

# **MSA 8040 Data Management for Analytics**

## **Final Project Report**

### **Data Analysis with Estimote.com**

**Team: EstiAmaze Teens**

Bharath Badri Venkata

Vaishnavi Mada

Srujani Mareddy

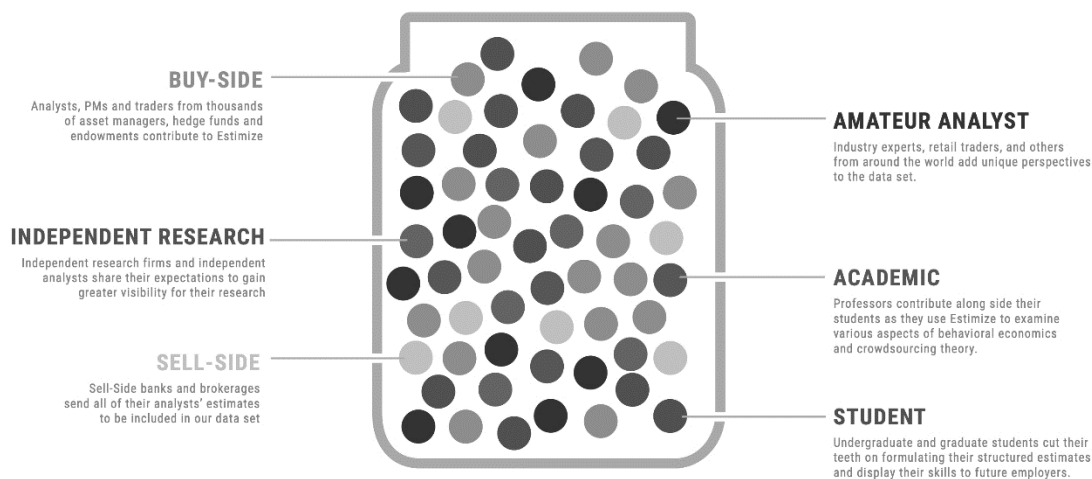


**Tools used: Selenium, MySQL, Python**



## About Estimize.com

Estimize.com is a free and open financial estimates platform where hedge fund, independent, and sell-side analysts, as well as private investors, industry professionals, and students, contribute their earnings per share (EPS) and revenue predictions for public companies. When compared to sell side only data sets, Estimize gives a more accurate and representative perspective of expectations by obtaining estimates from a diverse community of individuals.



Estimize, which was founded in 2011, is an open financial estimates platform that collects forward-looking financial estimates from independent, buy-side, and sell-side analysts, as well as private investors and academics.

Estimize provides a more accurate, timely, and representative perspective of expectations by obtaining estimates from a varied group of persons, as opposed to sell side exclusive data sets, which suffer from various major biases.

Estimize currently has 96,211 analysts contributing, resulting in quarterly coverage of approximately 2,200 equities and 80 economic indicators. Over 70% of the time, and by 15% on average, the Estimize consensus has proven to be more accurate than comparable sell side data sets.

## How does Estimize.com work?



In exchange for their honest pseudonymous contributions, Estimize platform contributors gain free access to view their data. Estimize manages contribution honesty and quality using a combination of machine learning algorithms and statistical methodologies, as well as a human layer of evaluation. Estimize ranks and scores analysts, making it simple for them to store, benchmark, and analyze their own accuracy.

Their data is available for purchase via the Estimize web platform, API, and FTP, with no contribution required. Their clientele includes institutional managers that run fully systematic quantitative strategies, fundamental managers who run quanta mental, long/short, and long only strategies, sell side market making and research desks, and macro investors and vol traders. Estimize represents the market's actual consensus, with its data frequently cited in prominent financial media outlets such as Bloomberg, The Wall Street Journal, CNN Money, The Street, Forbes, Barron's, Investor's Business Daily, and Business Week.

## Problem Statement:

- Extract comprehensive data for 29 stock tickers from Estimote.com, covering 2020,2021,2022 years across four quarters.
- Develop structured datasets to systematically store scraped information, encompassing EPS data, Company details, and Analyst insights.
- Construct a robust database leveraging the generated datasets, facilitating seamless queries to extract accurate and relevant information.

## Problem solution & procedure:

- The data on Estimote.com is unorganized and not easily downloaded or useable.
- Using Python and Selenium, we will scrape data from Estimote.com. The scraped data will then be saved in json or csv format.
- Finally, we will import the stored data into MYSQL workbench and query the data as needed.

### 1. Setting up the environment in Jupyter Notebook:

```
In [1]: import os
import re
import time
import csv
import json
import random
import requests
import pandas as pd
import numpy as np
from datetime import date, timedelta, datetime
import selenium
from selenium import webdriver
from bs4 import BeautifulSoup
```

```
In [2]: from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.support.ui import Select
from selenium.webdriver.support.select import Select
from selenium.webdriver.chrome.options import Options
```

```
In [3]: pip install selenium==3.141.0
```

```
Collecting selenium==3.141.0
  Downloading selenium-3.141.0-py2.py3-none-any.whl (904 kB)
----- 0.0/904.6 kB ? eta -:-:--
----- 10.2/904.6 kB ? eta -:-:--
----- 41.0/904.6 kB 495.5 kB/s eta 0:00:02
----- 122.9/904.6 kB 1.0 MB/s eta 0:00:01
----- 512.0/904.6 kB 2.9 MB/s eta 0:00:01
----- 904.6/904.6 kB 4.4 MB/s eta 0:00:00
Requirement already satisfied: urllib3 in c:\users\bvb\anaconda3\lib\site-packages (from selenium==3.141.0) (1.26.16)
Installing collected packages: selenium
```

