Report: Bank Data Analysis

1. Data Preparation

1.1. Check for data consistency and ensure all columns are formatted

I have initialised a Spark session and stored the Wikipedia page with the dataset into a variable called “url”. The Spark session allows creation of DataFrames, running SQL queries and manage the application.

```
# Initialize Spark session
spark = SparkSession.builder \
    .appName("BankingDataAnalysis") \
    .getOrCreate()
# URL of the Wikipedia page
url = "https://web.archive.org/web/20230908091635/https://en.wikipedia.org/wiki/List_of_largest_banks"
```

I also loaded the exchange rate dataset by mounting my Google drive and reading the dataset using *spark.read.csv()*. The “header=True” attribute tells Spark that the first row contains the names of the columns and the “inferSchema=True” attribute automatically detects data types that are present in the dataset. We see that this dataset has the exchange rates of 3 different currencies - Euros, Great Britain Pounds and Indian Rupees.

```
# Load the exchange rate data
from google.colab import drive
drive.mount('/content/drive')
exchange_rate_df = spark.read.csv("/content/drive/MyDrive/Assignment 2/exchange_rate.csv", header=True, inferSchema=True)

# Show the first few rows of the exchange rate data to verify
exchange_rate_df.show()

log_progress("Exchange Rate Data Loaded")
```

```
+--------+-----+
|Currency| Rate|
+--------+-----+
|     EUR| 0.93|
|     GBP|  0.8|
|     INR|82.95|
+--------+-----+
```

1.2. Apply sampling techniques if needed to extract a representative subset for analysis

Then, I used Pandas library to read all the HTML tables present in the page. There were 4 tables scrapped from Wikipedia and then stored into 4 different variables, namely df0, df1, df2 and df3. I loaded the 2nd table which in index form

is the "1st" table. From its output, we see that Table 1 has 10 rows and 3 columns present.

```
# Use pandas to read all HTML tables from the page
tables = pd.read_html(url)
print(f"Total number of tables scraped: {len(tables)}")
df0 = tables[0]
df1 = tables[1]
df2 = tables[2]
df3 = tables[3]

# Load the correct table (Table 1)
df = tables[1]
print(f'The Table[1] Data Frame has {df.shape[0]} rows and {df.shape[1]} columns')
```

```
Total number of tables scraped: 4
The Table[1] Data Frame has 10 rows and 3 columns
```

1.3. Structure and prepare the data for further processing and analysis

Next, I converted the Pandas DataFrame into PySpark DataFrame. A Pandas DataFrame works on a single machine and is very fast for local operation but may slow down or crash when the dataset is too large. A PySpark DataFrame, on the other hand, works on multiple machines and does not require all data to fit into memory which is suitable for large/big data. I then renamed the column names to "Rank", "Bank_name" and "Market_Cap_USD_billion" with the help of the *.withColumn()* function, so that it would be easier to understand the data. As shown in the output below the code, we see that there are top 10 banks and are arranged in descending order.

```
# Convert pandas DataFrame to PySpark DataFrame
spark_df = spark.createDataFrame(df)

# Rename columns
spark_df = spark_df.withColumnRenamed("Rank", "Rank") \
                   .withColumnRenamed("Bank name", "Bank_name") \
                   .withColumnRenamed("Market cap (US$ billion)", "Market_Cap_USD_billion")

# Show the first few rows of the PySpark DataFrame
spark_df.show()
```

```
+----+--------------------+----------------------+
|Rank|           Bank_name|Market_Cap_USD_billion|
+----+--------------------+----------------------+
|   1|      JPMorgan Chase|                432.92|
|   2|     Bank of America|                231.52|
|   3|Industrial and Co...|                194.56|
|   4|Agricultural Bank...|                160.68|
|   5|           HDFC Bank|                157.91|
|   6|         Wells Fargo|                155.87|
|   7|   HSBC Holdings PLC|                 148.9|
|   8|      Morgan Stanley|                140.83|
|   9|China Constructio...|                139.82|
|  10|       Bank of China|                136.81|
+----+--------------------+----------------------+
```

The *.printSchema()* function in the code below prints the structure of the DataFrame and shows each column's names and its data type in a tree format.

```
# Print the schema of the DataFrame
spark_df.printSchema()
```

```
root
 |-- Rank: long (nullable = true)
 |-- Bank_name: string (nullable = true)
 |-- Market_Cap_USD_billion: double (nullable = true)
```

Finally, I configured the logging function to record the progress of the code at different stages. The function takes a text message and logs it with a timestamp. A sample of the code is shown below in the output where the function *"log_progress()"* is called and shows the date, time and a message being recorded - "Data Loaded and Processed".

```
# Configure logging
import logging
from datetime import datetime

def log_progress(message):
    """Logs the progress of the script."""
    timeformat = '%Y-%h-%d-%H:%M:%S'
    timestamp = datetime.now().strftime(timeformat)
    log_message = f"{timestamp} : {message}"
    print(log_message)
    with open("code_log.txt", 'a') as f:
        f.write(log_message + '\n')
log_progress("Data Loaded and Processed")
```

```
2025-Nov-25-05:57:11 : Data Loaded and Processed
```

2. Data Cleaning

2.1. Handling Missing Values

2.1.1. Print schema to check data types

This step was done to check that Spark has correctly identified data types of all columns. This helps prevent errors in the future with other transformations and accurate missing-value handling.

```
# Print the schema to check data types
spark_df.printSchema()
```

```
root
 |-- Rank: long (nullable = true)
 |-- Bank_name: string (nullable = true)
 |-- Market_Cap_USD_billion: double (nullable = true)
```

The output above shows that “Rank” was correctly detected as a long integer, “Bank_name” was detected as a string and “Market_Cap_USD_billion” was detected as double. Hence, suggesting that there were no type-conversion issues.

2.1.2. Check for missing values

The code below helps check for missing values in the dataset. The steps followed are as such:

1) The *isnull()* function checks if “c” is NULL.
2) The *when()* function is used for each row where the column value will be returned, else it will return None.
3) The *count()* function will count how many times the condition above has returned something.

```
# Check for missing values
spark_df.select([count(when(isnull(c), c)).alias(c) for c in spark_df.columns]).show()
```

```
+----+---------+---------------------+
|Rank|Bank_name|Market_Cap_USD_billion|
+----+---------+---------------------+
|   0|        0|                    0|
+----+---------+---------------------+
```

As the output above shows, there are no missing values in the dataset.

2.1.3. Drop rows with missing values

The code below is used to drop any rows that have missing values in them. The steps taken to achieve the same are below:

1) The for loop is used to iterate over each column name in the “columns” list.
2) The *.filter()* function returns a new DataFrame which will contain rows only where the expression is true. The *.isNull()* function will create a boolean expression that is *true* when the column is NULL. The *.count()* function is an action that launches a job within the loop to count the number of missing values present.
3) A new DataFrame called “spark_df_cleaned” by dropping rows that contain NULL values. The *.dropna()* function will drop any row that has any NULL in any column (“any” here is a default).
4) Finally, I have displayed the output of the number of rows before and after dropping missing values using the *print()* function.

```
# Drop rows with missing values
columns = spark_df.columns
for column_name in columns:
    null_count = spark_df.filter(col(column_name).isNull()).count()
    print(f"Column '{column_name}': {null_count} missing values")
log_progress("Counted Null values in each column")
spark_df_cleaned = spark_df.dropna()
# Show the number of rows before and after dropping
print(f"Number of rows before dropping missing values: {spark_df.count()}")
print(f"Number of rows after dropping missing values: {spark_df_cleaned.count()}")

log_progress("Dropped rows with missing values")
```

```
Column 'Rank': 0 missing values
Column 'Bank_name': 0 missing values
Column 'Market_Cap_USD_billion': 0 missing values
2025-Nov-22-18:25:19 : Counted Null values in each column
Number of rows before dropping missing values: 10
Number of rows after dropping missing values: 10
2025-Nov-22-18:25:20 : Dropped rows with missing values
```

The output above shows that there are no missing values present in the dataset. Hence, after “dropping”, we see that there are no changes made to the rows.

2.2. Fixing Columns

The code below counts the total number of rows and checks if there are any duplicates. The steps done are as shown below:

1) The “total_rows” variable will run a spark action to compute the total number of rows in the cleaned DataFrame, which stores an integer value and is logged using the user-defined log_progress() function.
2) The *.groupBy()* function will group the DataFrame by all columns with the *.count()* function counting the number of rows per group and only keeping groups where count is more than 1, using the *.filter()* function. The overall result will be stored in the “duplicate_rows” variable.
3) The “num_duplicates” variable stores the count of how many duplicate groups were found, by executing a Spark action.
4) I have used an if-else condition where if the count of duplicate groups is more than 0, an output will be printed saying that duplicate rows have been found. Else (if no duplicates are present in the dataset), it will print “no duplicate rows found”.

```
# Count the total number of rows
total_rows = spark_df_cleaned.count()
print(f"Total number of rows: {total_rows}")
log_progress(f"Total number of rows: {total_rows}")

# Check if there are duplicates
duplicate_rows = spark_df_cleaned.groupBy(spark_df_cleaned.columns).count().filter("count > 1")
num_duplicates = duplicate_rows.count()
print(f"Number of duplicate rows: {num_duplicates}")
log_progress(f"Number of duplicate rows: {num_duplicates}")
if num_duplicates > 0:
    print("Duplicate rows found:")
    duplicate_rows.show()
else:
    print("No duplicate rows found.")
```

```
Total number of rows: 10
2025-Nov-22-17:53:19 : Total number of rows: 10
Number of duplicate rows: 0
2025-Nov-22-17:53:20 : Number of duplicate rows: 0
No duplicate rows found.
```

The output of the code shows that there are no duplicate rows found within the dataset. The same has been logged using the user-defined logging function.

