ELT Pipeline Design Document

GitHub: SChow-FinalProject-ELTPipeline/ProjectRepo (github.com)

ADS-507
Final Project
Connie Sau Chow

Table Of Contents

1. **ELT Pipeline Project Overview**
2. **Source Datasets**
    1. Consumer Expenditures
    2. Consumer Price Index
    3. Gross Domestic Product

3. Architectural Design
1. High Level Architectural Design
2. Python Application Software Design
3. 'Consumer Expenditures' Database Schema Design (EER Diagram)
4. Python Methods

4. ELT Pipeline Output
1. SQL View
2. Logging

5. Concluding Remarks
1. Limitations & Gaps In The System
2. New Features To Develop

1. ELT Pipeline Project Overview

The purpose of this project is to code a basic ELT pipeline implementation using Python & SQL. This pipeline extracts Consumer Expenditures data, as well as relevant economic indicators data (ie GDP and CPI) from various external sources and preprocess/cleans that data and then houses it in a "Business Warehouse", where it is ready to be used and consumed by end users such as data scientists, data analysts, executives who need dashboards to make business decisions, in our case maybe an economists or researcher. This application uses various python libraries, the main ones being SQLAlchemy and Pandas. The 'Staging Area' and 'Business Warehouse' utilizes a MySQL database locally installed on a personal laptop.

This design document covers software architecture and design of the pipeline, reasons for choosing ELT over ETL design, data sources, limitations, scalability and new features to be implemented to make this pipeline more robust and scalable in future iterations.

2. Source Datasets

This pipeline extracts three datasets from different external data sources. The first is the Consumer Expenditures dataset. The website took the data published by the Bureau of Labor Statistics Consumer Expenditures website and put it into a simple three table relational database available on their website. Refer to the link below. This dataset contains three tables 'EXPENDITURES', 'HOUSEHOLD_MEMBERS' and 'HOUSEHOLDS'. Each row in the 'EXPENDITURES' table contains information about the particular household, date of purchase, type of product, cost of that product and whether or not it was a gift. Each row in the 'HOUSEHOLD_MEMBERS' table contains information about the household making the purchase including marital status, sex, age and work status.

The group initially decided on using the Consumer Expenditures data from a website online that contains the Bureau of Labor Statistics Consumer Expenditures data formatted into a relational database. I thought it'd be useful to also incorporate some data from economic indicators like CPI and GDP to give a "bigger picture" to the spending habits of consumers.

All of these datasets were found by googling either the type of data we were looking for, googling "relational databases datasets" or searching directly on Kaggle website.

1. Consumer Expenditures

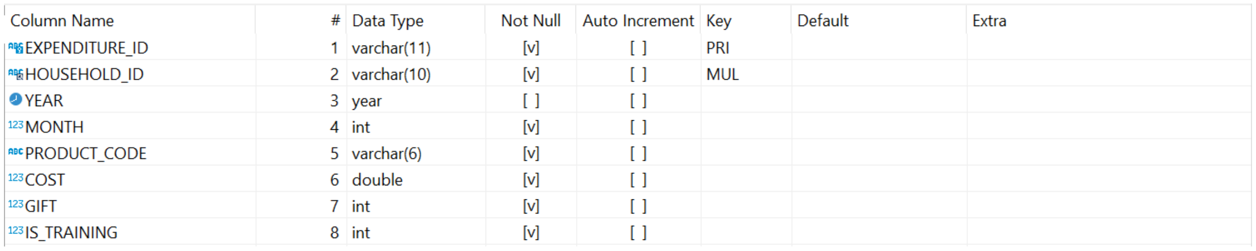
Link
https://relational.fit.cvut.cz/dataset/ConsumerExpenditures

Type

MySQL Database (MariaDB)

