Silicon Valley Bank (SVB) Failure: Predictable or Not?

Final Report

Team #21 | Yifei Long, Simón Caicedo, Rodney Nuñez

1. Introduction

Our project centers around the failures of Silicon Valley Bank (SVB) and First Republic Bank, which have prompted the formation of a collaborative effort between the Federal Deposit Insurance Corporation (FDIC) and the USA government. The objective is to provide a data warehouse that will be used by the data analysts of Fact Radar Inc to analyze financial data from publicly traded financial institutions.

Motivated by the need to gain insights from the financial data, our team will design and implement a robust data warehouse solution. This data warehouse will serve as the backbone for efficiently storing and managing vast amounts of data from various sources. By optimizing it for reporting and analysis, we aim to facilitate the creation of a comprehensive decision-making dashboard by the data analysts at Fact Radar Inc.

The significance of our work lies in its direct impact on the reliability and time sensitivity of the insights that are potentially generated from the established big-data ETL pipeline and data warehouse. Creating a data warehouse for investigating the predictability of SVB's or other financial institutions' failure offers valuable insights that can be applied to detect early warning signs in other financial institutions, potentially averting similar situations in the future. The use cases of our project can be attributed to several factors:

- Ensuring financial stability: By examining the factors that contributed to SVB's failure, we can better understand the risks and vulnerabilities within the financial sector. This knowledge will help in formulating strategies that promote financial stability, safeguarding the interests of depositors, investors, and the overall economy.
- Regulatory considerations: If we find that SVB's failure was predictable, it raises questions about the effectiveness of current regulations and supervisory practices. Our findings could be valuable for policymakers and regulators to

assess and improve existing regulatory frameworks, preventing similar failures in the future.
- Restoring investor confidence: Gaining insights into the factors that led to SVB's failure will help investors make more informed decisions when investing in financial institutions. By shedding light on potential risks and warning signs, we aim to help investors regain confidence in the financial markets.
- Learning opportunities for the industry: Our investigation into the circumstances surrounding SVB's failure will offer lessons for other financial institutions. By learning from these events, they can enhance their risk management practices and protect their operations from encountering similar issues.

In addition, the quality of the data warehouse will be a determining factor in the success of this project. We utilized our expertise in data modeling, data integration, and optimization to ensure the efficiency and effectiveness of the data warehouse. By delivering a well-architected data warehouse solution, our objective is to empower the data analysts at Fact Radar Inc with the means to extract meaningful insights from the complex financial data. This, in turn, will enable stakeholders to make informed decisions and take proactive measures to prevent similar failures in the future.

2. Data Sources

Our team measured various financial data and company information to create a data warehouse for analyzing the predictability of the Silicon Valley Bank (SVB) or other financial institutions failure:

- Financial statements of all publicly traded financial institutions, including balance sheets, income statements, and cash flow statements.
- Stock data, such as market capitalization and stock prices.
- Financial ratios like ROE, ROA, CAR, NIM, and NLP, can help in assessing the financial health of institutions.
- Basic company information, such as GICS code, sector of activity or industry, headquarters' location, and number of employees, among others.
- The end product for this pipeline will be a dashboard composed of the above information.

Some of the variables that we will be using include:

- Balance sheet items, such as assets, liabilities, and equity.
- Income statement items, including revenues, expenses, and net income.
- Market capitalization and stock prices.

- Financial ratios (ROE, ROA, CAR, NIM, and NLP)
- Company information: GICS code, sector, location data (latitude, longitude, county, zip code, etc), number of employees, and branch locations.

The time interval to be measured will cover the period from January 2017 to June 2023. This range not only allows us to analyze financial data and trends leading up to SVB's failure, but also includes the latest data to predict the future, providing a comprehensive view of the factors that could have contributed to the event.

In determining the optimal data sources for our purposes, we focused on three key attributes: reliability, depth, and accessibility. Our selection of Yahoo Finance API and StockAnalysis.com as primary data sources was a strategic decision made based on these considerations.

Yahoo Finance API emerged as our chosen source for company profiles and stock price data. It stands as a preeminent platform in the world of finance, renowned for its robustness and reputation. Yahoo Finance provides real-time and historical stock data, which is critical for tracking market movements and predicting trends. Moreover, it offers extensive information on company profiles, including data on thousands of publicly traded companies worldwide. This data is continuously updated, ensuring we're working with the most current information at all times.

StockAnalysis.com, on the other hand, was instrumental for extracting financial statements. StockAnalysis.com offers a comprehensive repository of financial data on a multitude of companies, making it a prime choice for sourcing balance sheets, income statements, and cash flow statements. The platform is highly valued for its accurate datasets that are meticulously curated and verified.

By leveraging these platforms, we ensured that the data we extracted was not only reliable and exhaustive but also readily accessible. This approach underscores our commitment to creating an accurate and reliable data warehouse for the analysis of financial institutions.

The details of our extraction process are discussed further in the ETL - Extract section of this report. For an in-depth understanding of the extracted data, we have included a data dictionary in the Appendix that elaborates on the various data types sourced from these platforms.