2.3. Handling Outliers

The code below is used to detect and analyse any outliers that are present in the dataset. As the “Market_Cap_USD_billion” column is already numeric, there is no conversion required. I have done the following steps to produce the relevant output.

1) Firstly, I calculated the 1st and 3rd quartiles, 25% and 75% respectively, using the .approxQuantile() function. These values are the basis to find outliers.
2) Next, I calculated the Inter Quartile Range (IQR) by subtracting the values in the previous step. This is done so that I can measure the spread of the data present. Based on the IQR, I have calculated both boundaries - lower bound and upper bound, using the following formulas respectively:
   lower_bound = q1 - 1.5 * iqr
   upper_bound = q3 + 1.5 * iqr
   Any value outside this range is considered to be an *outlier*.
3) Finally, I have used the *.filter()* function with conditions that will compare with the lower and upper bounds, where all the rows in Market Cap that are below the lower bound or above the upper bound are considered to be outliers. Hence, I have filtered the dataset to find the potential outliers present.
4) I have displayed the values of Q1, Q3, IQR, Lower and Upper boundaries and the potential outlier based on Market Cap, and have logged the action using the log_progress() function

```
# Write code for outlier analysis
q1 = spark_df_cleaned.approxQuantile("Market_Cap_USD_billion", [0.25], 0.0)[0]
q3 = spark_df_cleaned.approxQuantile("Market_Cap_USD_billion", [0.75], 0.0)[0]
iqr = q3 - q1
lower_bound = q1 - 1.5 * iqr
upper_bound = q3 + 1.5 * iqr
outliers = spark_df_cleaned.filter(
    (col("Market_Cap_USD_billion") < lower_bound) |
    (col("Market_Cap_USD_billion") > upper_bound)
)
print(f"Q1 (25th percentile): {q1}")
print(f"Q3 (75th percentile): {q3}")
print(f"IQR (Interquartile Range): {iqr}")
print(f"Lower Bound for Outliers: {lower_bound}")
print(f"Upper Bound for Outliers: {upper_bound}")
print("\nPotential Outliers based on Market Cap (US$ Billion):")
outliers.show()

log_progress("Outlier analysis performed")
```

```
Q1 (25th percentile): 140.83
Q3 (75th percentile): 194.56
IQR (Interquartile Range): 53.72999999999999
Lower Bound for Outliers: 60.23500000000003
Upper Bound for Outliers: 275.155

Potential Outliers based on Market Cap (US$ Billion):
+----+--------------+----------------------+
|Rank|     Bank_name|Market_Cap_USD_billion|
+----+--------------+----------------------+
|   1|JPMorgan Chase|                432.92|
+----+--------------+----------------------+

2025-Nov-22-17:53:31 : Outlier analysis performed
```

Based on the output above, we see that the 25th and 75th percentiles are 140.83 and 194.56 respectively, and the lower and upper bounds are 60.235 and 275.155 respectively. This means that any value within the Market Cap column is below or above the boundaries is considered to be an outlier.
A potential outlier that was filtered is JPMorgan Chase with its Market Cap value being 432.92.

3. EDA: Finding Patterns

3.1. Conversion from PySpark to Pandas DataFrame

The code below converts the PySpark DataFrame to a Pandas DataFrame using the *.toPandas()* function available. I have also logged the progress of it using the defined log_progress() function and have displayed the first 5 rows of the converted DataFrame using the *.head()* function, as well as using the *.info()* function to display information about the columns in the dataset.

```
# Convert PySpark DataFrame to Pandas DataFrame
pandas_df = spark_df_cleaned.toPandas()

log_progress("Converted PySpark DataFrame to Pandas DataFrame")
display(pandas_df.head())
pandas_df.info()
```

```
2025-Nov-24-03:51:03 : Converted PySpark DataFrame to Pandas DataFrame
```

	Rank	Bank_name	Market_Cap_USD_billion
0	1	JPMorgan Chase	432.92
1	2	Bank of America	231.52
2	3	Industrial and Commercial Bank of China	194.56
3	4	Agricultural Bank of China	160.68
4	5	HDFC Bank	157.91

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 3 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Rank                   10 non-null     int64
 1   Bank_name              10 non-null     object
 2   Market_Cap_USD_billion  10 non-null     float64
dtypes: float64(1), int64(1), object(1)
memory usage: 372.0+ bytes
```

The screenshot above displays the output for the first 5 rows of the converted dataset and information on the columns.
From the information on the columns, we see that there are 10 non-null rows for each column within the dataset and they are of “int”, “floa”t and “object” datatypes - which are easy to work with in Python using pandas.

3.2. Market Capitalisation

The code below is used to get the distribution of Market Cap using a histogram. I have done the following steps to get the output.

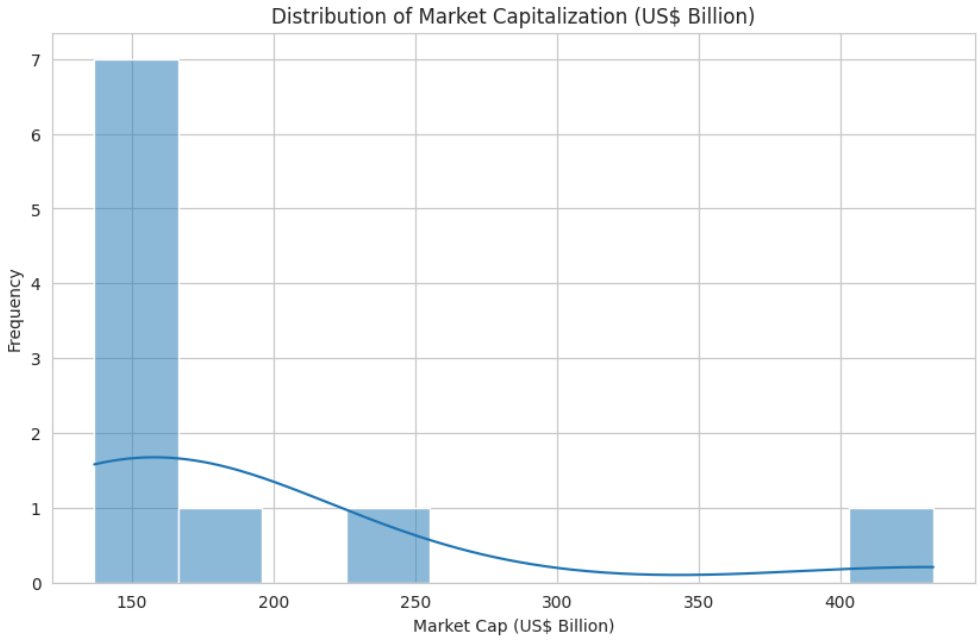
1) Firstly, I have set the style of the plot to “whitegrid” so that the background is white and has light gridlines which makes the visualisation easy to read.
2) Then, I have created a canvas of size 10 x 6 using figsize attribute.
3) Using the *.histplot()* function, I have created a histogram with the data from “pandas_df” and bins being 10 so that the data is split equally between the 10 intervals. I have also set the *kde* (Kernel Density Estimate) attribute to True - a line that estimates the probability distribution of the data.

```
# Distribution of Market Cap (US$ Billion)

# Set the style for seaborn
sns.set_style("whitegrid")

# Plot the distribution of market cap
plt.figure(figsize=(10, 6))
sns.histplot(pandas_df['Market_Cap_USD_billion'], kde=True, bins=10)
plt.title('Distribution of Market Capitalization (US$ Billion)')
plt.xlabel('Market Cap (US$ Billion)')
plt.ylabel('Frequency')
plt.show()

log_progress("Plotted distribution of Market Cap")
```



From the output above, we see that most companies fall within the US$150 billion range. This shows a right-skewed distribution as most companies have a small market cap while few companies have a very large market cap.
The KDE curve, which shows the probability distribution of the data, peaks around US$150-160 billion as most of the values are there.

3.3. Top 10 banks

The code below identifies the top 10 banks with reference to market capitalisation of them. I have followed the steps below.

1) I have sorted the dataset with banks in descending order such that the highest market capital will be displayed on top and so on. I have also used the .head(10) function to display the 10 rows - which means that the top 10 banks will be displayed.

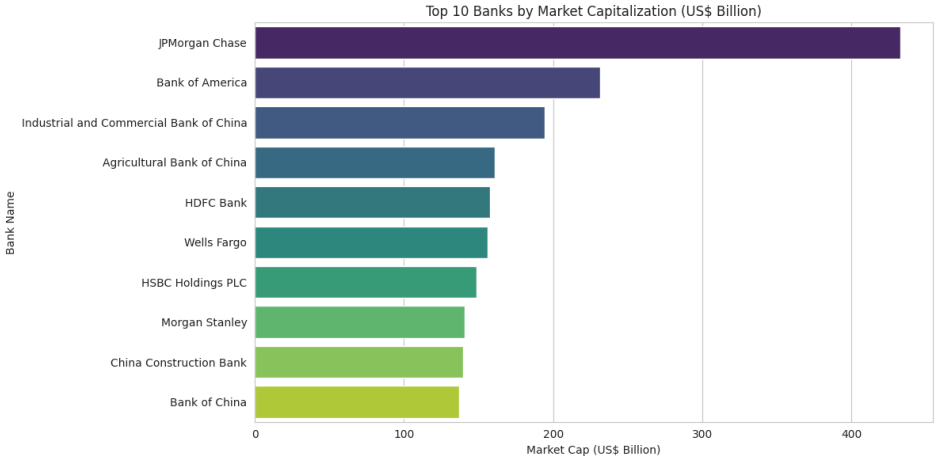
2) I have then plotted the above data into a barplot with the x-axis being Market capitalisation and y-axis being the name of the banks. The legend attribute is set to false to avoid any duplicate labels.
3) The .tight_layout() function was used to fix spacing issues so that the labels (name of the banks) do not get cut off.

