# Approach and Methodology

## 1. Data Acquisition and Bulk File Handling

- The dataset was downloaded from Kaggle and duplicated into multiple CSV files for simulating bulk file handling.
- All CSV files were loaded into a single PySpark DataFrame using the following process:
    - Directory paths were scanned for all CSV files.
    - The `spark.read.csv()` function was used to load and merge the files into a single DataFrame.

## 2. Data Exploration

- The total number of rows and columns were determined using `df.count()` and `len(df.columns)`.
- Checked using a combination of `col()` and `isNull()` functions.
- Key metrics such as mean, median, and mode for numerical columns were computed using the `describe()` method.

## 3. Data Cleaning

- Missing values were replaced with 0 to ensure consistency in the data.
- Columns were renamed for clarity and consistency using the `withColumnRenamed()` method.

## 4. Data Transformation

- A new column was created by normalizing the "Amount" column
- A new column was added with the natural logarithm of the "Amount" column

## 5. File Conversion

- The cleaned and transformed DataFrame was saved in Parquet format using the `write.parquet()` method for efficient storage and querying.

## 6. SQL Querying

The DataFrame was registered as a temporary SQL view, enabling SQL queries to extract insight

# Findings and Insights

1. **Exploratory Data Analysis**:
    - The dataset was well-structured, but missing values needed to be addressed.
    - Fraudulent transactions constituted a small percentage of the dataset, indicating class imbalance.
2. **Data Transformation**:
    - The normalized amount and logarithmic transformations helped mitigate the impact of outliers.
3. **SQL Queries**:
    - Key statistics about fraudulent and non-fraudulent transactions were extracted, providing insights for downstream analyses.

# Tools and Technologies

- **Apache Spark with PySpark**: For distributed data processing.
- **Jupyter Notebook**: For implementing and documenting the task.
- **Parquet Format**: For efficient storage and querying.

# Conclusion

This task demonstrated the end-to-end pipeline for processing and analyzing large-scale credit card transaction data. Bulk file handling, data cleaning, transformation, and SQL querying were successfully implemented, with key insights derived from the data.