## 2. Setting up the chrome driver and defining the browser object:

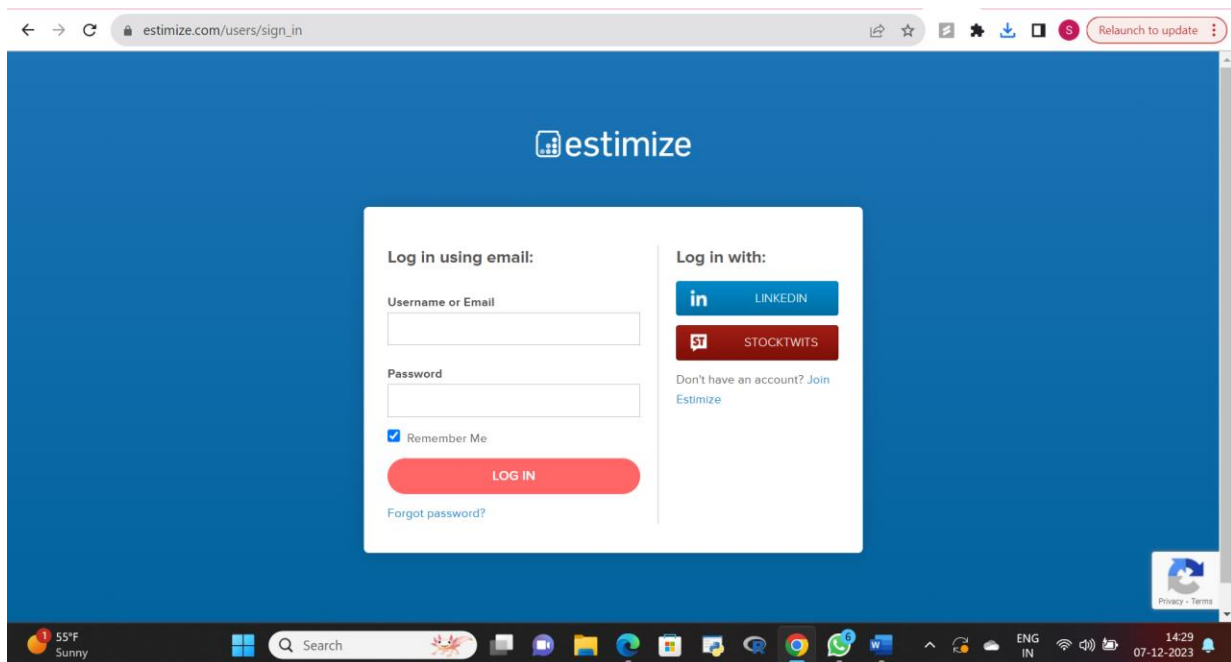
```
In [4]: pip install webdriver-manager
```

```
Collecting webdriver-manager
  Obtaining dependency information for webdriver-manager from https://files.pythonhosted.org/packages/b1/51/b5c11cf739ac4eecd611794a0ec9df420d0239d51e73bc19eb44f02b48b/webdriver_manager-4.0.1-py2.py3-none-any.whl.metadata
  Using cached webdriver_manager-4.0.1-py2.py3-none-any.whl.metadata (12 kB)
Requirement already satisfied: requests in c:\users\bvb\anaconda3\lib\site-packages (from webdriver-manager) (2.31.0)
Requirement already satisfied: python-dotenv in c:\users\bvb\anaconda3\lib\site-packages (from webdriver-manager) (0.21.0)
Requirement already satisfied: packaging in c:\users\bvb\anaconda3\lib\site-packages (from webdriver-manager) (23.1)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\bvb\anaconda3\lib\site-packages (from requests->webdriver-manager) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\bvb\anaconda3\lib\site-packages (from requests->webdriver-manager) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\bvb\anaconda3\lib\site-packages (from requests->webdriver-manager) (1.26.16)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\bvb\anaconda3\lib\site-packages (from requests->webdriver-manager) (2023.7.22)
Using cached webdriver_manager-4.0.1-py2.py3-none-any.whl (27 kB)
Installing collected packages: webdriver-manager
Successfully installed webdriver-manager-4.0.1
Note: you may need to restart the kernel to use updated packages.
```

```
In [5]: chrome_options = webdriver.ChromeOptions()
chrome_options.headless = False
browser = webdriver.Chrome(executable_path = 'C:\\Users\\BVB\\Downloads\\D\\WEbprj1\\chromedriver',
                           options = chrome_options)
url_head = 'https://www.estimize.com/users/sign_in'
browser.get(url_head)
time.sleep(random.uniform(1,5))
```

## 3. Chrome launches the Estimize user sign-in page in the browser:

[https://www.estimize.com/users/sign\\_in](https://www.estimize.com/users/sign_in)