```
# Top 10 Banks by Market Cap

# Sort the DataFrame by market cap in descending order
top_10_banks = pandas_df.sort_values(by='Market_Cap_USD_billion', ascending=False).head(10)

# Plot the top 10 banks by market cap
plt.figure(figsize=(12, 6))
sns.barplot(x='Market_Cap_USD_billion', y='Bank_name', data=top_10_banks, palette='viridis', hue='Bank_name', legend=False)
plt.title('Top 10 Banks by Market Capitalization (US$ Billion)')
plt.xlabel('Market Cap (US$ Billion)')
plt.ylabel('Bank Name')
plt.tight_layout()
plt.show()

log_progress("Plotted Top 10 Banks by Market Cap")
```



Based on the output above, we see that JPMorgan Chase has the highest market capital, almost twice at ~US$430 billion, compared to the other banks. This suggests that JPMorgan Chase holds dominance in global banking. Most of the banks fall around the US$140 to US$180 billion range. This could suggest that there is a competition amongst them for improvements in their ranking. It also shows that the market concentration is very high at the top with JPMorgan Chase & Bank of America leading.

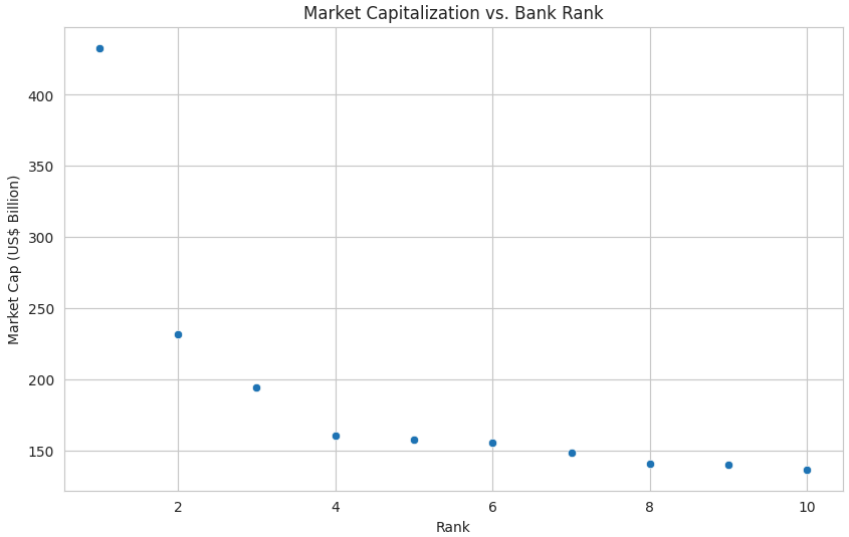
3.4. Market Cap vs Bank Ranking

The code below is used to compare the relationship between market capitalisation and bank ranking.
Firstly, I have created a canvas size of 10 x 6 such that all the data points and labels will be clearly visible. Then, I have used Seaborn to create a scatterplot, with the *.scatterplot()* function with x-axis being “Rank” and y-axis being “Market Capitalisation”. The scatterplot has been displayed using the *.show()* function.

```
# Market Cap vs Rank

# Plot market cap vs rank
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Rank', y='Market_Cap_USD_billion', data=pandas_df)
plt.title('Market Capitalization vs. Bank Rank')
plt.xlabel('Rank')
plt.ylabel('Market Cap (US$ Billion)')
plt.show()

log_progress("Plotted Market Cap vs Rank")
```



From the above output, we can see that the Market Capitalisation of US$400 billion is at rank 1 and is most likely JPMorgan Chase, as seen in the previous section. That single/outlier point at the top shows the bank's extreme market concentration. We also see that there is a sharp drop from Rank 1 to Rank 2, with the bank at this rank having a market capitalisation at ~US$240 billion. Continuing on, the market capitalisation steadily decreases, showing a smooth downward trend. Based on this, we can interpret that there is a clear negative correlation between the two variables where as the rank increases, the market capitalisation decreases.

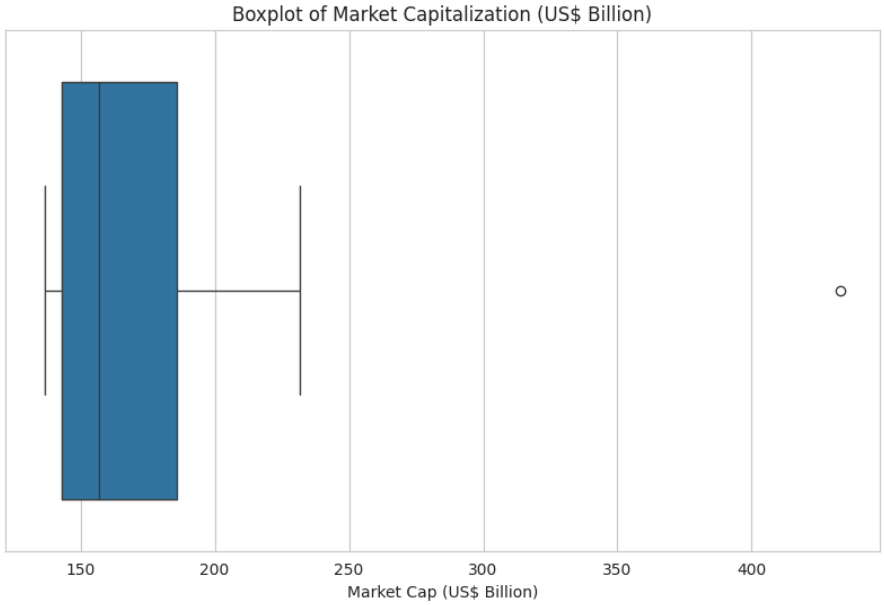
3.5. Market Cap Analysis

I have used the *.boxplot()* function in the Seaborn package to plot a boxplot to show the spread and outliers present in the dataset.

```
# Boxplot of Market Cap
#To show the spread and outliers in the market capitalization data.

# Plot a boxplot of market cap
plt.figure(figsize=(10, 6))
sns.boxplot(x=pandas_df['Market_Cap_USD_billion'])
plt.title('Boxplot of Market Capitalization (US$ Billion)')
plt.xlabel('Market Cap (US$ Billion)')
plt.show()

log_progress("Plotted Boxplot of Market Cap")
```



From the above output, we see that the median value of the dataset is ~US$160-170 billion, suggesting that most of the banks have a market capital less than ~US$165 billion. The interquartile range (IQR), given by the formula Q3-Q1, is between ~US$150 billion - 190 billion, suggesting that there is low variation among mid-tier large banks. The Q3 is the 75th percentile while Q1 is the 25th percentile. We also see that there is an outlier at ~US$430 billion, which is JPMorgan Chase bank.

The whiskers - at both minimum and maximum of the boxplot are approximately at US$140 billion and US$230 billion, respectively, suggesting that the banks that are falling between that range are not considered as outliers or unusual.

3.6. Market Cap Quartile Distribution

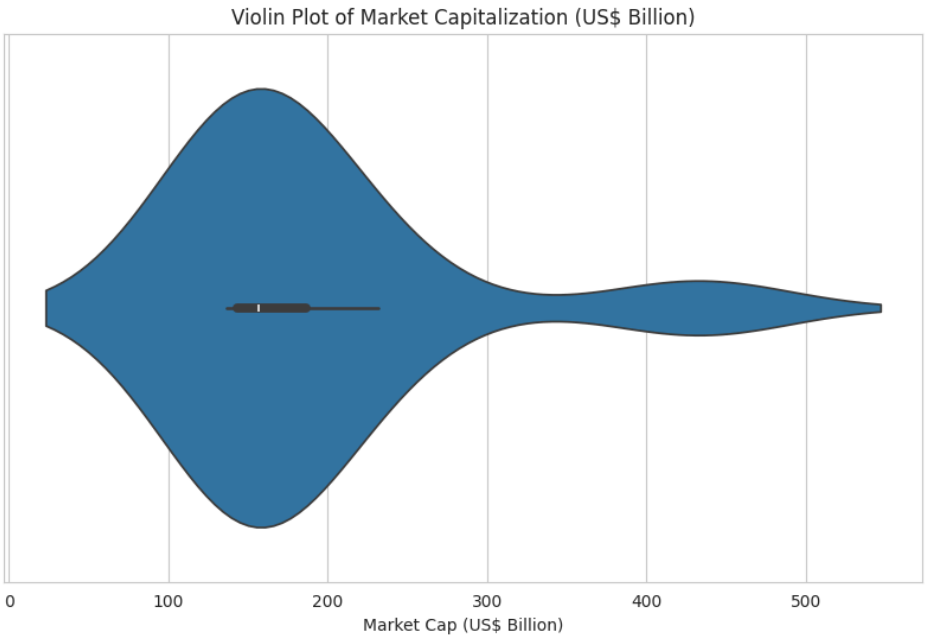
The violin plot below shows the quartile distribution of the market capital of each bank.

With the canvas size being 10 x 6, the function *.violinplot()* in the Seaborn library plots the distribution. The violin plot is used here as it shows the distribution by

combining a “box plot” with a curve and it also compares data spread, central tendency and shape across multiple groups.