3. Data Model

The data being modeled includes financial and stock market-related information for multiple companies. It covers aspects such as stock prices, financial ratios, income statements, balance sheets, and cash flows. The data model incorporates dimensions for time (date) and company profiles, enabling analysis of company performance, financial indicators, and market trends. The star schema designed for this project includes the following fact and dimension tables:

Facts & Dimensions

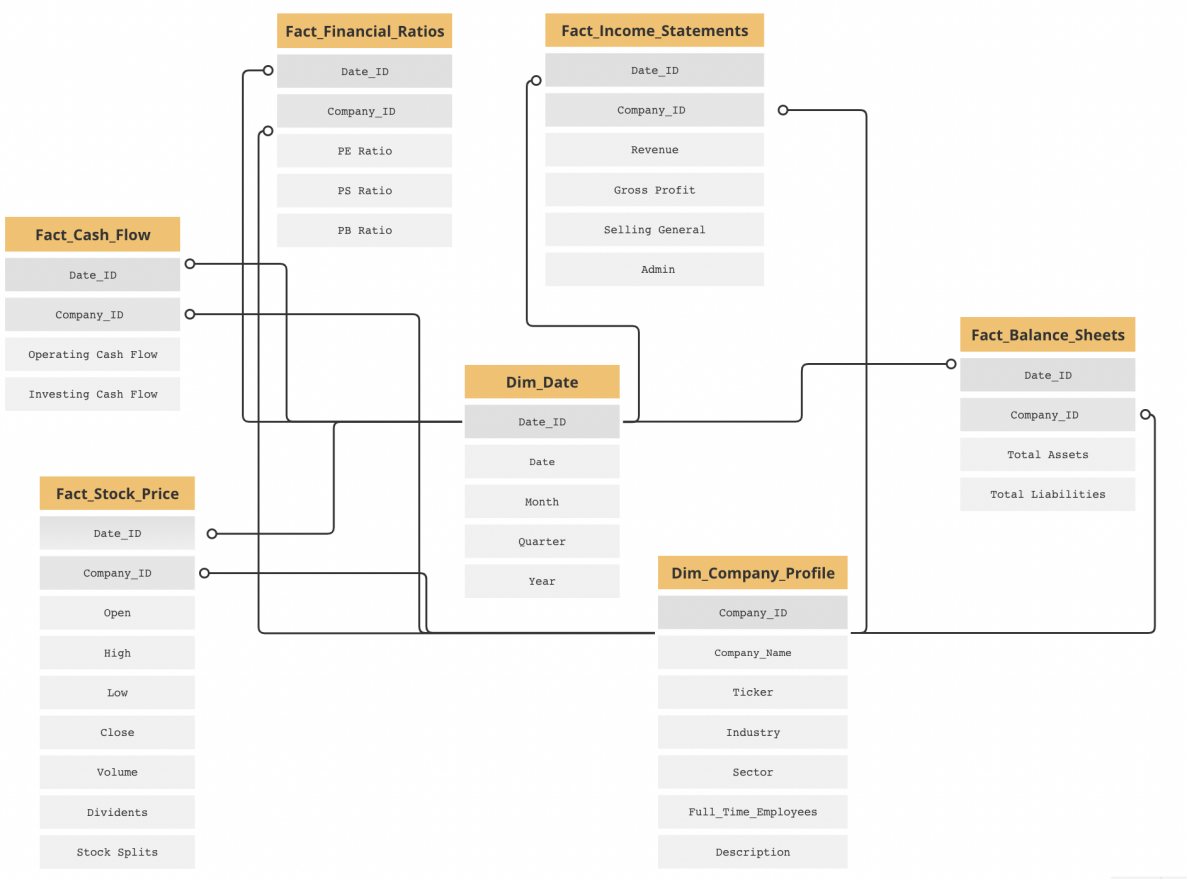
1. *Fact_Stock_Price:*
   - Date_ID (FK), Company_ID (FK), Open, High, Low, Close, Volume, Dividends, Stock Splits
2. *Fact_Financial_Ratios:*
   - Date_ID (FK), Company_ID (FK)
   - All the financial ratios (PE Ratio, PS Ratio, PB Ratio, etc.)
3. *Fact_Income_Statements:*
   - Date_ID (FK), Company_ID (FK)
   - All income statement items (Revenue, Gross Profit, Selling General & Admin, etc.)
4. *Fact_Balance_Sheets:*
   - Date_ID (FK), Company_ID (FK)
   - All balance sheet items (Total Assets, Total Liabilities, etc.)
5. *Fact_Cash_Flow:*
   - Date_ID (FK), Company_ID (FK)
   - All cash flow items (Operating Cash Flow, Investing Cash Flow, etc.)

1. *Dim_Date:*
   - Date_ID (PK), Date, Month, Quarter, Year
2. *Dim_Company_Profile:*
   - Company_ID (PK), Company_Name, Ticker, Industry, Sector, Full_Time_Employees, Description

Star Schema

The dimensions and facts have been selected based on their relevance to the business domain and the project's objectives. The dimension tables provide crucial contextual information such as dates, months, quarters, years, company names, tickers, industries,

sectors, and other descriptive attributes necessary for future data analysis. The facts encompass various financial aspects, including stock prices, financial ratios, income statements, balance sheets, and cash flows, enabling in-depth analysis of each company's performance.



Once our raw data is processed, the objective is to have data files that follow the structure of this star schema.

4. ETL Pipeline

ETL Step 0: Identification of financial institutions

The primary task in our data extraction process involved accurately identifying the companies from which we required data. Specifically, we focused on all publicly traded financial institutions. To accomplish this, we compiled comprehensive lists of financial institutions from reputable online resources, including but not limited to

StockAnalysis.com, Yahoo Finance, and renowned financial indices such as the S&P 500 and FTSE Russell 2000.