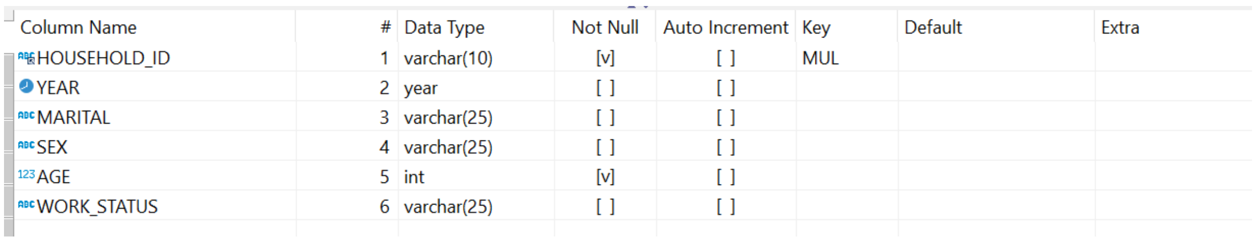
Reasons For Choosing This Dataset
Given that the economy is going into a recession and an inflationary environment, it would be interesting to build a repository to analyze and understand consumer behavior and trends. This dataset is in the form of a relational database, which I wanted to try as an exercise for extraction in building this pipeline.

Table Definitions
EXPENDITURES



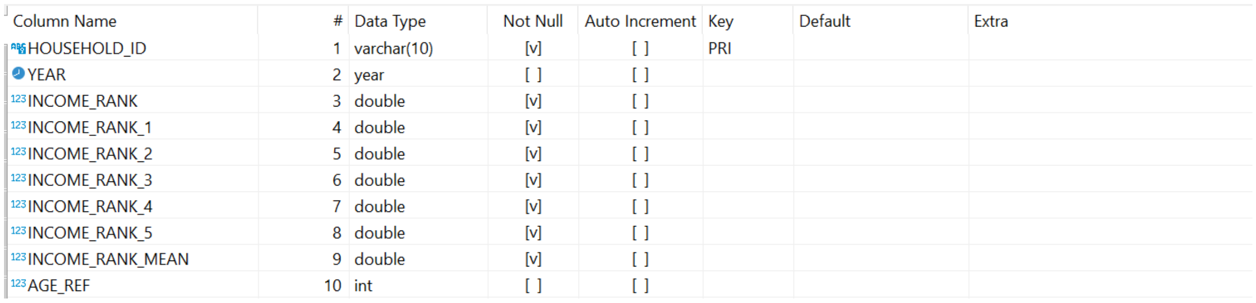
Column Name	#	Data Type	Not Null	Auto Increment	Key	Default	Extra
EXPENDITURE_ID	1	varchar(11)	[v]	[ ]	PRI		
HOUSEHOLD_ID	2	varchar(10)	[v]	[ ]	MUL		
YEAR	3	year	[ ]	[ ]			
MONTH	4	int	[v]	[ ]			
PRODUCT_CODE	5	varchar(6)	[v]	[ ]			
COST	6	double	[v]	[ ]			
GIFT	7	int	[v]	[ ]			
IS_TRAINING	8	int	[v]	[ ]			

HOUSEHOLD_MEMBERS



Column Name	#	Data Type	Not Null	Auto Increment	Key	Default	Extra
HOUSEHOLD_ID	1	varchar(10)	[v]	[ ]	MUL		
YEAR	2	year	[ ]	[ ]			
MARITAL	3	varchar(25)	[ ]	[ ]			
SEX	4	varchar(25)	[ ]	[ ]			
AGE	5	int	[v]	[ ]			
WORK_STATUS	6	varchar(25)	[ ]	[ ]			

HOUSEHOLD



Column Name	#	Data Type	Not Null	Auto Increment	Key	Default	Extra
HOUSEHOLD_ID	1	varchar(10)	[v]	[ ]	PRI		
YEAR	2	year	[ ]	[ ]			
INCOME_RANK	3	double	[v]	[ ]			
INCOME_RANK_1	4	double	[v]	[ ]			
INCOME_RANK_2	5	double	[v]	[ ]			
INCOME_RANK_3	6	double	[v]	[ ]			
INCOME_RANK_4	7	double	[v]	[ ]			
INCOME_RANK_5	8	double	[v]	[ ]			
INCOME_RANK_MEAN	9	double	[v]	[ ]			
AGE_REF	10	int	[ ]	[ ]			

2. Consumer Price Index (CPI)

Link
https://www.kaggle.com/datasets/varpit94/us-inflation-data-updated-till-may-2021

Reasons For Choosing This Dataset

This dataset is in CSV format, available as a file off of the Kaggle website. The main dataset is the Consumer Expenditures and it would be interesting to also see how the consumer price index changes as consumer spending changes as well. The consumer price index is an index calculated from a basket of consumer goods and services. This is different from the inflation rate in that this number is the change in overall price of everything in the economy, whereas CPI represents change in the price of the cost of living.

