ETL Final Analysis

We wanted to see if different interest rates had any predictive power to identify the forward returns of S&P 500. The following sources were used to extract the data and determine whether our hypothesis was true or if there was another theory that would affect the forward rate of return.

Extract:

The data was brought from 2 sources: 1) St. Louis Fred Database, 2) Yahoo Finance.

1) Fred Database: (https://fred.stlouisfed.org/)
   Data Extracted: CPI(Monthly), AAA bonds(Monthly), BAA Bonds(Monthly), 3-month T-Bill(Daily), 10 minus 2 year rates(Monthly)
   Date Range: 1/1/1995 - 11/1/2019

   Sample Code:
   ```
   cpi = web.DataReader("CPIAUCNS", "fred", sdt, edt)
   aaa = web.DataReader("AAA", "fred", sdt, edt)
   baa = web.DataReader("BAA", "fred", sdt, edt)
   aaa_baa = pd.merge(aaa,baa, on="DATE", how ="left")
   T10Y2YM = web.DataReader("T10Y2YM", "fred", sdt, edt)
   DTB3 = web.DataReader("DTB3", "fred", sdt, edt)
   ```

2) Yahoo Finance: (yahoo.com)
   Data Extracted: S&P500 Closing Prices(Daily)
   Date Range: 1/1/1994 - 11/1/2019

   ```
   SPY = web.get_data_yahoo('^GSPC', start=dt.datetime(1994, 1, 1),
   end=dt.datetime(2019, 12, 1))
   ```

Transform:

1) We removed NaN values from the datasets.
2) We transformed 3-month T-Bill and S&P 500 daily values to monthly value by taking the mean for the month.
3) We merged AAA bonds and BAA bonds in a single dataframe.
4) We removed a few columns that we did not need and calculated the 12 month forward returns for S&P500.

5) We calculated a new column to hold the 12-month forward returns and dropped rows that had NaN values

   Sample Code:

```
DTB3_C = DTB3.dropna()

SPY = SPY.drop(["High", "Low", "Open","Volume", "Adj Close"], axis=1)
SPY = SPY.resample('1M').mean()
SPY_Pct_Chg = SPY.pct_change(12)
SPY_Merged = pd.merge(SPY,SPY_Pct_Chg, on="Date", how ="left")

SPY_Merged.rename(columns = {'Close_x':'SPYClose', 'Close_y':'SPY12mFR'
                     }, inplace = True)

SPY_Merged_C = SPY_Merged.dropna()

SPY_Merged_C.index.names = ['DATE']
```

Load:

The codes listed below were used to create the tables in SQL and another code was used to verify that the databases for our sources were correctly added. PGAdmin was used to create the tables.

```
Create Table "Bonds"(
"DATE" date not Null,
  "AAA" varchar not Null,
  "BAA" varchar not Null,
  constraint "pk_Bonds" Primary Key (
  "DATE")
);

Create Table "CPI"(
"DATE" date not Null,
  "CPI" varchar not Null,

  constraint "pk_CPI" Primary Key (
  "DATE")
);

Create Table "3monthTBill" (
  "DATE" date NOT NULL,
  "3monthBill" varchar NOT NULL,
  CONSTRAINT "pk_3monthTBill" Primary Key (
  "DATE")
);

Create Table "10Minus2" (
  "DATE" date NOT NULL,
  "10Minus2" varchar NOT NULL,
  CONSTRAINT "pk_10Minus2" Primary Key (
  "DATE")
);

Create Table "S&PData" (
  "DATE" date NOT NULL,
  "SPYClose" varchar NOT NULL,
  "SPY12mFR" varchar NOT NULL,
  CONSTRAINT "pk_S&PData" Primary Key (
  "DATE")
);
```

PGAdmin was chosen over MongoDB planning on doing time series analysis where we need data as a primary key. Since the multiple tables were going to be joined, we needed to use the relational database for this process.

Final Tables that were used:
1)S&PData
2)10Minus2
3)3monthTBill
4)CPI
5)Bonds

SQLAlchemy was used in the process to load the data in the databases and load the tables. Code listed below was used for the connection string.

```
rds_connection_string = user + ":" + password + "@localhost:5432/ETLStock_db"
engine = create_engine(f'postgresql://{rds_connection_string}')

aaa_baa_df.to_sql(name='Bonds', con=engine, if_exists='append', index=True)
CPI.to_sql(name='CPI', con=engine, if_exists='append', index=True)
DTB3_M.to_sql(name='3monthTBill', con=engine, if_exists='append', index=True)
T10Y2YM.to_sql(name='10Minus2', con=engine, if_exists='append', index=True)
SPY_Merged_C.to_sql(name='S&PData', con=engine, if_exists='append', index=True)
```

The following queries were run to verify that the data was properly loaded.

```
pd.read_sql_query('select * from "S&PData"', con=engine).head()
pd.read_sql_query('select * from "10Minus2"', con=engine).head()
pd.read_sql_query('select * from "3monthTBill"', con=engine).head()
pd.read_sql_query('select * from "CPI"', con=engine).head()
pd.read_sql_query(‘select * from “Bonds”’, con=engine).head()
```