```
# Group data based on Product and determine the sum of EU Sales per Product.
EU_sales_summary <- new_sales %>%
  group_by(Product) %>%
  summarise(Total_EU_Sales = sum(EU_Sales))
```

```
# View the data frame for Total Global Sales.
head(EU_sales_summary)
```

A tibble: 6 × 2

Product	Total_EU_Sales
<chr>	<dbl>
1012	3.71
1031	2.14
107	23.80
1175	2.89
1183	3.21
1212	2.32

```
# Order the data frame in descending order of Total_EU_Sales
total_EU_sales_summary_ordered <- EU_sales_summary %>%
  arrange(desc(Total_EU_Sales))

# View the first 10 rows of the ordered data frame.
first_10_rows <- head(total_EU_sales_summary_ordered, 10)
print(first_10_rows)
```

```
# A tibble: 10 × 2
   Product Total_EU_Sales
   <chr>            <dbl>
 1 107               23.8
 2 515               18.9
 3 195               10.6
 4 3967              10.2
 5 2371               9.26
 6 876                9.25
 7 3645               9.14
 8 979                9.07
 9 231                9.03
10 399                9.02
```

]:
```r
# Create a new data frame 'total_sales' with the new aggregated sales columns.
total_sales <- data.frame(Product = global_sales_summary$Product,
                          Total_NA_Sales = NA_sales_summary$Total_NA_Sales,
                          Total_EU_Sales = EU_sales_summary$Total_EU_Sales,
                          Total_Global_Sales = global_sales_summary$Total_Global_Sales)

# View the first 10 rows of the new data frame 'total_sales'.
first_10_rows <- head(total_sales, 10)
print(first_10_rows)
```

```
   Product Total_NA_Sales Total_EU_Sales Total_Global_Sales
1     1012           5.73           3.71               10.75
2     1031           5.54           2.14               10.69
3      107          34.02          23.80               67.85
4     1175           2.09           2.89               10.06
5     1183           3.89           3.21               10.01
6     1212           6.54           2.32                9.95
7      123          26.64           4.01               37.16
8     1241           3.61           2.27                9.76
9     1307           9.84           4.89               18.29
10    1459           2.47           0.01                9.56
```

]:
```r
# Explore the total_sales data frame.
head(total_sales)
tail(total_sales)
dim(total_sales)
str(total_sales)
summary(total_sales)
```

A data.frame: 6 × 4

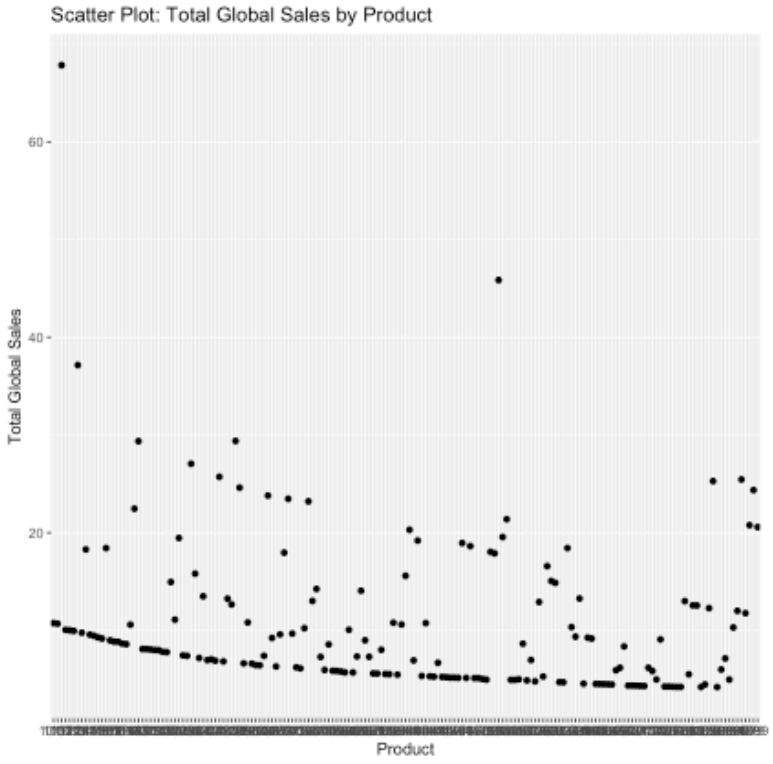
	Product	Total_NA_Sales	Total_EU_Sales	Total_Global_Sales
	<chr>	<dbl>	<dbl>	<dbl>
1	1012	5.73	3.71	10.75
2	1031	5.54	2.14	10.69
3	107	34.02	23.80	67.85
4	1175	2.09	2.89	10.06
5	1183	3.89	3.21	10.01
6	1212	6.54	2.32	9.95

A data.frame: 6 × 4

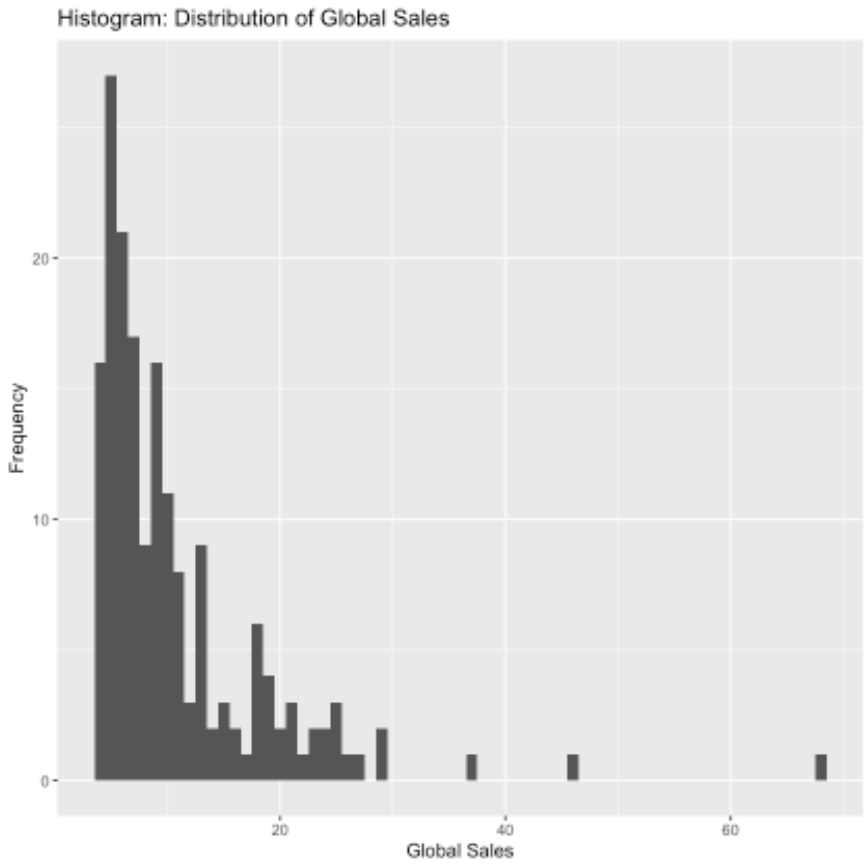
	Product	Total_NA_Sales	Total_EU_Sales	Total_Global_Sales
	<chr>	<dbl>	<dbl>	<dbl>
170	930	4.83	4.13	12.00
171	948	14.42	7.79	25.45
172	977	4.24	3.32	11.77
173	978	9.75	7.83	20.77
174	979	11.55	9.07	24.36
175	999	11.09	6.66	20.58

Create scatterplots, histograms, and boxplots to gain insights into the sales data.

```
## 2b) Determine which plot is the best to compare game sales.

# Create scatterplots.

# Scatter Plot: Total Global Sales by Product
ggplot(data = total_sales, aes(x = Product, y = Total_Global_Sales)) +
  geom_point() +
  labs(title = "Scatter Plot: Total Global Sales by Product",
       x = "Product",
       y = "Total Global Sales")
```



```
# Create histograms.

# Histogram: Distribution of Total Global Sales
ggplot(data = total_sales, aes(x = Total_Global_Sales)) +
  geom_histogram(binwidth = 1) +
  labs(title = "Histogram: Distribution of Global Sales",
       x = "Global Sales",
       y = "Frequency")
```

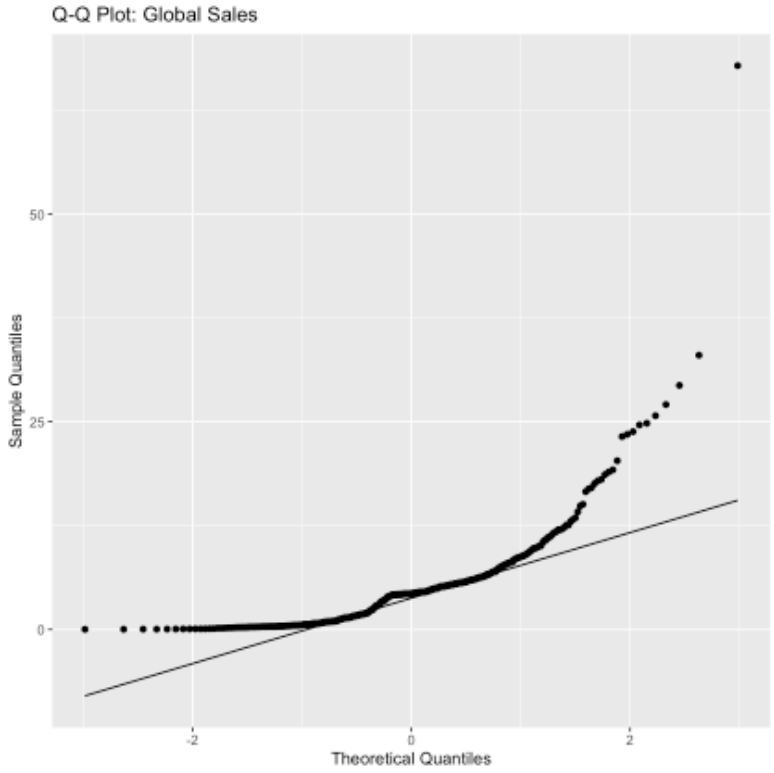


None of the above two plots seem to show a clear view of the Sales by product.

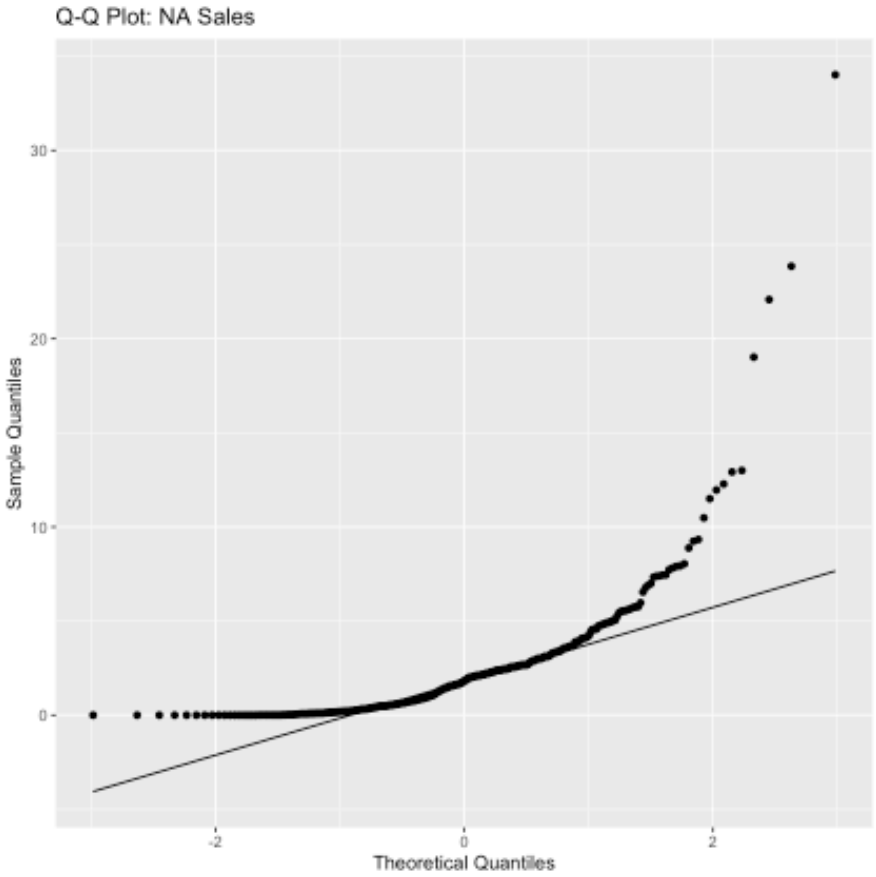
Since I was more familiar with R at this stage, I decided to apply my knowledge and review my work done in the assignment activity 4 to populate new plots, so I will use those plots for my presentation since they provide more insight into the data.