Through a meticulous process, we amalgamated the unique lists from these diverse sources, thereby integrating their collective information into a unified list. This rigorous methodology resulted in a comprehensive catalogue of 1093 companies, providing us with a robust list for our ETL, serving as the foundational layer for our data warehouse.

ETL Step 1: Extract

In the first stage of our ETL pipeline, we aimed to extract data from two primary sources: Yahoo Finance and stockanalysis.com. Yahoo Finance provided us with daily stock prices for various companies, and a detailed profile of each company. The second source, stockanalysis.com, provided us with various financial statements for each company, such as, balance sheet, cash flow statements, income statements and financial ratios, each of these financial statements can be found at the quarterly and annual level.

To carry out this process, we built a Python-based data extraction system leveraging two main libraries: *BeautifulSoup* and *yahooquery*. *BeautifulSoup* was employed to perform web scraping on stockanalysis.com, while *yahooquery* was used to connect to Yahoo Finance's API and download relevant data.

The primary Python classes used in this extraction process are the `StockData` and `DataExtractor` classes.

The `StockData` class focuses on retrieving data for a single stock symbol. Here's a brief summary of its functionality:

- It has an initialization method (`__init__`) that sets up the stock symbol and builds the URLs required for scraping the financial statements.

- It constructs a dictionary (`__create_dict_company`) to store all data related to the company.

- It provides methods for scraping financial statement data from stockanalysis.com (`__scrape_stock_analysis` and `scrape_financial_statements`).

- It retrieves company profile and stock price data from Yahoo Finance (`retrieve_profile_stock_price`).

- Finally, it saves all the scraped and retrieved data in JSON format (`save_to_json`).

The `DataExtractor` class operates on a list of stock symbols and uses the `StockData` class to retrieve and store data for each symbol.

- The `__extract_financial_price_data` method uses an instance of `StockData` to scrape financial data and retrieve stock price and company profile data.

- The `__extract_location` method takes the company's address and finds the latitude and longitude using the ArcGIS API.

- The `__update_stock_data` method checks if there's newer stock price data available. If there is, it retrieves the new data and appends it to the existing dataset.

- The `extract_data` method is the main driver method, which iterates over a list of stock symbols and extracts all the necessary data for each one.

An important aspect of this code is that it includes mechanisms for error handling. For instance, if data for a specific symbol is not available or if the system encounters a connection error, the code will print an appropriate message and continue with the next symbol. It also implements a retry system to attempt the data extraction a certain number of times before moving on.

Finally, to ensure our data is up-to-date, we also developed functionality to check for new stock price data and append it to the existing dataset, thus maintaining the completeness and relevance of our data. This is crucial, as financial data can change frequently, and having the most recent data is important for analysis.

ETL Step 2: Transform

In the data transformation stage of our ETL pipeline, we have created and utilized a series of Python classes to conduct the cleaning and processing of our raw financial data. This raw data was initially obtained from a collection of JSON files. Each Python

class was strategically designed to clean and transform a specific type of our raw financial data, namely financial statements, profiles, and stock data. These classes execute a range of cleaning operations, including transposing data, verifying duplicated data, cleaning column names, handling missing values, and others.

To perform these transformations, we make use of the `FinancialDataTransformer` class. This class takes as input the list of stock symbols to be processed and saves the transformed data into CSV files. This process is broken down into three key steps:

1. Initialization: Create an instance of the `FinancialDataTransformer` class with the list of stock symbols as an argument.

2. Transformation: Invoke the `transform` method on the instance. This method calls upon our data cleaning, deduplication, and formatting classes to perform the necessary transformations on the data.

3. Data Export: The transformed data for each stock symbol is then saved into CSV files using the `save_to_csv` method.

Our data transformation implementation encompasses several cleaning, deduplication, and format revision operations, which are briefly explained below:

- Financial statement ratio data cleaning (`FinStmtRatioCleaner`): This class transposes the financial ratio data, cleans the column names by replacing special characters with underscores, and converts column data to numeric values. It also formats the date column and resets the DataFrame index after cleaning.

- Company profile data cleaning (`ProfileCleaner`): Given a dictionary of profile data, this class selects specific keys and creates a DataFrame with the cleaned profile data.

- Stock data cleaning (`StockDataCleaner`): This class converts a dictionary of stock data into a DataFrame and formats the date column.

- Cleaning all data (`DataCleaner`): This class loads the financial data from a JSON file and applies the cleaning process using the aforementioned classes for each type of data. The cleaned data frames are then stored in the `data` attribute of the `DataCleaner` instance.

The `FinancialDataTransformer` class uses `DataCleaner` to load, clean, and transform the financial data for all stock symbols. It combines the data frames of the same type from all symbols and saves the resulting data frames to CSV files. Each CSV file corresponds to a key in the `data` dictionary, reflecting the type of financial data it contains.

Through these classes and their methods, we are able to perform various data operations such as transposing data, cleaning column names, converting data types, formatting dates, and joining data frames. The ultimate goal is to transform our raw financial data, initially in JSON format, into a more structured and standardized format that is easier to process and analyze. The transformed data is saved into CSV files that contains the structure of the star schema.

To summarize, the combination of these classes and their methods forms a powerful and flexible pipeline for the cleaning and transformation of financial data, ensuring the data is in the best format and structure for subsequent loading, processing, and analysis.