```
# Market Cap Distribution by Quartile
plt.figure(figsize=(10, 6))
sns.violinplot(x=pandas_df['Market_Cap_USD_billion'])
plt.title('Violin Plot of Market Capitalization (US$ Billion)')
plt.xlabel('Market Cap (US$ Billion)')
plt.show()

log_progress("Plotted Violin Plot of Market Cap Distribution")
```



The output above shows a wide and concentrated distribution on the left side which means most banks have a market capital between ~US$100 billion and US$200 billion. We also see that there is a long tail that extends towards the right, indicating a few banks have very high market capitals and are also considered as outliers.

The box inside the violin plot is a mini boxplot, where the:

- White dot represents the median
- Thick horizontal bar represents the interquartile range
- Thin line extending from the box represents the whiskers, which is data range excluding the outliers

From the box inside the violin plot, it reaffirms that the median is around US$150 - US$180 billion, suggesting again that most banks are around the lower range of market capital.

3.7. Cumulative Market Share Analysis

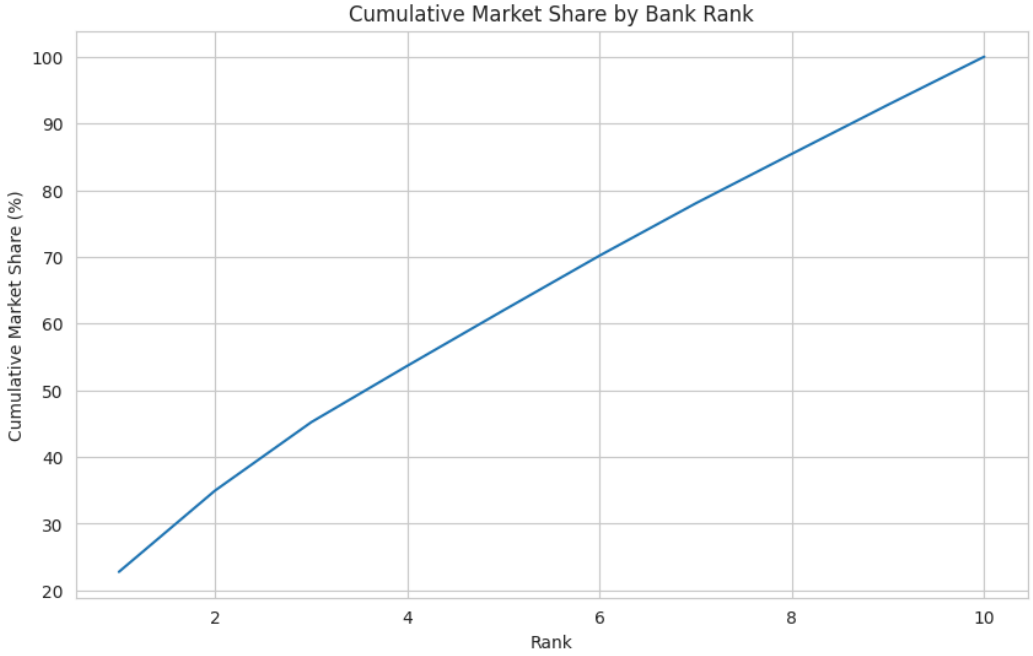
The code below calculates the cumulative market share, grouped by ranks. I have done the steps below to get the visualisation output.

1) Calculated the total market capitalisation using the .sum() function on the “Market_Cap_USD_billion” column in the “pandas_df” DataFrame.
2) Computed the cumulative market share for each ranked bank using .cumsum() function, which keeps adding each bank’s market capital in the order of their rank. I have also converted the result to percentage share by dividing total market capital and multiplying it by 100.
3) I plotted the result using Seaborn’s .lineplot() function with the x-axis being the rank while y-axis being the cumulative market share.

```
# Cumulative Market Share

# Plot
total_market_cap = pandas_df['Market_Cap_USD_billion'].sum()
pandas_df['Cumulative_Market_Share'] = pandas_df['Market_Cap_USD_billion'].cumsum() / total_market_cap * 100
plt.figure(figsize=(10, 6))
sns.lineplot(x='Rank', y='Cumulative_Market_Share', data=pandas_df)
plt.title('Cumulative Market Share by Bank Rank')
plt.xlabel('Rank')
plt.ylabel('Cumulative Market Share (%)')
plt.show()

log_progress("Plotted Cumulative Market Share")
```



From the output above, we see that the cumulative market share jumps from 23% at rank 1 to 35% at rank 2, and so on. This suggests that the first few top-ranked banks hold a very large chunk of the market.

From rank 4, the line increases at a steady pace (which means growth slows down). This could be interpreted as mid-tier banks contributing smaller amounts to overall market capital share.

3.8. Categorizing Banks

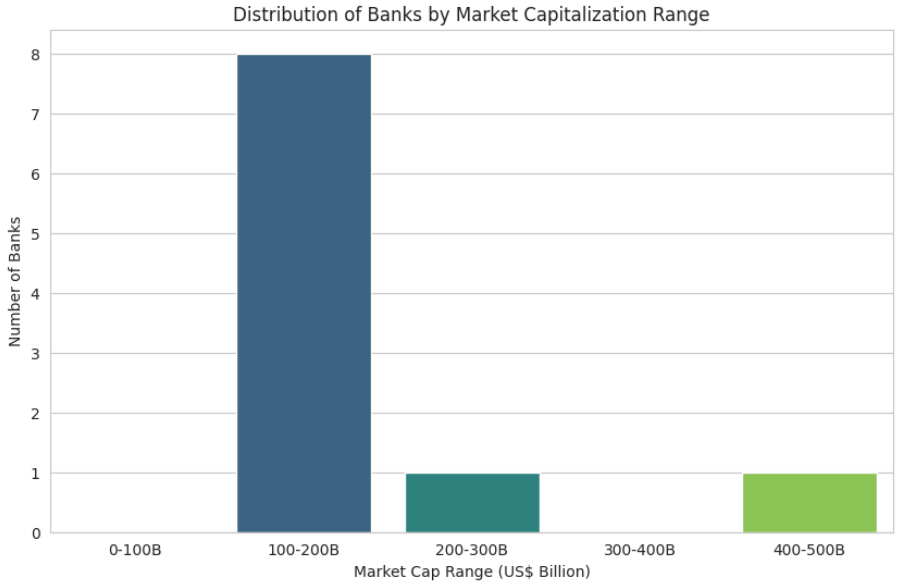
I have created a bar chart to visualise the market capitalisation ranges with bank categorisation, using the following steps:

1) Defining the bins for the bar chart by creating 5 ranges: 0-100B, 100-200B, 200-300B, 300-400B and 400-500B. This represents the different categories of bank market capitals.
2) Categorising each back into the defined market capital ranges using the *pd.cut()* function, which places each bank into one of the bins based on its market capital. The “right=False” attribute means that the bin includes the left value but excludes the right value. For example, there are 100 counts in the range 100-200B, but not in 0-100B.
3) The “range_counts” variable counts how many banks fall into each range and the *.sort_index()* functions will ensure that the range appears in the correct order.
4) The bar chart is plotted with the x-axis showing the bins/ranges and y-axis showing the number of banks in each range.

```
# Market Cap Range Distribution
# Create market cap ranges
bins = [0, 100, 200, 300, 400, 500]
labels = ['0-100B', '100-200B', '200-300B', '300-400B', '400-500B']
pandas_df['Market_Cap_Range'] = pd.cut(pandas_df['Market_Cap_USD_billion'], bins=bins, labels=labels, right=False)
range_counts = pandas_df['Market_Cap_Range'].value_counts().sort_index()

# Plot
plt.figure(figsize=(10, 6))
sns.barplot(x=range_counts.index, y=range_counts.values, palette='viridis', hue=range_counts.index, legend=False)
plt.title('Distribution of Banks by Market Capitalization Range')
plt.xlabel('Market Cap Range (US$ Billion)')
plt.ylabel('Number of Banks')
plt.show()

