SAMPLE THESIS TITLE WITH A LONG TITLE THAT DESCRIBES THE

THESIS IN A CLEAR, CONSISE MANNER

by

Axel Eschholz

Thesis submitted in partial fulfillment of the

requirements for the Degree of

Bachelor of Computer Science

Acadia University

March, 2024

© Copyright by Axel Eschholz, 2024

The author retains copyright in this thesis. Any substantial copying or any other actions that exceed fair dealing or other exceptions in the Copyright Act require the permission of the author.

Table of Contents

[List of Tables vi](#_Toc163044862)

[List of Figures vii](#_Toc163044863)

[Abstract viii](#_Toc163044864)

[CHAPTER 1: Introduction 1](#_Toc163044865)

[1.1 Background 1](#_Toc163044866)

[1.2 Motivation 2](#_Toc163044867)

[CHAPTER 2: Investigation & Design 3](#_Toc163044868)

[2.1 Requirements 3](#_Toc163044869)

[2.1.1 Data Requirements 3](#_Toc163044870)

[2.1.2 Pipeline Requirements 3](#_Toc163044871)

[2.2 Source Investigation 3](#_Toc163044872)

[2.3 Metadata Report 4](#_Toc163044873)

[2.4 Pipeline Design 4](#_Toc163044874)

[2.5 Tools 6](#_Toc163044875)

[2.5.1 Source Investigation - Postman 6](#_Toc163044876)

[2.5.2 Pipeline Dependencies 6](#_Toc163044877)

[2.5.3 Visualization – Tableau 7](#_Toc163044878)

[CHAPTER 3: Implementation 8](#_Toc163044879)

[3.1 Data Retrieval 8](#_Toc163044880)

[3.2 Analysis 9](#_Toc163044881)

[CHAPTER 4: Results 10](#_Toc163044882)

[CHAPTER 5: Conclusion 11](#_Toc163044883)

[Appendix A: Full Meta Data Report 12](#_Toc163044884)

[Appendix B: Encodings 13](#_Toc163044885)

[Glossary 14](#_Toc163044886)

[References 15](#_Toc163044887)

# List of Tables

**No table of figures entries found.**

# List of Figures

[Figure 1: Postman Interface 6](#_Toc163044889)

# Abstract

The abstract should state concisely and lucidly the objectives, the method of procedure and the findings or conclusions of the thesis. It should not exceed one page in length.

1. Introduction
   1. Background

In the digital age, data is a ubiquitous resource. The continued development of the Internet of Things produces an ever-growing deluge from the mass of sensors and devices connected to the internet. [SOURCE] This unprecedented scale of information produces new sets of challenges and opportunities in the handling and analysis of what is colloquially known as ‘big data’. These data can be used to train large language models, produce highly accurate predictive analyses, and derive valuable insights at scale across just about every field and industry. [SOURCE]

The foremost challenge in this new paradigm is that the vast majority of these data are messy and disparate, making them unusable in their raw form. Therefore, infrastructure must be developed to effectively control the flow and quality of data. This infrastructure takes the form of processes that automate the movement and control of data, known as data pipelines. Data pipelines work by first ingesting data, then processing, storing, analyzing, and performing quality control on them to produce a usable dataset for the relevant application. [SOURCE] Given the varied applications of big data, the corresponding data pipelines can have wildly varying structure, complexity, and roles depending on the nature of the data and the requirements of the application. Therefore, a significant step in implementing a data pipeline is considering the data at hand and designing the automated steps to fit these requirements.

* 1. Motivation

This project is motivated by stakeholders in the finance industry concerning the implementation of a data pipeline to convert messy open-source data into insights for cattle futures markets. Futures markets are financial exchanges that facilitate the trade of standardized contracts to buy or sell commodities or financial instruments at a specified price with delivery set at a specified time in the future. The prices of these contracts, known as futures contracts, is dynamically influenced by a number of factors, including market sentiment, economic indicators, and physical market conditions of the underlying asset. However, these factors are only priced into the market to the extent that they are apparent to a majority of stakeholders. If there is a factor with the potential to influence the market, but not yet readily apparent or widely considered, it can give individual market participants an edge. The increasing digitalization of markets has amplified the importance and value of data in these decisions. A substantial quantity of data sources are openly accessible yet underutilized. This is due to their often disparate, unstructured, and raw nature, making it challenging for investors to leverage effectively.

