**POTATO – Take Home Task**

I am using 50 MB small dataset for this task and below I am proving you a step-by-step solution for this assignment:

1st we have to install Docker to our system:

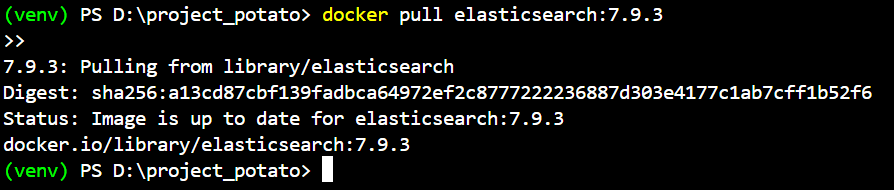
And I am using VS code for running the code in Python:

**Step 1:**

**Run the below command:**

docker pull elasticsearch:7.9.3

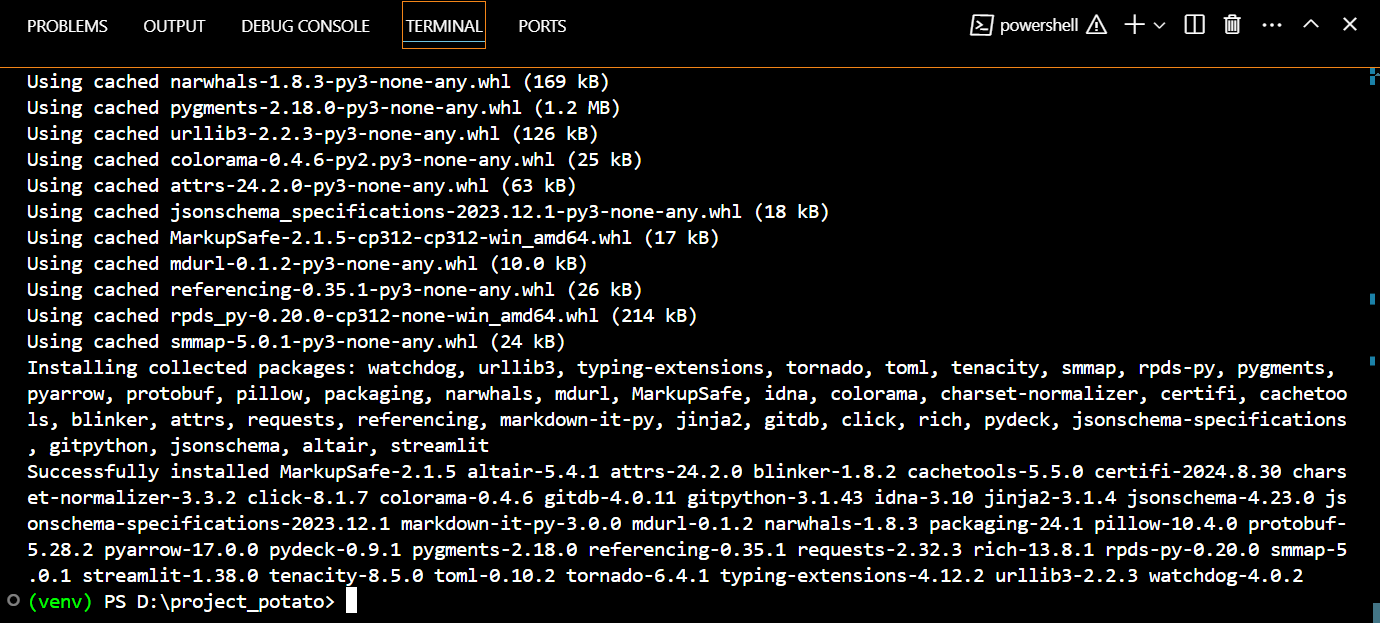
*Running docker pull elasticsearch:7.9.3 will download the Elasticsearch version 7.9.3 Docker image to your local machine, allowing you to run an Elasticsearch instance using Docker.*