3. Determine the normality of the data set

```
# 3a) Create Q-Q Plots.
# Create Q-Q Plot for Global Sales.
ggplot(data = new_sales, aes(sample = Global_Sales)) +
  stat_qq() +
  stat_qq_line() +
  labs(title = "Q-Q Plot: Global Sales",
       x = "Theoretical Quantiles",
       y = "Sample Quantiles")
```



```
# Create Q-Q Plot for NA Sales.
ggplot(data = new_sales, aes(sample = NA_Sales)) +
  stat_qq() +
  stat_qq_line() +
  labs(title = "Q-Q Plot: NA Sales",
       x = "Theoretical Quantiles",
       y = "Sample Quantiles")
```



Interpreting the Q-Q Plot:

As we can see, the points for the three Sales columns deviate from the line in an S-shape, this indicates that the data might have heavy tails, indicating non-normality.

```r
# 3b) Perform Shapiro-Wilk test.
# Install and import Moments.
install.packages("moments")

# Load the moments package
library(moments)

# Perform Shapiro-Wilk test.

# Perform Shapiro-Wilk test on all sales columns
shapiro_results <- lapply(new_sales[, c("NA_Sales", "EU_Sales", "Global_Sales")], shapiro.test)

# Create a data frame to store the results
shapiro_results_new_sales <- data.frame(
  Column = names(new_sales[, c("NA_Sales", "EU_Sales", "Global_Sales")]),
  p_value = sapply(shapiro_results, function(x) x$p.value),
  is_normal = sapply(shapiro_results, function(x) x$p.value > 0.05)
)

# View the output.
print(shapiro_results_new_sales)
```

```
The downloaded binary packages are in
        /var/folders/fz/gk1vc_0x66369mjlyppr8rh80000gn/T//RtmpFG0plg/downloaded_packages
                    Column      p_value is_normal
NA_Sales         NA_Sales 7.001091e-27     FALSE
EU_Sales         EU_Sales 2.393884e-26     FALSE
Global_Sales Global_Sales 3.200364e-25     FALSE
```

Shapiro-Wilk test result:

The p-value for the three sales columns is ≤ 0.05, then we reject the null hypothesis. This indicates that there is significant evidence to suggest that the data deviates from a normal distribution. In this case, the data is not normally distributed.

```r
# 3c) Determine Skewness and Kurtosis.
# Skewness and Kurtosis.

# Calculate skewness and kurtosis of sales columns.
skewness_values <- sapply(new_sales[, c("NA_Sales", "EU_Sales", "Global_Sales")], skewness)
kurtosis_values <- sapply(new_sales[, c("NA_Sales", "EU_Sales", "Global_Sales")], kurtosis)

# Create a data frame to store the results.
skewness_kurtosis_new_sales <- data.frame(
  Column = names(new_sales[, c("NA_Sales", "EU_Sales", "Global_Sales")]),
  Skewness = skewness_values,
  Kurtosis = kurtosis_values
)

# View the output.
print(skewness_kurtosis_new_sales)
```

```
                    Column Skewness Kurtosis
NA_Sales         NA_Sales 4.309210 31.36852
EU_Sales         EU_Sales 4.818688 44.68924
Global_Sales Global_Sales 4.045582 32.63966
```

Interpreting the Skewness and Kurtosis values:

Skewness:

The skewness values for the three sales columns is positive (greater than 0), meaning that the distribution is skewed to the right, in other words, the tail on the right side is longer than the left side.

Kurtosis:

The Kurtosis values for the sales columns are greater than 3 (positive kurtosis), meaning that the distribution has heavier tails and a sharper peak compared to the normal distribution.

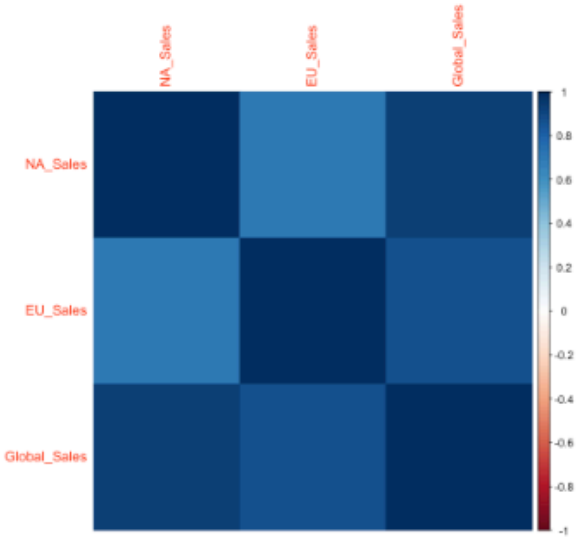
```r
# 3d) Determine correlation between the three sales columns.

# Calculate the correlation matrix.
sales_correlation <- cor(new_sales[, c("NA_Sales", "EU_Sales", "Global_Sales")])

# View the output.
print(sales_correlation)
```

```
              NA_Sales  EU_Sales Global_Sales
NA_Sales     1.0000000 0.7055236    0.9349455
EU_Sales     0.7055236 1.0000000    0.8775575
Global_Sales 0.9349455 0.8775575    1.0000000
```

```r
# The sales_correlation matrix will display the correlation coefficients between the speci
# The correlation coefficient ranges from -1 to 1, where -1 indicates a perfect negative c
# 1 indicates a perfect positive correlation, and 0 indicates no correlation.

# Install corrplot package.
install.packages("corrplot")

# Load the corrplot package.
library(corrplot)

# Visualize the correlation matrix as a heatmap.
corrplot(sales_correlation, method = "color")
```

```
The downloaded binary packages are in
        /var/folders/fz/gklvc_0x66369mjlyppr8rh80000gn/T//RtmpFG0plg/downloaded_packages
```

```
corrplot 0.92 loaded
```

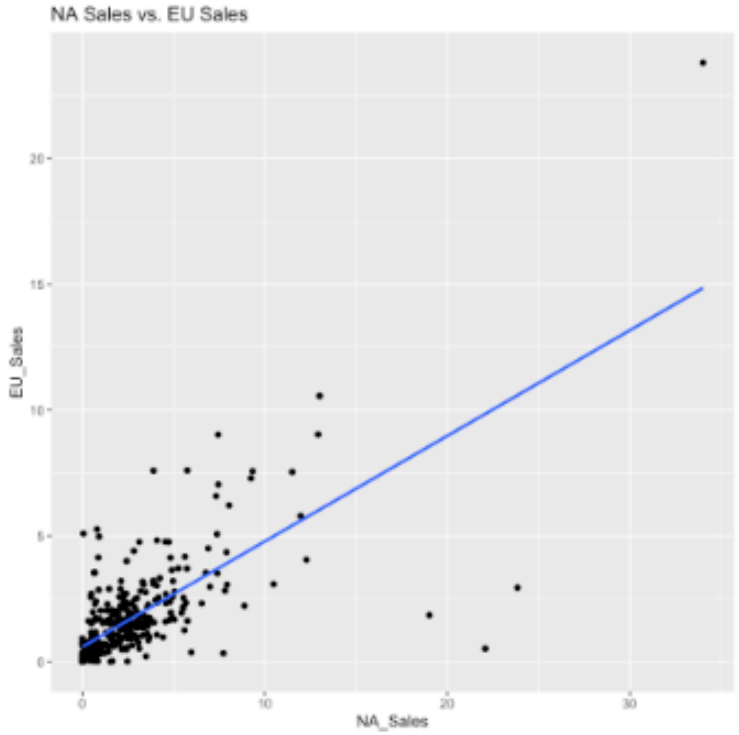


As expected, there is a positive correlation between the global sales and the NA sales and EU sales, since global sales in the world is the sum of EU sales, NA sales and other sales.

For future analysis, it might worth digging into the 'other sales' category and calculate the sales under this uncategorised group. Turtle Games should properly collect and label the 'other sales' category to identify the market / markets that belong to this category. This would be very useful to inform future marketing campaigns too since we would be able to see socioeconomic and demographic profile of the customers under this category.

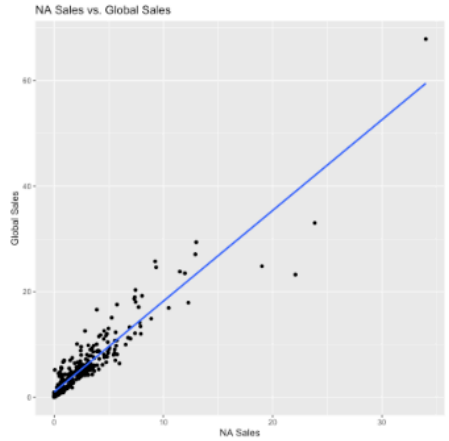
4. Plot the data

]:
```r
# Create plots to gain insights into the data.
# Scatter plot for NA_Sales vs. EU_Sales.
ggplot(new_sales, aes(x = NA_Sales, y = EU_Sales)) +
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ x, se = FALSE) +
  labs(title = "NA Sales vs. EU Sales",
       x = "NA_Sales",
       y = "EU_Sales")
```



Overall, NA Sales are greater than EU sales.

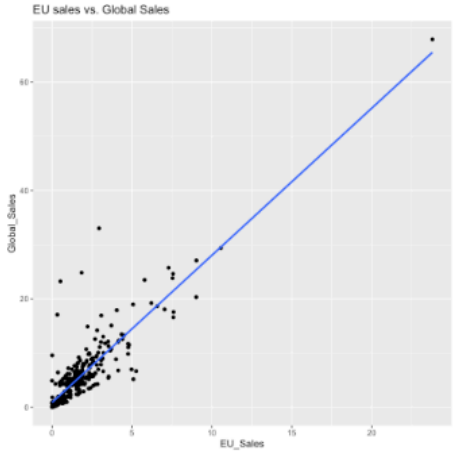
```r
# Scatter plot for NA_Sales vs. Global_Sales
ggplot(new_sales, aes(x = NA_Sales, y = Global_Sales)) +
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ x, se = FALSE) +
  labs(title = "NA Sales vs. Global Sales",
       x = "NA Sales",
       y = "Global Sales")
```



Overall, we can see that the global sales values a much greater than the EU sales values, that makes sense.

We can also see how there are some global sales values that deviate more from the normal distribution, meaning that these outliers are the greatest sales values observed within the global sales column, such as the black dot shown way over the 60M mark.

```r
# Scatter plot for EU_Sales vs. Global_Sales
ggplot(new_sales, aes(x = EU_Sales, y = Global_Sales)) +
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ x, se = FALSE) +
  labs(title = "EU sales vs. Global Sales",
       x = "EU_Sales",
       y = "Global_Sales")
```

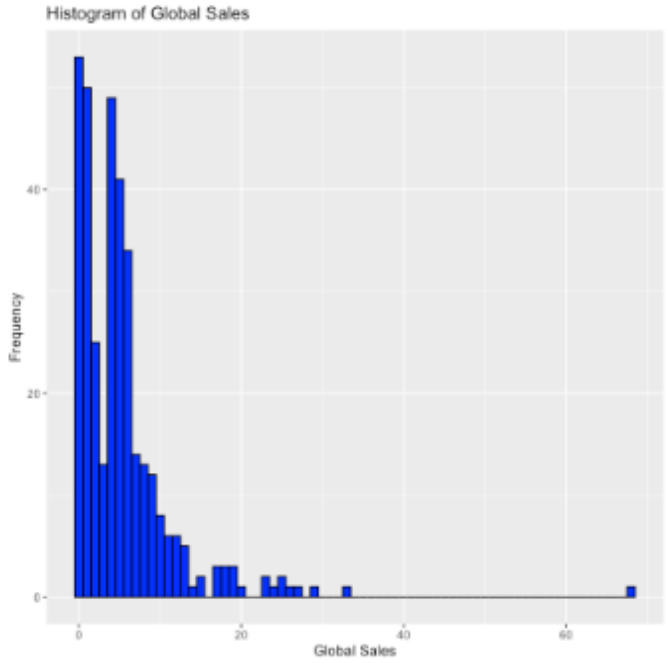


Overall, we can see that the global sales values a much greater than the EU sales values, that makes sense.

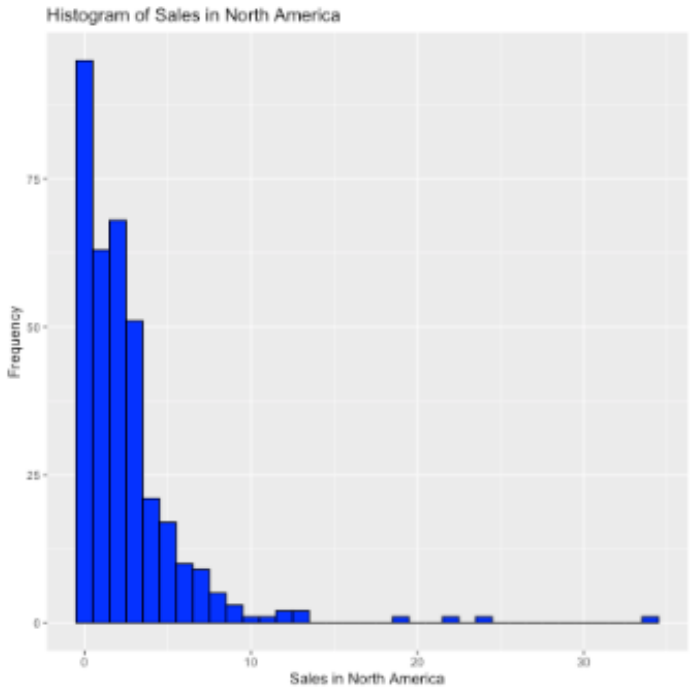
We can also see how there are some global sales values that deviate more from the normal distribution, meaning that these outliers are the greatest sales values observed within the global sales column.

Next, I'll explore how sales are distributed at a global scale and by market (EU, NA), using histograms and barplots.

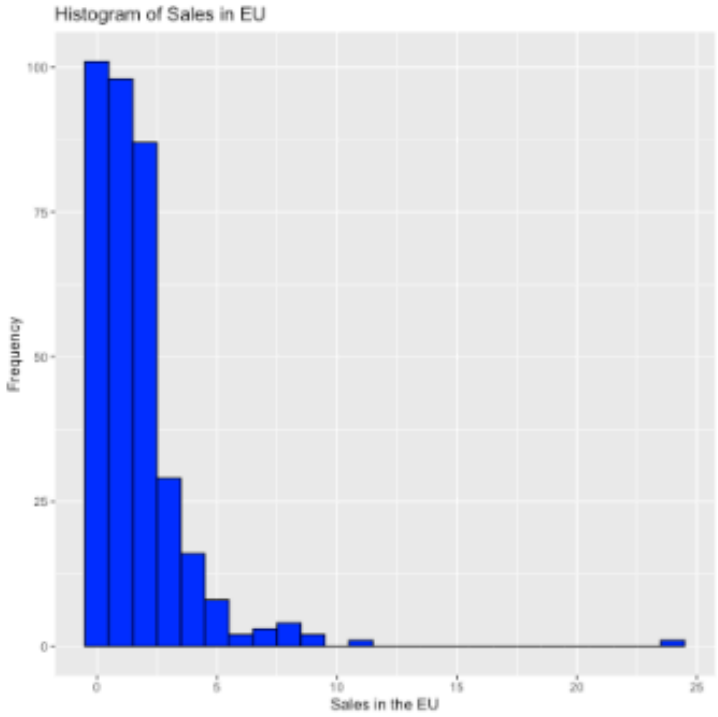
]:
```r
# Histogram of Global Sales.
ggplot(new_sales, aes(x = Global_Sales)) +
  geom_histogram(binwidth = 1, fill = "blue", color = "black") +
  labs(title = "Histogram of Global Sales",
       x = "Global Sales",
       y = "Frequency")
```



]:
```r
# Histogram of Sales in North America.
ggplot(new_sales, aes(x = NA_Sales)) +
  geom_histogram(binwidth = 1, fill = "blue", color = "black") +
  labs(title = "Histogram of Sales in North America",
       x = "Sales in North America",
       y = "Frequency")
```



```r
# Histogram of Sales in European Union.
ggplot(new_sales, aes(x = EU_Sales)) +
  geom_histogram(binwidth = 1, fill = "blue", color = "black") +
  labs(title = "Histogram of Sales in EU",
       x = "Sales in the EU",
       y = "Frequency")
```



F. Making recommendations to the business

I'll investigate any possible relationship(s) in the sales data by creating a simple and multiple linear regression model.