log_progress("Plotted Market Cap Range Distribution")
```



From the output above, we see that the majority of the banks fall in the 100-200B range, showing 8 out of 10 banks are in this range. This suggests that most of the banks are mid-sized scale. There are no banks in the 300-400B and 0-100B range, which shows a jump from medium-scaled banks to large banks in the market capital. All this shows that there are a few giants like JPMorgan Chase and Bank of America that dominate the market while the other banks are competitive within the mid-tier.

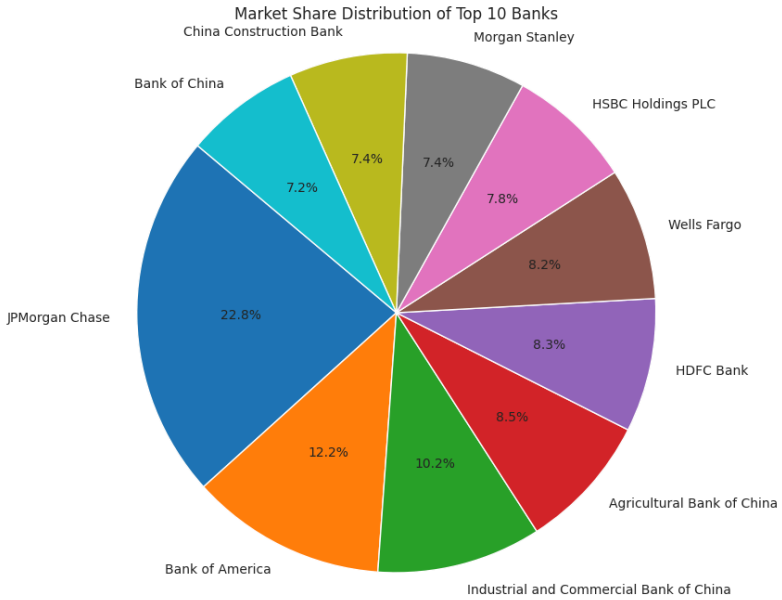
3.9. Visualize Market Share Distribution

1) The *.sum()* function is used to do the summation of the market capitalisation of the top 10 banks, using the "top_10_banks" DataFrame that already has these banks. This will be used to calculate the percentage share of the market each bank has, in the next step.
2) To compute each bank's market share, we divide the market capital by the total and multiply by 100 to convert it to percentage. The new column "Market_Share_Percentage" in the DataFrame will represent each bank's proportion of the top 10 banks.
3) A pie chart was used as the slices will correspond correctly to the market share percentages. The "autopct='%1.1f%'" attribute is used for formatting the percentage labels on each slice in the chart. Similarly, the "startangle=140" attribute will rotate the chart for better visual alignment.
4) The *.axis('equal')* function is used to ensure that the pie chart is a perfect circle and it is finally displayed using the *plt.show()* function.

```
# Top 10 Banks Market Share
# Calculate market share percentage for top 10 banks
total_market_cap_top10 = top_10_banks['Market_Cap_USD_billion'].sum()
top_10_banks['Market_Share_Percentage'] = (top_10_banks['Market_Cap_USD_billion'] / total_market_cap_top10) * 100

# Plot
plt.figure(figsize=(10, 8))
plt.pie(top_10_banks['Market_Share_Percentage'], labels=top_10_banks['Bank_name'], autopct='%1.1f%%', startangle=140)
plt.title('Market Share Distribution of Top 10 Banks')
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
plt.show()

log_progress("Plotted Market Share Distribution of Top 10 Banks")
```



From the pie chart above, we see that JPMorgan Chase has a share of 22.8% in the market, still being among the top 10 banks. This shows its global presence and brand value, since it's significantly higher than the other banks. Coming in the 2nd place is Bank of America with a share of 12.2% which is still above than the other banks.

The remaining 8 banks have a market share between the range of 7 to 10%, suggesting that none of the banks in this group have a huge dominance over the other and the market is fairly competitive for these 8 banks.

4. Banking Data & ETL Querying

4.1. Perform Advanced Market Capitalization Analysis with Growth Metrics

This analysis was done to understand how the market value of the bank changes in relation to the previous bank (in its ordered ranks). It is to help reveal any gaps or similarities that are occurring between multiple banks, and identify the market share concentration. The code below follows the steps to produce an analysis

1) Creating a window specification will show where PySpark should look at rows relative to each other, which have been ordered ascending in their rank.
2) Adding a column for previous bank's market capital using the .lag() function, which fetches the market capital of the previous row/bank. Rank 1 will have NULL for some calculations as there is no previous bank for it.
3) The absolute growth is calculated using the formula Current Market Capital - Previous Market Capital, giving the change in market capital from the previous rank.

4) The percentage growth was calculated using the formula (Growth / Previous Market Capital) * 100. This is saved into a new column, which shows the percentage increase (positive value) or decrease (negative value) in the market capital between the banks.

```
# Query: Advanced Market Cap Analysis with Growth Metrics
window_spec = Window.orderBy(asc("Rank"))
market_cap_analysis = spark_df_cleaned.withColumn(
    "Previous_Market_Cap", lag("Market_Cap_USD_billion", 1).over(window_spec)
)
market_cap_analysis = market_cap_analysis.withColumn(
    "Market_Cap_Growth_USD", col("Market_Cap_USD_billion") - col("Previous_Market_Cap")
)
market_cap_analysis = market_cap_analysis.withColumn(
    "Market_Cap_Growth_Pct",
    round((col("Market_Cap_Growth_USD") / col("Previous_Market_Cap")) * 100, 2)
)
print("Advanced Market Capitalization Analysis with Growth Metrics:")
market_cap_analysis.show()

log_progress("Advanced Market Cap Analysis Performed")
```

```
Advanced Market Capitalization Analysis with Growth Metrics:
+----+--------------------+--------------------+------------------+--------------------+--------------------+
|Rank|           Bank_name|Market_Cap_USD_billion|Previous_Market_Cap|Market_Cap_Growth_USD|Market_Cap_Growth_Pct|
+----+--------------------+--------------------+------------------+--------------------+--------------------+
|   1|      JPMorgan Chase|              432.92|              NULL|                NULL|                NULL|
|   2|     Bank of America|              231.52|            432.92|              -201.4|              -46.52|
|   3|Industrial and Co...|              194.56|            231.52|  -36.96000000000001|              -15.96|
|   4|Agricultural Bank...|              160.68|            194.56|  -33.879999999999995|              -17.41|
|   5|           HDFC Bank|              157.91|            160.68|  -2.7700000000000102|               -1.72|
|   6|         Wells Fargo|              155.87|            157.91|  -2.039999999999992|               -1.29|
|   7|   HSBC Holdings PLC|               148.9|            155.87|  -6.969999999999999|               -4.47|
|   8|      Morgan Stanley|              140.83|             148.9|  -8.069999999999993|               -5.42|
|   9|China Constructio...|              139.82|            140.83|  -1.0100000000000193|               -0.72|
|  10|       Bank of China|              136.81|            139.82|  -3.009999999999991|               -2.15|
+----+--------------------+--------------------+------------------+--------------------+--------------------+
```

From the above output, we see there are 6 columns, each defined below:

- Rank - in descending order
- Bank name
- Market_Cap_USD_billion - Current market capital of the bank
- Previous_Market_Cap - The market capital of the bank above it in rank
- Market_Cap_Growth_USD - Difference between this bank and the previous bank
- Market_Cap_Growth_Pct - Percentage difference compared to previous bank

Each value for each bank has a meaning, as shown below.

Bank Name	Value	Interpretation
JPMorgan Chase	Market Cap = 432.92B Previous Market Cap = NULL Growth = NULL	This bank is the highest among the group and there is no bank above it, hence growth of this bank cannot be calculated.
Bank of America	Market Cap = 231.52B Previous Market Cap = 432.92B	It is nearly US$200 billion smaller than JPMorgan Chase which is a 46.5% drop - showing the difference between

	Growth = -201.4B Growth % = -46.52%	ranks 1 and 2.
Industrial and Commercial Bank of China (ICBC)	Market Cap = 194.56B Previous Market Cap = 231.52B Growth = -36.96B Growth % = -15.96%	The market capital continues to drop but at a slower rate. The gap here is much smaller than that between ranks 1 & 2
Agricultural Bank of China	Market Cap = 160.68B Previous Market Cap = 194.56B Growth = -33.88B Growth % = -17.41%	There is another drop (negative growth) of about 16-17%, similar to the rate of growth as the bank in rank 3
HDFC	Market Cap = 157.91B Previous Market Cap = 160.68B Growth = -2.77B Growth % = -1.72%	The market size for HDFC is almost the same as Agricultural Bank of China (rank 4)
Wells Fargo	Market Cap = 155.87B Previous Market Cap = 157.91B Growth = -2.04B Growth % = -1.29%	The bank at this rank and the previous 2 ranks are all very close to each other, suggesting that the market has a stable valuation band and no single bank dominates in this group.
HSBC	Market Cap = 148.90B Previous Market Cap = 155.87B Growth = -6.97B Growth % = -4.47%	There is a slightly bigger difference here, when compared to the banks in the previous 3 rankings, but still smaller compared to the top banks (ranks 1 and 2)
Morgan Stanley	Market Cap = 140.83B Previous Market Cap = 148.90B Growth = -8.07B Growth % =-5.42%	There is a moderate drop in growth within the mid-tier cluster of banks.
China Construction Bank	Market Cap = 139.82B Previous Market Cap = 140.83B Growth = -1.01B Growth % = -0.72%	This has a very small gap when compared to the bank in the previous bank - having similar market caps.

Bank of China	Market Cap = 136.81B Previous Market Cap = 139.82B Growth = -3.01B Growth % = -2.15%	Along with the banks in ranks 8 and 9, this bank is also extremely close in the market capitalisation, indicating a dynamic and competitive lower tier of banks within the top 10 itself.

4.2. Analyze Market Concentration and Categorize Banks Based on Market Share Tiers.

Analysing the market concentration and categorising these banks helps us understand the dominance of the largest banks and identify their competitiveness, if possible, within the banking and financial industry. Hence, I have done the below steps to understand the analysis.

1) Calculating the total market capitalisation using the *.sum()* function will do the summation of market capital across all banks and *.collect()[0][0]* extracts the numerical value from the row.
2) Computing the individual market share will show the percentage contribution of each bank in the market. This is done by using the formula: Market share = (bank market cap/total market cap) * 100. The result is also rounded to 2 decimal places using the *round()* function.
3) Calculating the cumulative market share helps to identify how quickly the market gets concentrated within the top 10 banks. It uses a window function to do the summation of all the market shares.
4) Categorising the banks into market share tiers help to understand their dominance. There are 3 tiers, where:
   Tier 1 = major/dominant players (>= 10% market share)
   Tier 2 = mid-level/less dominant players (5-10%)
   Tier 3 = small/least dominant players (<5%)