**

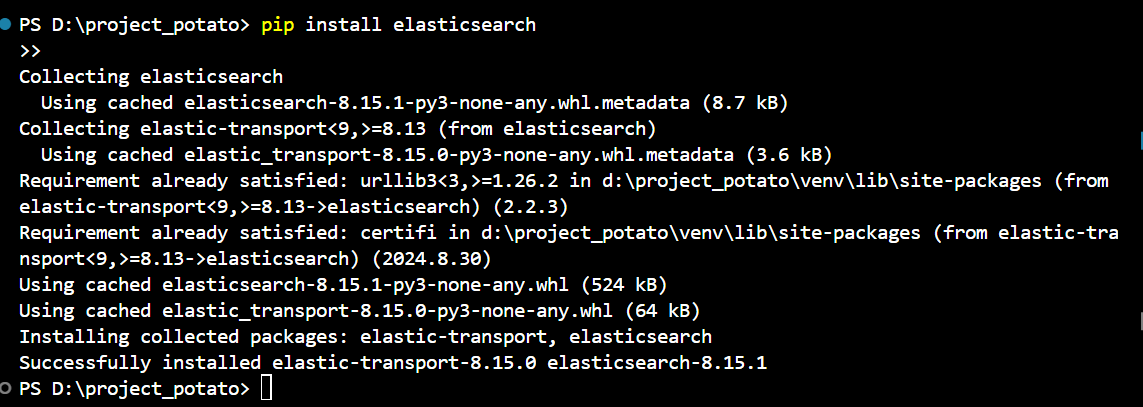
**Step 2:**

Install Streamlit and Elasticsearch

pip install streamlit & pip install elasticsearch



Streamlit is Successfully installed



Elasticsearch is successfully installed

**Creating a virtual Environment:**

**Run below commands in Python Terminal**

python -m venv venv

.\venv\Scripts\Activate

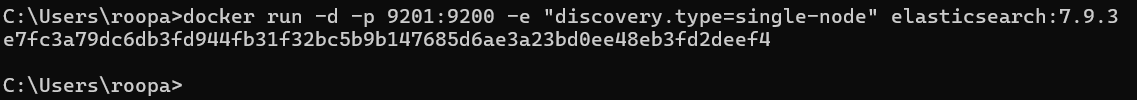
This command will create a virtual environment



**Step 3:**

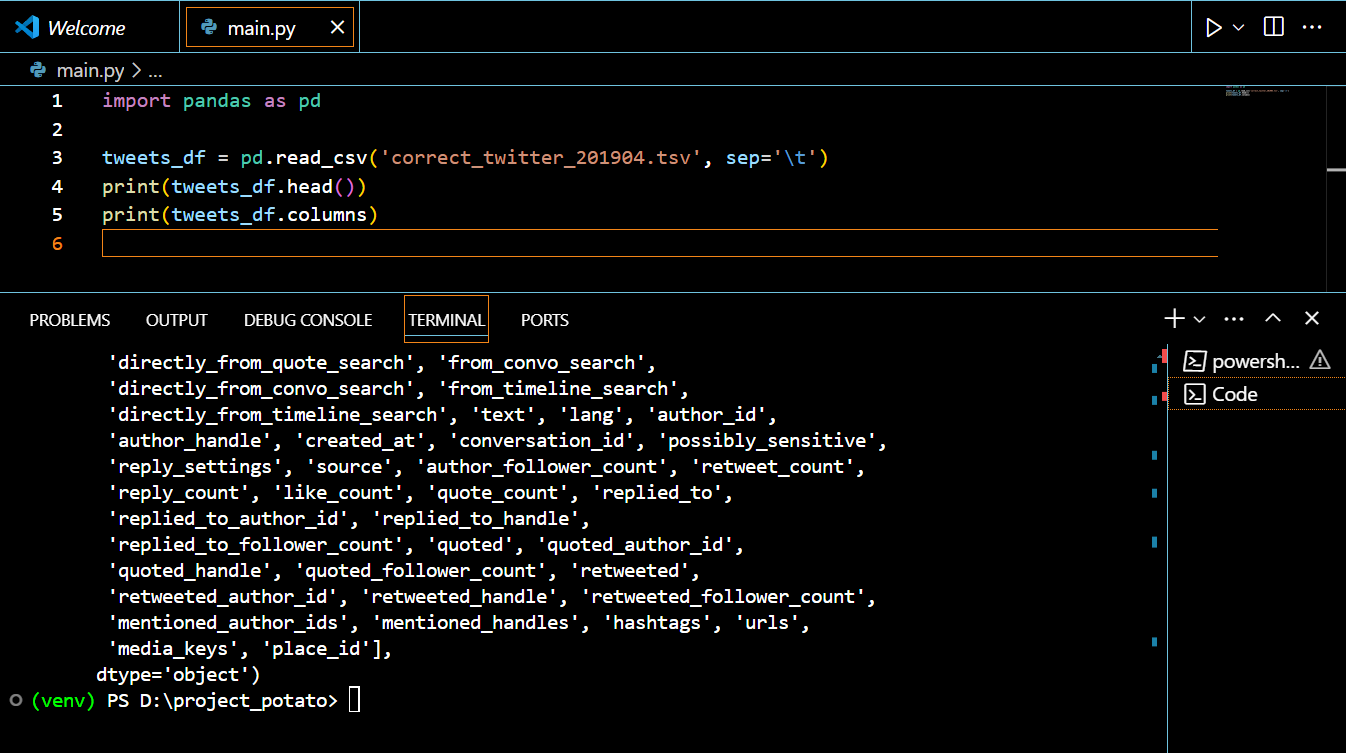
Now I have to setup Elasticsearch using Docker by using the below command:

docker run -d -p 9201:9200 -e "discovery.type=single-node" elasticsearch:7.9.3

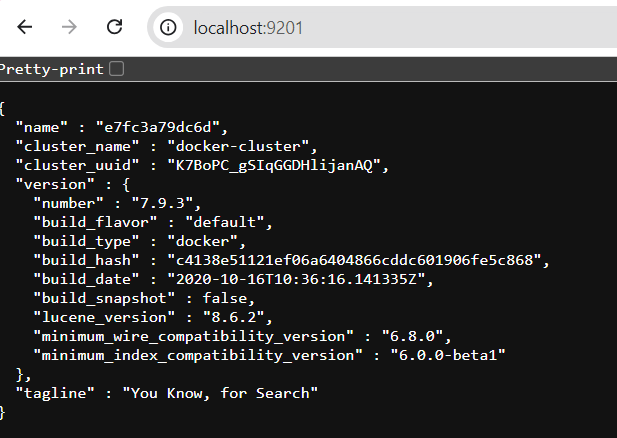


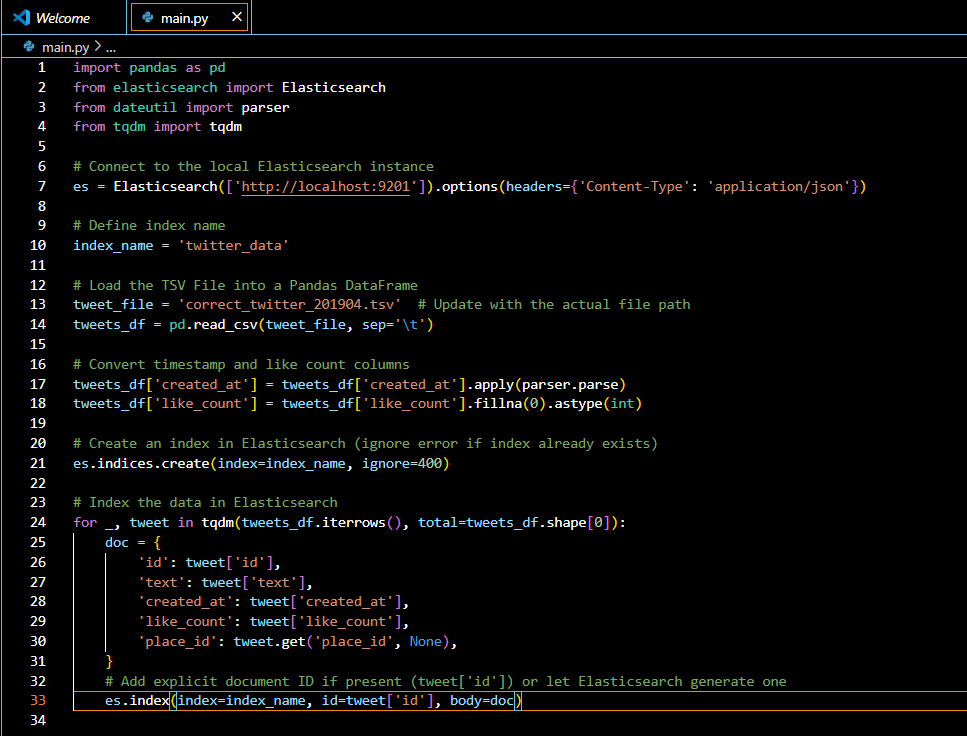
**Step 4:**

**Now Load the Dataset**

****

**The port is running properly:**

****

****

Library Imports: Imports Pandas for data handling, Elasticsearch for indexing, parser for date parsing, and tqdm for tracking progress.

Elasticsearch Connection: Establishes a connection to Elasticsearch with the correct content-type headers.

TSV File Loading: Loads the tweet data from a TSV file into a Pandas DataFrame for easier manipulation.

Data Preprocessing: Converts the created\_at column to datetime format and fills missing like\_count values with 0.

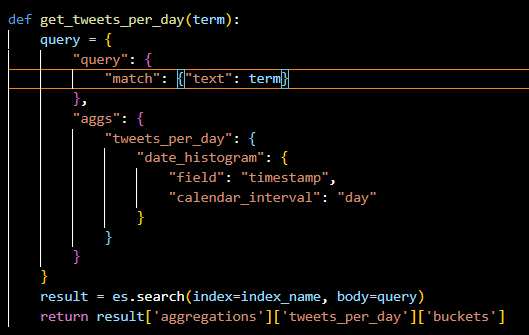
Index Creation: Creates an Elasticsearch index named twitter\_data, ignoring errors if the index already exists.

Data Iteration: Iterates over each tweet in the DataFrame, preparing a document for Elasticsearch.

Document Indexing: Indexes each tweet document into Elasticsearch, ensuring a unique id for each entry.