ETL Step 3: Load

Step 3.1: Load Data to Amazon S3:

The load phase of our ETL process involved taking our carefully processed financial data, structured as per the defined star schema, and securely storing it for subsequent analysis and use.

We leveraged the services of Amazon S3, known for its scalability, data availability, and security, for this critical step. Amazon S3 provides a robust and resilient platform, enabling large scale data storage and easy retrieval, qualities essential to our operation.

Within our designated S3 bucket, we established two distinct subfolders - one for 'raw data' and another for 'processed data'. This dual-folder approach was essential for a couple of reasons.
The 'raw data' subfolder stores our original data in its unaltered form, saved as JSON files. By maintaining this original version of our data, we ensure a permanent record that can serve as a reference point, or be used for auditing or troubleshooting

purposes. It also allows us the flexibility to reprocess the data if needed in the future, without having to re-extract it.

Our 'processed data', following the structure defined by the star schema, is saved in CSV format and loaded into the 'processed data' subfolder. This separate storage of processed data ensures a streamlined access to cleaned, structured, and analysis-ready data, thus facilitating rapid and efficient query execution.

This procedure of loading data into S3 forms the first part of our load process. It guarantees that both our original and processed data are organized, secured, and ready for further loading into our final data warehouse, Amazon Redshift, the specifics of which will be discussed in the following section.

The Python script for the loading phase utilizes the boto3 library, Amazon's own software development kit (SDK), to interact with AWS services. This library makes it easy to integrate our Python script with AWS S3.

Here are the implementation of data extraction in details:

1. The script begins with the definition of a class named DataLoader. This class includes several crucial methods for establishing and manipulating data within Amazon S3, all of which leverage the functionality of boto3.

2. The `create_bucket` method, for instance, employs the boto3 client for S3 to create a new bucket that will serve as our primary repository for raw and processed data. Similarly, the `create_folder` method uses boto3 to establish subfolders within this bucket, allowing for organized segregation of raw and processed data.

3. Once our data storage structure is set, we proceed with the data loading operation, again using boto3. The `upload_file` method uploads files from our local directories to the appropriate S3 bucket. It iterates through all JSON files containing raw data and CSV files of processed data, uploading these to their corresponding subfolders within the S3 bucket.

Besides these operations, the DataLoader class also houses methods for file downloading, file deletion, and bucket deletion - `download_file`, `delete_file`, and `delete_bucket` respectively. These operations, although not utilized in the primary

loading process, are available for future need, demonstrating the versatility and control offered by our code created with boto3.

Overall, the boto3 library plays a central role in our loading process, providing a direct, efficient link between our Python environment and Amazon S3 storage. As a result, we successfully accomplish the first part of our load operation, positioning our raw and processed data securely in Amazon S3 for subsequent transfer to Amazon Redshift.

Step 3.2: Load Data to Amazon S3:

In the second phase of the Load part of our ETL process, we shift our focus towards building the data warehouse, which serves as the repository for our transformed data. For this, we take advantage of AWS's scalable data warehouse service, Amazon Redshift.

Amazon Redshift is a fully managed, petabyte-scale data warehouse service in the cloud. It allows us to analyze data using standard SQL and existing Business Intelligence (BI) tools. Specifically, we utilized Amazon Redshift's serverless variant, which provides on-demand, scalable capacity without the need to manage the infrastructure.

Our first step in constructing the data warehouse was to create a database. This database serves as the primary container for our processed financial data. Within this database, we then set up a schema called ‘financial_radar’ that adheres to our data structure. The schema provides a logical framework for our data and facilitates efficient data retrieval and manipulation.

With our database and schema established, we then turned to the creation of our tables. We employed the Redshift Query Editor to create multiple tables corresponding to the different segments of our financial data, such as stock prices, income statement data, company profiles, and cash flow statements. The table structure was designed in accordance with our chosen star schema.

Each table was meticulously crafted to contain specific financial information. For instance, the 'balance_sheet_quarterly' table contained columns for aspects like cash and equivalents, short-term investments, and receivables, among others. On the other hand, the 'cash_flow_statement_quarterly' table detailed net income, depreciation and amortization, share-based compensation, and other elements of a standard cash flow statement.

Similarly, the 'income_statement_quarterly' and 'ratios_quarterly' tables encapsulated essential components of an income statement and financial ratios, respectively. The 'profile' table contained fields for each company's industry sector, long business summary, address, and even geographic coordinates. The 'stock_price' table, crucial for any financial analysis, recorded daily open, high, low, and closing prices, alongside trade volumes and dividend details.

Importantly, if these tables existed from previous loads, we ensured to first drop the old tables before creating new ones, guaranteeing clean, up-to-date information in our data warehouse.

With the tables in place, we could finally perform the critical task of loading the processed data from Amazon S3 into our Redshift data warehouse. Using SQL COPY commands, we loaded data from the 'processed' folder within our S3 bucket to the respective tables in the 'financial_radar' schema. Each COPY command specified the relevant table in Redshift, the data file's location in the S3 bucket, and the IAM role with required permissions to access and transfer data between S3 and Redshift.

The IAM role, identified by the Amazon Resource Name (ARN), was specified in the COPY command as 'arn:aws:iam::aws_account_id:role/redshift_s3_role'. This role authorized the necessary actions in S3 and Redshift, ensuring the smooth flow of data between these two services.
Additionally, the COPY command included formatting instructions such as 'FORMAT AS CSV', 'IGNOREHEADER 1', and 'DELIMITER ','', which guided Redshift on how to interpret the incoming data.