Type
CSV file

Table Definition

Table Field Name	Data Type	Example Value
Yearmon	Date	1/1/1913
CPI	Double	9.8

3. Gross Domestic Product (GDP)

Link
https://www.kaggle.com/datasets/federalreserve/interest-rates?resource=download

Reasons For Choosing This Dataset
This dataset is in CSV format, available as a file off of the Kaggle website. The main dataset is the Consumer Expenditures and it would be interesting to also see how the real GDP and relevant federal funds rate coincide with changes as consumer spending changes. The federal funds rate is the overnight lending rate for depository institutions. So it basically will affect how much money and at what cost it is lent out at and this propagates to the rest of the economy in terms of trickle down from big companies down to the consumer.

Type
CSV file

Table Definition

Table Field Name	Data Type	Example Value

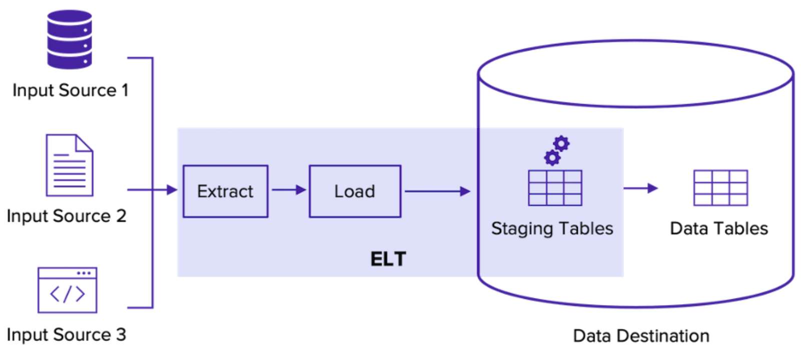
Year	Date	1913
Month	Int	1
Day	Int	3
Federal Funds Target Rate	Double	5.75
Federal Funds Upper Target	Double	5.75
Federal Funds Lower Target	Double	5.75
Effective Federal Funds Rate	Double	0.8
Real GDP (Percent Change)	Double	4.6
Unemployment Rate	Double	4.1
Inflation Rate	Double	3.2

3. Architectural Design

High Level Architectural Design

A basic ELT pipeline design was chosen for this project. Refer to the diagram below Figure 1 for a general overview of the ELT pipeline design. With the ELT pipeline, data is extracted and then first loaded into the 'Staging Area' database before transformations are performed on it. After this, the data is loaded into a 'Business Warehouse', a data repository ready to be consumed by end users such as data scientists, data analysts, researchers, executive management, marketing and sales etc. End users need the data in a ready-to-go state with data that is already cleaned and preprocessed.

