Dave Friesen & Lane Whitmore

ADS-507

Design Document

February 27, 2023

BIS Pipeline – Bank for International Settlements (BIS)

**GitHub:**[**https://github.com/lanewhitmore/BIS\_Data\_Pipeline**](https://github.com/lanewhitmore/BIS_Data_Pipeline)

**Opportunity and Solution**

The Bank for International Settlements (BIS) is an international “bank for central banks” supporting monetary and financial cooperation among its central bank owners around the globe (BIS, 2023). Among its roles, the BIS compiles and publicly publishes a "gold mine" of data: statistics rates, and metrics that inform analysis of global financial stability and liquidity. This information spans banking, currency, debt, derivatives, credit, property, consumers, and financial market domains. Leverage of this data can be accelerated through automation and structure.

**BIS Pipeline** demonstrates this leverage. It provides a production-ready, automated data pipeline to extract, load, transform, and persist select BIS datasets to a relational database for further analysis and flexible "consumption." Data assets "surfaced" through the baseline version of **BIS Pipeline** include US dollar exchange rates, consumer prices, and policy rates.

Consumption opportunities using **BIS Pipeline** as a baseline range from simple descriptive analytics and visualization to advanced predictive models. For example, the base pipeline provided here demonstrates automated output of Consumer Price Index (CPI) vs. the US federal discount rate, North American currency exchange rate comparisons, and exchange rate views across countries of interest. A simple and natural extension to these examples might be a time-series predictive model (e.g., ARIMA-based) to forecast CPI changes from the federal discount rate (as a leading indicator). Further opportunities exist through code and relational database schema extensions to this "open source" code base, and even third-party data augmentation.

**Source Data**

As summarized under Opportunity and Solution, **BIS Pipeline** data includes US dollar exchange rates (monthly, quarterly and annual), consumer prices, and policy rates (monthly). These datasets are sourced from BIS’ statistics download page located at <https://www.bis.org/statistics/full_data_sets.htm>, summarized in Table 1 as follows:

**Table 1**  
BIS Datasets

|  |  |  |
| --- | --- | --- |
| **Dataset Name** | **File Name and Format** | **Size (dimensional)** |
| US Dollar exchange rates  (monthly, quarterly, annually) | WS\_XRU\_csv\_col.csv | 1,150 rows (less header)  3,960 columns |
| Consumer prices | WS\_LONG\_CPI\_csv\_col.csv | 240 rows (less header)  1,696 columns |
| Policy rates (monthly) | WS\_CBPOL\_M\_csv\_col.csv | 39 rows (less header)  937 columns |

This "raw" source data is extracted, loaded, transformed, and ultimately persisted into a MySQL relational database for further analysis and "consumption." The following Pipeline Functional and Non-Functional Overview describes this process in further detail.

**Pipeline Functional and Non-Functional Overview**

**Pipeline Architecture and Process**

Figure 1 overviews **BIS Pipeline**'s end-to-end architecture and data flow, followed by a summary of each step and data stage in the process:

**Figure 1**  
BIS Pipeline Architecture and Data Flow

Diagram, timeline

Description automatically generated

1. **Online Data Source** – Per the Data section above, BIS Pipeline automation sources data from the following URL: <https://www.bis.org/statistics/full_data_sets.htm> (BIS, 2023).
2. A. **Pipeline Trigger –** Automation Directions are available at the top of the README on GitHub in addition to pipeline setup directions in general. The pipeline trigger is unfortunately only available within Window's Operating System. The pipeline has been created by constructing a batch (.bat) file in NotePad that contains four line items; pathing to an Anaconda environment that has been used to construct the pipeline, current working directory pathing to the 'src' folder within the repository, pathing to the Anaconda python.exe file, and, finally, pathing to the python pipeline file. The batch file is then used within Window's Task Scheduler to create a new task that runs on the second of every month at 10am, as the BIS datasets are updated every month on the first at any given time. The task will open Window's Command Prompt at that time and date and run the commands outlined earlier to begin updating the database with the pipeline. This will either populate the database, if the pipeline is running for the first time, or extract only rows that have not been populated within the database to update the tables with. Doing so will show print functions tracking the pipeline's progress in the command prompt. Once the pipeline has completed, within the 'src' folder, that has been set as the working directory, a pipeline log will be populated with recent updates or any expected errors that may have occurred.