```
# Market Concentration Analysis
total_market_cap = spark_df_cleaned.agg(sum("Market_Cap_USD_billion")).collect()[0][0]
market_concentration = spark_df_cleaned.withColumn(
    "Market_Share", round((col("Market_Cap_USD_billion") / total_market_cap) * 100, 2)
).withColumn(
    "Cumulative_Market_Share", sum("Market_Share").over(window_spec.rowsBetween(Window.unboundedPreceding, 0))
)
market_concentration = market_concentration.withColumn(
    "Market_Share_Tier",
    when(col("Market_Share") >= 10, "Tier 1 (>= 10%)")
    .when((col("Market_Share") >= 5) & (col("Market_Share") < 10), "Tier 2 (5%-10%)")
    .otherwise("Tier 3 (< 5%)")
)
print("Market Concentration Analysis:")
market_concentration.show()

log_progress("Market Concentration Analysis Performed")
```

Market Concentration Analysis:

Rank	Bank_name	Market_Cap_USD_billion	Market_Share	Cumulative_Market_Share	Market_Share_Tier
1	JPMorgan Chase	432.92	22.79	22.79	Tier 1 (>= 10%)
2	Bank of America	231.52	12.19	34.98	Tier 1 (>= 10%)
3	Industrial and Co...	194.56	10.24	45.22	Tier 1 (>= 10%)
4	Agricultural Bank...	160.68	8.46	53.68	Tier 2 (5%-10%)
5	HDFC Bank	157.91	8.31	61.99	Tier 2 (5%-10%)
6	Wells Fargo	155.87	8.2	70.19	Tier 2 (5%-10%)
7	HSBC Holdings PLC	148.9	7.84	78.03	Tier 2 (5%-10%)
8	Morgan Stanley	140.83	7.41	85.44	Tier 2 (5%-10%)
9	China Constructio...	139.82	7.36	92.8	Tier 2 (5%-10%)
10	Bank of China	136.81	7.2	100.0	Tier 2 (5%-10%)

From the above output, we see that only the top 3 banks - JPMorgan Chase, Bank of America and ICBC are in tier 1 (each holding more than 10% market share), controlling approximately 30% of the entire market. This suggests that they are dominant players and have a large customer base and brand value.
The remaining 7 banks are all in tier 2, holding market shares between 7-8.5% each. This suggests that the market for this tier has a stable structure with the strongest mid-tier banks at the top of the group.

4.3. Examine Statistical Distribution of Market Capitalization Using Quartile Analysis.

Quartile analysis helps to understand how market capitalisation values are spread across the dataset and identify any patterns/trends. It also helps to classify the banks into separate tiers based on their similarities.

1) Importing the required PySpark functions and giving the window specifications will help in calculations, ranking and analysis in the upcoming steps.
2) Calculating the first 3 quartiles - 25%, 50%, 75% estimates the market capital for each range, where 50% is the median.
3) Dividing the number of banks using the *ntile(4)* function such that equal-sized groups can be created and the order of those will be defined by their ranks.
4) The 3 quartiles with their threshold values are added to each row for reference and comparison using the *.withColumn()* function where “Q1_value”, “Q2_value”, “Q3_value” represents quartiles each respectively.
5) The .show() function prints the full quartile distribution table with the assigned quartiles and their respective threshold values.

```
# Import required functions
from pyspark.sql.functions import (col, avg, sum, count, desc, asc, round, lag,
                                   dense_rank, ntile, when, first, lead, min, max, lit)
from pyspark.sql.window import Window

# Query 3: Statistical Distribution Analysis
Q1 = spark_df_cleaned.approxQuantile("Market_Cap_USD_billion", [0.25], 0.05)[0]
Q2 = spark_df_cleaned.approxQuantile("Market_Cap_USD_billion", [0.50], 0.05)[0]
Q3 = spark_df_cleaned.approxQuantile("Market_Cap_USD_billion", [0.75], 0.05)[0]
quartile_analysis = spark_df_cleaned.withColumn(
    "Quartile", ntile(4).over(window_spec)
)
quartile_analysis = quartile_analysis.withColumn("Q1_Value", lit(Q1)) \
                                     .withColumn("Q2_Value", lit(Q2)) \
                                     .withColumn("Q3_Value", lit(Q3))
print("Quartile Distribution Analysis:")
quartile_analysis.show()

log_progress("Completed Quartile Distribution Analysis")
```

```
Quartile Distribution Analysis:
+----+--------------------+----------------------+--------+--------+--------+--------+
|Rank|           Bank_name|Market_Cap_USD_billion|Quartile|Q1_Value|Q2_Value|Q3_Value|
+----+--------------------+----------------------+--------+--------+--------+--------+
|   1|      JPMorgan Chase|                432.92|       1|  140.83|  155.87|  194.56|
|   2|     Bank of America|                231.52|       1|  140.83|  155.87|  194.56|
|   3|Industrial and Co...|                194.56|       1|  140.83|  155.87|  194.56|
|   4|Agricultural Bank...|                160.68|       2|  140.83|  155.87|  194.56|
|   5|           HDFC Bank|                157.91|       2|  140.83|  155.87|  194.56|
|   6|         Wells Fargo|                155.87|       2|  140.83|  155.87|  194.56|
|   7|   HSBC Holdings PLC|                 148.9|       3|  140.83|  155.87|  194.56|
|   8|      Morgan Stanley|                140.83|       3|  140.83|  155.87|  194.56|
|   9|China Constructio...|                139.82|       4|  140.83|  155.87|  194.56|
|  10|       Bank of China|                136.81|       4|  140.83|  155.87|  194.56|
+----+--------------------+----------------------+--------+--------+--------+--------+
```

From the above output, we see that the top 10 banks have been assigned a quartile (from 1 to 4). The banks in ranks 1 to 3 are in the 1st quartile as they have market capitals above the 75th percentile, suggesting that they are the highest-value banks in the dataset. Next, the banks in ranks 4 to 6 are in the 2nd quartile represent banks who are not the top performers, but do significantly well. Thirdly, the 3rd quartile with banks ranking 7 and 8 represent the lower segment and the last quartile (Quartile 4) represent the lowest segment of banks, those in ranks 9 and 10.

4.4. Conduct Comparative Size Analysis to Classify Banks by Relative Market Size.

Comparative size analysis helps in categorising banks into different groups so that it allows for a quick comparison of their relative market share and competitiveness in the banking industry. To achieve this, I have done the following steps:

1) Calculating the key statistics such as minimum, maximum and average for statistical analysis. The *.alias()* function is used to rename the column in the Spark DataFrame to appear as a new name in the output. It stores

these values into “stats” and specifically extracts the average market capital into “avg_cap”.

2) Classification of banks based on their market capital is done using the *.agg()* function with conditional logic where:
if market capital >= average, the bank is a large bank
else if market capital >= 0.75 (75%), the bank is a mid-size bank
otherwise, the bank is a small bank.
I have then displayed the results using the *.show()* function.

```
# Comparative Size Analysis
stats = spark_df_cleaned.agg(
    min("Market_Cap_USD_billion").alias("Min_Cap"),
    max("Market_Cap_USD_billion").alias("Max_Cap"),
    avg("Market_Cap_USD_billion").alias("Avg_Cap")
).collect()[0]
avg_cap = stats["Avg_Cap"]
size_analysis = spark_df_cleaned.withColumn(
    "Size_Category",
    when(col("Market_Cap_USD_billion") >= avg_cap, "Large Bank")
    .when(col("Market_Cap_USD_billion") >= avg_cap * 0.75, "Mid-Sized Bank")
    .otherwise("Small Bank")
)
print("Comparative Size Analysis:")
size_analysis.show()

log_progress("Completed Comparative Size Analysis")
```

```
Comparative Size Analysis:
+----+--------------------+----------------------+--------------+
|Rank|           Bank_name|Market_Cap_USD_billion| Size_Category|
+----+--------------------+----------------------+--------------+
|   1|      JPMorgan Chase|                432.92|    Large Bank|
|   2|     Bank of America|                231.52|    Large Bank|
|   3|Industrial and Co...|                194.56|    Large Bank|
|   4|Agricultural Bank...|                160.68|Mid-Sized Bank|
|   5|           HDFC Bank|                157.91|Mid-Sized Bank|
|   6|         Wells Fargo|                155.87|Mid-Sized Bank|
|   7|   HSBC Holdings PLC|                 148.9|Mid-Sized Bank|
|   8|      Morgan Stanley|                140.83|    Small Bank|
|   9|China Constructio...|                139.82|    Small Bank|
|  10|       Bank of China|                136.81|    Small Bank|
+----+--------------------+----------------------+--------------+
```

From the output above, we see that the distribution is skewed, with a few banks dominating the market. JPMorgan Chase is greatly above the average, which has caused the average of the DataFrame to increase. Since the average increased, fewer banks, other than JPMorgan Chase, have fallen into the category of “Large Bank”.

Banks like HDFC, Wells Fargo fall below the average, although they are large in reality. They are classified as a mid-sized bank due to the average market capital of the dataset being higher. Some banks are classified as “small” although they are globally significant - showing that the comparative analysis depends on the calculated average that was defined.

4.5. Evaluate Market Growth and Identify Gaps Between Consecutive Banks.

This is done to understand how much the market size drops from one bank to the next, to see the concentration of market amongst them and identify any competitiveness present. Hence, I have done the following steps:

1) The new column, created by using the *lag()* function, "Previous_Market_Cap" stores that market capital of each bank in the previous row. For rank 1, there is no previous bunk hence it returns NULL.
2) The gap between each bank is calculated using the formula: current_market_cap - previous_market_cap. If there is a negative value, it means that the current bank is smaller than the previous one, which helps in quantifying how much difference exists between each bank.
3) The percentage of the gap is calculated using the formula: (Market_Cap_Gap_USD/Previous_Market_Gap)*100. The percentage gives a normalised comparison between the 2 columns.
4) Finally, the percentage is being rounded to 2 decimal places using the round() function, so as to make the percentage presentable and clean.