One such case of open-source data with potential to inform futures markets is a database of cattle auction reports maintained by the United States Department of Agriculture (USDA). The database is free to access, but, due to its structure as disparate auction house reports, it is difficult to extract meaningful data from. However, if it were to be aggregated and effectively engineered, it could potentially provide valuable insights on the future supply of cattle. Therefore, this project will attempt to design and implement a data pipeline to extract, process, and maintain a relevant dataset for the use in predicting cattle futures.

1. Investigation & Design
   1. Requirements
      1. Data Requirements

Given the intended use of the data as a metric for conducting financial analysis, completeness, accuracy, and integrity of the data were all important requirements for the project.

accurately reflecting all recorded auction sales in the database across a given timeframe. In order to effectively be used for financial decisions, the data must also be as recent as possible.

* + 1. Pipeline Requirements

The pipeline must be able to store, manage, and update data to reflect the requirements of the data. It must be able to do so with minimal computation power and time. Therefore, data will need to be stored in an efficient and modular architecture. The pipeline must also be able to easily and rapidly update with new reports to maintain the relevancy and recency requirement of the data.

* 1. Source Investigation

The source provided for this project by the stakeholders is a list of reports across 31 states for every cattle auction held in each state. These reports contain a weekly aggregated state summary for 27 of the 31 states. The form of each report is a PDF with tables and comments about the market conditions. Given the difficulty of extracting meaningful data from a PDF, further investigation into the source of these data was required. This investigation yielded a database lookup tool to query specific reports.

[Image of database lookup tool]

By extracting the network request used by the online tool, one could retrieve report-specific data in JSON format. However, due to the limitations imposed on the browser-specific request, only 1000 rows could be returned at a time. This quota is significantly inadequate to properly retrieve the roughly 900,000 rows across all reports. Digging further into the source of these data reveals a separate dedicated API endpoint that can be accessed with a freely available API key that provides access to the same data but without the imposed quota. Thus, the source for this pipeline implementation is the API combined with the report ids scraped from the original document.

Expand on full scope of API

* 1. Metadata Report

In raw form, the returned data contains 48 attributes. The full list of attributes is listed in Appendix A.

* 1. Pipeline Design

The pipeline to be constructed will be based on the Extract Transform Load Transform (ETLT) model [SOURCE]. In the ETLT model the source data is retrieved and acted upon before being stored or ‘loaded’ into the data sink. It is then further processed before being displayed or used.

For the initial retrieval of the data, the python requests library will be used in conjunction with pandas to retrieve data from each report slug within a specified timeframe. Then certain attributes will be stripped or transformed to fit the shape and expectations of the data sink. Each report is stored separately to silo data and to allow for more distributed storage and processing. Therefore, the data sink consists of 27 csv files supervised by a central python interface to ensure integrity and adherence to a uniform structure. After each report is loaded into the data sink, the data is reviewed for integrity and transformed ahead of analysis and visualization.

When required for visualization and analysis, data is combined from each report silo into a single csv. These data can be filtered during this step if not all data is required for the particular analysis. The new dataset can then be applied as needed, but the data sink remains the source of truth and the new dataset can be discarded when no longer in use.

As new reports are published, the pipeline will extract the new report and merge it into the appropriate silo, ensuring that no instances are duplicated in the process. This allows for lightweight continuous updates to ensure any analysis reflects the most recent available data.

* 1. Tools
     1. Source Investigation - Postman

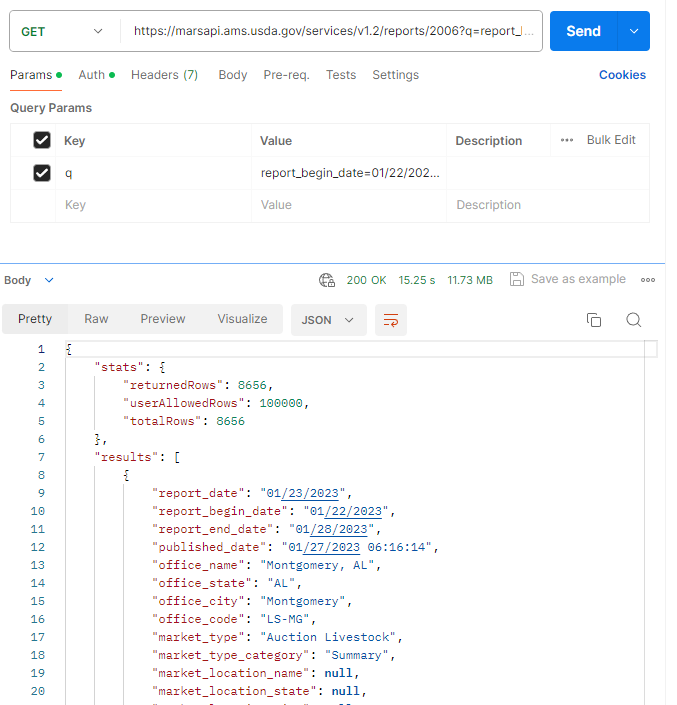


Figure 1: Postman Interface

For the source investigation, the API request tool Postman was used to analyze API endpoints and source data structure. Postman can easily store and edit request endpoints, request types, headers, authentication, and body parameters. These functionalities make it a useful tool for API analysis. Figure 1 depicts an example of the interface handling a request with returned JSON.

