ETL

Super Financial Bros

2/14/2022

Introduction

This project examines the domestic stock market in the United States as represented by Vanguard proxy index funds. Specifically, it investigates how the stock market responds to different domestic economic contexts and how those responses may vary by sector. It also analyzes how investment risk and return vary by sector and how historical sector trends compare to historical national total revenue trends for the years 2006 through 2020. The project sources data from the *Economic Indicators* and the *Stock Time Series* APIs provided by Alpha Vantage (Alpha Vantage, 2022a; Alpha Vantage, 2022b) as well as Census data from the *Annual Survey of State Government Finances Tables* for the years 2006 through 2020 (United States Census Bureau, 2021; United States Census Bureau, 2022). Each of these datasets contributes to the project question but is insufficient on its own. They also contain extra data irrelevant to the project. Data transformation is necessary to effectively filter the datasets for appropriate data and merge them into one comprehensive table for analytics.

Data Sources

Alpha Vantage. (2022a). *Economic Indicators* [API Dataset]. Retrieved from API Documentation | Alpha Vantage.

Alpha Vantage. (2022b). *Stock Time Series* [API Dataset]. Retrieved from API Documentation | Alpha Vantage.

United States Census Bureau. (2021a). *2006 Annual Survey of State Government Finances Tables* [Data set]. Retrieved January 31, 2022, from 2006 Annual Survey of State Government Finances Tables (census.gov).

United States Census Bureau. (2021b). *2007 Annual Survey of State Government Finances Tables* [Data set]. Retrieved January 31, 2022, from 2007 Annual Survey of State Government Finances Tables (census.gov).

United States Census Bureau. (2021c). *2008 Annual Survey of State Government Finances Tables* [Data set]. Retrieved January 31, 2022, from 2008 Annual Survey of State Government Finances Tables (census.gov).

United States Census Bureau. (2021d). *2009 Annual Survey of State Government Finances Tables* [Data set]. Retrieved January 31, 2022, from 2009 Annual Survey of State Government Finances Tables (census.gov).

United States Census Bureau. (2021e). *2010 Annual Survey of State Government Finances Tables* [Data set]. Retrieved January 31, 2022, from 2010 Annual Survey of State Government Finances Tables (census.gov).

United States Census Bureau. (2021f). *2011 Annual Survey of State Government Finances Tables* [Data set]. Retrieved January 31, 2022, from 2011 Annual Survey of State Government Finances Tables (census.gov).

United States Census Bureau. (2021g). *2012 Annual Survey of State Government Finances Tables* [Data set]. Retrieved January 31, 2022, from 2012 Annual Survey of State Government Finances (census.gov).

United States Census Bureau. (2021h). *2013 Annual Survey of State Government Finances Tables* [Data set]. Retrieved January 31, 2022, from 2013 Annual Survey of State Government Finances Tables (census.gov).

United States Census Bureau. (2021i). *2014 Annual Survey of State Government Finances Tables* [Data set]. Retrieved January 31, 2022, from 2014 Annual Survey of State Government Finances Tables (census.gov).

United States Census Bureau. (2021j). *2015 Annual Survey of State Government Finances Tables* [Data set]. Retrieved January 31, 2022, from 2015 Annual Survey of State Government Finances Tables (census.gov).

United States Census Bureau. (2021k). *2016 Annual Survey of State Government Finances Tables* [Data set]. Retrieved January 31, 2022, from 2016 Annual Survey of State Government Finances Tables (census.gov).

United States Census Bureau. (2022a). *2017 Annual Survey of State Government Finances Tables* [Data set]. Retrieved January 31, 2022, from 2017 Annual Survey of State Government Finances Tables (census.gov).

United States Census Bureau. (2022b). *2018 Annual Survey of State Government Finances Tables* [Data set]. Retrieved January 31, 2022, from 2018 Annual Survey of State Government Finances Tables (census.gov).