```
# Growth and Gap Analysis
growth_gap = spark_df_cleaned.withColumn(
    "Previous_Market_Cap", lag("Market_Cap_USD_billion", 1).over(window_spec)
).withColumn(
    "Market_Cap_Gap_USD",
    col("Market_Cap_USD_billion") - col("Previous_Market_Cap")
).withColumn(
    "Gap_Percentage",
    round((col("Market_Cap_Gap_USD") / col("Previous_Market_Cap")) * 100, 2)
)
print("Market Growth and Gap Analysis Between Consecutive Banks:")
growth_gap.show()
log_progress("Completed Market Growth & Gap Analysis")
```

```
Market Growth and Gap Analysis Between Consecutive Banks:
+----+--------------------+----------------------+-------------------+--------------------+--------------+
|Rank|           Bank_name|Market_Cap_USD_billion|Previous_Market_Cap| Market_Cap_Gap_USD|Gap_Percentage|
+----+--------------------+----------------------+-------------------+--------------------+--------------+
|   1|      JPMorgan Chase|                432.92|               NULL|                NULL|          NULL|
|   2|     Bank of America|                231.52|             432.92|              -201.4|        -46.52|
|   3|Industrial and Co...|                194.56|             231.52| -36.96000000000001|        -15.96|
|   4|Agricultural Bank...|                160.68|             194.56|-33.879999999999995|        -17.41|
|   5|           HDFC Bank|                157.91|             160.68|-2.7700000000000102|         -1.72|
|   6|         Wells Fargo|                155.87|             157.91| -2.039999999999992|         -1.29|
|   7|   HSBC Holdings PLC|                 148.9|             155.87| -6.969999999999999|         -4.47|
|   8|      Morgan Stanley|                140.83|              148.9| -8.069999999999993|         -5.42|
|   9|China Constructio...|                139.82|             140.83|-1.0100000000000193|         -0.72|
|  10|       Bank of China|                136.81|             139.82| -3.009999999999991|         -2.15|
+----+--------------------+----------------------+-------------------+--------------------+--------------+
```

From the above output, we see that there is a huge drop from rank 1 to rank 2 with almost 46.52%. This suggests that JPMorgan Chase is ahead of all others and the market capital and share is dominated by this bank. From then on, we see a continuous decline across the ranking as the percentage is negative, meaning that each subsequent bank has a smaller market capital than the previous one with small and random fluctuations in between.

The gap analysis highlights where the biggest competitive shares are, which blanks are closely matched and how the market share and capital is distributed across the ranks.

4.6. Assess Market Dominance by Measuring Cumulative Share and Dominance Score.

Assessing the market dominance to understand how much of the total market share is controlled by each bank and how the top banks dominate the banking industry. The steps taken to do so are as below:

1) Calculating the total market capital by computing the sum of market capital with the *.sum()* function. The total is needed to calculate each bank's percentage share.
2) Adding a cumulative dominance column uses a running window to add all the market shares from ranks 1 to 10, showing how much market share has been captured up to each bank. Finally, the *.show()* function was used to display the market share and cumulative dominance.

```
# Market Dominance Analysis
total_cap = spark_df_cleaned.agg(sum("Market_Cap_USD_billion")).collect()[0][0]
dominance = spark_df_cleaned.withColumn(
    "Market_Share",
    round((col("Market_Cap_USD_billion") / total_cap) * 100, 2)
)
dominance = dominance.withColumn(
    "Cumulative_Dominance",
    sum("Market_Share").over(window_spec.rowsBetween(Window.unboundedPreceding, 0))
)
print("Market Dominance Analysis:")
dominance.show()

log_progress("Completed Market Dominance Analysis")
```

```
Market Dominance Analysis:
+----+--------------------+----------------------+------------+--------------------+
|Rank|           Bank_name|Market_Cap_USD_billion|Market_Share|Cumulative_Dominance|
+----+--------------------+----------------------+------------+--------------------+
|   1|      JPMorgan Chase|                432.92|       22.79|               22.79|
|   2|     Bank of America|                231.52|       12.19|               34.98|
|   3|Industrial and Co...|                194.56|       10.24|               45.22|
|   4|Agricultural Bank...|                160.68|        8.46|               53.68|
|   5|           HDFC Bank|                157.91|        8.31|               61.99|
|   6|         Wells Fargo|                155.87|         8.2|               70.19|
|   7|   HSBC Holdings PLC|                 148.9|        7.84|               78.03|
|   8|      Morgan Stanley|                140.83|        7.41|               85.44|
|   9|China Constructio...|                139.82|        7.36|                92.8|
|  10|       Bank of China|                136.81|         7.2|               100.0|
+----+--------------------+----------------------+------------+--------------------+
```

From the output above, we see that a few banks - JPMorgan Chase and Bank of America dominate the market, with JPMorgan Chase holding 22.79% of the entire market - dominating heavily. This shows a high concentration of market share at the top, while still amongst the top 10 banks.

Banks in ranks 1 to 7 have a cumulative market dominance of 78.03%, showing that the banking industry is not distributed evenly and the top half has more power/influence in the market. The banks in ranks 8 to 10 have a cumulative

portion of 14%, which suggests that market dominance does not shift significantly although these banks matter as they are in the top 10.

4.7. Analyze Segment-Wise Bank Performance Based on Market Capitalization Ranges.

The analysis done is helpful in classifying banks into meaningful market-cap ranges to compare their performance within similar size groups. It can also help to identify how many banks fall into each segment and see where the market is dominated. To produce the output, I performed these sequence of steps

1) Creation of market capital segments such that each bank in the top 10 banks DataFrame will be assigned to one of the 3 fixed categories on the basis of its market capitalisation. The categories are:
   - low segment: below US150 billion
   - medium segment: US150-250 billion
   - high segment: above US250 billion
2) A segment label is added to the DataFrame as a new column called “Market_Cap_Segment”, to classify each bank accordingly.
3) The *.groupBy()* function helps in grouping the bank by the segment category and the *.count()* function counts the number of banks in each group.

```
# Segment Performance Analysis
segment_analysis = spark_df_cleaned.withColumn(
    "Market_Cap_Segment",
    when(col("Market_Cap_USD_billion") < 150, "Low (0 - 150B)")
    .when((col("Market_Cap_USD_billion") >= 150) & (col("Market_Cap_USD_billion") < 250),
          "Medium (150B - 250B)")
    .otherwise("High (250B+)")
)
segment_counts = segment_analysis.groupBy("Market_Cap_Segment").count()
print("Segment-wise Performance Analysis:")
segment_counts.show()

log_progress("Completed Segment-wise Performance Analysis")
```

```
Segment-wise Performance Analysis:
+--------------------+-----+
|  Market_Cap_Segment|count|
+--------------------+-----+
|        High (250B+)|    1|
|Medium (150B - 250B)|    5|
|      Low (0 - 150B)|    4|
+--------------------+-----+
```

From the output above, we see that the market is truly dominated by one bank (JPMorgan Chase), suggesting that the top of the market is not crowded. Most banks (5 of them) fall into the medium segment, indicating a cluster of mid-tier banks and are competitive with each other in the same cluster. Finally, there are 4 banks in the low segment. Although they hold large market capitals, they have

less influence compared to the other 2 segments. This analysis can help decision-makers understand the current competition levels and find opportunities for growth or investment.

4.8. Generate a Comprehensive Performance Dashboard for Bank Rankings and Metrics.

The code below is a full and final dashboard of all the banks rankings and metrics that were used previously. This dashboard helps to combine everything into a single view, making it easier for comparison across multiple dimensions. The steps to produce this dashboard are given below.

1) I calculated each bank's percentage contribution to total market capitalisation and rounded most of the values to 2 decimal places for easy understanding and viewing. The formula used to calculate the percentage is: ("Market_Cap_USD_billion"/total_cap)*100
2) Then, I created a running total of the market share of the banks in descending order, to show how quickly the market dominance accumulates.
3) Next, I used the .lag() function to retrieve the market capital of the previous bank in the ranking for comparison, which helps to analyse gaps or differences between them.
4) Then, I computed how much larger or smaller a bank is relative to the rank above it. This is useful for spotting jumps or any tight competition.
5) Finally, I divided all banks into 4 equal groups based on their rank to highlight their relative positioning across the distribution.