B. **File Download** – The pipeline iterates (flexibly, extensibility) through the set of source data files, using URL handling package urllib to retrieve each from their specified URL. Bis.org maintains consistent access points and filing naming conventions so this step should continue to function without exception, including when adding even more datasets. However, the pipeline tracks, logs, and elegantly handles potential issues with either the HTTP transport or URL itself. (Note that success is also logged.)

1. **Archive Data** – Using on baseline **BIS Pipeline** scope, Step 2.b results in the following archive files being replicated directly from bis.org: full\_xru\_csv.zip, full\_cbpol\_m\_csv.zip, and full\_long\_cpi\_csv.zip.
2. **A. File Extraction** – Once (to the extent) archive files are staged - and actually (efficiently) in parallel with the download of each file - the pipeline automatically extracts archive contents for further processing. Simlar to download, this is done using common Python package zipfile, and success is logged or exceptions are cleanly handled and logged. Exceptions may include corrupt or extremely large files. This step within the baseline script will produce the following files for continued processing: WS\_XRU\_csv\_col.csv, WS\_LONG\_CPI\_csv\_col.csv, WS\_CBPOL\_M\_csv\_col.csv.

B. **File Load** – Following file extraction, the pipeline "ingests" the [n] .csv's into Pandas data frame(s). This is a normal .csv load with a couple of key extensions: 1) a file count control is implemented to confirm full load of .csv contents; and, 2) the loaded data frames are split into respective "context" and "measurement" components. This data frame split is done because the quantitative values in BIS data may be extremely "wide", spanning granular time-series over many years. Segregating these values into a separate data frame allows for a Pandas melt() (transposition) of these values from wide to narrow, for simpler load into the **BIS Pipeline** relational database.

1. **Source Data** – As summarized in Step 4.b, at this point in the pipeline process, data frames are staged and ready for further processing including transformation and relational database population.
2. **Data Transformation and Database Load** – Following extraction and load, the pipeline then executes a transformation stage. This part of the process has been constructed by using custom built commands within Python that employ the use of the package PyMySQL and sqlalchemy’s create\_engine function. The first formula creates the connection between the python script and the database cursor and closes the connection once the formula has run the SQL script in the cursor. The second function can be used to create tables on the database through the python script, this is mostly for future usability if needed. The third function uses the first connection function to push the data to the database. The formula creates a connection to the database, pulls the table’s columns from the database, uses those column names to extract the important columns from the pandas data frame, and creates a data frame column to be filled by the auto incremental IDs within the MySQL schema. The formula also pulls the existing index from the schema and uses the difference function to find indexes that have not yet been posted to the databases to extract then ultimately push to the schema. By comparing and extracting indexes, it removes the possibility of reposting the same data and end up with duplicate data throughout the database. In total, as Figure 1 points out, the transformation portion creates six tables, two for each CSV, and ten views that are to be used for easier access to the data and/or security purposes.
3. **Entity Relationship Diagram (ERD)** – The following Figure 2 visualizes the physical model for BIS\_Pipeline's relational database:

**Figure 2**  
BIS Entity Relationship Diagram

Graphical user interface, text, application, chat or text message

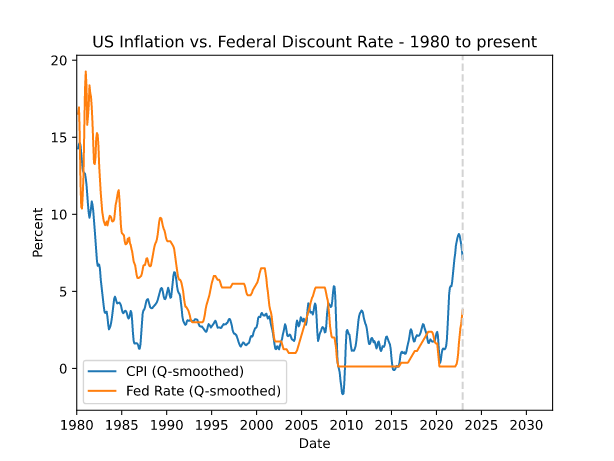
Description automatically generated

Following are just a few examples of the possibilities with BIS data using the baseline script. As highlighted elsewhere in this document, BIS Pipeline is easily extensible to expand these possibilities.

