Here's a breakdown of the code used in different files:

1. SQLAlchemy and Database Models:

The code uses SQLAlchemy to define a NewsArticle model, which corresponds to a news_articles table in a PostgreSQL database. Each article has fields such as id, title, content, publication_date, source_url, category, and created_at. The id is the primary key, and source_url is unique, ensuring no duplicate articles are stored.

2. Database Connection:

The create_engine function connects to the PostgreSQL database using the specified DATABASE_URL. Base.metadata.create_all(engine) creates the news_articles table in the database if it doesn't exist. SessionLocal is set up to create session objects for interacting with the database.

3. Functionality:

The insert_article function takes a database session and an article dictionary as input. It first checks for existing articles with the same source_url to avoid duplicates using SQLAlchemy's query functionality. If a duplicate is found, it logs the information and exits the function.

4. Classification:

If the article is new, it's classified using the classify_article function from the nlp module. A new NewsArticle instance is created with the article's details and added to the session. Finally, the session is committed to persist the changes to the database, and the insertion is logged.

5. Using spaCy:

The classify_article function uses the spaCy library to process the article's content. It loads a small English language model (en_core_web_sm) to analyze the text.

6. Classification Logic:

The function checks for keywords associated with each predefined category in the article's content. If specific keywords (like "terrorism," "happy," "earthquake," etc.) are found, it returns the corresponding category. If no keywords match, it classifies the article as "Others."

7. RSS Feed Fetching:

The code uses the feed parser library to parse a list of RSS feeds. The fetch_rss_feed function attempts to fetch articles from a given feed URL and returns a list of articles formatted as dictionaries, including title, content, publication date, and source URL.

8. Error Handling:

If an error occurs while fetching or parsing a feed, it logs the error and returns an empty list.

9. Fetching All Feeds:

The fetch_all_feeds function iterates over all defined RSS feeds, logs the fetching process, and aggregates all articles into one list.

10. Main Execution:

When the script is run directly, it fetches all articles and inserts them into the database through the insert_article function. A scheduled job is defined to fetch and insert articles every hour using the schedule library.

11. Continuous Execution:

The script runs indefinitely, checking for any scheduled jobs to execute, which allows for regular updates to the database with the latest articles.

To Run the files:

First Step up PSQL by creating a database named news_db and add the username and password of your server in the models.py

Then run models.py by using **python models.py**

Then run rss_feed.py by using **python rss_feed.py**

You can check the data stored in the DATA.csv as it generates a CSV file too