Create a simple linear regression model.

```
# A. Create a linear regression model to compare Global Sales vs NA Sales.

# Step 1: Create a simple linear regression model
model_A <- lm(Global_Sales ~ NA_Sales, data = new_sales)

# Step 2: Get the correlation between columns
correlation_A <- cor(new_sales$Global_Sales, new_sales$NA_Sales)

# Step 3: Print the model summary and correlation.
summary(model_A)
print(correlation_A)
```

```
Call:
lm(formula = Global_Sales ~ NA_Sales, data = new_sales)

Residuals:
     Min       1Q   Median       3Q      Max
-15.7352  -1.0341  -0.5555   0.6247   8.8676

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.01232    0.14752   6.862 3.09e-11 ***
NA_Sales     1.71797    0.03485  49.300  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.226 on 350 degrees of freedom
Multiple R-squared:  0.8741,    Adjusted R-squared:  0.8738
F-statistic:  2430 on 1 and 350 DF,  p-value: < 2.2e-16

[1] 0.9349455
```

```r
# B. Create a linear regression model to compare Global Sales vs Total EU Sales.

# Step 1: Create a simple linear regression model
model_B <- lm(Global_Sales ~ EU_Sales, data = new_sales)

# Step 2: Get the correlation between columns
correlation_B <- cor(new_sales$Global_Sales, new_sales$EU_Sales)

# Step 3: Print the model summary and correlation
summary(model_B)
print(correlation_B)
```

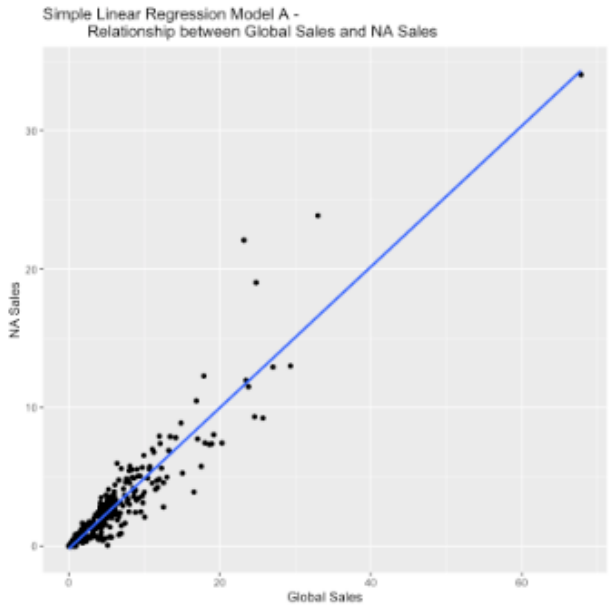
```
Call:
lm(formula = Global_Sales ~ EU_Sales, data = new_sales)

Residuals:
    Min      1Q  Median      3Q     Max
-9.5377 -1.2173 -0.6040  0.8755 24.1474

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.87350    0.20660   4.228 3.01e-05 ***
EU_Sales     2.71399    0.07926  34.241  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.008 on 350 degrees of freedom
Multiple R-squared:  0.7701,    Adjusted R-squared:  0.7695
F-statistic:  1172 on 1 and 350 DF,  p-value: < 2.2e-16

[1] 0.8775575
```

```r
# C. Create a linear regression model to compare Total NA Sales vs Total EU Sales.

# Step 1: Create a simple linear regression model
model_C <- lm(NA_Sales ~ EU_Sales, data = new_sales)

# Step 2: Get the correlation between columns
correlation_C <- cor(new_sales$NA_Sales, new_sales$EU_Sales)

# Step 3: Print the model summary and correlation
summary(model_C)
print(correlation_C)
```

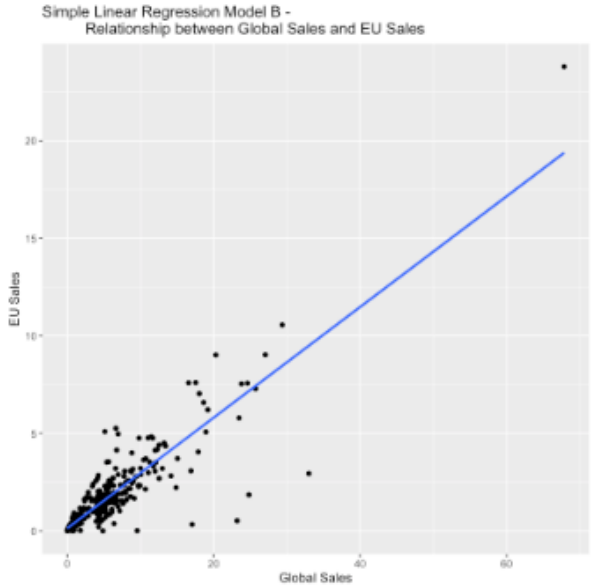
```
Call:
lm(formula = NA_Sales ~ EU_Sales, data = new_sales)

Residuals:
    Min      1Q  Median      3Q     Max
-6.5482 -0.8419 -0.3678  0.5287 20.8985

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.56407    0.16618   3.394 0.000767 ***
EU_Sales     1.18744    0.06376  18.625  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.42 on 350 degrees of freedom
Multiple R-squared:  0.4978,    Adjusted R-squared:  0.4963
F-statistic: 346.9 on 1 and 350 DF,  p-value: < 2.2e-16

[1] 0.7055236
```

Create plots to view the linear regression.

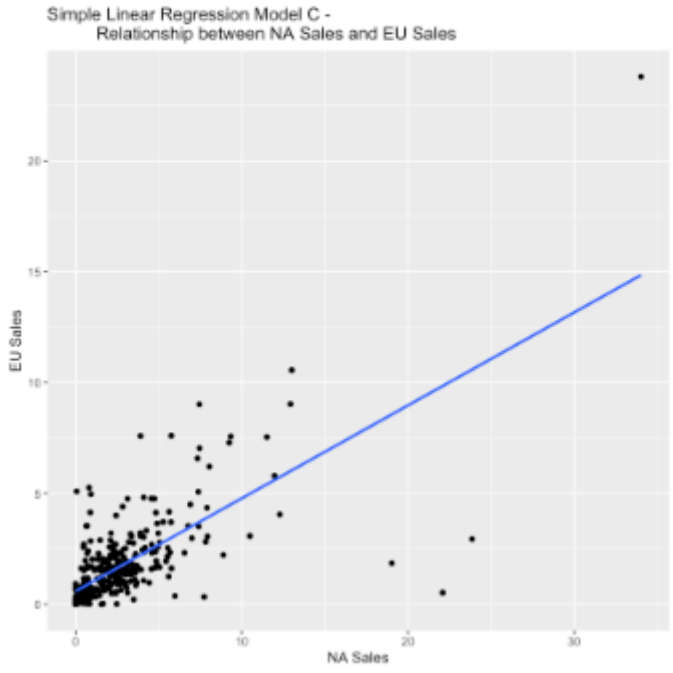
```
# Create the plot for the lineal regression model A.
ggplot(new_sales, aes(x = Global_Sales, y = NA_Sales)) +
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ x, se = FALSE) +
  labs(x = "Global Sales", y = "NA Sales") +
  ggtitle("Simple Linear Regression Model A -
          Relationship between Global Sales and NA Sales")
```




```
# Create the plot for the lineal regression model B.
ggplot(new_sales, aes(x = Global_Sales, y = EU_Sales)) +
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ x, se = FALSE) +
  labs(x = "Global Sales", y = "EU Sales") +
  ggtitle("Simple Linear Regression Model B -
          Relationship between Global Sales and EU Sales")
```



```
# Create the plot for the lineal regression model C.
ggplot(new_sales, aes(x = NA_Sales, y = EU_Sales)) +
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ x, se = FALSE) +
  labs(x = "NA Sales", y = "EU Sales") +
  ggtitle("Simple Linear Regression Model C -
          Relationship between NA Sales and EU Sales")
```



Create a multiple linear regression model and determine the correlation between the sales columns.

Select only numeric columns from the original data frame.

```
# Import the original data frame.
sales <- read.csv("turtle_sales.csv", header = TRUE)

# Print the data frame.
head(sales)
```

A data.frame: 6 × 9

	Ranking	Product	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	Global_Sales
	<int>	<int>	<chr>	<int>	<chr>	<chr>	<dbl>	<dbl>	<dbl>
1	1	107	Wii	2006	Sports	Nintendo	34.02	23.80	67.85
2	2	123	NES	1985	Platform	Nintendo	23.85	2.94	33.00
3	3	195	Wii	2008	Racing	Nintendo	13.00	10.56	29.37
4	4	231	Wii	2009	Sports	Nintendo	12.92	9.03	27.06
5	5	249	GB	1996	Role-Playing	Nintendo	9.24	7.29	25.72
6	6	254	GB	1989	Puzzle	Nintendo	19.02	1.85	24.81

```
# Convert the 'product' column to a character type.
sales$Product <- as.character(sales$Product)

# Check the data types after conversion.
str(sales)
```

```
'data.frame':   352 obs. of  9 variables:
 $ Ranking     : int  1 2 3 4 5 6 7 8 9 10 ...
 $ Product     : chr  "107" "123" "195" "231" ...
 $ Platform    : chr  "Wii" "NES" "Wii" "Wii" ...
 $ Year        : int  2006 1985 2008 2009 1996 1989 2006 2006 2009 1984 ...
 $ Genre       : chr  "Sports" "Platform" "Racing" "Sports" ...
 $ Publisher   : chr  "Nintendo" "Nintendo" "Nintendo" "Nintendo" ...
 $ NA_Sales    : num  34.02 23.85 13 12.92 9.24 ...
 $ EU_Sales    : num  23.8 2.94 10.56 9.03 7.29 ...
 $ Global_Sales: num  67.8 33 29.4 27.1 25.7 ...
```

```r
# Multiple linear regression model.

# Select numeric columns for the model
numeric_cols <- sales[, c("NA_Sales", "EU_Sales", "Global_Sales")]

# Create the multiple linear regression model
model_multiple <- lm(Global_Sales ~ NA_Sales + EU_Sales, data = numeric_cols)

# Print the summary of the model
summary(model_multiple)
```

```
Call:
lm(formula = Global_Sales ~ NA_Sales + EU_Sales, data = numeric_cols)

Residuals:
    Min      1Q  Median      3Q     Max
-3.6186 -0.4234 -0.2692  0.0796  7.4639

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.22175    0.07760   2.858  0.00453 **
NA_Sales     1.15543    0.02456  47.047  < 2e-16 ***
EU_Sales     1.34197    0.04134  32.466  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.112 on 349 degrees of freedom
Multiple R-squared:  0.9687,    Adjusted R-squared:  0.9685
F-statistic:  5398 on 2 and 349 DF,  p-value: < 2.2e-16
```

 # Multiple linear regression model interpretation:

The adjusted R-squared value of 0.9685 indicates that approximately 96.85% of the variance in the Global Sales is explained by the EU Sales and NA Sales in the linear regression model. This is a high value and suggests that the model fits the data well and is likely to be a good predictor of the Global Sales.

Estimate 2: B1, Intercept (NA Sales): it refers to the estimated coefficient of the NA Sales in the model. Specifically, it represents the change in the Global Sales for a one-unit change in the NA Sales, holding all other variables constant (EU Sales). In this case the estimated coefficient of 1.15 means that for each one-unit increase in the NA Sales, the predicted value of the Global Sales is expected to increase by 1.15 units, while keeping the EU Sales and any other variables constant.

Estimate 2: B1, Intercept (EU Sales): In this case the estimated coefficient of 1.34 means that for each one-unit increase in the EU Sales, the predicted value of the Global Sales is expected to increase by 1.34 units, while keeping the NA Sales and any other variables constant.

NA Sales: A large t-value, such as 47.047, indicates that the coefficient estimate of NA Sales is highly statistically significant, suggesting a strong and statistically significant relationship between the NA Sales and the Global Sales. It means that the coefficient is unlikely to be zero by chance alone and provides evidence that the variable (NA Sales) has a significant impact on the outcome variable (Global Sales).

EU Sales: A t-value of 32.466 suggests a strong and statistically significant relationship between the Eu Sales and the Global Sales in the multiple linear regression model.

Predict global sales based on provided values and compare your prediction to the observed value(s).