1. **A. Consumption Sample One Using Base Schema –**

The consumption sample below has been created at the end of the pipeline by pulling the consumer price index and federal rates from the United States. It acts as a sample of a data analytics dashboard that could automatically be constructed during the pipeline.

**Figure 3**  
Consumer Price Index vs. Federal Rates - 1980 to Present



**B. Consumption Samples Two and Three Using Built-in Views –**

The consumption samples below are created by pulling the pre-filtered views that have been constructed within the schema. This serves as an example of potential use cases when using views. Views can be used for convenience when a query is used often, but, in addition to that, views can serve as security in the database. More on this topic is explained in the security section. Figures 4 and 5 below can offer insight into how the U.S. economy is performing by evaluating the U.S. Dollar in proportion to various other countries. Figure 4 highlights the exchange rate with two other North American countries Canada and Mexico over the span of 50 years. It points out that the relationship between the American Dollar and the Canadian Dollar have stayed static over the years while the exchange rate has steadily climbed going to Mexican Peso. For instance, this could point out that while Canadian and U.S. economies have stayed strong, or at least stayed relative to one another, the Mexican economy has struggled in comparison.

**Figure 4**  
U.S. Dollar North American Exchange Rates - 1970 to Present

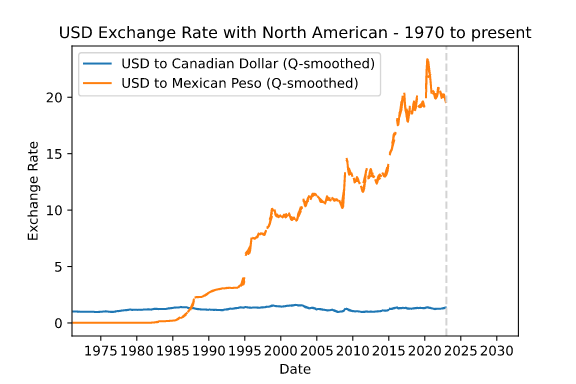
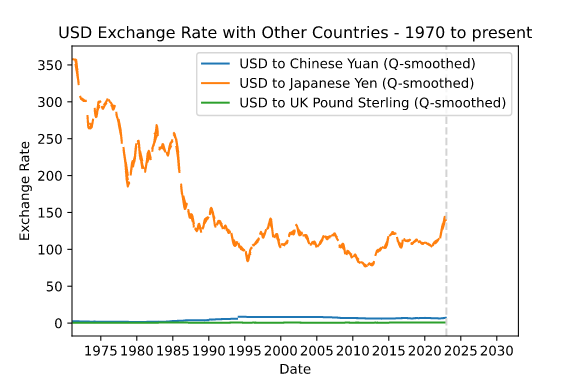


Figure 5 offers more examples from the views about potential analytics use cases from the database. The U.K., China, and Japan all have been world powers at some point in economic history. Figure 5 shows that the Japanese economy has become much stronger in comparison to the United States during the 21st Century. This could be due to the technological boom that has occurred as Japan has been at the forefront with multiple tech companies. The Pound and Yuan have stayed static for the last 50 years. These views are just options for how much the database can be expanded depending on potential use case. Although, based on the datasets already hosted within the database, and the way the schema has been setup, the best option moving forward will be evaluating the strength of the U.S. economy using analytical dashboarding and time series projecting.