Ultimately, our ETL pipeline successfully created a comprehensive data warehouse in Amazon Redshift, loaded with data processed in Amazon S3, and ready for querying and data analysis. By leveraging AWS's robust and scalable services, we created an efficient, reliable, and scalable infrastructure for our financial data management and analytics, enabling us to perform deep, meaningful analyses on financial data.

5. Query Examples

In this section we present some simple SQL queries that one might use to identify firms with low performance using our data warehouse:

1. Identify firms with low Return on Assets (RoA) from ratios_quarterly:

```
8
9   SELECT
10      DISTINCT symbol
11  FROM
12      financial_radar.ratios_quarterly
13  WHERE
14      return_on_assets_roa < 0.01
15  LIMIT 5;
16
```

Result 1 (5)

symbol
BANX
BPRN
ING
CET

2. Identify firms with low Earnings Yield from ratios_quarterly:

```
32  SELECT
33      DISTINCT symbol,
34      earnings_yield
35  FROM
36      financial_radar.ratios_quarterly
37  WHERE
38      earnings_yield < 0.02
39  LIMIT 5;
```

Result 1 (4)

symbol	earnings_yield
CET	0
TYG	0
ING	0
BANX	0

3. Identify firms with high Debt to Equity Ratio in their last quarter available in the data:

```
WITH latest_quarters AS (
    SELECT
        symbol,
```

```
        date,
        debt_or_equity_ratio,
        ROW_NUMBER() OVER (PARTITION BY symbol ORDER BY date DESC) as
rn
    FROM
        financial_radar.ratios_quarterly
)
SELECT
    symbol,
    date,
    debt_or_equity_ratio
FROM
    latest_quarters
WHERE
    rn = 1 AND
    debt_or_equity_ratio > 2;
```

symbol	date	debt_or_equity_ratio
AGM	2023-03-31	20.28
AMBC	2023-03-31	2.84
ARES	2023-03-31	8.13
BKU	2023-03-31	3.33
BSIG	2023-03-31	52.72
COHN	2023-03-31	9.79
FFWM	2023-03-31	2.02
GOGN	2023-03-31	2.69
GS	2023-03-31	6.48

4. Finds companies where the closing price at the end of the last month was at least 10% lower than at the beginning of the mont:

```
WITH monthly_prices AS (
    SELECT
        symbol,
        DATE_TRUNC('month', date::date) as month,
```

```
        FIRST_VALUE(close) OVER (PARTITION BY symbol, DATE_TRUNC('month',
date::date) ORDER BY date ROWS BETWEEN UNBOUNDED PRECEDING AND
CURRENT ROW) as start_price,
        LAST_VALUE(close) OVER (PARTITION BY symbol, DATE_TRUNC('month',
date::date) ORDER BY date ROWS BETWEEN CURRENT ROW AND
UNBOUNDED FOLLOWING) as end_price
    FROM
        financial_radar.stock_price
),
last_month_prices AS (
    SELECT
        symbol,
        month,
        start_price,
        end_price,
        ROW_NUMBER() OVER (PARTITION BY symbol ORDER BY month DESC) as
rn
    FROM
        monthly_prices
)
SELECT
    symbol,
    month,
    start_price,
    end_price
FROM
    last_month_prices
WHERE
    rn = 1 AND
    ((start_price - end_price) / start_price) >= 0.10;
```

☐	symbol	month	start_price	end_price
☐	BUR	2023-06-01 00:00:00	13.520000457763672	12.010000228881836
☐	PMTS	2023-06-01 00:00:00	26.309999465942383	22.850000381469727
☐	AVAC	2023-06-01 00:00:00	10.569999694824219	9.100000381469727
☐	OCCI	2023-06-01 00:00:00	9.609999656677246	8.40999984741211

6. Conclusion & Future Work

What went well

1. The project had a well-defined topic and objective. We had a clear understanding of the required data and its desired format for the final data warehouse. This allowed us to develop an appropriate and efficient plan for data collection, transformation, and integration.

2. We were fortunate to have access to adequate and reliable data sources. This included financial statements, stock data, financial ratios, and company information from trustworthy sources like Yahoo Finance and a Stock Analysis Webpage. The availability of relevant and reliable data was instrumental in the success of our project.

3. Our team members had clearly defined roles and responsibilities. We effectively divided tasks based on individual strengths and skills. Transparent and open communication among team members fostered collaboration and ensured that everyone remained aligned and informed.

4. Throughout the project, we had the opportunity to learn and gain expertise in various data engineering tools and techniques. This hands-on experience provided valuable insights and enhanced our knowledge in the field of data engineering.

What would you do differently

1. Incorporate additional data sources: Considering storage limitations, we made the decision not to incorporate every data detail as outlined in the Appendix. However, in the future, we can explore the possibility of adding additional data feeds to further enrich our analysis. By incorporating more data sources, we can enhance the depth and breadth of our analysis, leading to a more comprehensive and robust understanding of the factors that influence the predictability of financial institution failures.

2. Expand the time range: We can extend the time range of our analysis to cover a longer period. By incorporating historical data, you can identify long-term trends

and patterns that may contribute to a better understanding of the factors leading to financial institution failures.

3. Implement advanced analytics techniques: We could consider leveraging advanced analytics techniques such as machine learning, natural language processing, or predictive modeling to extract deeper insights from the data in the data transformation step.