United States Census Bureau. (2022c). *2019 Annual Survey of State Government Finances Tables* [Data set]. Retrieved January 31, 2022, from 2019 Annual Survey of State Government Finances Tables (census.gov).

United States Census Bureau. (2022d). *2020 Annual Survey of State Government Finances Tables* [Data set]. Retrieved January 31, 2022, from 2020 Annual Survey of State Government Finances Tables (census.gov).

Extraction

Economic Indicators and *Stock Time Series* APIs (Alpha Vantage, 2022a; Alpha Vantage, 2022b):

These data were found through a web search for APIs that have data on the stock market. The Alpha Vantage *Economic Indicators* and *Stock Time Series* APIs were chosen because they contain data about the economic features (GDP, interest, unemployment rate, consumer sentiment, and average treasury yield) and index funds (VTI, VGT, VCR, VIS, VHT, VFH) of interest they are provided by the same source and thus have similar documentations and usages. Data from these sources were collected through API calls in Python and returned in JSON format.

[2006-2020] *Annual Survey of State Government Finances Tables* (United States Census Bureau, 2021; United States Census Bureau, 2022):

> A link was provided to the United States Census Bureau website. These data sets were found by searching the Census website for economic data on the national level. Data for the years 2006 through 2020 were downloaded and kept in the original .xls or .xlsx formats.

Transformation

[2006-2020] *Annual Survey of State Government Finances Tables* (United States Census Bureau, 2021; United States Census Bureau, 2022):

1. Rename each file in the format: [year].xls or [year].xlsx, depending on the file type.
   Ex. 2006.xls

   In 2006.xls, merge cell A1 with cell B1 in Excel so that “State Government Finances: 2006” spans the two cells. For reference, look at cell A1 in the other spreadsheets and match the formatting.
2. Upload all files to the storage container.
3. In a Databrick, create an empty list to which data rows will be appended.

   For each file:

   1. Read in the file as a Spark DataFrame from the storage container.
   2. Drop all columns except for the column containing data about the United States.
   3. Add a column called “Year” that includes the dataset year as given in the filename.
   4. Rename the column containing United States data to “United States”.
   5. If the Total Revenue cell includes commas, remove the commas.
   6. Cast the United States column to be float data.
   7. Extract the row containing Total Revenue data from the DataFrame and append this to a list created in Step 3.
4. Create a Spark DataFrame from the list of rows created in Step 3.
5. Write the compiled DataFrame to a csv file in the storage container.

***Stock Time Series* (Alpha Vantage, 2022b):**

1. In a Databrick, create an empty list to which Pandas DataFrames will be appended.

For stock tickers: VTI, VGT, VHT, VIS, VCR, VFH:

   1. Call the Stock Time Series API to receive monthly data for the ticker in JSON format.

    2. Save the JSON object as a Pandas DataFrame.
    3. Transpose the DataFrame.
    4. Rename each column to maintain the original column names.
    5. Cast each column to be float data using pandas.to_numeric().
    6. Append the DataFrame to the list created in Step 1.

To prepare data for comparison with [2006-2020] ***Annual Survey of State Government Finances Tables*** **(United States Census Bureau, 2021; United States Census Bureau, 2022):**

1. Create an empty dictionary to which other dictionaries will be added.
2. For each DataFrame in the list of DataFrames created in Step 1:
    1. Create a dictionary where the keys are the dates, and the values are the associated closing prices for the given stock.
    2. Save the dictionary created in Step 2a. to be a value in the dictionary created in Step 1 with the key being the name of the stock.
        1. Example output: {‘VTI': {'2022-01-01':'55', ...}, ‘VGT’: {…}, ...}
3. Create a Pandas DataFrame from the dictionary that was filled in Step 2b.
4. Cast all columns in the DataFrame to be float data.
5. Impute any null values as the mean for each column
6. Create a new column for this DataFrame labeled “Year” and retrieve the year from the index in the format YYYY and an integer datatype.
7. Aggregate the Year column of the DataFrame by the average and save this to a new DataFrame.
8. Create a Year column for the new DataFrame by using the index of this DataFrame.
9. Rename the name of the index axis for the new DataFrame to be a blank string.
    1. This is to avoid merging conflicts between index ‘Year’ and the column ‘Year’.
10. Read in the annual US revenue CSV file previously saved in the storage container and create a DataFrame from it.

