## By: Elif Evrim Polat, Juan Castaneda, Nichole Edet

## ETL PROJECT

## Summary:

As the popularity among cryptocurrencies rise, we selected 10 cryptocurrencies, pulled data on their hourly and daily price changes for the past year. The database that we have can be used by data analysts to find the trends in these cryptocurrency prices and possibly predict what may happen in the upcoming months and/or years.

## Extraction:

We used 2 different sources of data. We downloaded the CSV files from Crypto Data Download website; [www.CryptoDataDownload.com](http://www.CryptoDataDownload.com) and we got our API requests in JSON format from Coin API website; <https://www.coinapi.io/>

* Webscraping using pandas, obtained today’s 10 different cryptocurrencies table at <https://coinmarketcap.com>.
* Webscraping using BeautifulSoup to obtain the available cryptocurrencies on the following website: <https://www.cryptodatadownload.com/data/binance/>. We found a paragraph that contained all the links of the available historical data. We had to find which of the daily cryptocurrencies were on this website.
* For the cryptocurrencies that were not available on the above website, we had to extract the data using the API from: <https://www.coinapi.io/>

This the API key that was used to get historical data for the cryptocurrency es that were not on the crypto data download website: /v1/ohlcv/{asset\_id\_base}/{asset\_id\_quote}/history?period\_id={period\_i d}&time\_start={time\_start}&time\_end={time\_end}&limit={limit}

These are the required parameters for the use of the API:

Asset\_id\_base: Represents the coin symbol, example, Bitcoin=“BTCT” , Etherium= “ETH”

Asset\_id\_quote: Represents the currency equivalence, example: “ USD” , “ BTC”, etc.

Period\_id: Represents the period of time, example: “1Day, 1Hr”

Limit: Specifies how many records the API is going to return.

Time start: Specifies the start date.

Time end: Specifies the end date.

## Transform:

In order to transform the publicly found data, we performed the following to transform into our database:

* The CSV files that we downloaded from [www.CryptoDataDownload.com](http://www.CryptoDataDownload.com), we transformed them to pandas data frames in Jupyter Notebook.
* The JSON data that we obtained from the API, we converted into a pandas data frame.
* We removed unnecessary columns that were not needed from our cryptos\_df dataframe;

| ‘24h % | 7d % | Market Cap | Volume(24h) | Circulating Supply | Last 7 Days | Unnamed: 10’ |
| --- | --- | --- | --- | --- | --- | --- |

Table

Description automatically generated

We kept the following columns: '#', 'Name', 'Price'. Name was renamed to ‘coin\_name’, ‘#’ was renamed to ‘ID’, Price was renamed to latest\_price. We added ‘coin\_symbol’ to our data frame. We selected the first 10 coins .

Table

Description automatically generated

* **Cleaning up CSV Files from** [**www.CryptoDataDownload.com**](http://www.CryptoDataDownload.com)**;**

We removed unnecessary columns df dataframe; unix, symbol, Volume BTC, Volume USDT, tradecount;

## Table Description automatically generated

We kept the following columns; coin\_id, date, open, high, low, close to match our SQL tables structure.

Table

Description automatically generated

Since some of the dates in the csv file are in the next form "2020-11-20 07-AM", we will need to reformat them so SQL can read it as a Date format, Example: "2020-11-20 07-AM" after formatting would be ""2020-11-20 07:00:00""

Table

Description automatically generated

**Cleaning up CSV Files from** <https://www.coinapi.io/>

We removed unnecessary columns df1 dataframe

Table

Description automatically generated

We renamed and removed unnecessary columns from our df1 dataframe, we converted date to a datetime format, we sorted df1 dataframe in descending order (latest date first).

Table

Description automatically generated

* We utilized for loop to automate the clean-up process.

## Load:

* We used quick database diagrams to create SQL schemas. We created coins table, hourly price and daily price on Postgres using the SQL schema generated on www.quickdatabasediagrams.com

Graphical user interface, application, Teams

Description automatically generated

* We put cryptocurrency dataframes into coins table in PgAdmin server.

This was achieved utilizing pandas built in method to\_sql().

The following example represents the process of loading our dataframe into PostgreSQL using this method

***df.to\_sql(name=table\_name, con=engine, if\_exists='append', index=False)***

where:

* table\_name specifies the SQL table in our database,
* con=engine, is the engine we created using SQLAlchemy, this object creates a connection to Pgadmin server using the next parameters.

connection\_string = f"postgres:{password}@localhost:5433/cryptos\_db"

engine = create\_engine(f'postgresql://{connection\_string}')

* index=False tells pandas not to send the dataframe index column

Table

Description automatically generated

After pulling and cleaning the CSV files, we connected the dataframes to the Postgres database in PgAdmin. We chose to use PostgreSQL because we needed a relational database system where we were able to relate two different records using foreign key. We came to the conclusion that PostgreSQL is efficient at handling large amount of data. Our datasets were very large as it was listing hourly and daily change for 10 cryptocurrencies in 1 year or available historical data.