Appendix

Data obtained in the extract part of the ETL pipeline:

Stock Information:

- Symbol (VARCHAR):
    - The unique identifier or ticker symbol for the publicly traded company.
- Company Name (VARCHAR):
    - The official name of the publicly traded company.
- Industry Name (VARCHAR):
    - The specific industry in which the company operates.

Company Profile: Obtained via Yahoo Finance API

- Basic Company Information:
    - Symbol (VARCHAR):
        - Identifier of the company
    - address1 (VARCHAR):
        - The street address of the company's HQ location.
    - city (VARCHAR):
        - The city in which the company's HQ reside.
    - state (VARCHAR):
        - The state in which the company's HQ reside.
    - zip (VARCHAR):
        - The zip code of the company's HQ.
    - country (VARCHAR):
        - The country in which the company's HQ reside.
    - phone (VARCHAR):

        - The main contact phone number for the company.
    - website (VARCHAR):
        - The official website URL for the company.
    - latitude (NUMERIC):
        - The latitude of the company's HQ.
    - longitude (NUMERIC):
        - The longitude of the company's HQ.
- Company Industry and Sector:
    - industry (VARCHAR):
        - The specific industry in which the company operates.
    - industryDisp (VARCHAR):
        - A displayed version of the specific industry in which the company operates.
    - sector (VARCHAR):
        - The general sector in which the company operates.
- Company Overview:
    - longBusinessSummary (TEXT):
        - A detailed description of the company's business model and operations.
- Company Employee Information:
    - fullTimeEmployees (NUMERIC):
        - The number of full-time employees working for the company.
- Company Risk Information:
    - auditRisk (NUMERIC):
        - The risk level associated with the company's audit, higher numbers imply higher risk.
    - boardRisk (NUMERIC):
        - The risk level associated with the company's board of directors, higher numbers imply higher risk.
    - compensationRisk (NUMERIC):
        - The risk level associated with the company's compensation structure, higher numbers imply higher risk.
    - shareHolderRightsRisk (NUMERIC):
        - The risk level associated with shareholder rights, higher numbers imply higher risk.
    - overallRisk (NUMERIC):
        - The overall risk level associated with the company, higher numbers imply higher risk.

Stock Prices Data: Obtained via Yahoo Finance API

- Stock Trading Data:
    - symbol (VARCHAR):
        - The unique identifier or ticker symbol for the publicly traded company.
    - date (DATE):
        - The specific date of the recorded trading data.
    - open (NUMERIC):
        - The opening price of the stock for the trading day.
    - high (NUMERIC):
        - The highest price of the stock reached in the trading day.
    - low (NUMERIC):
        - The lowest price of the stock reached in the trading day.
    - close (NUMERIC):
        - The closing price of the stock for the trading day.
    - volume (NUMERIC):
        - The number of shares of the stock that were traded during the trading day.
    - adjclose (NUMERIC):
        - The adjusted closing price of the stock for the trading day. This is typically adjusted for corporate actions such as dividends, stock splits, and new stock offerings.
    - dividends (NUMERIC):
        - The amount of dividends paid out on that trading day.
    - splits (NUMERIC):
        - The split ratio applied to the stock on that trading day. A split ratio of 2, for example, would represent a 2-for-1 stock split.

Financial Statements: We obtained data both at the annual and quarterly level

- Balance Sheet Data:
    - Symbol (VARCHAR):
        - The unique identifier or ticker symbol for the publicly traded company.
    - Year (DATE):

        - If it is the annual data, the year in which the data was recorded.
    - Quarter (DATE):
        - If it is the quarterly data, the quarter in which the data was recorded.
    - Cash & Equivalents (NUMERIC):
        - The company's cash on hand, as well as assets that can be readily converted into cash.
    - Short-Term Investments (NUMERIC):
        - Investments that are expected to be converted into cash within a year.
    - Cash & Cash Equivalents (NUMERIC):
        - The sum of cash on hand and short-term investments.
    - Cash Growth (NUMERIC):
        - The percentage growth in cash and cash equivalents from the previous year.
    - Receivables (NUMERIC):
        - Money owed to the company by its customers from sales made on credit.
    - Other Current Assets (NUMERIC):
        - Other assets that can or will be converted into cash within one year.
    - Total Current Assets (NUMERIC):
        - The total value of all assets that are expected to be converted into cash within one year.
    - Property, Plant & Equipment (NUMERIC):
        - The company's tangible fixed assets, such as land, buildings, and equipment.
    - Goodwill and Intangibles (NUMERIC):
        - The value of a company's brand name, solid customer base, good customer relations, good employee relations, and any patents or proprietary technology.
    - Other Long-Term Assets (NUMERIC):
        - Other assets that cannot be converted into cash within one year.
    - Total Long-Term Assets (NUMERIC):
        - The total value of long-term assets.
    - Total Assets (NUMERIC):
        - The total value of all the company's assets.
    - Accounts Payable (NUMERIC):