Figure 1. ELT Pipeline Generic Design

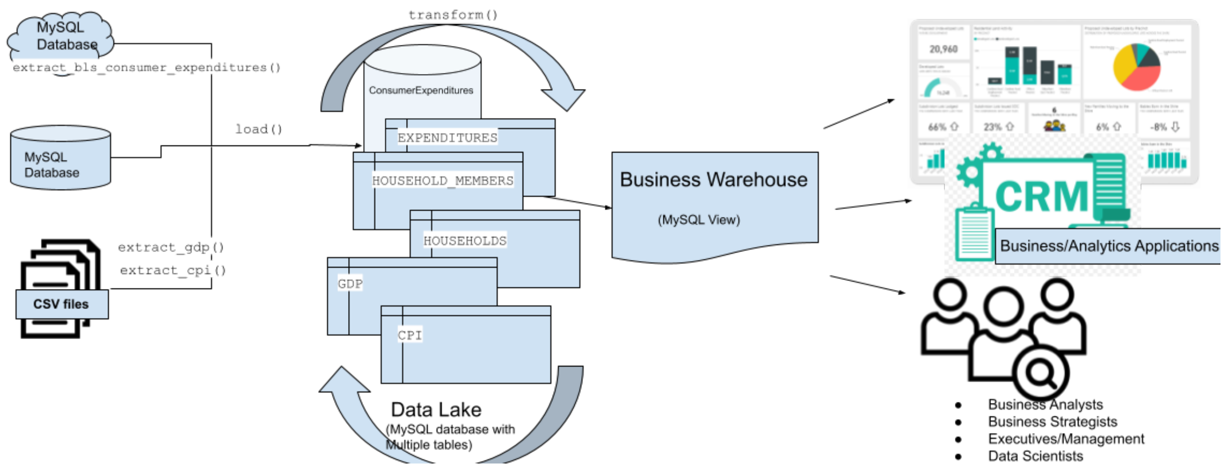


As opposed to the traditional ETL design, the ELT design is more fit for continuously integrating and streaming in real-time (or non-static) data. While the three datasets this pipeline currently extracts are static (updated annually), I wanted to setup the pipeline so that in the future it can also integrate new forms of real-time data ie day-to-day E-commerce purchases etc.

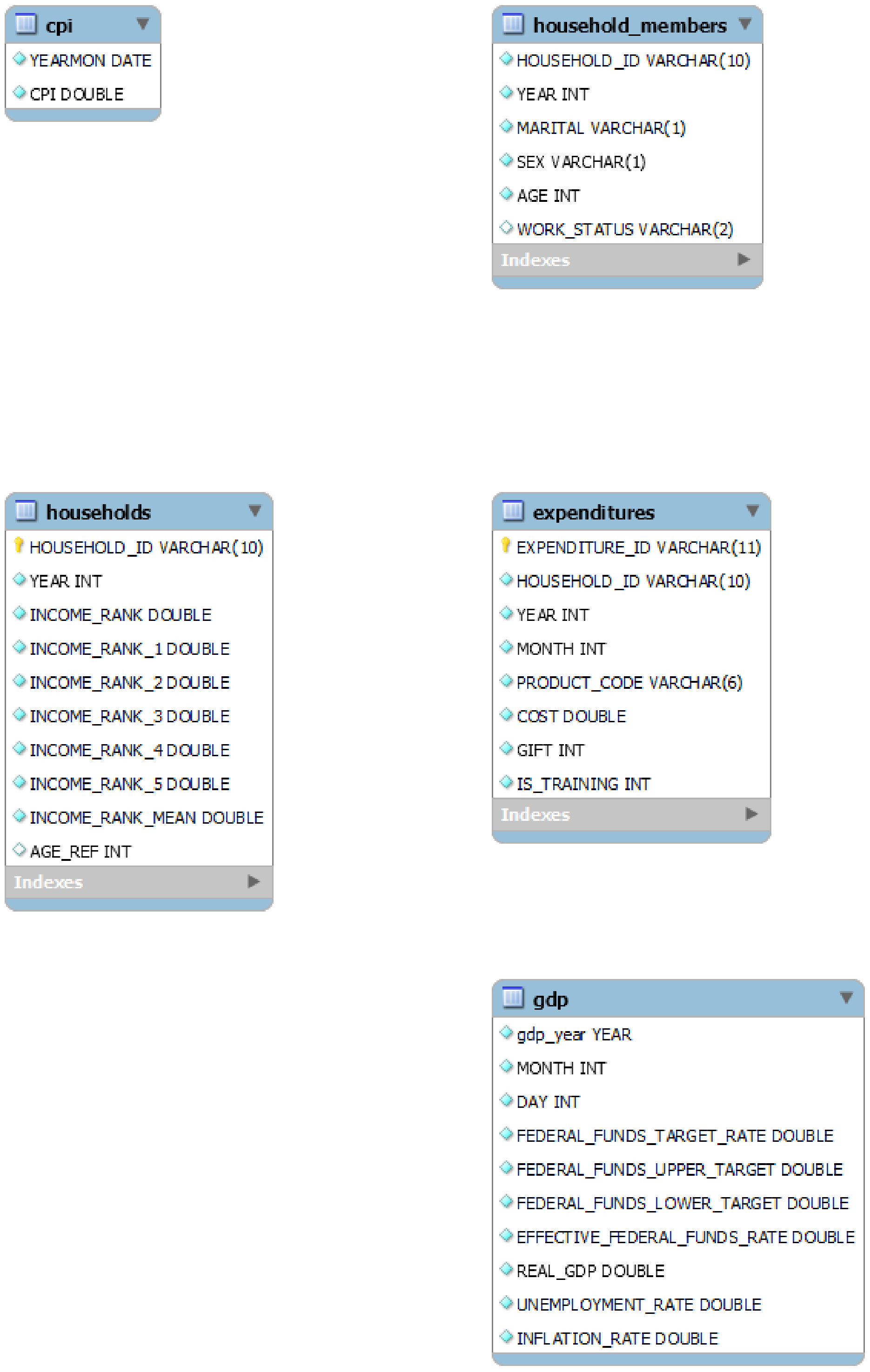
The extract and load implementation was already also meshed together so it naturally took on the ELT paradigm.