4. Predict global sales based on provided values.

Compare your prediction to the observed value(s).

```
# Provided NA_Sales_sum and EU_Sales_sum values.
# NA_Sales_sum of 34.02 and EU_Sales_sum of 23.80.
# NA_Sales_sum of 3.93 and EU_Sales_sum of 1.56.
# NA_Sales_sum of 2.73 and EU_Sales_sum of 0.65.
# NA_Sales_sum of 2.26 and EU_Sales_sum of 0.97.
# NA_Sales_sum of 22.08 and EU_Sales_sum of 0.52.
```

```
# Create a data frame with the provided NA_Sales_sum and EU_Sales_sum values
predicted_global_sales <- data.frame(Provided_NA_Sales = c(34.02, 3.93, 2.73, 2.26, 22.08),
                        Provided_EU_Sales = c(23.80, 1.56, 0.65, 0.97, 0.52))

# Predict global sales using the multiple linear regression model.
predicted_global_sales <- predict(model_multiple, predictedglobalsales = predicted_global_sales)

# Print the predicted global sales.
predicted_global_sales
```

```
1
71.4685719565562
2
31.7242592920501
3
29.4136250420139
4
27.2679725186325
5
20.6809417295976
6
24.6807609296029
7
21.1606831021403
8
23.6277167123172
```

```
# Create a data frame with the predicted sales.
predicted_scenarios <- data.frame(
  Predicted_NA_Sales_Sum = c(34.02, 3.93, 2.73, 2.26, 22.08),
  Predicted_EU_Sales_Sum = c(23.80, 1.56, 0.65, 0.97, 0.52),
  Predicted_Other_Sales_Sum = c(0, 0, 0, 0, 0)  # We could add the sum of other sales if available.
)

# Calculate the predicted Global_Sales
predicted_scenarios$Predicted_Global_Sales <- predicted_scenarios$Predicted_NA_Sales_Sum
+ predicted_scenarios$Predicted_EU_Sales_Sum + predicted_scenarios$Predicted_Other_Sales_Sum

# Print the data frame with predicted values
predicted_scenarios
```

A data.frame: 5 × 4

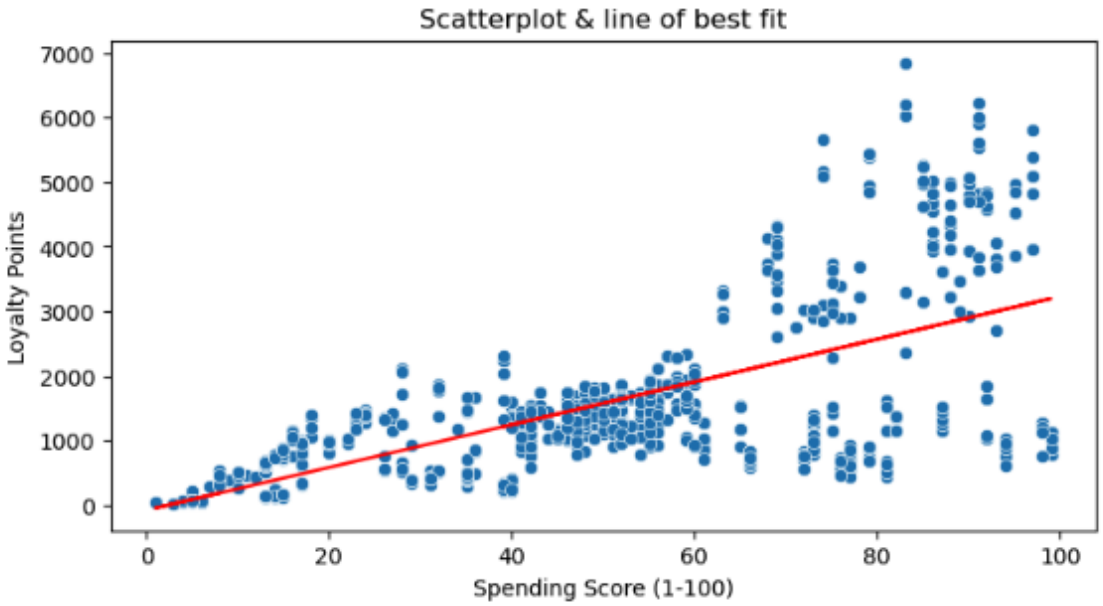
Predicted_NA_Sales_Sum	Predicted_EU_Sales_Sum	Predicted_Other_Sales_Sum	Predicted_Global_Sales
<dbl>	<dbl>	<dbl>	<dbl>
34.02	23.80	0	57.82
3.93	1.56	0	5.49
2.73	0.65	0	3.38
2.26	0.97	0	3.23
22.08	0.52	0	22.60

```
# View the descriptive statistics of the predicted_scenarios data frame.
summary(predicted_scenarios)
str(predicted_scenarios)
```

```
 Predicted_NA_Sales_Sum Predicted_EU_Sales_Sum Predicted_Other_Sales_Sum
 Min.   : 2.26          Min.   : 0.52          Min.   :0
 1st Qu.: 2.73          1st Qu.: 0.65          1st Qu.:0
 Median : 3.93          Median : 0.97          Median :0
 Mean   :13.00          Mean   : 5.50          Mean   :0
 3rd Qu.:22.08          3rd Qu.: 1.56          3rd Qu.:0
 Max.   :34.02          Max.   :23.80          Max.   :0
 Predicted_Global_Sales
 Min.   : 3.23
 1st Qu.: 3.38
 Median : 5.49
 Mean   :18.50
 3rd Qu.:22.60
 Max.   :57.82

'data.frame':   5 obs. of  4 variables:
 $ Predicted_NA_Sales_Sum   : num  34.02 3.93 2.73 2.26 22.08
 $ Predicted_EU_Sales_Sum   : num  23.8 1.56 0.65 0.97 0.52
 $ Predicted_Other_Sales_Sum: num  0 0 0 0 0
 $ Predicted_Global_Sales   : num  57.82 5.49 3.38 3.23 22.6
```

3. Visualisations and Insights - Patterns, Predictions and Recommendations

Question 1: How customers accumulate loyalty points?

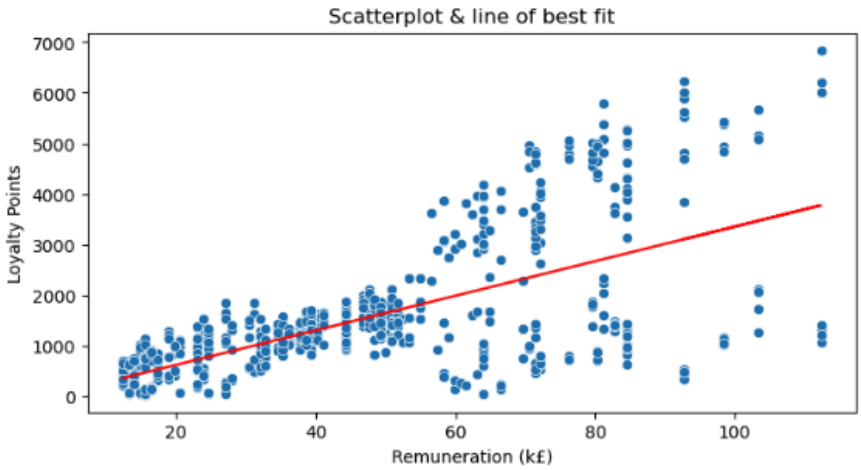


Correlation between loyalty points and spending score: 67%.

This indicates a positive linear relationship between the two variables, showing that as the spending score increases, the loyalty points tend to increase as well. In other words, customers with higher spending scores tend to accumulate more loyalty points, however this does not imply causation between these two variables, since there are customers with high spending scores that do not accumulate many loyalty points, (all those blue dots below the red trend line.

By looking at the OLS results, we can say that 45% of the variability of the Loyalty Points is explained by the Spending Score.

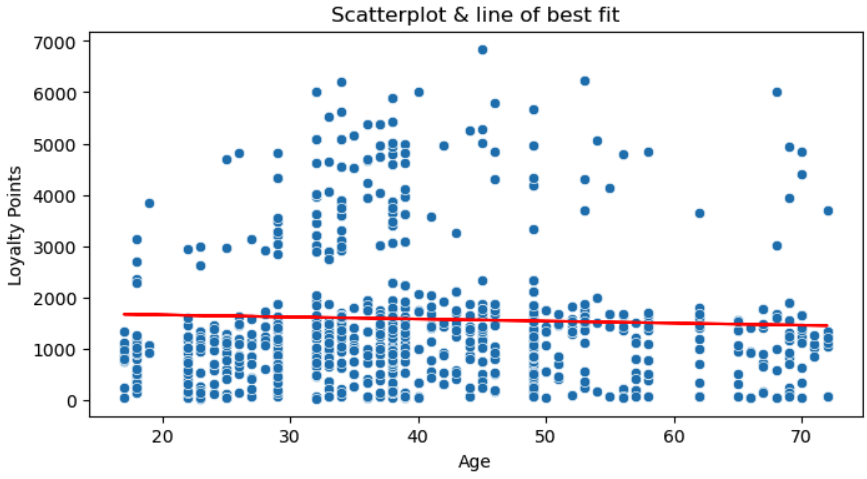
Overall, spending score is not a very significant estimate of the loyalty points, and do not have a strong effect on the loyalty points in this linear regression model.



Correlation between loyalty points and remuneration: 62%.

This indicates a positive and moderately strong linear relationship between the two variables, meaning that customers with higher remunerations tend to accumulate more loyalty points, however this does not indicate causation between these two variables, since there are customers with hight remunerations who do not accumulate many loyalty points.

Remuneration is not a very significant estimate of the loyalty points and do not have a strong effect on the loyalty points in this linear regression model.



Correlation between loyalty points and age: -4.2%.

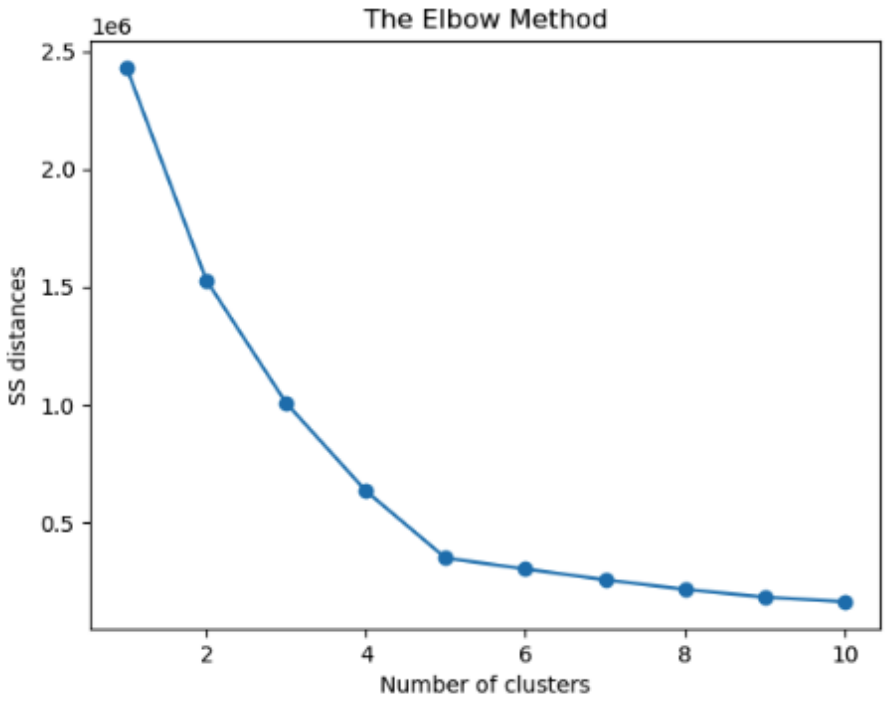
This indicates that there is a weak negative linear relationship between age and loyalty points. As age increases, loyalty points tend to slightly decrease, and viceversa. However, since the correlation is negative this linear relationship is very weak, and this why we see the points scattered around without following a clear pattern.

It's important to note that a weak correlation does not imply causation. The correlation coefficient only measures the strength and direction of the linear relationship between age and loyalty points. It does not imply that age is the cause of changes in loyalty points or viceversa.

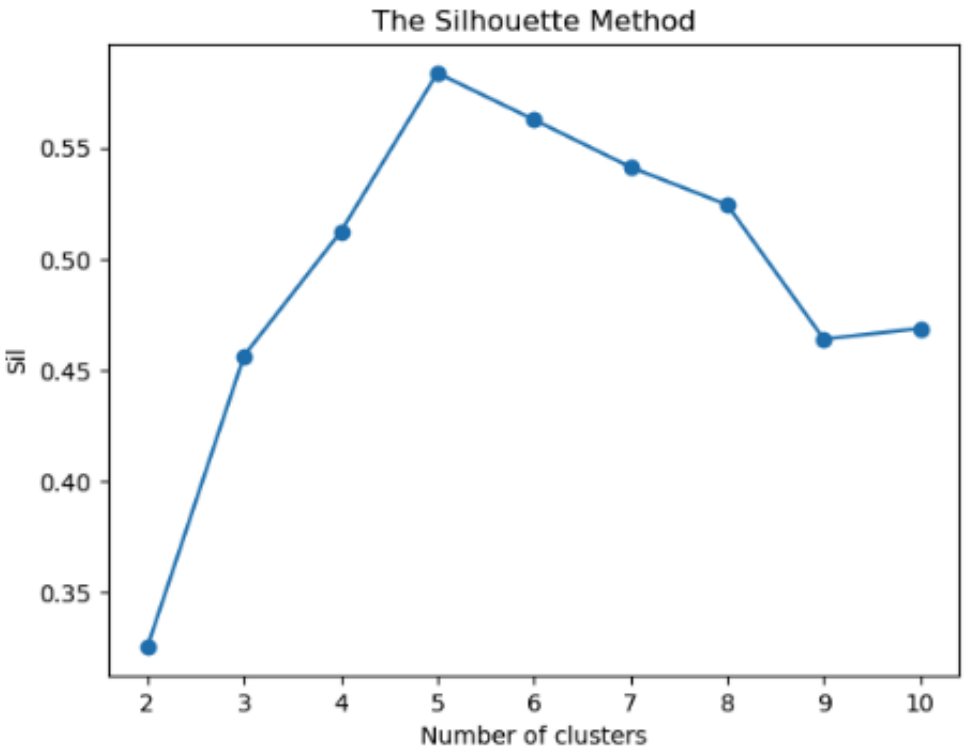
The weak correlation may indicate that age alone is not a strong predictor of loyalty points. There may be other factors or interactions with age that better explain variations in loyalty points, hence we can say that age is not a significant estimate of the loyalty points.

Question 2: How useful are remuneration and spending scores data in providing data for analysis and in finding groups within the customer base that can be used to target specific market segments?

Considering that Turtle Games is looking to identify groups within the customer base that can be used to target specific market segments for their marketing campaigns, our job is to identify the optimal number of clusters for this purpose.



Based on the Elbow method, five clusters have been identified. Next, I'll apply the Silhouette method.



Based on the Silhouette method, five main clusters have been identified, meaning that the 5 cluster solution has the highest average silhouette coefficient. Based on our data set, we will have to judge what number of clusters is the most appropriate to consider.