**Figure 5**  
U.S. Dollar International Exchange Rates - 1970 to Present



**Security**

There are multiple steps of security that can be taken when using MySQL as the host for a relational database. Currently, there is no sensitive information that may need to have authorization to access within the database, but the pipeline and database is highly scalable. Due to this, it is within best practice to give privileges to users only when required. In this case, we have stored the data in 6 tables and made 10 views for those tables. Within the structure of the company, permissions can be given to access the connected database allowing the team concerned with tracking long term exchange rate trends to evaluate the strength of the U.S. Dollar in comparison to other countries. Doing so could add context to evaluating the economic strength for the United States, for example. Views have been created for just this purpose. In Figure above, the group of USD views define the long-term relationships between the U.S. Dollar and various other currencies in potential countries of interest. Permission could be added for certain users working on such a project to access only these views rather than the entirety of the database to protect any future sensitive information that may be added to the database in addition to protecting the integrity of the database from any mistaken queries that may add false data to the structure.

In addition to having views to protect the database from security or structural risks, steps have been taken within the pipeline itself to hide credentials. Credentials in this case are stored locally within the machine as environment variables. Another step of security we have taken into consideration is backing up the storage. In the case of this pipeline, CSV backups are stored within the “data” folder of the repository. In the event that the database is attacked and wiped in addition to the website being taken down or attacked in some way, the most recent version of the CSV files hosted on the website are stored as backups.

**Data Integrity Controls and Logging**

The Pipeline Architecture and Process section highlights logging of successful processing or exceptions, with process Step 4.B explicitly noting a data completeness control. These are simply emphasized here as key "non-functional" considerations for an automated pipeline.

**Pipeline Architecture and Process**

**Scalability**

Given that the nature of the data and ETL pipeline is storing the data as a structured relational database within MySQL, the database will be highly scalable. To cement this scalable construct, as the CSV files from BIS comes wide, with dates as columns rather than rows, each CSV is stored as two tables within the database with matching keys to call back. Doing so allows one table to be smaller, in the thousands or hundreds in rows, with more computationally expensive information like descriptions, country, and title. Meanwhile, the larger table, in the hundreds of thousands of rows, stores only row key, data, and value. Establishing the schema in this way allows for sub-querying to be more optimized as the smaller table, with more expensive information, can be filtered then the keys can be matched to inner join the much larger table containing dates and values. This process will make the database more scalable as it grows each month. For example, WS\_LONG\_CPI\_csv\_col.csv becomes two tables, Figure 1 above highlights this more clearly. Table one is the smaller table with the columns; consumer\_prices\_id, frequency, reference\_area, unit\_of\_measure, and series. Table two is the longer table with consumer\_prices\_values\_id, consumer\_prices\_id, date, and values. An example of the sub-query filtering is the view united\_states\_cp that grabs the IDs associated with United States reference\_area, which, are then used to pull just under 1000 rows of dates and values in table 2. This greatly reduces the computation time to grab potentially thousands of rows.

**Gaps and Opportunities (Extensibility)**

BIS Pipeline gaps may be summarized in two areas: baseline-only scope and potential pipeline environment limitations.

As highlighted above, BIS Pipeline was created as an automated pipeline framework for BIS - and, prospectively, complementary - data. Considering this, output examples are limited to baseline data and analytical visualizations. That said, pipeline extensibility was considered in design and implementation, and is encouraged.

Regarding potential environment limitations, executing the pipeline on a local (e.g., vs. scalable cloud) environment may limit options when expanding the scope of the database to include most or all the datasets housed on BIS’s website. For instance, that substantial increase in data will result in more space to house the data which may lead to further investment in hardware. So, while the pipeline itself will be highly scalable moving forward, the limitations of on-site hardware could potentially put hold on materially expanding the database.

In conclusion, BIS Pipeline was developed as a base framework to enable and extend analytics and other usage of global financial data provided by the Bank of International Settlements. As noted above under Opportunity and Solution, possibilities range from descriptive analytics and visualization to advanced predictive models and others. The pipeline's architecture, implementation approach, and automation support data and consumption pattern extensions limited only by user interest and ambition.

**References**

BIS. (2023). *Statistics – Download BIS statistics in a single file.* Retrieved from:

https://www.bis.org/statistics/full\_data\_sets.htm