Refer to Figure 2 below. This is a diagram of the ELT pipeline implementation and contains specifics of what will be developed and in what forms it will take. First, the pipeline will send a request to the various data sources and read in the data into pandas dataframes and load them immediately into the local MySQL database 'Consumer Expenditures'. From there, database tables will be created to store this data as you can see with the labeled rectangles. From there the pipeline will perform SQL transformations via python SQLAlchemy calls to execute explicit SQL queries. Next, an SQL view is created from those tables. This view aggregates all the data from the tables into a comprehensive single table ready to be accessed and consumed by end users like analytics applications and other relevant people who need cleaned and preprocessed data.

Figure 2. ELT Pipeline Implementation Diagram



'Consumer Expenditures' Database Schema Design (EER Diagram)



cpi
- YEARMON DATE
- CPI DOUBLE

household_members
- HOUSEHOLD_ID VARCHAR(10)
- YEAR INT
- MARITAL VARCHAR(1)
- SEX VARCHAR(1)
- AGE INT
- WORK_STATUS VARCHAR(2)

Indexes

households
- HOUSEHOLD_ID VARCHAR(10)
- YEAR INT
- INCOME_RANK DOUBLE
- INCOME_RANK_1 DOUBLE
- INCOME_RANK_2 DOUBLE
- INCOME_RANK_3 DOUBLE
- INCOME_RANK_4 DOUBLE
- INCOME_RANK_5 DOUBLE
- INCOME_RANK_MEAN DOUBLE
- AGE_REF INT

Indexes

expenditures
- EXPENDITURE_ID VARCHAR(11)
- HOUSEHOLD_ID VARCHAR(10)
- YEAR INT
- MONTH INT
- PRODUCT_CODE VARCHAR(6)
- COST DOUBLE
- GIFT INT
- IS_TRAINING INT