The number of predicted values per class seems to indicate a better distribution for k=5 than k=7 and k=3, hence I selected five clusters as the optimal number of clusters based on all the analysis and insights mentioned above.

We can clearly see that there is a distinct demarcation in terms of remuneration and spending scores, however here's not a causative relationship between these two variables.

Clusters two and four are the groups with the highest reunertation, however this is not an indication of a high spending score.

And clusters three and one are the customers with the lowest remuneration, but again this is not an indication of a lower spending sore.

The fifth group is cluster zero, which might be the easier segment to taget for Turtle Games, since this group of customers seems to behave more homogenously and isolated in terms of all of the obserbations within our data set grouping customers with a remuneration between (£35K - (£55K) approx. and a spending score between 35 - 60

points), with no other clusters showing this combination of reemuneration and spendig score, unlike the rest of the clusters that share some similarities of behaviour in terms of sharing either the same remuneration level with another cluster or sharing the same spending score with another cluster.

So, my proposal of segments for Turtle Games to target in their marketing campaigns would be as follow:

Segment 1: customers with a remuneration between (£35K - (£55K) and a spending score between 35 - 60 points)

Segment 2: customers with a remuneration between (£15K - (£35K) and a spending score between 60 - 100 points)

Segment 3: customers with a remuneration between (£55K - (£100K+) and a spending score between 60 - 100 points)

Segment 4: customers with a remuneration between (£15K - (£35K) and a spending score between 5 - 40 points)

Segment 5: customers with a remuneration between (£55K - (£100K+) and a spending score between 5 - 40 points)

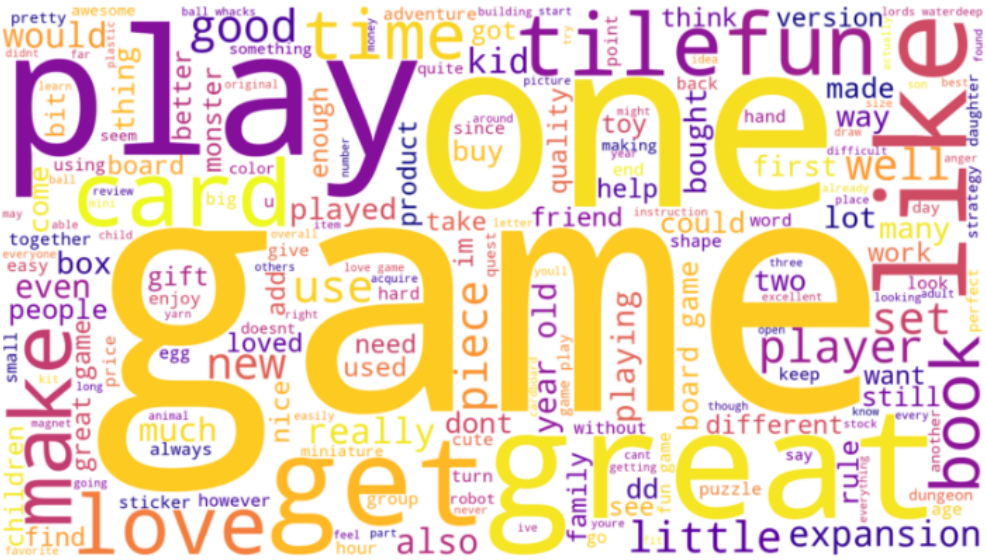
By grouping Turtle Games within the above segments, we could say that the spending score is an indication of the number of purchases or the amount of money spent by customers during a specific range of time, indicating that regardless of the customer remuneration, there are customers with a low remuneration that tend to have a high spending score (we can assume that this is because they spend more), these customers belong to segment 2, and then to give another example of how customers behave, we can see that for instance, customers within segment 5, with a high remuneration above 60K, tend to have a lower spendig score, below 40 points (we can assume that this is because they spend less).

In future analysis, it might be worth looking at the behaviour of these segments by considering a specific period of time, let's say one year, one month, or even a quarter, so we can then compare the customers behaviour across different years, months or quarters.

It might also be interesting for future analysis, to introduce a third variable into our clustering analysis, such as the customer age, gender or education.

Question 3: How customer reviews can be used to inform marketing campaigns?

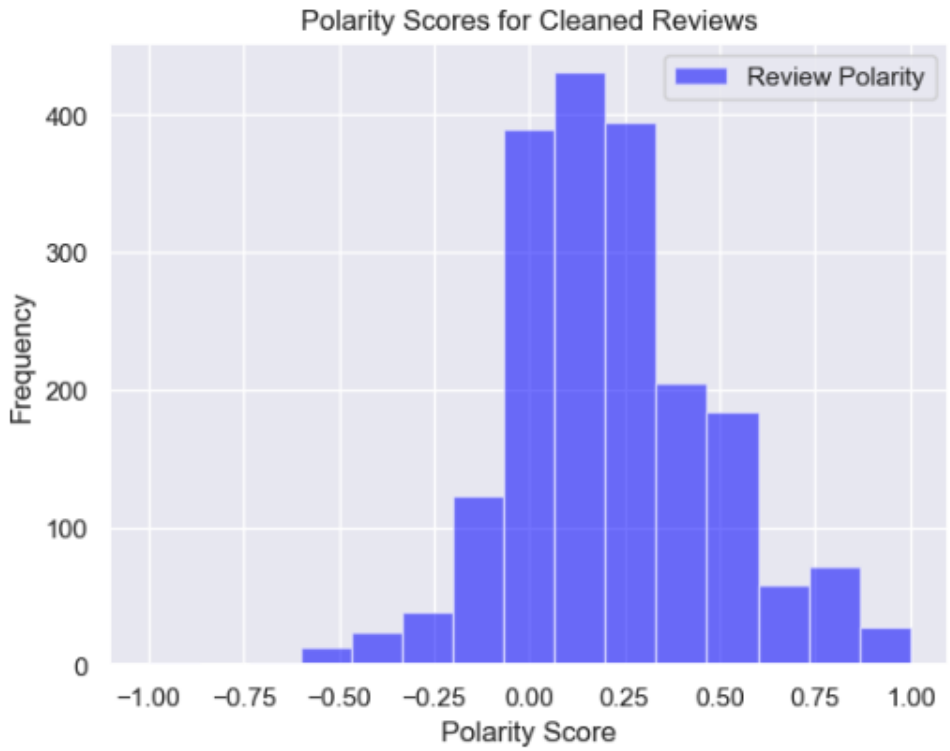
Review Word Cloud



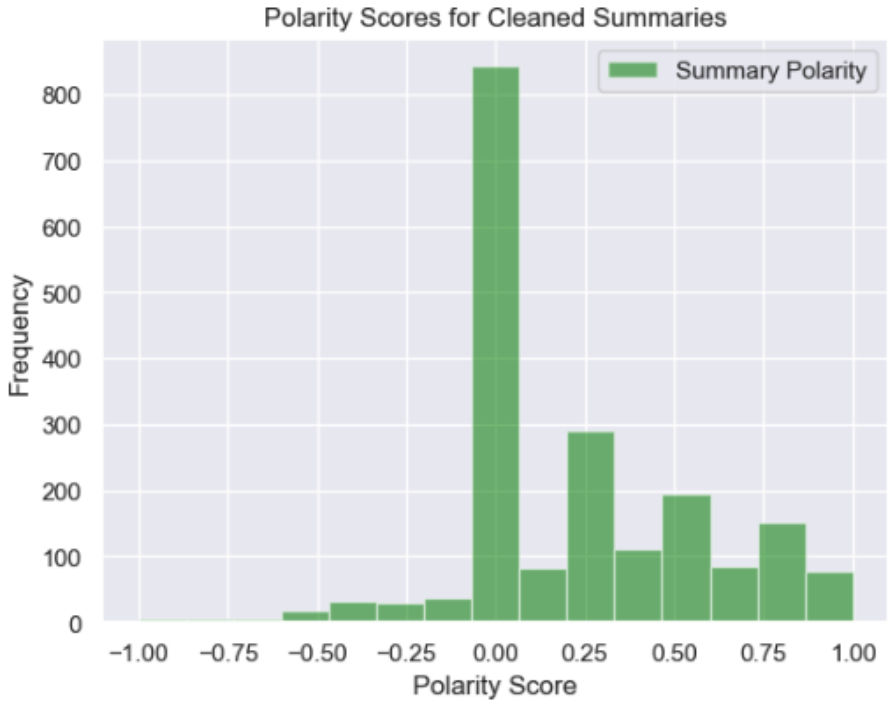
Review_Words	Frequency
game	1671
great	580
fun	552
one	530
play	502
like	414
love	323
really	319
get	319
cards	301
tiles	297
time	291
good	289
would	280
book	273

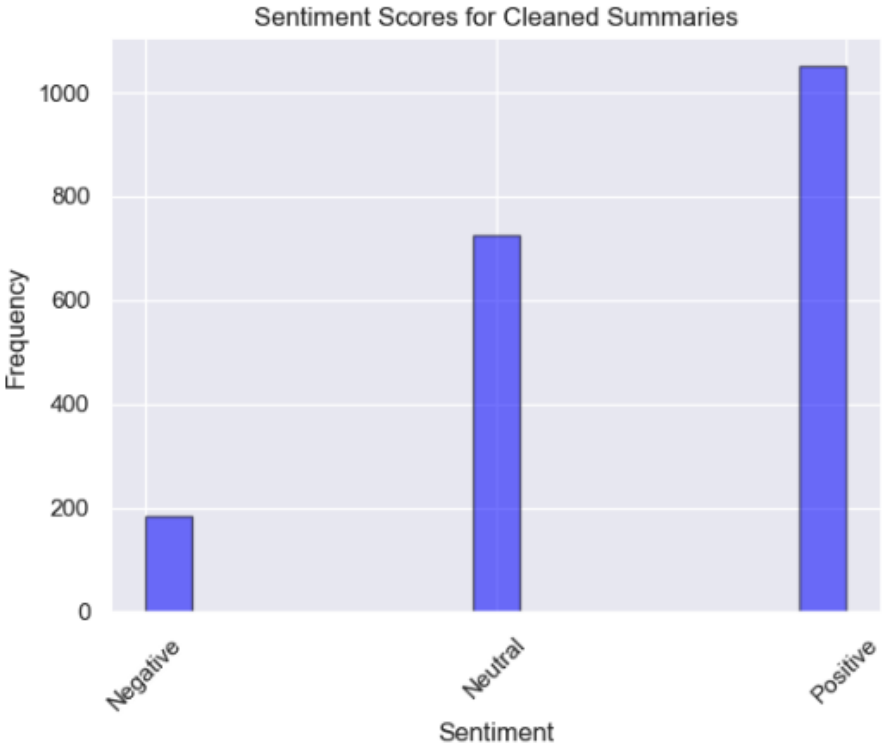
Summary Word Cloud

	Frequency
Summary_Words	
stars	427
five	342
game	319
great	295
fun	218
love	93
good	92
four	58
like	54
expansion	52
kids	50
cute	45
book	43
one	38
awesome	36


Polarity Scores for Cleaned Reviews
Review Polarity
Frequency
0
100
200
300
400
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
Polarity Score


Sentiment Scores for Cleaned Reviews
Frequency
0
200
400
600
800
1000
1200
1400
1600
Negative
Positive
Neutral
Sentiment


Polarity Scores for Cleaned Summaries
Summary Polarity
Frequency
0
100
200
300
400
500
600
700
800
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
Polarity Score


Sentiment Scores for Cleaned Summaries
Frequency
0
200
400
600
800
1000
Negative
Neutral
Positive
Sentiment

Top 20 negative reviews:

```
['im sorry find product boring frank juvenile',
 'comes dms screen space screen absolute premium fact  space wasted art terribly informative needed art well makes co
mpletely useless reason gave  stars  technically speaking least still stand block notes dice rolls drops ball complet
ely',
 'really cheaply produced cardboard playing pieces cost',
 'high hopes game big fan fourth edition dd though like e enjoy im going tell one better ill tell one prefer also enj
oy games modular tiles really went game biased towards enjoying sadly couldnt rescue game exceedingly repetitive play
turns youll get entire experience game offer nothing new exciting game hopelessly shallow based much chance strategy
nonexistent despite modular tiles game offers virtually customization pieces seem wellmade fortunately might supply u
se actual dd since store bought doesnt returns however quality pieces matched game really wanted enjoy game played  w
hole games uncounted number aborted attempts accepting fact simply doesnt work extremely disappointing',
 'wish id watched gameplay videos ashardalon buying others mentioned rpg board game board game particularly bad board
game punishingly difficult little theme begs house rules clean terrible encounter card system tried playing aggressiv
ely quickly overwhelmed critters kill tried playing cautiously party wrecked encounter cards doesnt feel like proper
dungeon crawl youve killed five monsters lost half characters hp youve left sight staircase like combat game set wack
y arena bad stuff happens time seen another way like puzzle game balance risk generally much reward generally little
hopefully survive long enough reach objective way feel like dungeon crawl importantly poor introduction rpg elements
clearly minority wellreviewed game simply wish offer dissenting opinion ashardalon fun play im selling copy im going
try mice mystics instead looks like hoping ashardalon would balance teenfriendly story adventure combat need dm im al
so eager try arcadia quest grant two stars review instead one bits box fantastic excellent miniatures board pieces to
pnotch presentation looks fantastic take box',
 'book  pages size x note card much fun',
 'great game kids love play',
 'game tiles board tile stands made paper using times sustain paper board tiles move board making game messy inconven
ient manage shame done brilliant game',
 'guess look closely information game impression similar acquire used play quality one use looks cheap would paid get
better quality',
 'found card game opposite intended actually kids focusing ways get angry etc instead teaching calm act better really
tested sale better game would calm dragon tried game kids absolutely behavior anger problems began behaving badly get
ting angry second round dont recommend therapist work kids anger issues day long thought might good tool wrong',
 'kids grew peg bench hammer loved bought brand grandson disappointed pegs fit loosely bench even use hammer pound pu
sh hand sometimes fall automatically suggestion make pegs fit little tighter kids learn skills coordination etc pound
ing pegs nice thick little hands snug enough fitting really use toy intended',
 'mom already owned acquire game always commented poorly made thought would get new one christmas quality one much be
tter old one cards player see much hotel cost buy according many tiles one even expected better quality price paid di
dnt even come bag tiles think disappointed',
```