        - The amount of short-term debt the company owes to its suppliers or creditors.
    - Current Debt (NUMERIC):
        - The portion of debt due to be paid within one year.
    - Other Current Liabilities (NUMERIC):
        - Other obligations due within one year.
    - Total Current Liabilities (NUMERIC):
        - The total amount of liabilities due within one year.
    - Long-Term Debt (NUMERIC):
        - The portion of debt that is due more than one year from now.
    - Other Long-Term Liabilities (NUMERIC):
        - Other obligations due after one year.
    - Total Long-Term Liabilities (NUMERIC):
        - The total amount of liabilities due after one year.
    - Total Liabilities (NUMERIC):
        - The total amount of both current and long-term liabilities.
    - Total Debt (NUMERIC):
        - The total amount of debt, both short-term and long-term.
    - Debt Growth (NUMERIC):
        - The percentage growth in debt from the previous year.
    - Common Stock (NUMERIC):
        - The total value of all outstanding common shares.
    - Retained Earnings (NUMERIC):
        - The portion of net profits not paid out as dividends but instead reinvested in the core business or used to pay off debt.
    - Comprehensive Income (NUMERIC):
        - The total change in equity for a reporting period other than transactions with owners (like dividends and share repurchases).
    - Shareholders' Equity (NUMERIC):
        - The net value of the company, calculated by subtracting total liabilities from total assets.
    - Total Liabilities and Equity (NUMERIC):
        - The sum of total liabilities and total equity. This should be equal to total assets.
    - Net Cash / Debt (NUMERIC):
        - The difference between a company's cash and its total debt.
    - Net Cash Per Share (NUMERIC):
        - The net cash per outstanding share of common stock.
    - Working Capital (NUMERIC):

        - A measure of a company's operational liquidity. It's calculated as current assets minus current liabilities.
    - Book Value Per Share (NUMERIC):
        - The value of a company if all its assets were sold and all its liabilities were paid off, divided by the number of outstanding shares.
- Cash Flow Statement Data:
    - Symbol (VARCHAR):
        - The unique identifier or ticker symbol for the publicly traded company.
    - Year (DATE):
        - If it is the annual data, this variable is the year in which the data was recorded.
    - Quarter (DATE):
        - If it is the quarterly data, this variable is the quarter in which the data was recorded.
    - Net Income (NUMERIC):
        - The company's total profit or loss.
    - Depreciation & Amortization (NUMERIC):
        - The decrease in value of the company's fixed assets due to wear and tear, and the gradual reduction in the value of intangible assets.
    - Other Operating Activities (NUMERIC):
        - The cash inflows or outflows related to other operations that are not categorized under net income or depreciation & amortization.
    - Operating Cash Flow (NUMERIC):
        - The cash generated by a company's regular business operations.
    - Operating Cash Flow Growth (NUMERIC):
        - The percentage growth in operating cash flow from the previous year.
    - Change in Investments (NUMERIC):
        - The net change in the company's investments over the period.
    - Other Investing Activities (NUMERIC):
        - The cash inflows or outflows related to other investing activities that are not categorized under change in investments.
    - Investing Cash Flow (NUMERIC):
        - The cash inflows or outflows from investments.
    - Dividends Paid (NUMERIC):
        - The total dividends paid to shareholders during the period.

    - Share Issuance / Repurchase (NUMERIC):
        - The net cash inflow (issuance) or outflow (repurchase) from equity financing.
    - Debt Issued / Paid (NUMERIC):
        - The net cash inflow (debt issued) or outflow (debt paid) from debt financing.
    - Other Financing Activities (NUMERIC):
        - The cash inflows or outflows related to other financing activities that are not categorized under dividends paid, share issuance/repurchase, or debt issued/paid.
    - Financing Cash Flow (NUMERIC):
        - The net cash inflow or outflow from all financing activities.
    - Net Cash Flow (NUMERIC):
        - The total change in a company's cash and cash equivalents during a period.
    - Free Cash Flow (NUMERIC):
        - The cash a company generates from its operations that is free to be distributed to investors, calculated as operating cash flow minus capital expenditures.
    - Free Cash Flow Growth (NUMERIC):
        - The percentage growth in free cash flow from the previous year.
    - Free Cash Flow Margin (NUMERIC):
        - The free cash flow divided by total revenue, expressed as a percentage. It represents how much free cash flow is generated for each dollar of revenue.
    - Free Cash Flow Per Share (NUMERIC):
        - The free cash flow divided by the total number of shares outstanding. It indicates the amount of free cash flow available per share.
- Income Statement Data:
    - Symbol (VARCHAR):
        - The unique identifier or ticker symbol for the publicly traded company.
    - Year (DATE):
        - If it is the annual data, this variable is the year in which the data was recorded.
    - Quarter (DATE):
        - If it is the quarterly data, this variable is the quarter in which the data was recorded.

- Revenue (NUMERIC):
  - The total money made from selling goods and services.
- Revenue Growth (YoY) (NUMERIC):
  - The percentage change in revenue from the previous year.
- Gross Profit (NUMERIC):
  - The profit a company makes after deducting the costs associated with making and selling its products, or the costs associated with providing its services.
- Selling, General & Admin (NUMERIC):
  - The sum of all direct and indirect selling expenses and all general and administrative expenses of a company.
- Other Operating Expenses (NUMERIC):
  - The category of expenditure that a business incurs as a result of performing its normal business operations.
- Operating Expenses (NUMERIC):
  - The sum of a business's operating expenses for a period of time.
- Operating Income (NUMERIC):
  - The profit realized from a business's operations.
- Other Expense / Income (NUMERIC):
  - Expenses or incomes that are unusual, not recurring, or not directly tied to the company's normal operations.
- Pretax Income (NUMERIC):
  - The income that a company makes before deducting the income taxes it owes.
- Income Tax (NUMERIC):
  - The amount of tax owed to the government based on reported income.
- Net Income (NUMERIC):
  - The company's total earnings or profit.
- Net Income Growth (NUMERIC):
  - The year-over-year growth rate of net income.
- Shares Outstanding (Basic) (NUMERIC):
  - The number of shares that are currently held by all its shareholders, including institutional investors.
- Shares Outstanding (Diluted) (NUMERIC):
  - The number of shares that would be outstanding if all possible sources of conversion, such as convertible bonds and stock options, were exercised.
- Shares Change (NUMERIC):