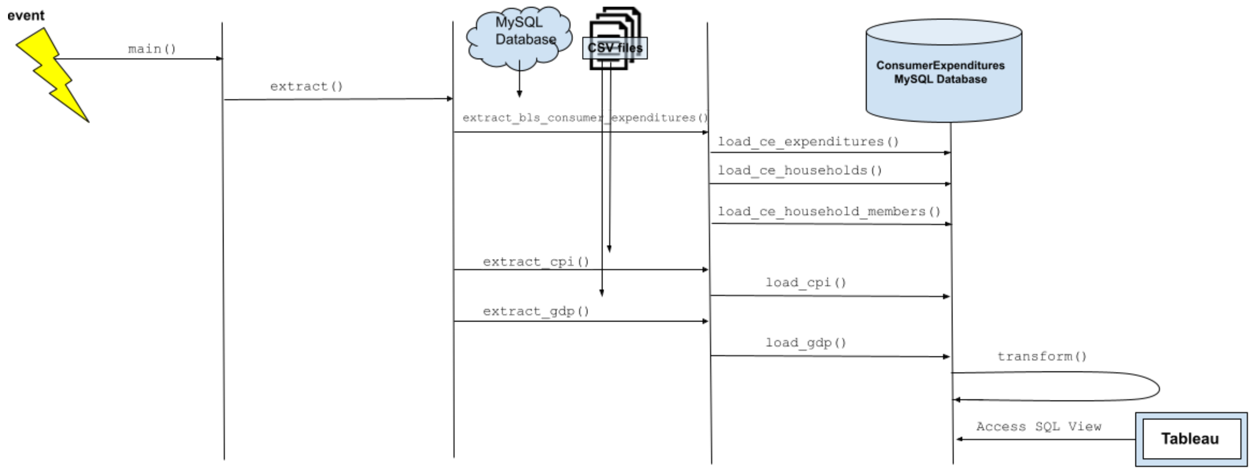
Indexes

gdp
- gdp_year YEAR
- MONTH INT
- DAY INT
- FEDERAL_FUNDS_TARGET_RATE DOUBLE
- FEDERAL_FUNDS_UPPER_TARGET DOUBLE
- FEDERAL_FUNDS_LOWER_TARGET DOUBLE
- EFFECTIVE_FEDERAL_FUNDS_RATE DOUBLE
- REAL_GDP DOUBLE
- UNEMPLOYMENT_RATE DOUBLE
- INFLATION_RATE DOUBLE

Python Application Software Design

Overview
The pipeline will be implemented in Python for the application level. SQLAlchemy and pyodbc libraries are utilized to interact with and execute explicit SQL statements and queries. Below, Figure 3 shows the order of execution of specific methods in the python program. Note that this diagram also shows the application's interaction with different objects such as the external data sources, MySQL database, and Tableau dashboard.

Figure 3. Python ELT Pipeline Application Sequence Diagram



Python Methods

Method	Description
`main()`	ELT pipeline application execution starts here. Calls extract(), load() and transformation().
`extract()`	Contains all calls to subroutines to extract data from all three data sources.
`extract_bls_consumer_expenditures()`	Extracts data from the online relational database for Consumer Expenditures data. Reads in tables into pandas dataframes, executes SQL statements using SQLAlchemy library call to create the same tables and copy over the data to these tables.
`extract_gdp()`	Opens the CSV file on the local drive and reads them in row by row into a dataframe which is then written to the database table.
`extract_cpi()`	Opens the CSV file on the local drive and reads them in

	row by row into a dataframe which is then written to the database table.
`load_ce_expenditures(df_datasource_table, tbl)`	Takes the dataframes containing the data from external data source, creates relevant SQL database tables and then writes the dataframe to them.
`load_ce_households(df_datasource_table, tbl)`	Takes the dataframes containing the data from external data source, creates relevant SQL database tables and then writes the dataframe to them.
`load_ce_household_members(df_datasource_table, tbl)`	Takes the dataframes containing the data from external data source, creates relevant SQL database tables and then writes the dataframe to them.
`transformation()`	Converts code values in columns like ‘MARITAL’ status from numeric values into descriptive readable values like ‘Married’ or ‘Divorced’.

4. ELT Pipeline Output

The goal of the pipeline is to take various external or internal data sources, clean and preprocess them and load the data in a final “business warehouse” type of repository. The repository is the output of this pipeline and the place where all the data is made accessible in a digestible format for end users’ work processes.

In this particular ELT pipeline, the output is a SQL View which pulls together the different database tables (data lake) containing the data from the various data sources. This view is made accessible as a portal into the data available from the pipeline. This data is useful because each row contains a particular purchase, the cost and information on the purchaser and also the date of purchase and the relevant economic indicators for that particular year.

End users will be able to easily query the View or connect to the View through their analytics applications and answer questions like:

What are the spending habits of generation X from 2000 to 2020?

A marketer at Target corporation might ask “In a period of inflation or high consumer goods prices, which products sell the best or most frequently and in which price range?”

An economist might ask “What are the consumer spending habits by consumer good category and by generation?”

A data scientists working for a retail shopping mall developer might want to predict changes in the shopping habits of consumers and observe changes in patterns in spending habits by generation.

SQL View

Figure 3 below shows a diagram of the ELT pipeline as a whole. This is shown to give context to the SQL View which is the output of the ELT pipeline. The output of the pipeline is a comprehensive single SQL View available to the outside world to access. Usually, you want the ‘Business Warehouse’ to be an actual database and this is the case for real-world implementations. However, for the sake of efficiency and memory, the ‘Business Warehouse’ for this project is presented as an SQL View.

As an SQL View, it would not require extra memory to be allocated as you would with a business warehouse. It is presented “virtually” as an SQL view. Another good thing about this being a view is that it has implicit read-only rights so that any user accessing the data cannot change it. End users of the business warehouse/pipeline output include Business and Data Analytics applications as well as end users like business analysts, data scientists, executives who usually refer to dashboards to make management/business decisions.