```
# Comprehensive Performance Dashboard
dashboard = spark_df_cleaned \
    .withColumn("Market_Share", round((col("Market_Cap_USD_billion") / total_cap) * 100, 2)) \
    .withColumn("Cumulative_Share", sum("Market_Share").over(window_spec)) \
    .withColumn("Previous_Cap", lag("Market_Cap_USD_billion").over(window_spec)) \
    .withColumn("Gap_USD", col("Market_Cap_USD_billion") - col("Previous_Cap")) \
    .withColumn("Quartile", ntile(4).over(window_spec)) \
    .withColumn(
        "Size_Category",
        when(col("Market_Cap_USD_billion") >= avg_cap, "Large Bank")
        .when(col("Market_Cap_USD_billion") >= avg_cap * 0.75, "Mid-Sized Bank")
        .otherwise("Small Bank")
    )
print("Comprehensive Performance Dashboard:")
dashboard.show()

log_progress("Comprehensive Performance Dashboard Generated")
```

Comprehensive Performance Dashboard:

Rank	Bank_name	Market_Cap_USD_billion	Market_Share	Cumulative_Share	Previous_Cap	Gap_USD	Quartile	Size_Category
1	JPMorgan Chase	432.92	22.79	22.79	NULL	NULL	1	Large Bank
2	Bank of America	231.52	12.19	34.98	432.92	-201.4	1	Large Bank
3	Industrial and Co...	194.56	10.24	45.22	231.52	-36.96000000000001	1	Large Bank
4	Agricultural Bank...	160.68	8.46	53.68	194.56	-33.879999999999995	2	Mid-Sized Bank
5	HDFC Bank	157.91	8.31	61.99	160.68	-2.7700000000000102	2	Mid-Sized Bank
6	Wells Fargo	155.87	8.2	70.19	157.91	-2.039999999999992	2	Mid-Sized Bank
7	HSBC Holdings PLC	148.9	7.84	78.03	155.87	-6.969999999999999	3	Mid-Sized Bank
8	Morgan Stanley	140.83	7.41	85.44	148.9	-8.069999999999993	3	Small Bank
9	China Constructio...	139.82	7.36	92.8	140.83	-1.0100000000000193	4	Small Bank
10	Bank of China	136.81	7.2	100.0	139.82	-3.009999999999991	4	Small Bank

5. Conclusions

5.1. Final Insights and Recommendations

The whole analysis has provided a clearer and structured understanding of how the world's top banks compare in terms of market capitalisation, their positioning and competitiveness between each bank. The JPMorgan Chase bank has been consistently at the top of the global banking market with its market capital being so large that it shifts the entire statistical average upward to create a significant gap between the remaining ranks.

The statistical analysis of the data also shows the same where only a few banks dominate heavily, most banks are operating in a balanced yet competitive section and there are small differences that exist at the lower end of the ranking.

From a business and decision-making perspective,

5.2. Suggestions to use cross-currency analysis (USD, GBP, EUR, INR) for consistent benchmarking of financial institutions across regions

As global banks operate in different regions and currencies, using a single currency is difficult due to variations in different countries. This proposed cross-currency approach would be more realistic to allow analysts to categorise banks by adjusting the exchange rates and local economic conditions of certain regions/markets like Asia, Europe and America.

This method, to an extent, allows for a fairer global market benchmark as some currencies could be weak while some could be stronger. Hence instead of misrepresenting the growth and operational efficiency of the currency and banks, these currencies can be used where possible without relying on only one type of currency.

An analysis of currency can help in long-term strategy planning for investments done by a bank. It helps them in understanding the different economic environments in different countries, highlighting trends and dangers that may or may not be easily visible/understandable.

5.3. Propose continuous monitoring of market share concentration to identify growth opportunities for mid-tier banks

To continuously monitor and identify growth opportunities while staying competitive in such a market, some of the following proposed solutions can be done:

- Regularly assess cumulative market dominance of the top banks so that mid-tier banks can identify where they may get or lose opportunities to improve.
- Monitoring yearly and quarterly growth rates will allow these banks to spot any expansion that can be done by them.

- There may be points where market share within the mid-tier banks start getting competitive and cause differences in growth rates. Hence, some of these banks can learn and adopt the strategies taken to raise their ranks.
- Monitor and evaluate the impact of economic trends across the globe and within regions. This can help influence decisions taken for market share distribution, hence improving the accuracy of opportunity detection within the market.
- Dashboards, similar to the one in previous sections, can be implemented at their convenience so that decision-makers can monitor the banks' positioning in the market, check for anomalies without waiting for a while before a report is generated.

5.4. Identify potential regions or banking segments for expansion by analysing gaps between tiers of banks and regional trends

As seen from the analysis done in previous sections there are three different tiers - dominant global leaders, strong mid-tier competitors and lower-tier (but still within the top 10). The size and segment analysis of these banks show that some of these banks have similar market capitalisations and shares. Some potentials for expansion of these banks can be done by doing the following:

- Targeting high-growth markets that have financial demand such as India, Africa and Southeast Asia. These regions have been showing an increasing economic growth with their digitalisation aim growing exponentially simultaneously.
- The gaps that occur between different areas can be covered by mid-tier banks with similar market shares and capitalisations by entering different segments like wealth management and cross-boarder payments. This can help to differentiate between other banks within the tier and promote them as a "unique" bank.
- Similarly, mid and lower-tier banks can expand into markets like Europe and Asia-Pacific where the top-tier banks are unable to reach due to their stagnant nature. This will allow them to strengthen their presence within the region.
- Regions like Europe and Asia are pushing for sustainable green financing and investments. Hence, some banks which enter these tiers can also use government incentives to meet the incoming demands.
- Cross-border payment systems can be incorporated especially in regions like ASEAN countries, where trading is constant and opportunities for corporate finance usually prevail.
- Strengthening their retail presence in booming digital markets like India show high potential for new retail-based banking expansion.

6. Visualisation Integration

6.1. Prepare data for visualisation platforms

The steps taken to prepare data for visualisation platforms are as below:

1) Verify data
    - Ensure that all the missing values have been handled.
    - Ensure that there are no duplicate rows existing.
    - Ensure consistent formatting of fields in the dataset.
2) Rename column names
    - Rename specific column names to labels that are readable and easily understandable.
    - This is done to avoid issues with regular expressions (regex) or special characters.
3) Sort and structure the dataset
    - Organise or group the data by market capital such that the visualisation tools are able to generate clean dashboards.
    - Create derived fields for certain columns where needed.
4) Export the dataset
    - Export the dataset to CSV (.csv) or Excel (.xls or .xlsx) formats as these are widely used by many visualisation tools.
5) Document additional information
    - Ensure metadata of each column has been attached as a “README” file for easy understanding of each column and metric.

6.2. Generate Tableau Connection Instructions

1) Open Tableau desktop and select “Connect to a file” (since the exported data is in a CSV/Excel format).
2) Choose the type of data format - CSV, Excel, SQL, etc.
3) Browse to the location of the processed data and load it into Tableau.
4) Tableau automatically detects the data types and allows creation of worksheets using the drag-and-drop elements that Tableau is known for having.
5) Ensure appropriate credentials and drivers are installed when using database connection, to connect to databases.

6.3. Generate Power BI Connection Instructions

1) Open Power BI Desktop and select “Get data”.
2) Choose the type of data format to import - CSV, Excel, SQL, etc.
3) Browse to the location of the processed data and load it into Power BI.
4) The Query Editor within Power BI can be used to apply more transformations to the data when necessary.

5) After loading the data and model, visualisations can be created within the Power BI tool. Some visualisations include bar charts, tables and key performance indicators (KPIs).

6.4. Execute Visualisation Setup

1) Load the dataset
    - Import the cleaned or exported dataset according to their format.
    - Ensure all fields are loaded correctly and match their data types.
2) Define relationships
    - Establish relationships between tables if multiple datasets are used.
    - Ensure there are primary and foreign keys to relate them to multiple tables, where necessary.
3) Select appropriate visualisations
    - Choose the correct visualisation type that is needed such as histograms, pie charts, bar charts, etc.
    - These visualisations should be able to reflect the data correctly - showing trends, correlations and relationships between the variables.
4) Configure visual properties
    - Apply consistent formatting to the visualisation such as colours, labels and titles.
5) Arrange dashboard layout
    - Organise the view of the visualisations in the dashboard so that it is clean and easy to understand.
6) Export the dashboard
    - Publish or share the dashboard by providing permissions to relevant people.

6.5. Sample Dashboard Layout (Documentation)

1) Overview
    - KPI Cards that include “Total Market Capitalisation”, “Top Bank by Market Cap”, “Cumulative Market Share”.
    - Summary block that includes dataset size and context (in terms of currency).
2) Ranking Analysis Section
    - Horizontal bar chart can be used for “Top 10 Banks by Market Capitalisation’.
    - Scatter plot can be used for “Rank vs Market Cap” to highlight negative correlations.
    - Tabular View can be used for showing the Bank, its rank, its market cap, its growth percentage and which tier it is in.
3) Distribution & Outlier Insights

- Histogram can be used to show the market capital distribution among the top 10 banks.
- Box plots can be used to show outliers, median and the spread of data/quartiles.
- Violin plots can be used to identify the distribution density with quartile markers.
- The visuals mentioned above can help in identifying the skewness of distribution and outliers that are present within the dataset.

4) Market Concentration & Dominance Section
    - It is to display how market share and dominance is amongst the top 10 banks.
    - Line charts can be used to show the “Cumulative market share by rank”.
    - Similarly, pie charts can be used to display the “Percentage market share” of the top 10 banks.
    - Coloured bands for tier categorisation.
5) Growth and Gap metrics
    - The purpose of this is to show how each bank compares with one another.
    - Delta charts and arrow indicators can be used to show the market capital growth percentage, when compared to the bank in the rank above it.