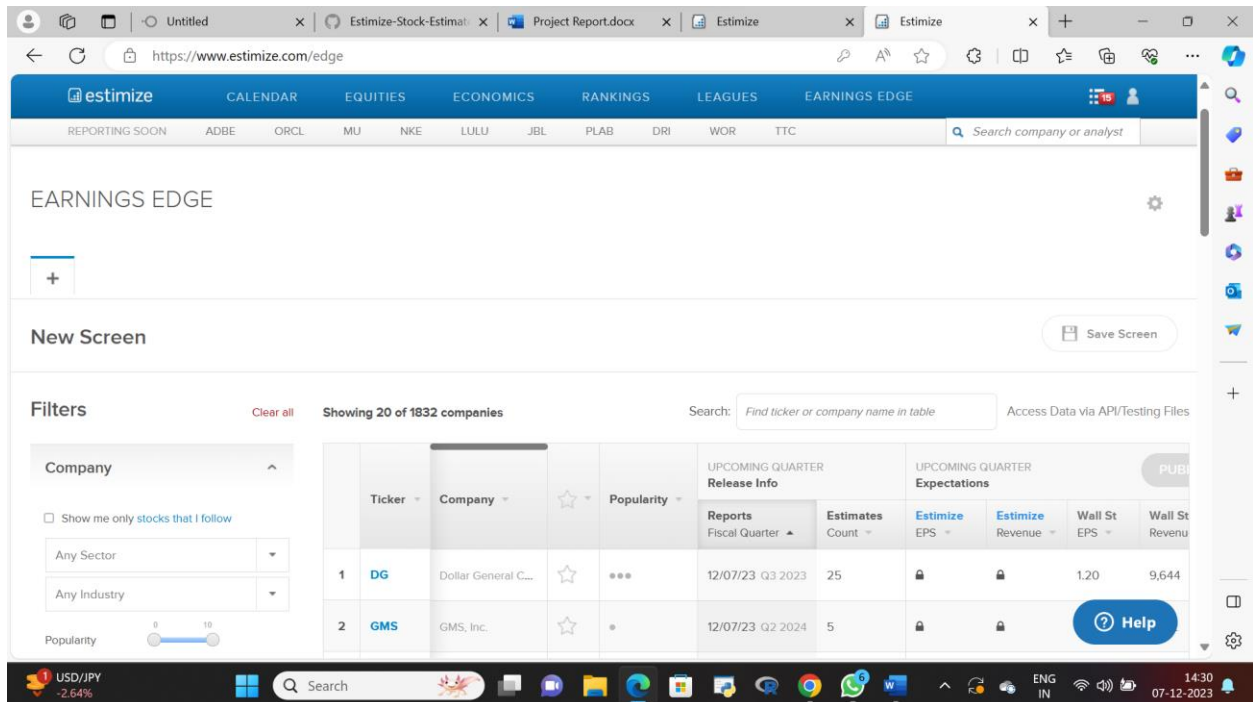
#### 4. User authentication using credentials in the sign-in page:

```
In [6]: import time
from selenium import webdriver

def sign_in(username, password):
    url = 'https://www.estimize.com/users/sign_in'
    browser = webdriver.Chrome()
    browser.get(url)
    email_field = browser.find_element_by_id('user_login')
    password_field = browser.find_element_by_id('user_password')
    submit_btn = browser.find_element_by_xpath('//input[@type="submit"]')
    email_field.send_keys(username)
    password_field.send_keys(password)
    submit_btn.click()
    time.sleep(1)
    browser.quit()

username = 'bharathbv8@gmail.com'
password = 'qwerty123asdzxc'
sign_in(username, password)
```

#### 5. User page is displayed upon successful login:



The screenshot shows the Estimize website interface. The top navigation bar includes links for CALENDAR, EQUITIES, ECONOMICS, RANKINGS, LEAGUES, and EARNINGS EDGE. Below this, there's a section titled "EARNINGS EDGE" with a "New Screen" button and a "Save Screen" button. A "Filters" section on the left allows users to filter by Company, Sector, and Industry. The main content area displays a table of 20 companies, showing their Ticker, Company Name, Popularity, and upcoming quarter release information. The table includes columns for Reports, Estimates, and Expectations.

	Ticker	Company	Popularity	UPCOMING QUARTER Release Info		UPCOMING QUARTER Expectations			
				Reports Fiscal Quarter	Estimates Count	Estimize EPS	Estimize Revenue	Wall St EPS	Wall St Revenue
1	DG	Dollar General C...	***	12/07/23 Q3 2023	25			1.20	9,644
2	GMS	GMS, Inc.	*	12/07/23 Q2 2024	5				

## 6. Ticker and quarter selection:

```
In [8]: tickers=["AMZN","AAPL","MSFT","GOOG","TSLA","JNJ","PG","NVDA","CSCO","BABA","HD",
"BIDU","WMT","CRM","LULU","TGT","PANW","ADBE","VMW","MU","NKE","ORCL","BB","HPQ","COST","AMAT",
"BAC","CVX","AMGN","PG"]

quarters = ['fq1-2020', 'fq2-2020', 'fq3-2020', 'fq4-2020']
```

## 7. Building the Company table:

The following section of the webpage was scraped to generate the company data fields.

[https://www.estimize.com/amzn/fq4-2023?metric\\_name=eps&chart=historical](https://www.estimize.com/amzn/fq4-2023?metric_name=eps&chart=historical)

REPORTING SOON   ADBE   ORCL   MU   NKE   LULU   JBL   PLAB   DRI   WOR   TTC										Search company or analyst
>										
<b>AMZN</b> Amazon.com Inc. Consumer Discretionary > Internet & Catalog Retail		✓ FOLLOWED		FOLLOWERS 10,804		ANALYSTS 6,874		TOP ANALYST Wx_Torn		NEXT REPORT FQ4 '23 56 days 1 h 27 m 02/01/24 AMC (estimated)

The company basic information, such as Ticker, company name, Sectors, Industries, number of followers, number of analysts.