Figure 3. ELT Pipeline Diagram

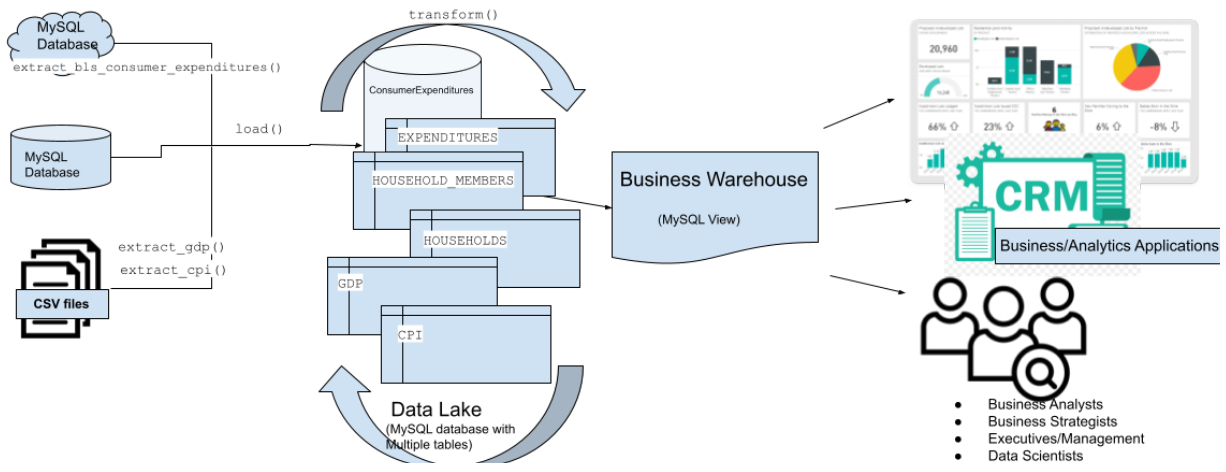
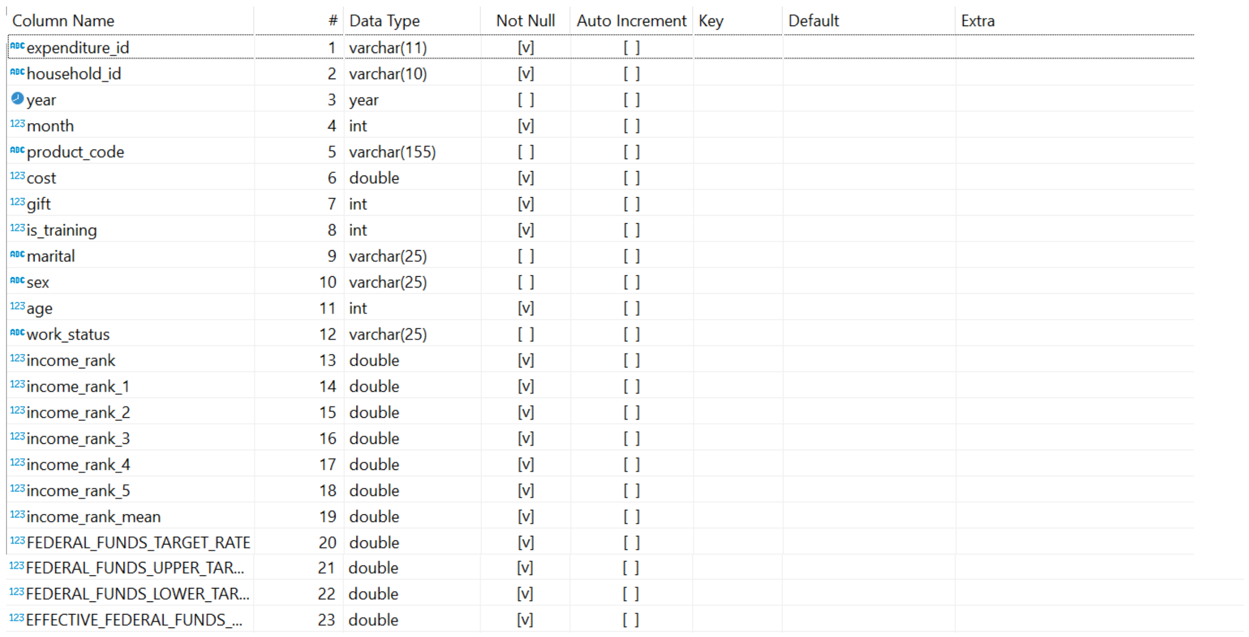


Figure 5 below shows the fields available in the SQL View. These are the fields that will be available to any end user of this pipeline’s repository.