Top 20 negative summaries:

```
['disappointing',
 'fact  space wasted art terribly informative needed art',
 'disappointing',
 'disappointing',
 'disappointing',
 'small boring',
 'mad dragon',
 'bad qualityall made paper',
 'bad expecting',
 'promotes anger instead teaching calming methods',
 'disappointed',
 'disappointed',
 'disappointed',
 'disappointed',
 'another worthless dungeon masters screen galeforce',
 'hated running rpg campaign dealing towns',
 'boring',
 'horrible nothing say would give zero stars',
 'boring unless craft person',
 'worst value ive ever seen']
```

Top 20 positive reviews:

```
['make game come alive battles go much quickly smoothly nicely designed great way give detail depth battles',
 'excellent expansion lords waterdeep importantly adds sixth player option game',
 'pigeon books elementary school library dont pigeon missing students love pigeon',
 'disclaimer one villain cards came factory defect machine burned rarity far rest best miniatures ive ever cave bears
wifes favorite love otyugh insane number orcs duergar extra kobolds devils frickin awesome seriously well balanced ev
en new game features stand even better previous castle ravenloft game',
 'bought doll  year old boy class preschool teacher perfect children love animals separately also book trio last year
s always favorite hands story preschoolers used one previous school belonged school time bought set',
 'impressed quality puzzle easy fun put together',
 'great resource bhis care coordinators works well kids teens says',
 'opinion best dungeon crawler setup played tore  hours makes fun ride fun kids mine  ',
 'wonderful ball manipulate consists  sided pieces fit together sided ball approximately inches diameter assemblies m
ade',
 'love expand current game new additions completely change half expansion two different expansions',
 'grandson could put wants play',
 'daughter loves little books theyre perfect size keep car diaper bag purse keep hand times stuck waiting doctors off
ice anywhere else',
 'far favorite dd board games fun easy learn even got family play',
 'quick fun easy learn wide age range fast play great family game',
 'great gift wedding party dr themed wedding everyone loved gift reasonably priced much fun',
 'yes quick wonderful accurate',
 'love game playing  years best played  people',
 'loads fun youve played dd boardgames set wrath ashardalon opinion wellrounded among includes gold tokens spend buyi
ng treasures adventures balanced set heroes memorable unique abilities without overpowered make game easy relatable s
etting havent played boardgames dont one standalone game rules board pieces cards characters interchangeable game dun
geon crawl turn explore new room dungeon deal things find monster trap encounter anyall fully cooperative need one pl
ayer dungeon master plays different rules others acts villain even play game solo really wanted ai rules monsters enc
ounter game determined card dont worry making people feel like theyre singled ie game wont break friendships unlike r
isk monopoly approachable dont ever played game dungeons dragons understand even appreciate hand could actually great
way introduce people dd youre inclined rules distilled simplified versions actual tabletop roleplaying game typical r
ound play including setup packup could last anywhere  minutes  hours extreme end',
 'daughter loves stickers awesome seller thank',
 'somuchfun seriously addictive small card set family skeptical first prying away table almost midnight one wants qui
t playing best stocking stuffer gift ever good sense buy']
```

Top 20 positive summaries:

```
['awesome',
 'adds six player option excellent expansion',
 'pigeon perfect addition school library',
 'best boardgame ever',
 'perfect preschoolers',
 'excellent puzzle',
 'perfect',
 'best dungeon crawler',
 'wonderful ball manipulate',
 'awesome expansion',
 'awesome',
 'theyre perfect size keep car diaper',
 'best one series',
 'excellent',
 'perfect gift',
 'wonderful',
 'one best games ever',
 'best among dd boardgames',
 'awesome seller thank',
 'one best games ever']
```

Question 4: What is the impact that each product has on sales?

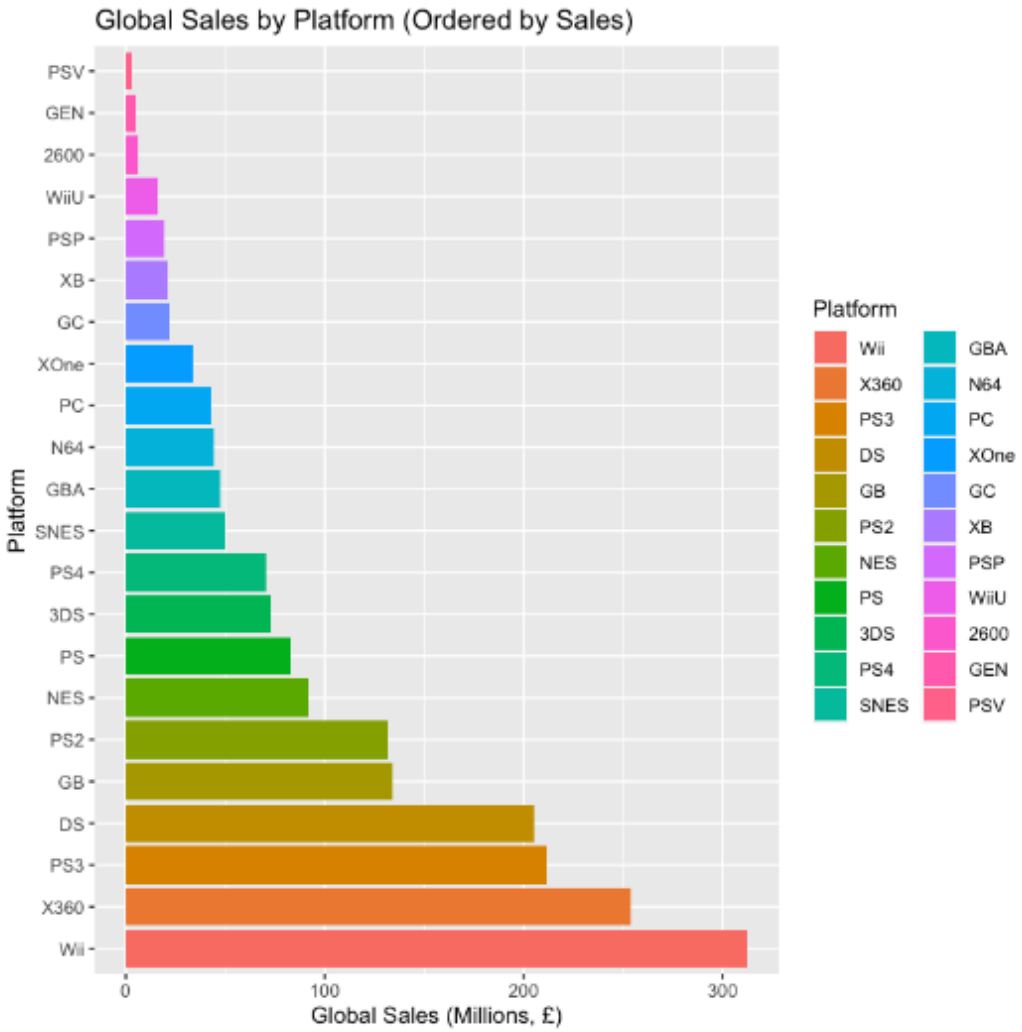
Product-ID Global_Sales
1 107 67.8
2 515 45.9
3 123 37.2
4 254 29.4
5 195 29.4
6 231 27.1
7 249 25.7
8 948 25.4
9 876 25.3
10 263 24.6

Product-ID NA Sales
1 107 34.0
2 123 26.6
3 326 22.1
4 254 21.5
5 515 19.2
6 948 14.4
7 535 13.1
8 195 13
9 231 12.9
10 876 12.8

Product-ID EU Sales
1 107 23.8
2 515 18.9
3 195 10.6
4 3967 10.2
5 2371 9.26
6 876 9.25
7 3645 9.14
8 979 9.07
9 231 9.03
10 399 9.02

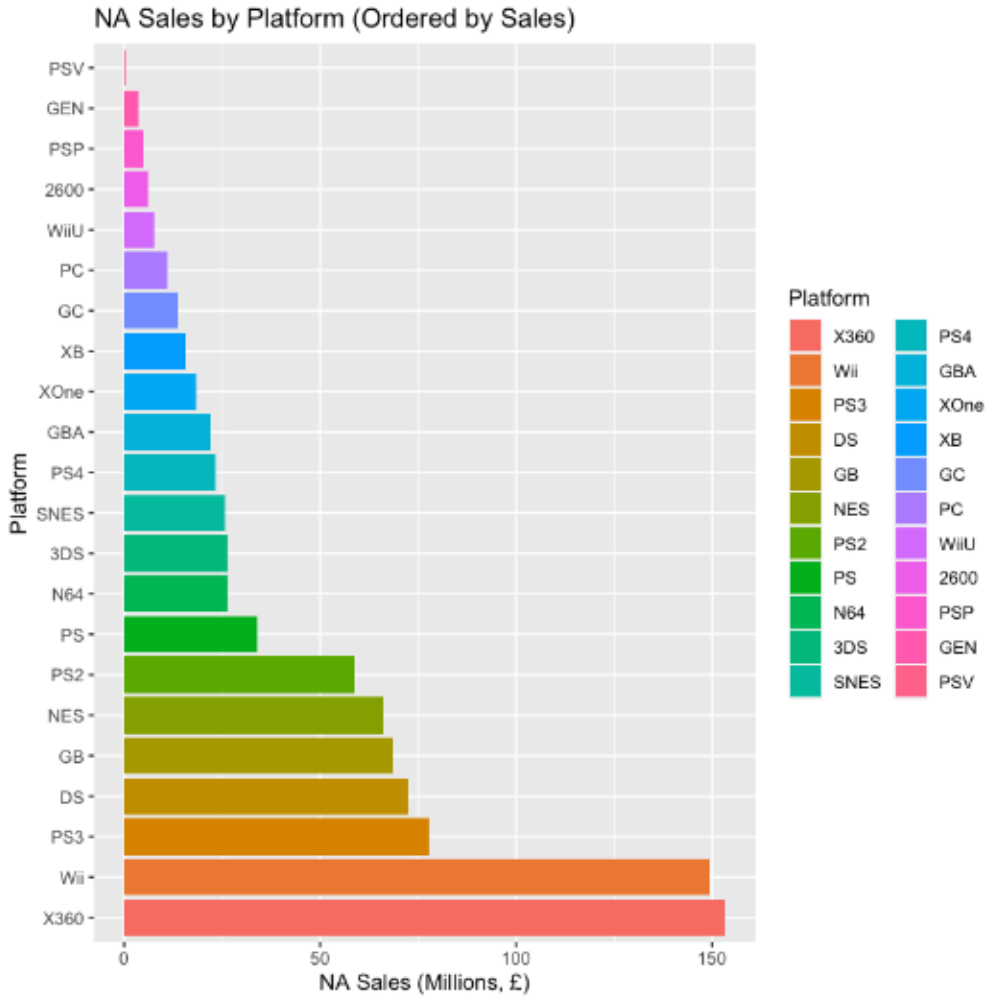
Impact of Sales per Platform:

Platform Global_Sales
1 Wii 313.
2 X360 254.
3 PS3 212.
4 DS 205.
5 GB 134.
6 PS2 132.
7 NES 91.4
8 PS 82.9
9 3DS 73.2
10 PS4 70.5