- i. ticker: AMZN
- ii. name: Amazon.com Inc.
- iii. Sectors: Consumer Discretionary
- iv. Industries: Internet and Catalog Retail
- v. number of followers: 10804
- vi. number of analysts: 6874

## Python and Selenium code for generating data for the Company table:

```
In [9]: def scrape_overview_data(tickers, quarters, browser):
        overview_all = {}

        try:
            for ticker in tickers:
                for quarter in quarters:
                    url = f'https://www.estimate.com/{ticker}/{quarter}?metric_name=eps&chart=historical'
                    browser.get(url)
                    time.sleep(1)
                    overview = {
                        'quarter': quarter,
                        'Ticker': browser.find_element_by_class_name('release-header-information-title').text,
                        'Company Name': browser.find_element_by_class_name('release-header-information-description').text
                    }
                    sectors = browser.find_element_by_class_name('release-header-information-breadcrumb')
                    overview['Sectors'] = sectors.text.split('>')[0]
                    overview['Industries'] = sectors.text.split('>')[1]
                    numbers = browser.find_elements_by_class_name('summary-sub-title')
                    overview['Number of Followers'] = numbers[0].text
                    overview['Number of Analysts'] = numbers[1].text
                    overview_all[len(overview_all)] = overview
        except Exception as e:
            print(f"An error occurred: {e}")

        return overview_all
```

```
In [10]: company_info = scrape_overview(browser)

        with open('Company.json', 'w') as outfile:
            json.dump(company_info, outfile, indent=4)

        df=pd.DataFrame(company_info)
        df_trans = df.T
        df_trans.to_csv('Company.csv')
```

## Company.csv



	A	B	C	D	E	F	G	H	I
1		quarter	Ticker	Company	Sectors	Industries	Number of	Number of	Analysts
2	0	fq1-2020	AMZN	Amazon.co	Consumer	Internet &	10,781	6,874	
3	1	fq2-2020	AMZN	Amazon.co	Consumer	Internet &	10,781	6,874	
4	2	fq3-2020	AMZN	Amazon.co	Consumer	Internet &	10,781	6,874	
5	3	fq4-2020	AMZN	Amazon.co	Consumer	Internet &	10,781	6,874	
6	4	fq1-2020	AAPL	Apple Inc.	Informatic	Computer	38,285	12,178	
7	5	fq2-2020	AAPL	Apple Inc.	Informatic	Computer	38,285	12,178	
8	6	fq3-2020	AAPL	Apple Inc.	Informatic	Computer	38,285	12,178	
9	7	fq4-2020	AAPL	Apple Inc.	Informatic	Computer	38,285	12,178	
10	8	fq1-2020	MSFT	Microsoft	Informatic	Software	15,996	4,459	
11	9	fq2-2020	MSFT	Microsoft	Informatic	Software	15,996	4,459	
12	10	fq3-2020	MSFT	Microsoft	Informatic	Software	15,996	4,459	
13	11	fq4-2020	MSFT	Microsoft	Informatic	Software	15,996	4,459	
14	12	fq1-2020	GOOGL	Alphabet Inc.	Informatic	Internet S	21,812	5,653	
15	13	fq2-2020	GOOGL	Alphabet Inc.	Informatic	Internet S	21,812	5,653	
16	14	fq3-2020	GOOGL	Alphabet Inc.	Informatic	Internet S	21,812	5,653	
17	15	fq4-2020	GOOGL	Alphabet Inc.	Informatic	Internet S	21,812	5,653	
18	16	fq1-2020	TSLA	Tesla Inc.	Consumer	Automobi	9,367	7,914	
19	17	fq2-2020	TSLA	Tesla Inc.	Consumer	Automobi	9,367	7,914	
20	18	fq3-2020	TSLA	Tesla Inc.	Consumer	Automobi	9,367	7,914	
21	19	fq4-2020	TSLA	Tesla Inc.	Consumer	Automobi	9,367	7,914	

## 8. Building the EPS table:

This table contains the EPS information, including:

- i. Reported Earnings
- ii. Estimate Consensus
- iii. Estimate Mean
- iv. Wall Street Consensus
- v. EPS estimations of all available analysts

The following section of the webpage shown below was used to scrape the EPS Analyst data fields.

[https://www.estimize.com/amzn/fq1-2020?metric\\_name=eps&chart=historical](https://www.estimize.com/amzn/fq1-2020?metric_name=eps&chart=historical)

Estimize

CALENDAR EQUITIES ECONOMICS RANKINGS LEAGUES EARNINGS EDGE

REPORTING SOON ADBE ORCL MU NKE LULU JBL PLAB DRI WOR TTC

Search company or analyst

### EPS Analysts: FQ1'20

Showing 30/402 estimates [View all-time analyst rankings for AMZN](#) ☐ Show only my followed analysts

Chart	Analyst	Rank	Points	Value	Confidence	Last Revised
	<b>AMZN Reported Earnings</b> Amazon.com Inc.			0.25		04/30/20
	<b>Estimize Consensus</b> 385 estimates weighted			0.33		04/30/20
	<b>Estimize Mean</b> 386 estimates averaged			0.34		04/30/20
	Wall Street Consensus			0.31		04/30/20
	<b>Flanner</b> Flanner	1	25	0.25	5.0	04/20/20
	<b>Analyst_9989915</b> Analyst_9989915	2	25	0.25	5.0	04/21/20
	<b>Analyst_5637861</b> Analyst_5637861	3	25	0.25	5.0	04/21/20