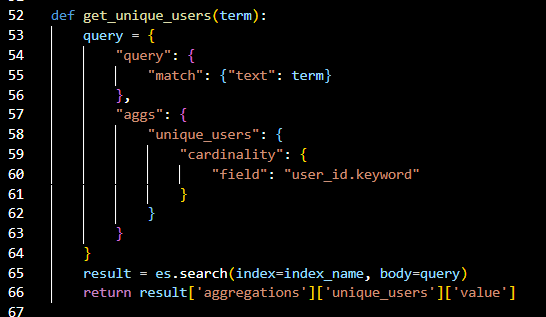
**Step 5: (PART 2)**

**Get the number of tweets per day containing the term**

****

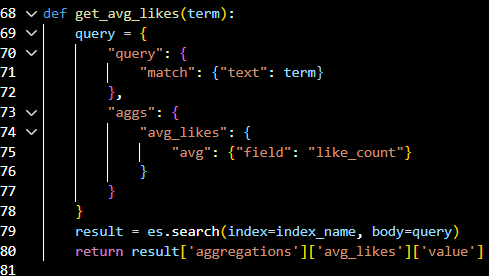
The get\_tweets\_per\_day function queries an Elasticsearch index for tweets that match a specific search term. It uses a date histogram aggregation to count the number of tweets per day based on the timestamp field, returning the results as buckets containing the tweet counts for each day.

**Get the number of unique users who posted a tweet containing the term**

****

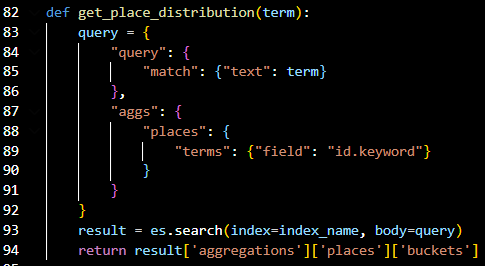
The get\_unique\_users function queries an Elasticsearch index to find tweets that contain a specified search term. It uses a cardinality aggregation on the user\_id.keyword field to count the number of unique users who posted those tweets, returning the total count as the result.

**Get the average number of likes for tweets containing the term**

****

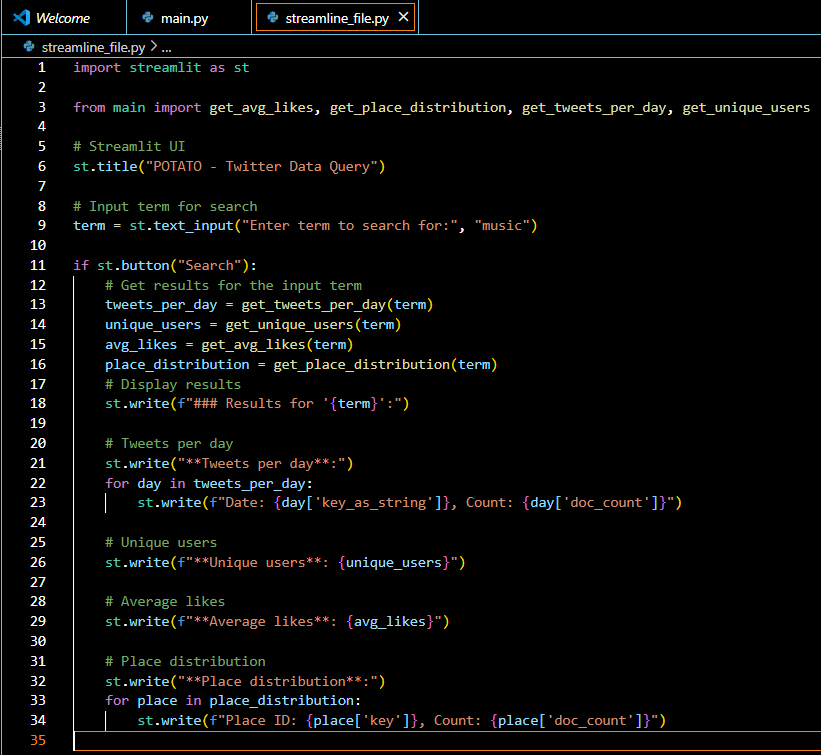
The get\_avg\_likes function queries an Elasticsearch index for tweets that include a specified term in the text field. It calculates the average number of likes received by these tweets using an average aggregation on the like\_count field, returning the computed average as the result.

**Get the distribution of tweets by place (id)**

****

The get\_place\_distribution function searches an Elasticsearch index for tweets containing a specified term in the text field. It then uses a terms aggregation on the id.keyword field to group and count tweets by their unique places (IDs), returning the distribution of these places as buckets, where each bucket represents a place and its corresponding tweet count.

**Now I created a Streamlit UI**

****

**This is the code for the Assignment**