Figure 5. SQL View Available Fields



Column Name	#	Data Type	Not Null	Auto Increment	Key	Default	Extra
expenditure_id	1	varchar(11)	[v]	[ ]			
household_id	2	varchar(10)	[v]	[ ]			
year	3	year	[ ]	[ ]			
month	4	int	[v]	[ ]			
product_code	5	varchar(155)	[ ]	[ ]			
cost	6	double	[v]	[ ]			
gift	7	int	[v]	[ ]			
is_training	8	int	[v]	[ ]			
marital	9	varchar(25)	[ ]	[ ]			
sex	10	varchar(25)	[ ]	[ ]			
age	11	int	[v]	[ ]			
work_status	12	varchar(25)	[ ]	[ ]			
income_rank	13	double	[v]	[ ]			
income_rank_1	14	double	[v]	[ ]			
income_rank_2	15	double	[v]	[ ]			
income_rank_3	16	double	[v]	[ ]			
income_rank_4	17	double	[v]	[ ]			
income_rank_5	18	double	[v]	[ ]			
income_rank_mean	19	double	[v]	[ ]			
FEDERAL_FUNDS_TARGET_RATE	20	double	[v]	[ ]			
FEDERAL_FUNDS_UPPER_TAR...	21	double	[v]	[ ]			
FEDERAL_FUNDS_LOWER_TAR...	22	double	[v]	[ ]			
EFFECTIVE_FEDERAL_FUNDS_...	23	double	[v]	[ ]			

Logging

Etl-pipeline.log
This file contains all the log messages from the application execution. Log messages include SQL Query results, Time to perform a task or method call, and execution points starting and stopping.

Concluding Remarks

Gaps In The System & Limitations

This is a preliminary proof of concept of an ELT pipeline. So all development was done centralized around a local MySQL database installed on a personal laptop. The 'Staging Area' database as well as 'Business Warehouse' database were actually separate tables in the same database on this local MySQL database. Usually, out in the real world, the 'Staging Area' and 'Business Warehouse' would be separate databases.

The python application for building this ELT pipeline certainly has some limitations.

1. Extracting data from data sources needs to be batched
2. Writing data to the local MySQL tables needs to be optimized. Currently with about 2M rows of data, the current implementation requires about 10-20 minutes to run. This can be optimized with the use of batch reading and writing for pd.to_sql() and pd.read_sql() calls in the SQLAlchemy library.

3. Extracting data from data sources needs to be

New Features To Implement

As this was a first proof of concept of an ELT pipeline, I discuss below some improvements that can be made in future iterations to make the pipeline more **robust** and **scalable**.

1. The current local MySQL for which the extracted data sources are being written to should be housed on a remote server somewhere. A server like hosted MySQL databases which have on-demand capacity and resources to scale the MySQL database and processing capacity if needed.
2. Data from the data sources needs to be extracted serially and in batches and for each batch, reading what is needed from the other data sources to combine before writing to the database permanently. This is one way to optimize the extraction process, rather than the current implementation where each data source is extracted and read into the database serially.
3. The implementation and logic for extracting data from external sources needs to be streamlined such that it can generically read in data. For example, if Data Source 1, in the next year, adds a new table or adds new columns to old tables, we want the logic in our extract() method to be able to dynamically read whatever columns are in these tables etc. In the current implementation, the reading of these tables from each external data source is hardcoded and will break if any of these data sources change.
4. Logging can be more streamlined so as to better aid in future development and troubleshooting or optimization tasks. Currently the logs are being written to separate log files and all types of log messages are written to them. They should be separated into types of log files ie time/duration of method calls of interest, errors during query execution or runtime execution should be separated, record of data in rows that were successful or unsuccessfully written to the database tables etc. A worker who needs to check why the ELT pipeline takes so long to refresh the business warehouse, might want to only check the logs for time spent on method calls etc.