55°F Sunny

Search

ENG IN 14:37 07-12-2023

## Python and Selenium code for generating data for the EPS table:

```
In [11]: def scrape_eps(browser):
eps_data = {}

for ticker in tickers:
    for quarter in quarters:
        url = f'https://www.estimize.com/{ticker}/{quarter}?metric_name=eps&chart=historical'
        browser.get(url)
        time.sleep(1)
        analysts_generic = browser.find_elements_by_class_name('estimates-tbl-consensus-column')
        values_generic = browser.find_elements_by_class_name('estimates-tbl-consensus-eps')
        for generic_count in range(len(analysts_generic)):
            eps_generic = {
                'quarter': quarter,
                'Ticker': ticker,
                'Company Name': browser.find_element_by_class_name('release-header-information-title').text,
                'Name': analysts_generic[generic_count].text.split('\n')[0],
                'Type': "Generic",
                'Estimated Value': values_generic[generic_count].text
            }
            eps[len(eps_data)] = eps_generic
        analysts = browser.find_elements_by_class_name('username')
        analysts_values = browser.find_elements_by_class_name('estimates-tbl-eps')

        for analyst_count in range(len(analysts)):
            eps_analyst = {
                'quarter': quarter,
                'Ticker': ticker,
                'Company Name': browser.find_element_by_class_name('release-header-information-title').text,
                'Name': analysts[analyst_count].text,
                'Type': "Analyst",
                'Estimated Value': analysts_values[analyst_count].text
            }
            eps[len(eps_data)] = eps_analyst
        return eps_data
```

```
In [12]: eps_info = scrape_eps(browser)

with open('EPS.json', 'w') as outfile:
    json.dump(eps_info, outfile, indent=4)

df=pd.DataFrame(eps_info)
df_trans = df.T
df_trans.to_csv('EPS.csv')
```

## Eps.csv

	A	B	C	D	E	F	G	H
1		quarter	Ticker	Company I	Name	Type	Estimated	Value
2	0	fq1-2020	AMZN	Amazon.co	AMZN Rep	Generic	0.25	
3	1	fq1-2020	AMZN	Amazon.co	Estimize C	Generic	0.33	
4	2	fq1-2020	AMZN	Amazon.co	Estimize M	Generic	0.34	
5	3	fq1-2020	AMZN	Amazon.co	Wall Stree	Generic	0.31	
6	4	fq1-2020	AMZN	Amazon.co	Flanner	Analyst	0.25	
7	5	fq1-2020	AMZN	Amazon.co	Analyst_9	Analyst	0.25	
8	6	fq1-2020	AMZN	Amazon.co	Analyst_5	Analyst	0.25	
9	7	fq1-2020	AMZN	Amazon.co	Analyst_4	Analyst	0.25	
10	8	fq1-2020	AMZN	Amazon.co	Analyst_1	Analyst	0.24	
11	9	fq1-2020	AMZN	Amazon.co	Analyst_2	Analyst	0.27	
12	10	fq1-2020	AMZN	Amazon.co	Analyst_7	Analyst	0.27	
13	11	fq1-2020	AMZN	Amazon.co	Shawshan	Analyst	0.28	
14	12	fq1-2020	AMZN	Amazon.co	Analyst_1	Analyst	0.22	
15	13	fq1-2020	AMZN	Amazon.co	Analyst_5	Analyst	0.29	
16	14	fq1-2020	AMZN	Amazon.co	Analyst_8	Analyst	0.29	
17	15	fq1-2020	AMZN	Amazon.co	Analyst_5	Analyst	0.29	
18	16	fq1-2020	AMZN	Amazon.co	Analyst_9	Analyst	0.29	
19	17	fq1-2020	AMZN	Amazon.co	Analyst_9	Analyst	0.3	
20	18	fq1-2020	AMZN	Amazon.co	Analyst_8	Analyst	0.3	

## 9. Building the Analyst table:

This table contains the information about each analyst, including Analyst name, Roles, Join date, Analyst Confidence score, number of estimates, Stocks Covered, pending estimates, Scored estimates.

- i. name: Wx\_Torn
- ii. roles: Non-Professional Financials Professional Services
- iii. Join Date: Oct 2015
- iv. Analyst Confidence Score: 7.9
- v. error rate: 7.70 %
- vi. Accuracy Percentile: 60%
- vii. points: 3,501
- viii. points/Estimate: 15.5
- ix. stocks: 227
- x. pending: 11

## Scrape All covered stock estimates by the analyst

[illegible]

### All pending stock estimates by the analyst

[illegible]