* + 1. Pipeline Dependencies

The pipeline was implemented in Python due to its readability, usability, and extensive support for data analytics. The packages used in the implementation of the pipeline are enumerated below in table 1.

**Table 1: Dependencies**

|  |  |  |
| --- | --- | --- |
| Name | Version | Description |
| Python | 3.11.3 | Python version |
| pandas | 2.2.1 | A library for handling data in python, used extensively across the project to store all data. |
| numpy | 1.26.4 | General math and data library used in data preparation and analysis |
| requests | 2.31.0 | Used for creating API queries and retrieving initial data. |
| click | N/A | Native python library used to develop cli interface. |
| scikit-learn | 1.4.1 | Machine learning library used for the preprocessing and modelling of data. |

* + 1. Visualization – Tableau

Data visualization was conducted entirely in the Tableau visualization software. Tableau is a powerful and intuitive tool that enables easy visualization across data types.

* 1. Analysis

Given the scope of the project is to

1. Implementation
   1. Data Retrieval

Database retrieval is linked to individual report ids, referred to in the API schema as report ‘slugs’. These slugs were pulled manually from the report list and represent the 27 state summary reports being aggregated. Each slug functions as a separate endpoint that contains all historical data from all editions of that report. Query parameters were then used to specify the *report\_end\_date* attribute to retrieve data from a specific timeframe. The query was called using the python requests library, with the API key specified as the authentication parameter. The returned data was then loaded into a pandas dataframe to facilitate cleaning and storage.

url = f'https://marsapi.ams.usda.gov/services/v1.2/reports/{slug}

?q=report\_end\_date={start\_date}:{end\_date}'

response = requests.get(url, auth=(‘api\_key\_goes\_here\_xxxxxxxx’, ''))

**Code Snippet 1: API Request**

Before storing the raw data down, initial transformations were applied to fit the structure and expectations of the data sink. First, redundant and useless attributes were dropped immediately (See Appendix A). Examples of dropped attributes might include comments, which are too ambiguous to be of real use, or further iterations of market location, which are redundant to these summary data past the state code.

Next, all date-related attributes were converted to the datetime data type using the pandas.to\_datetime() method. Then all instances were sorted by *report\_end\_date* in ascending order. Once the dates were formatted and sorted, the *final\_ind* attribute was translated into a binary attribute to indicate whether the instance represents final data. All remaining nominal columns were then factorized using a uniform encoding scheme (See Appendix B). These transformations excluded the attributes *report\_title*, ­*slug\_name*, and *slug\_id* since these are administrative values used in data control but not analysis.

Once the initial transformations had been applied, the data was stored in the data sink as a csv file. Each report slug was individually retrieved, transformed, and stored. Then an integrity check was run across the entire data sink to verify that transformations had been correctly and completely applied.

* 1. Application of Data

Before conducting analysis, data from all reports was combined into a central dataset. During this process, the aforementioned administrative attributes were dropped to create a uniform set of numeric data ready for analysis and visualization.

* + 1. Visualization

Once the combined dataset was loaded into Tableau, labels needed to be linked back to the nominal data to display readable graphs. This was accomplished by converting the encodings to Tableau’s native logic and loading them as calculated values in Tableau’s data scheme. These labels allowed for insightful analysis through visualization. Figure 2 illustrates one such example where the labeled data was able to be filter to reveal a significant portion of non-cattle animals that somehow got included in the report data.

A graph with numbers and text

Description automatically generated

Figure 2: Non-Cattle Commodities

* + 1. Analysis

PCA Analysis

* Model application

1. Results

Enumerate the results of the data analysis, as well as the insights gained from the data visualization.

1. Conclusion

Evaluate success criteria of the pipeline.

Evaluate success criteria of analysis.

Evaluate success criteria of the data as it pertains to the stakeholders.

Define potential paths for further research and development.

# Appendix A: Full Meta Data Report

[Import of Attributes from excel]

# Appendix B: Encodings

# Glossary

USDA – The United States Department of Agriculture

API – Application Programming Interface

# References

Cite API/Source Database