11. Merge the cleaned DataFrame from Step 9 with the annual US revenue Data Frame on the ‘Year’ column.
12. Save the file to the storage container as a backup file.

Combining the Monthly Stock info with the features from *Economic Indicators* (Alpha Vantage, 2022a):

1. Create a dataframe from the VTI stock information.
2. Transpose the dataframe so that the dates are the indices, and the features are the prices/volume
3. Iterate through each of the columns of the DataFrame and rename each of the columns to exclude the first 3 characters.
    1. This will remove the ‘#. ‘ from each of the column headers.
4. Cast each of the columns to a ‘float32’ data type similarly as in Step 5 of the annual US revenue CSV file creation section.
5. Create a Series for the dates that read in the index of VTI stock DataFrame
6. Apply a lambda function to this Series that converts the date strings from the VTI stock DataFrame indices into datetime.date objects, setting the day to be 1 for each object.
7. Set the index of the VTI stock Data Frame to this Series.

Combining each economic feature into one DataFrame:

1. Create a for loop that selects each of the economic features JSON objects
2. Using the map method, use a lambda function that reads the ‘date’ key of the economic features JSON object and selects every character up to the 7th character.
3. Using the map method, use a lambda function that reads in the ‘value’ key from the economic features JSON object.
4. Create a Data Frame for the economic feature,

    1. Set the data equal to the map object that contains all the values for the economic feature
    2. Set the index to the map object that contains all the dates
    3. Set the column header equal to the ‘name’ key of the economic feature JSON object
5. Replace any instances of “.” with numpy.nan (null values)
6. Set the indices of the economic feature DataFrame similarly to Steps 5-7 of the “Combining the Monthly stock info” section.
7. Cast the economic feature values to be a ’float’ data type
8. Join the economic feature data frame with the VTI stock price dataframe.
    1. Set the how parameter to ‘outer’
    2. Steps 2-8 should be within the for loop, resulting in a combined dataframe that contains the monthly stock prices for VTI and all the Economic Features information.
9. Create a ‘Year’ column similarly to Step 6 of the “prepare data for comparison...” section
    1. Make sure the format is YYYY and set as an integer data type
10. Filter out rows that are older than 2006.
11. Impute any null values with the mean for every column of the merged dataframe
    1. This was done using Scikit Learn’s impute.SimpleImputer module
12.

Combine each stock price with the merged DataFrame

1. Using the DataFrame constructed from Step 6 of the “prepare data for comparison...” section, filter out rows that are older than 2006.
2. Join this Data Frame with the merged economic features DataFrame
    1. Drop the Year column first
    2. Set the how parameter to ‘outer’

3. Drop the “open”, “close”, “high”, ”low”, and “volume” columns

4. Create a new column named ‘date’ and set it equal to the index of the newly merged DataFrame

5. Reset the index of the newly merged DataFrame by using the .reset_index() method.

6. Drop the column labeled ‘index’.

7. Save the DataFrame to the storage container in JSON format.

Load

1. Convert the annual DataFrame that has the annual stock prices and US revenue data into a Py Spark DataFrame. Also do this for the monthly economic features and percentage change DataFrame.
2. With a server name, database name, username, and password defined for a cloud-based SQL database, write the spark dataframes to your SQL database.
    1. Set the format to be ‘jdbc’
    2. Use mode “overwrite”

Conclusion

This project sources data from multiple APIs and Census data tables to examine the United States domestic stock market and how it responds to different economic contexts. It analyzes how different sectors of the US economy impact market response. Through extraction and transformation, all necessary data can be obtained, filtered, and combined to produce comprehensive tables for stock market analysis.