## All scored stock estimates by the analyst

### Scored Estimates

Showing 5/7306 scored estimates

Ticker	Quarter	Reported	Rank	EPS Points	Revenue Points	Total Points
DG	Q3 2023	Dec 7, 2023	2 / 25	25	19	44
CIEN	Q4 2023	Dec 7, 2023	7 / 10	9	11	20
VEEV	Q3 2024	Dec 6, 2023	14 / 28	15	-2	13
CHWY	Q3 2023	Dec 6, 2023	7 / 20	-2	-2	-4
CPB	Q1 2024	Dec 6, 2023	7 / 14	10	-1	9
Show 20 more						

```
In [14]: def scrape_analysts_list(tickers, quarters, browser):
analysts_list = {}

for ticker in tickers:
    for quarter in quarters:
        url = f'https://www.estimize.com/{ticker}/{quarter}?metric_name=eps&chart=historical'
        browser.get(url)
        time.sleep(1)
        links = browser.find_elements_by_tag_name("a")
        for link in links:
            href = link.get_attribute('href')
            if href and '/users/' in href and 'sign_out' not in href:
                analysts_list[len(analysts_list)] = href

ana_list_final = Remove_dup(ana_list)
print (len(ana_list_final))

return ana_list_final
```

```
In [15]: def scrape_analysts_info(browser):
analysts_urls = scrape_analysts_list(browser)
analyst_profile_metrics = ['Error rate', 'Accuracy Percentile', 'Points', 'Points/Estimate', 'Stocks', 'Pending']
analysts_overview_final = {}

for analyst_url in analysts_urls:
    browser.get(analyst_url)
    time.sleep(1)

    analyst_overview = {}
    try:
        analyst_overview['Name'] = browser.find_element_by_class_name('profile-display-name').text
        analyst_overview['User ID'] = browser.find_element_by_class_name('profile-username').text
        analyst_overview['Role'] = browser.find_element_by_class_name('profile-bio-categorizations').text

        date_info = browser.find_element_by_class_name('profile-activity-stats').text.split(' ')
        analyst_overview['Join Date'] = date_info[2] + ' ' + date_info[3]
        analyst_overview['Analyst Confidence Score'] = browser.find_element_by_class_name('value').text

        profile_stats = browser.find_elements_by_class_name('profile-stat')
        for i in range(len(profile_stats)):
            analyst_overview[analyst_profile_metrics[i]] = profile_stats[i].text

        analysts_overview_final[len(analysts_overview_final) + 1] = analyst_overview
    except Exception as e:
        # Handle exceptions (you might want to Log the exception or handle it differently)
        print(f"An error occurred: {e}")

return analysts_overview_final
```

```
In [16]: analysts_info = scrape_analysts_info(browser)

with open('Analysts_Info.json', 'w') as outfile:
    json.dump(analysts_info, outfile, indent=4)

df=pd.DataFrame(analysts_info)
df_trans = df.T
df_trans.to_csv('Analysts_Info.csv')
```

1504

## Analyst.csv

	A	B	C	D	E	F	G	H	I	J	K	L
1	Name	User ID	roles	Join Date	Analyst Confidence Score	Error rate	Accuracy Percentile	Points	Points/Estimate	Stocks	Pending	
2	1 Wx_Torn	Wx_Torn	Non Professional Information Technology Internet Software & Services	Oct-15	7.9	7.70%	60%	3,501	15.5	227	11	
3	2 Analyst_9989915	Analyst_9989915	Non Professional Other Other	Apr-20	7.1	-	-	-	-	-	0	
4	3 Flanner	Flanner	Non Professional Consumer Discretionary Textiles, Apparel & Luxury Goods	Apr-20	6.9	-	-	-	-	-	0	
5	4 Analyst_5637861	Analyst_5637861	Non Professional Health Care Biotechnology	Apr-20	6.7	-	-	-	-	-	0	
6	5 Analyst_4012279	Analyst_4012279	Non Professional Other Other	Apr-20	6.9	-	-	-	-	-	0	
7	6 Analyst_1390165	Analyst_1390165		Jul-17	7.3	5.60%	63%	200	22.2	9	0	
8	7 Analyst_2951	Analyst_2951	Non Professional Information Technology IT Services	Jul-18	8.2	-	-	-	-	-	0	
9	8 Analyst_7493397	Analyst_7493397	Non Professional Other Other	Apr-20	7.1	-	-	-	-	-	0	
10	9 Shawshank_redmp	Shawshank_redmp	Non Professional Information Technology Internet & Catalog Retail	Jul-18	7.4	-	-	-	-	-	0	
11	10 Analyst_140007	Analyst_140007	Non Professional Other Other	Sep-18	4.5	-	-	-	-	-	0	
12	11 Analyst_583258	Analyst_583258	Non Professional Information Technology Software	Apr-20	6.2	-	-	-	-	-	0	
13	12 Analyst_8198364	Analyst_8198364	Non Professional Other Other	Apr-20	7	-	-	-	-	-	0	
14	13 Analyst_530056	Analyst_530056		Jun-17	5.1	8.60%	42%	2	0.5	4	3	
15	14 Analyst_9651559	Analyst_9651559		Oct-19	7	-	-	-	-	-	0	
16	15 Analyst_9311958	Analyst_9311958	Financial Professional Independent Other	Jan-16	7.7	-	-	-	-	-	0	
17	16 Analyst_8113162	Analyst_8113162	Non Professional Other Other	Apr-20	7.3	-	-	-	-	-	0	
18	17 Scott Hendricks	shendricks	Non Professional Industrials Commercial Services & Supplies	Oct-12	8.1	12.20%	60%	4,157	13.9	298	24	
19	18 Barren_Wuffet	Barren_Wuffet	Non Professional Financials Real Estate Development	Mar-18	6.9	16.60%	43%	-13	-6.5	2	0	
20	19 Analyst_2971031	Analyst_2971031	Non Professional Other Other	Apr-20	6.9	-	-	-	-	-	0	
21	20 Analyst_7905880	Analyst_7905880	Non Professional Student	Oct-18	7.2	8.70%	53%	341	13.6	25	33	