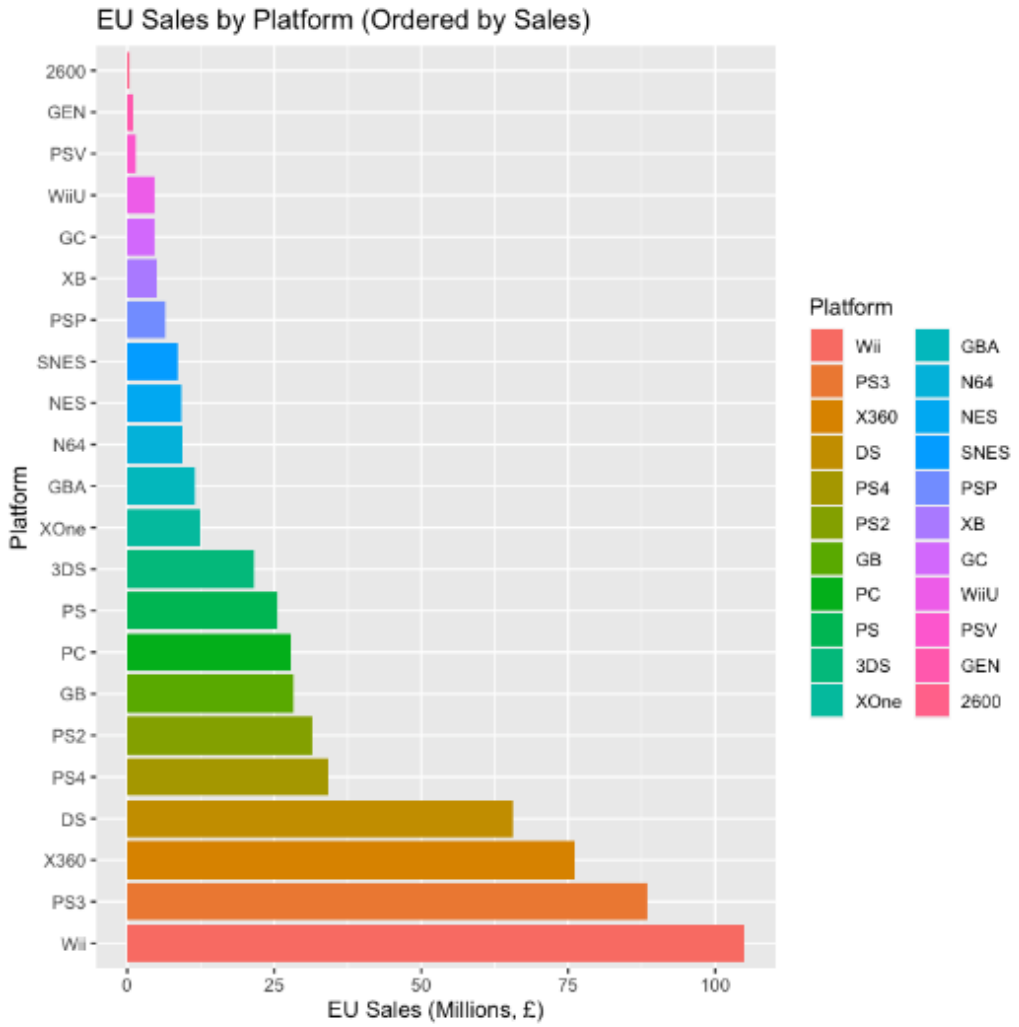
Platform NA_Sales

1 X360 153.
2 Wii 150.
3 PS3 77.8
4 DS 72.6
5 GB 68.7
6 NES 66.0
7 PS2 58.7
8 PS 34.0
9 N64 26.4
10 3DS 26.4

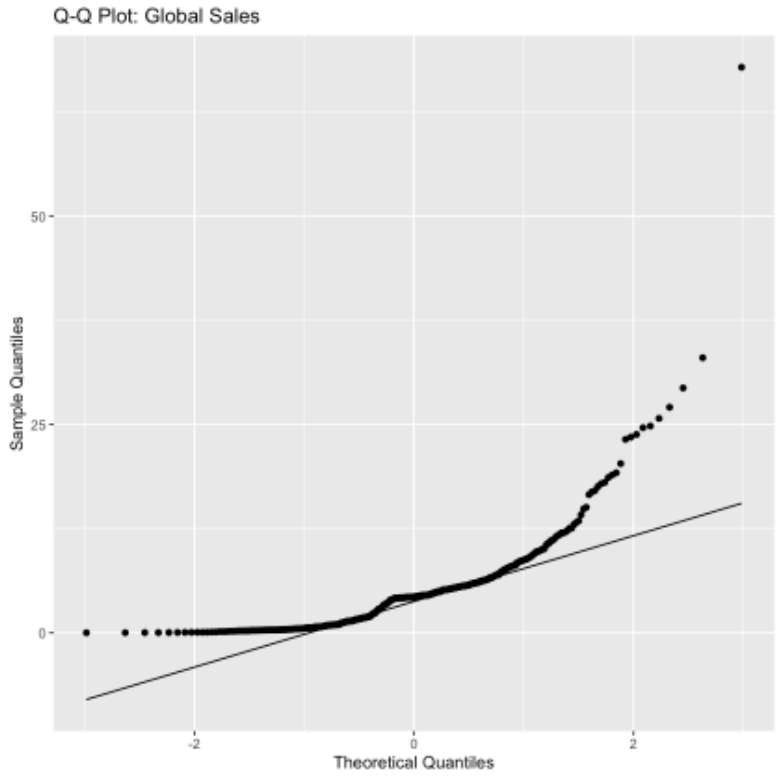


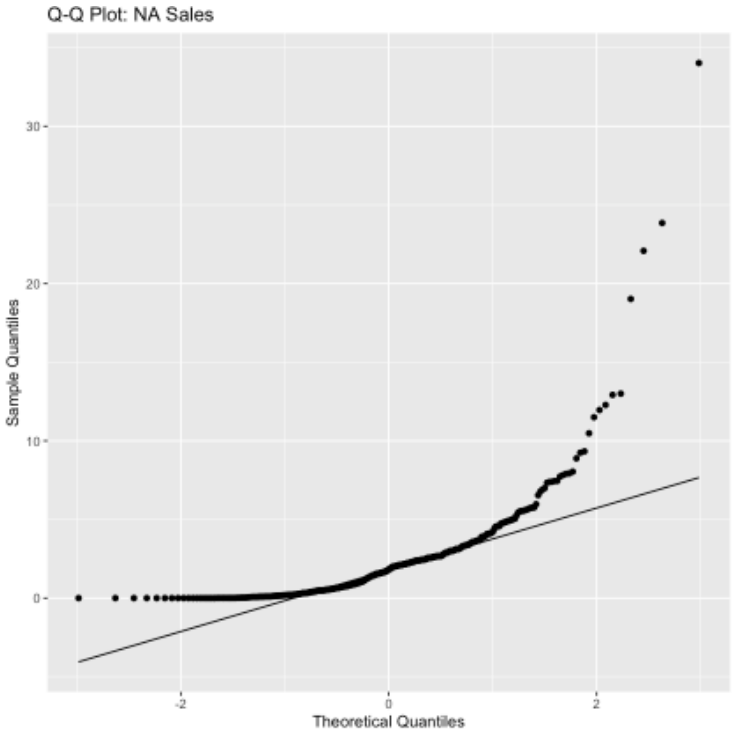
Platform EU_Sales

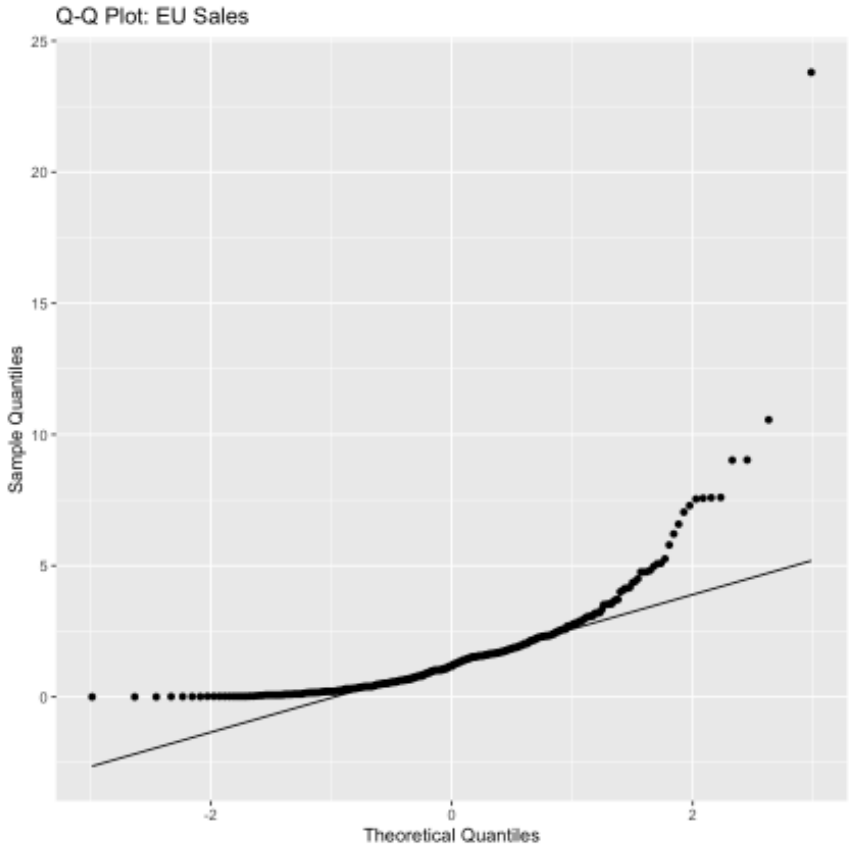
1 Wii 105.
2 PS3 88.5
3 X360 76.0
4 DS 65.6
5 PS4 34.1
6 PS2 31.5
7 GB 28.2
8 PC 27.9
9 PS 25.6
10 3DS 21.6



Question 5: How reliable the data is (e.g. normal distribution, skewness, or kurtosis)?







As we can see, the points for the three Sales columns deviate from the line in an S-shape, this indicates that the data might have heavy tails, indicating non-normality.

Shapiro-Wilk test result:

	Column	p_value	is_normal
NA_Sales	NA_Sales	7.001091e-27	FALSE
EU_Sales	EU_Sales	2.393884e-26	FALSE
Global_Sales	Global_Sales	3.200364e-25	FALSE

Shapiro-Wilk test result:

The p-value for the three sales columns is ≤ 0.05, then we reject the null hypothesis. This indicates that there is significant evidence to suggest that the data deviates from a normal distribution. In this case, the data is not normally distributed.

	Column	Skewness	Kurtosis
NA_Sales	NA_Sales	4.309210	31.36852
EU_Sales	EU_Sales	4.818688	44.68924
Global_Sales	Global_Sales	4.045582	32.63966

Interpreting the Skewness and Kurtosis values:

Skewness:

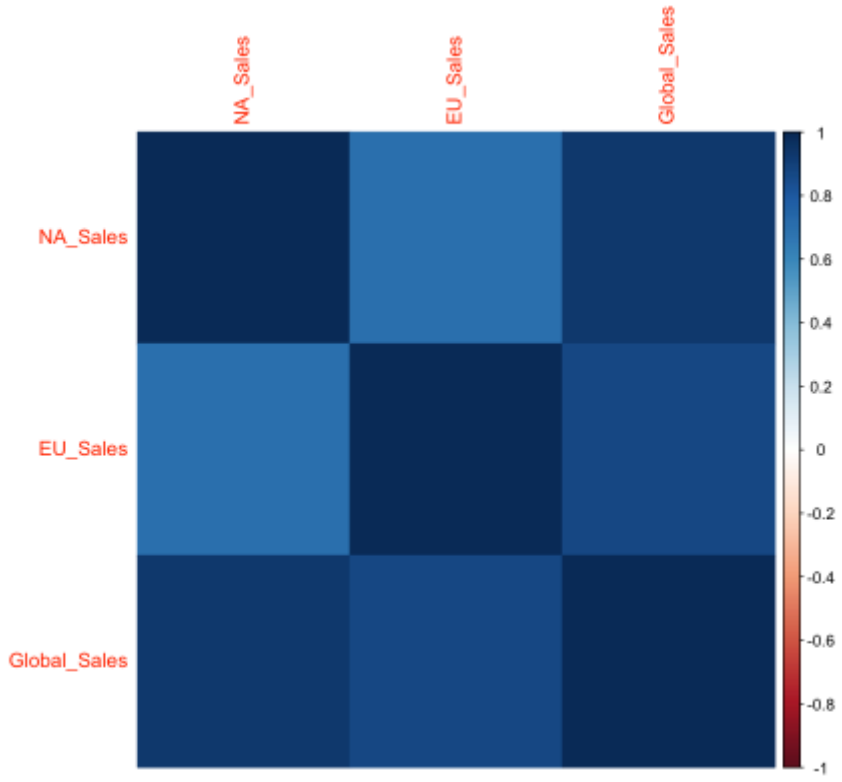
The skewness values for the three sales columns is positive (greater than 0), meaning that the distribution is skewed to the right, in other words, the tail on the right side is longer than the left side.

Kurtosis:

The Kurtosis values for the sales columns are greater than 3 (positive kurtosis), meaning that the distribution has heavier tails and a sharper peak compared to the normal distribution.

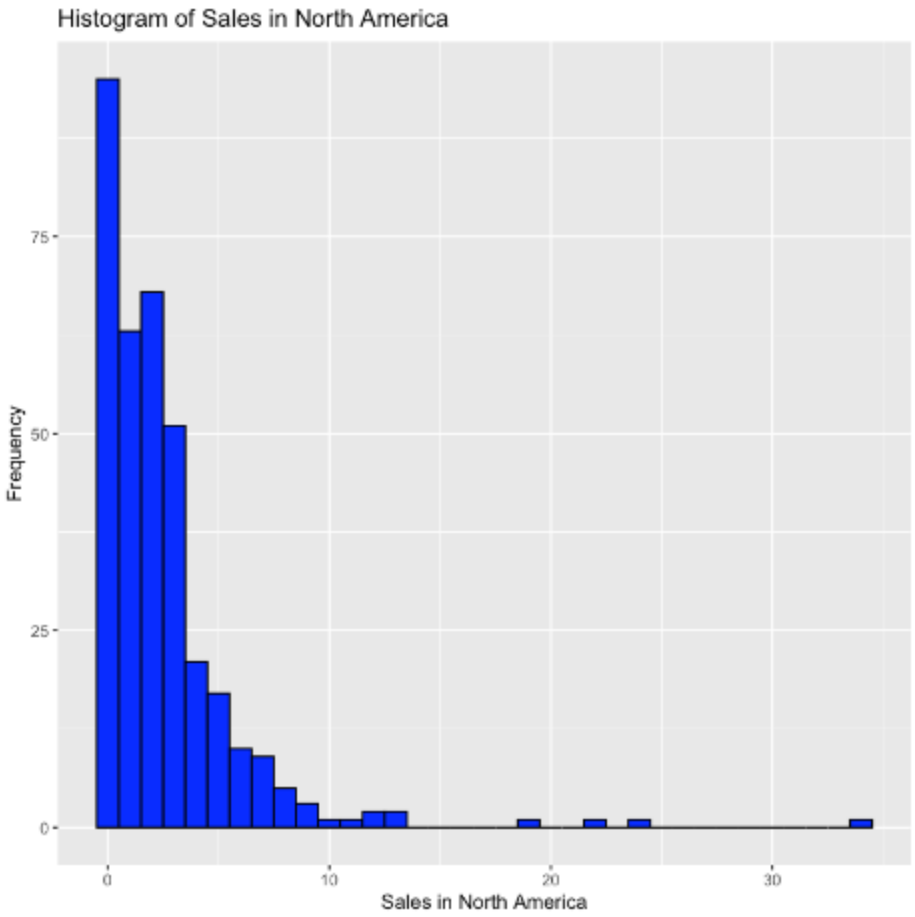
Correlation between the three Sales groups:

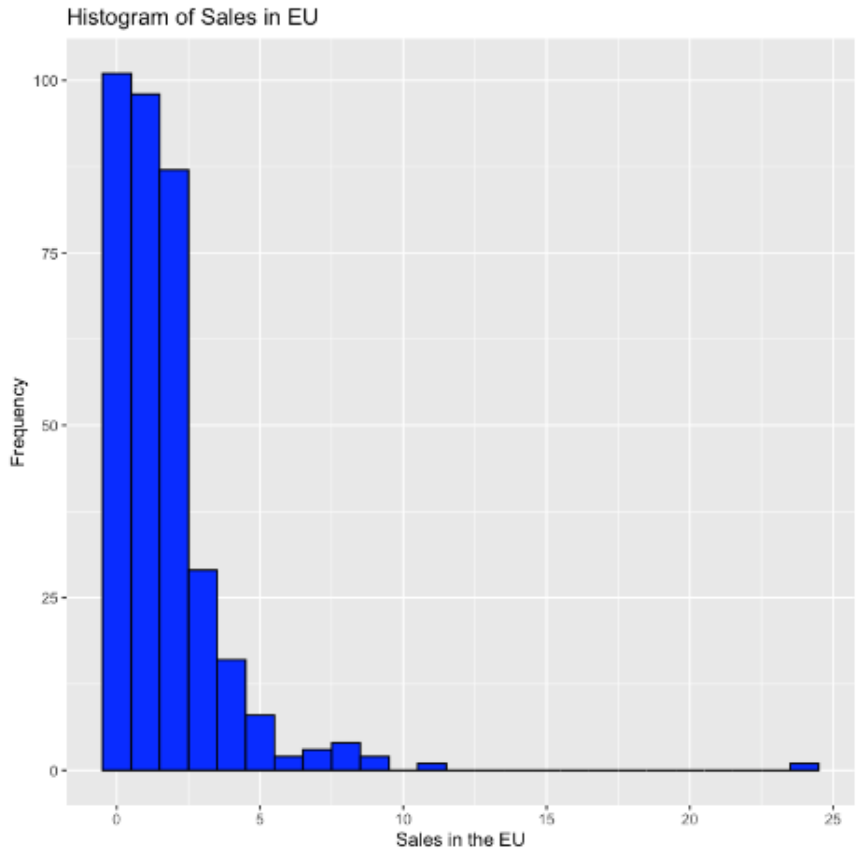
	NA_Sales	EU_Sales	Global_Sales
NA_Sales	1.0000000	0.7055236	0.9349455
EU_Sales	0.7055236	1.0000000	0.8775575
Global_Sales	0.9349455	0.8775575	1.0000000

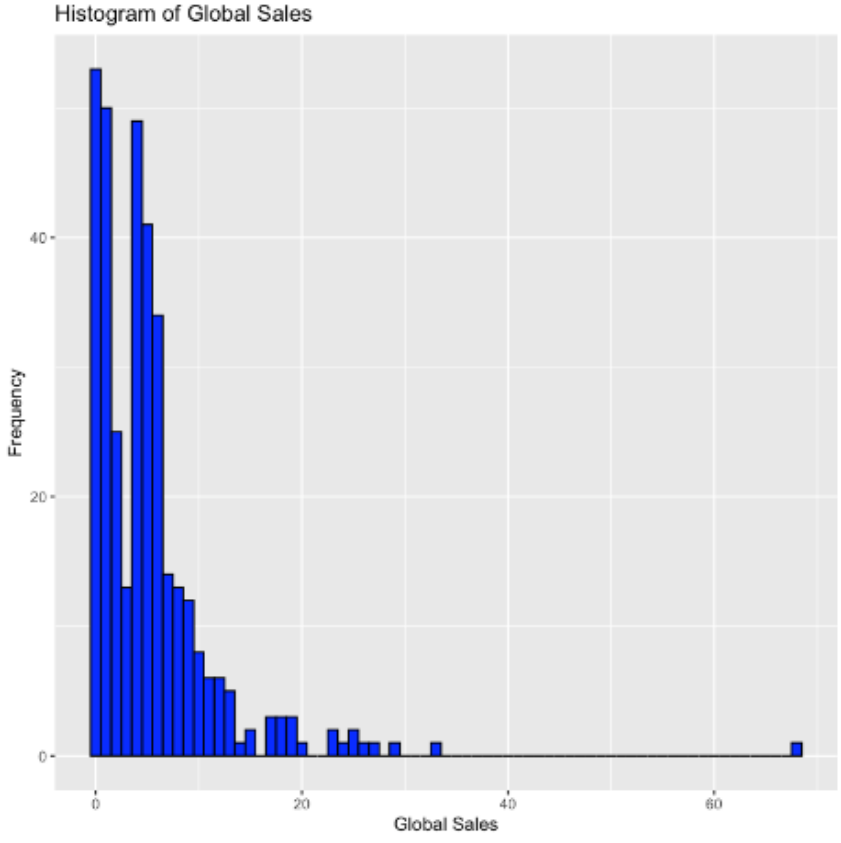


As expected, there is a positive correlation between the global sales and the NA sales and EU sales, since global sales in the world is the sum of EU sales, NA sales and other sales.

For future analysis, it might worth digging into the 'other sales' category and calculate the sales under this uncategorised group. Turtle Games should properly collect and label the 'other sales' category to identify the market / markets that belong to this category. This would be very useful to inform future marketing campaigns too since we would be able to see socio-economic and demographic profile of the customers under this category.




Histogram of Sales in EU
Frequency
100
75
50
25
0
0
5
10
15
20
25
Sales in the EU

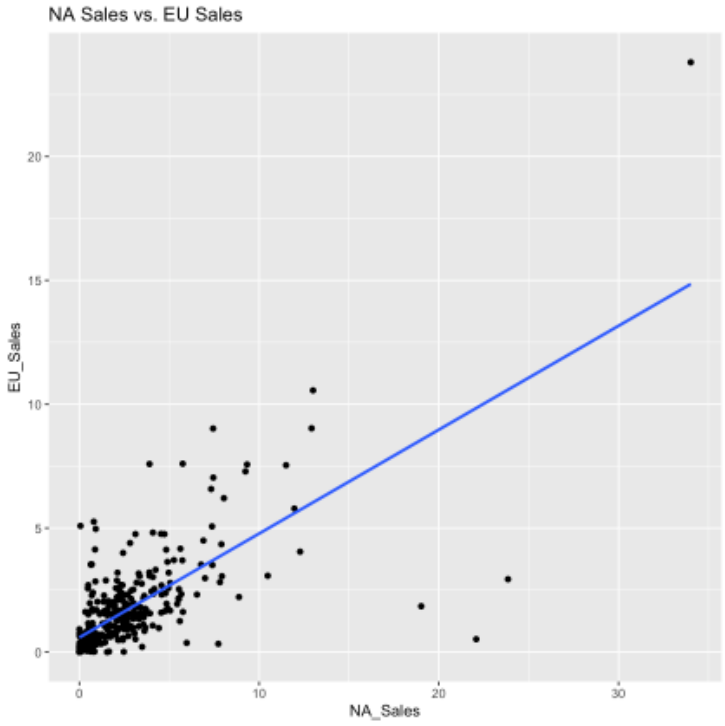

Histogram of Global Sales
Frequency
40
20
0
0
20
40
60
Global Sales

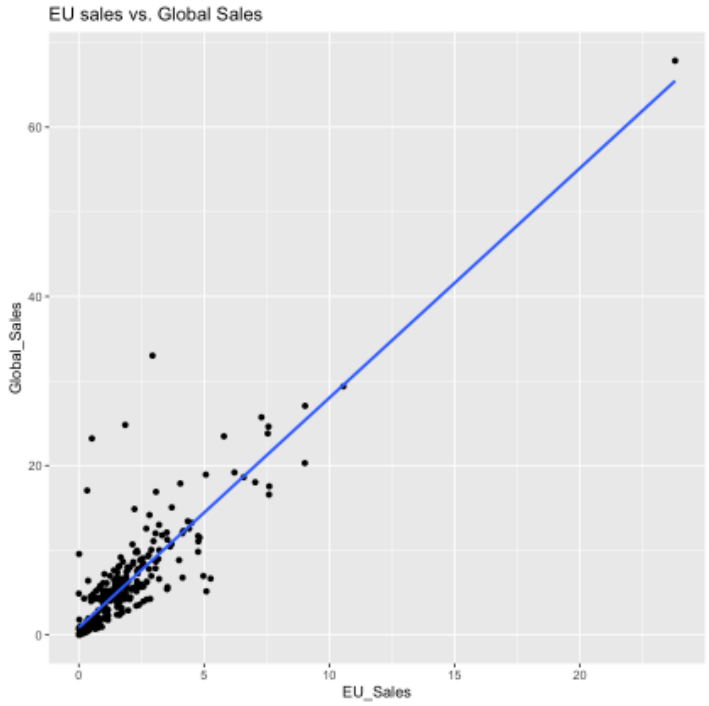
After looking at the histograms, we can see that the width of the bars for Global Sales is narrower than the width of the bars for NA and EU. This might be an indication of the different range of products sold at a global scale. In other words, we expect to see more variety of products sold when we look at Global Sales than when we look at NA or EU individually.

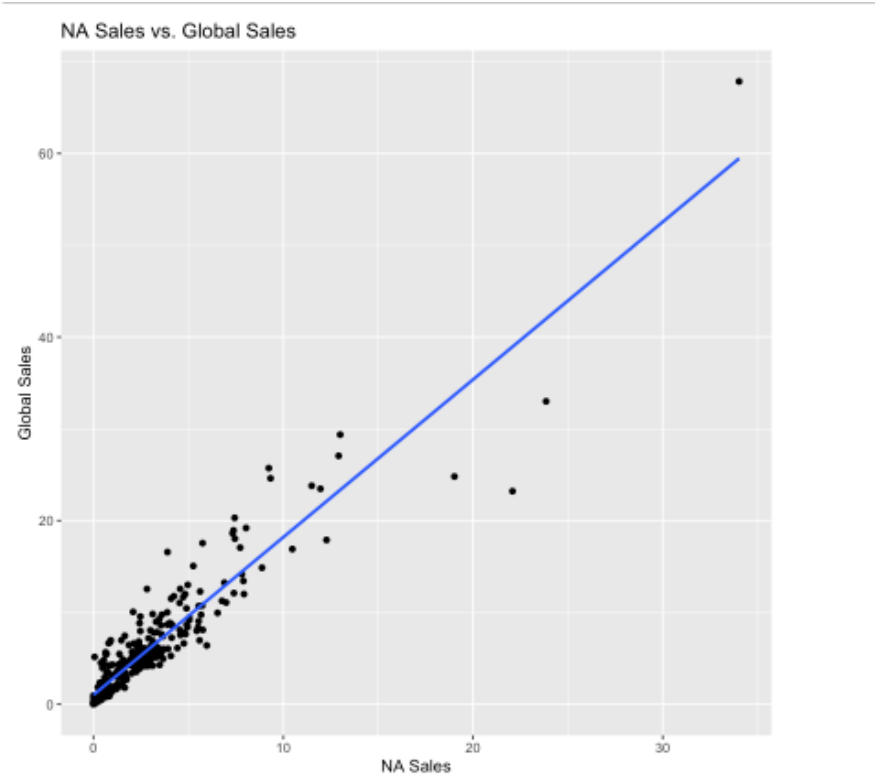
The length of the bars corresponds to the number of sales. The highest sales (bars) might be a sign of those products that influence sales the most in each market.

Question 6: What the relationship(s) is/are (if any) between North American, European, and global sales?

NA Sales are greater than EU sales.







Overall, we can see that the global sales are much greater than the EU and NA sales values, and that makes sense.

We can also see how there are some global sales values that deviate more from the normal distribution, meaning that these outliers are the greatest sales values observed within the global sales column, such as the black dot shown way over the 60M mark for Global Sales.

I managed to calculate the predicted global sales values, however, due to time constraints I could not investigate further to compare these predicted global sales values with the current global sales values to evaluate the performance of the model and assess how well the model fits the data.

I also wanted to plot a scatter plot showing the predicted global sales versus the actual global sales from our current data set, but due to some errors in my code, I could not continue this analysis further.

Appendix

Appendix 1

- **Five Whys framework approach:**

Q1: Why Turtle Games are not seeing better overall sales performance?

A1: Because they do not have a clear understanding of their overall sales performance nor their markets or segments within their customer base.

Q2: Why Turtle Games do not have a clear understanding of their overall sales performance nor their customer base?

A2: Because they do not have an analytical approach of their business in place.

Q3: Why Turtle Games do not have an analytical approach of their business in place?

A3: Because, historically, they never thought of the importance of their first party data and how they could leverage these data sets to improve business efficiencies.

Q4: Why Turtle Games could leverage these data sets to improve business efficiencies?

A4: Because they already have half of the job done, which is collecting their sales and customers data. They just need to keep updating these data sets periodically to enable regular quality data analysis.

Q5: Why Turtle Games should be performing regular data analysis?

A5: Because they could inform their future marketing and investment decisions based on the patterns and trends observed in their historical sales data and the online reviews submitted by customers who purchased and used their products.

Appendix 2

- **How customer reviews can be used to inform marketing campaigns?**

1. **Understand Customer Sentiments:** Positive sentiments can be leveraged to reinforce positive aspects of the brand, while negative sentiments can highlight areas that need improvement.

2. **Discover Customer Preferences:** Commonly used words related to product features or benefits can be used by marketers to tailor marketing campaigns that highlight these specific attributes.

3. **Uncover Competitor Insights:** If reviews mention competitors or similar products, marketers can gain insights into how customers compare the company's offerings with others in the market. This information can help refine the company's positioning and competitive strategy.

4. **Find Opportunities for Improvement:** Word clouds can reveal issues or pain points frequently mentioned by customers. Addressing these concerns can lead to improvements in products or services, enhancing customer satisfaction and loyalty.

5. **Support Content Marketing:** Word clouds and commonly used positive words can be incorporated into taglines, advertisements, and promotional content to resonate with the target audience.

6. **Customise Brand Message:** The language used by customers in reviews can be used by marketers to adapt their brand language to align better with their customer’s communication style.

Appendix 3

- **To explore further in the future:**

a. Which markets constitute ‘Other Sales’ and what is the % share of ‘Other Sales’ in relation to ‘Global Sales’, ‘NA Sales’ and ‘EU Sales’.

b. Demographic and socioeconomic profiles of those customers that bought games outside of NA and the EU regions, currently categorised as ‘Other Sales’ by Turtle Games.

c. Explore other social media platforms where Turtle Games has an official business account, (if any), with the aim of collecting and analysing customers reviews.

d. I noticed that there are only video games in the sales data set, however the reviews dataset contains reviews from across the entire range of products that Turtle Games sells (board games, video games, books and toys).

Predicting Future Outcomes

Turtle Games

Data Analyst & Project lead

Cristina Alonso Robles

aroblescristina@gmail.com

https://www.linkedin.com/in/cristina-alonso-robles/

https://github.com/crialorob

July 2023