        - The percentage change in the number of outstanding shares from the previous year.
    - EPS (Basic) (NUMERIC):
        - The portion of a company's profit allocated to each outstanding share of common stock.
    - EPS (Diluted) (NUMERIC):
        - A calculation for earnings per share if all convertible securities were exercised.
    - EPS Growth (NUMERIC):
        - The percentage change in EPS from the previous year.
    - Free Cash Flow Per Share (NUMERIC):
        - The amount of free cash flow (cash a company has left over after it has paid expenses, interest, taxes, and long-term investments) allocated to each outstanding share of common stock.
    - Dividend Per Share (NUMERIC):
        - The total dividends declared divided by the number of outstanding shares.
    - Dividend Growth (NUMERIC):
        - The year-over-year growth rate of dividends per share.
    - Gross Margin (NUMERIC):
        - Gross profit divided by revenues, expressed as a percentage.
    - Operating Margin (NUMERIC):
        - Operating income divided by revenues, expressed as a percentage.
    - Profit Margin (NUMERIC):
        - Net income divided by revenues, expressed as a percentage.
    - Free Cash Flow Margin (NUMERIC):
        - Free cash flow divided by total revenue, expressed as a percentage.
    - Effective Tax Rate (NUMERIC):
        - The average rate at which a company's pre-tax profits are taxed.
    - EBITDA (NUMERIC):
        - Earnings before interest, taxes, depreciation, and amortization.
    - EBITDA Margin (NUMERIC):
        - EBITDA divided by total revenue, expressed as a percentage.
    - Depreciation & Amortization (NUMERIC):
        - The decrease in value of the company's assets over time.
    - EBIT (NUMERIC):

        - Earnings Before Interest and Taxes, a measure of a firm's profit that includes all incomes and expenses except interest expenses and income tax expenses.
    - EBIT Margin (NUMERIC):
        - EBIT divided by total revenue, expressed as a percentage.
- Ratios:
    - Symbol (VARCHAR):
        - The unique identifier or ticker symbol for the publicly traded company.
    - Year:
        - If it is the annual data, this variable is the year in which the data was recorded.
    - Quarter:
        - If it is the quarterly data, this variable is the quarter in which the data was recorded.
    - Market Capitalization:
        - This is the total value of a company's outstanding shares of stock, calculated by multiplying the share price by the number of outstanding shares.
    - Market Cap Growth:
        - This shows the percentage change in the market capitalization from year to year.
    - Enterprise Value:
        - This is a measure of a company's total value, considering not only its equity (like market capitalization does) but also its debt and cash.
    - PE Ratio (Price/Earnings Ratio):
        - This ratio compares a company's current share price to its earnings per share (EPS).
    - PS Ratio (Price/Sales Ratio):
        - This ratio compares a company's market capitalization with its annual revenue.
    - PB Ratio (Price/Book Ratio):
        - This ratio compares a company's market capitalization to its book value (which is assets minus liabilities).
    - P/FCF Ratio (Price/Free Cash Flow Ratio):
        - This compares a company's market capitalization to its free cash flow.
    - P/OCF Ratio (Price/Operating Cash Flow Ratio):

        - This compares a company's market capitalization to its operating cash flow.
    - EV/Sales Ratio:
        - This compares a company's enterprise value to its annual revenue.
    - EV/EBITDA Ratio:
        - This compares a company's enterprise value to its earnings before interest, tax, depreciation, and amortization (EBITDA).
    - EV/EBIT Ratio:
        - This compares a company's enterprise value to its earnings before interest and tax (EBIT).
    - EV/FCF Ratio:
        - This compares a company's enterprise value to its free cash flow.
    - Debt / Equity Ratio:
        - This measures a company's financial leverage by comparing its total debt with its total equity.
    - Debt / EBITDA Ratio:
        - This compares a company's total debt to its EBITDA.
    - Debt / FCF Ratio:
        - This compares a company's total debt to its free cash flow.
    - Asset Turnover:
        - This ratio measures how efficiently a company uses its assets to generate revenue.
    - Return on Equity (ROE):
        - This ratio measures the profitability of a company in relation to the equity held by shareholders.
    - Return on Assets (ROA):
        - This ratio measures the profitability of a company in relation to its total assets.
    - Return on Capital (ROIC):
        - This ratio measures the return that an investment generates for capital contributors, i.e., bondholders and shareholders.
    - Earnings Yield:
        - This is the inverse of the P/E ratio and shows the percentage of each dollar invested in the stock that was earned by the company.
    - FCF Yield:
        - This is the inverse of the P/FCF ratio and shows the percentage of free cash flow relative to the market capitalization of the company.
    - Dividend Yield:

        - This shows how much a company pays out in dividends each year relative to its share price.
    - Payout Ratio:
        - This measures the proportion of earnings a company pays shareholders in dividends.
    - Buyback Yield / Dilution:
        - This measures the percentage change in the number of outstanding shares due to stock buybacks or dilution.
    - Total Shareholder Return:
        - This measures the return of an investment in a company's stock, including both price appreciation and dividends.