## **10. Importing data into MySQL**



### **I. Create a database in MySQL named: msa8040**

```
/* drop a database if exists */  
drop database if exists msa8040;  
/* create a database with name 'msa8040_finalproject_sql' */  
create database msa8040;
```

### **II. Use the database: msa8040**

```
/* Use database */  
use msa8040;
```

### **III. Using sqlalchemy to Import all the three csv files.**

Eps.csv, Company.csv and Analyst.csv for all the three years in the database: msa8040.

```

In [1]: from sqlalchemy import create_engine
import pandas as pd

In [2]: !pip install pymysql

Collecting pymysql
  Obtaining dependency information for pymysql from https://files.pythonhosted.org/packages/e5/30/20467e39523d0cfc2b6227902d3687a16364307260c75e6a1cb4422b0c62/PyMySQL-1.1.0-py3-none-any.whl.metadata
  Using cached PyMySQL-1.1.0-py3-none-any.whl.metadata (4.4 kB)
  Using cached PyMySQL-1.1.0-py3-none-any.whl (44 kB)
  Installing collected packages: pymysql
  Successfully installed pymysql-1.1.0

In [3]: from sqlalchemy import create_engine
from urllib.parse import quote

# Replace 'your_username', 'your_password', 'your_host', and 'your_database' with your MySQL credentials
username = 'root'
password = 'bvb1801'
host = '127.0.0.1'
database = 'msa8040'

# URL-encode the password
password = quote(password)

# Create the engine
engine = create_engine(f'mysql+pymysql://{username}:{password}@{host}/{database}')

In [9]: df1 = pd.read_csv('EPSDATA.csv')
df2 = pd.read_csv('COMPANYDATA.csv')
df3 = pd.read_csv('ANALYSTINFODATA.csv', encoding='latin1')

In [10]: df1.to_sql('EPSDATA', con = engine, if_exists = 'append', chunksize = 1000, index=False)

C:\Users\BVB\AppData\Local\Temp\ipykernel_19108\3956606583.py:1: UserWarning: The provided table name 'EPSDATA' is not found exactly as such in the database after writing the table, possibly due to case sensitivity issues. Consider using lower case table names.
  df1.to_sql('EPSDATA', con = engine, if_exists = 'append', chunksize = 1000, index=False)

Out[10]: 11861

In [11]: df2.to_sql('COMPANYDATA', con = engine, if_exists = 'append', chunksize = 1000, index=False)

C:\Users\BVB\AppData\Local\Temp\ipykernel_19108\1421973142.py:1: UserWarning: The provided table name 'COMPANYDATA' is not found exactly as such in the database after writing the table, possibly due to case sensitivity issues. Consider using lower case table names.
  df2.to_sql('COMPANYDATA', con = engine, if_exists = 'append', chunksize = 1000, index=False)

Out[11]: 352

In [12]: df3.to_sql('ANALYSTINFODATA', con = engine, if_exists = 'append', chunksize = 1000, index=False)

C:\Users\BVB\AppData\Local\Temp\ipykernel_19108\3514334223.py:1: UserWarning: The provided table name 'ANALYSTINFODATA' is not found exactly as such in the database after writing the table, possibly due to case sensitivity issues. Consider using lower case table names.
  df3.to_sql('ANALYSTINFODATA', con = engine, if_exists = 'append', chunksize = 1000, index=False)

Out[12]: 3791

```

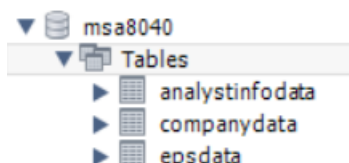
#### IV. Check for the successful import of all the three csv files

```

/*Show tables in the database msa8040_finalproject_sql */

show tables;

```





## V. Check for the number of records imported in each table.

```
/* Check for number of records in each table */
select * from eps;
select * from company;
select * from analyst;
```

### Results:

✓	3	16:12:43	Select * from epsdata LIMIT 0, 50000	11861 row(s) returned
✓	4	16:12:44	Select * from analystinfodata LIMIT 0, 50000	3791 row(s) returned
✓	5	16:12:44	Select * from companydata LIMIT 0, 50000	352 row(s) returned

## VI. Query the database to retrieve answers to the following questions.

- (a) Given a ticker, how many analysts have made estimations for its EPS? Rank them by their confidence score, total points, error rate or accuracy percentile?

- SELECT  
    E.Ticker,  
    E.Company,  
    E.Name,  
    A.UserID,  
    A.AnalystConfidenceScore,  
    A.Accuracy\_Percentile,  
    SUM(A.Points) as Total\_points  
FROM epsdata E  
INNER JOIN analystinfodata A  
    ON E.Name = A.Name  
WHERE E.Ticker = 'NKE'  
GROUP BY E.Ticker, E.Company, E.Name, A.UserID,  
A.AnalystConfidenceScore, A.Accuracy\_Percentile  
ORDER BY A.AnalystConfidenceScore DESC,  
Total\_points, A.Accuracy\_Percentile;

	Ticker	Company	Name	UserID	AnalystConfidenceScore	Accuracy_Percentile	Total_points
►	NKE	Nike Inc.	Bill_Maurer	Bill_Maurer	9.4	73%	2544
	NKE	Nike Inc.	Analyst_6452210	Analyst_6452210	8.7	-	0
	NKE	Nike Inc.	Mista	Mista	8.5	-	0
	NKE	Nike Inc.	agent4036	agent4036	8.5	-	0
	NKE	Nike Inc.	Analyst_6762933	Analyst_6762933	8.5	-	0
	NKE	Nike Inc.	Zilvinas Speteliunas	Spekoliunas	8.5	-	0
	NKE	Nike Inc.	Analyst_9676000	Analyst_9676000	8.5	62%	18
	NKE	Nike Inc.	Sandhu	Sandhu	8.4	-	0
	NKE	Nike Inc.	Analyst_6348688	Analyst_6348688	8.4	-	0
	NKE	Nike Inc.	Paw vet	Paw vet	8.4	-	0

/\* We are choosing Ticker 'NKE' and UserID who have made estimation for its EPS. \*/

- (b) Given an industry, how many companies are covered, the average number of analysts, the average bias between the Estimize Consensus and the Reported Earnings?

```

• SELECT
    count(distinct(C.Ticker_BI)) as Num_Companies,
    avg(Ticker_NumAnalysts) as Avg_Analysts,
    avg(E.ReportedEarnings_EPS +
E.EstConsensus_EPS)/2 as Avg_bias,
    Ticker_Industries
FROM
    company_details C
INNER JOIN
    eps_data E on C.Ticker_BI = E.Ticker_EPS
WHERE
    C.Ticker_Industries = 'Food & Staples Retailing'
GROUP BY
    C.Ticker_Industries;

```

	Num_Companies	Avg_Analysts	Avg_bias	Ticker_Industries
►	2	1576.0000	2.268125	Food & Staples Retailing

```
/* We are choosing Ticker_Industries - 'Food & Staples  
Retailing' Avg_Analysts and Avg_bias between the  
Estimize Consensus and the Reported Earnings */
```

(c) Which company have the largest number of analysts with confidence score greater than 7?

- ```
SELECT  
    DISTINCT C.Ticker_BI,  
    C.Ticker_NumAnalysts,  
    COUNT(I.AnalystUserName) AS No_of_Analysts  
FROM  
    company_details C  
INNER JOIN  
    analyst_eps P ON C.Ticker_BI = P.Ticker_AE  
INNER JOIN  
    analyst_data I ON P.AnalystSecondaryName_AE =  
    I.AnalystUserName  
WHERE  
    I.AnalystConfidenceRate > 7  
GROUP BY  
    C.Ticker_BI, C.Ticker_NumAnalysts  
ORDER BY  
    C.Ticker_NumAnalysts DESC  
LIMIT 1;
```

|   | Ticker_BI | Ticker_NumAnalysts | No_of_Analysts |
|---|-----------|--------------------|----------------|
| ▶ | AAPL      | 12177              | 52             |

(d) Who has the largest number of followers?

- ```
SELECT
    Ticker_BI
FROM
    company_details
WHERE
    Ticker_numFollowers =
    (SELECT MAX(Ticker_numFollowers) FROM
    company_details);
```

	Company	NumberofFollowers
▶	Apple Inc.	38,285

## 11. Regression Analysis.

```
In [62]: import pandas as pd
         from sklearn import linear_model

         # Assuming X and y are your features and target variable as Pandas DataFrames
         #subset_size = 10000 # Adjust this based on your available memory

         # Create subsets
         #X_subset = X_reset.iloc[:subset_size, :]
         #y_subset = y_reset.iloc[:subset_size]

         # Run the regression model on the subset
         lm = linear_model.LinearRegression()
         model = lm.fit(X, y)

In [63]: # This is the R2 score of our model. 95.3% of variance of the predictions have been explained by our model.
         lm.score(X,y)

Out[63]: 0.7972493243608243

In [64]: # Adding predicted EPS to our dataset.
         # As you can see our model is able to better predict as compared with Estimize Consensus and Wall Street Consensus overall.
         EPS_INFO["EPS_prediction"] = lm.predict(X)
         EPS_INFO
```

Linear Regression analysis has been done. The important independent variables are Ticker and Company which affects the independent variable EstimatedValue. So, given all the features constructed and scraped linear model has been fitted. And an accuracy of 79.7% is obtained. Thereby, finding the EPS\_prediction as shown below.

	quarter	Ticker	Company	Name	Type	EstimatedValue	EPS_prediction
0	fq1-2020	AMZN	Amazon.com Inc.	AMZN Reported Earnings	Generic	0.25	0.132080
1	fq1-2020	AMZN	Amazon.com Inc.	Estimize Consensus	Generic	0.33	-0.014404
2	fq1-2020	AMZN	Amazon.com Inc.	Estimize Mean	Generic	0.34	0.180908
3	fq1-2020	AMZN	Amazon.com Inc.	Wall Street Consensus	Generic	0.31	0.047119
4	fq1-2020	AMZN	Amazon.com Inc.	Flanner	Analyst	0.25	-0.360107
...	...	...	...	...	...	...	...
11856	fq4-2022	AMGN	Amgen Inc.	LA_Engineer	Analyst	4.34	4.471924
11857	fq4-2022	AMGN	Amgen Inc.	Mike Rietbrock	Analyst	4.35	4.384033
11858	fq4-2022	AMGN	Amgen Inc.	Alethia Young	Analyst	4.35	4.483643
11859	fq4-2022	AMGN	Amgen Inc.	Real2022	Analyst	4.35	4.083252
11860	fq4-2022	AMGN	Amgen Inc.	bknapp	Analyst	4.46	4.239502

11861 rows x 7 columns