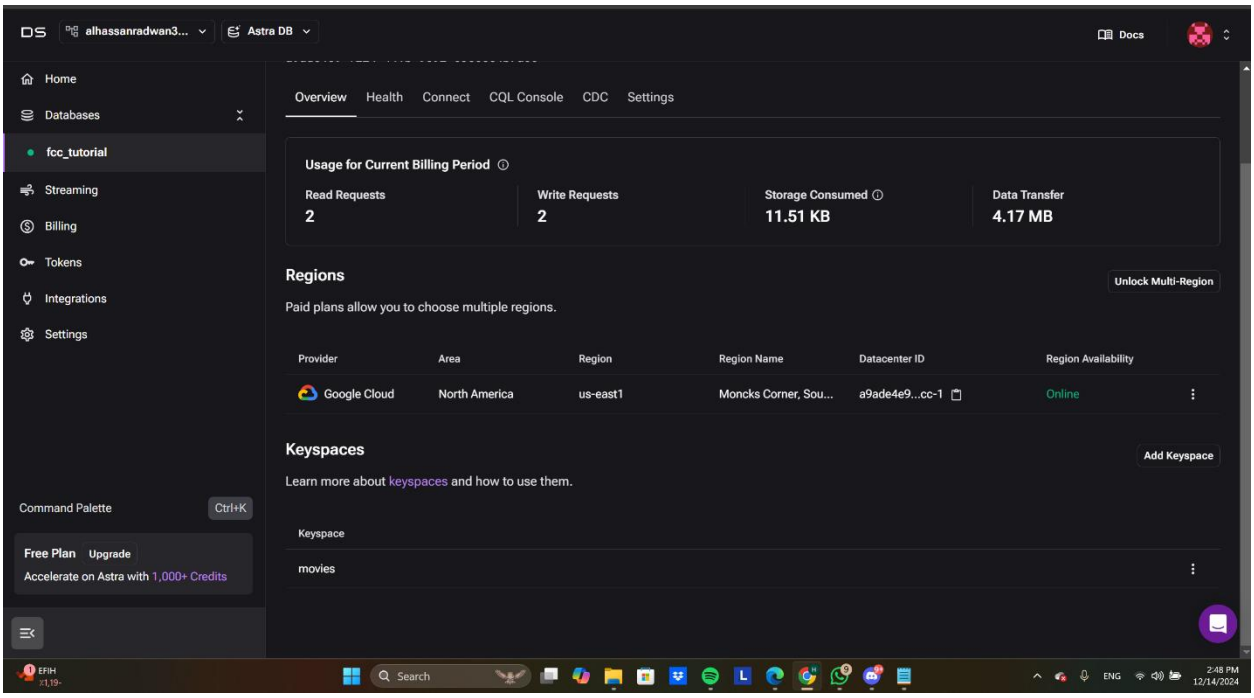
Advanced Database
Assignment 2

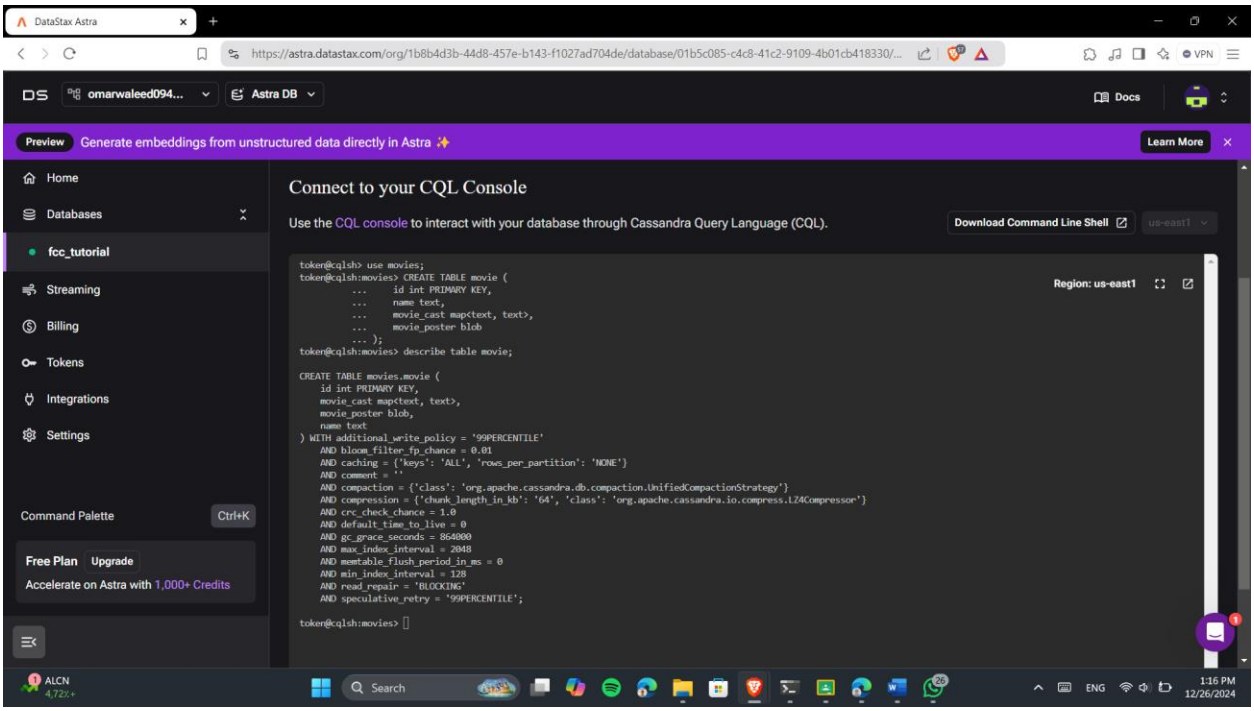
Team:

Omar Waleed	20227019
Al Hassan Ahmed	20227041
Wael Hossam	20227043
Eslam Khaled	20226013
Susana Ayman	20227015
Nardine Naguib	20227027

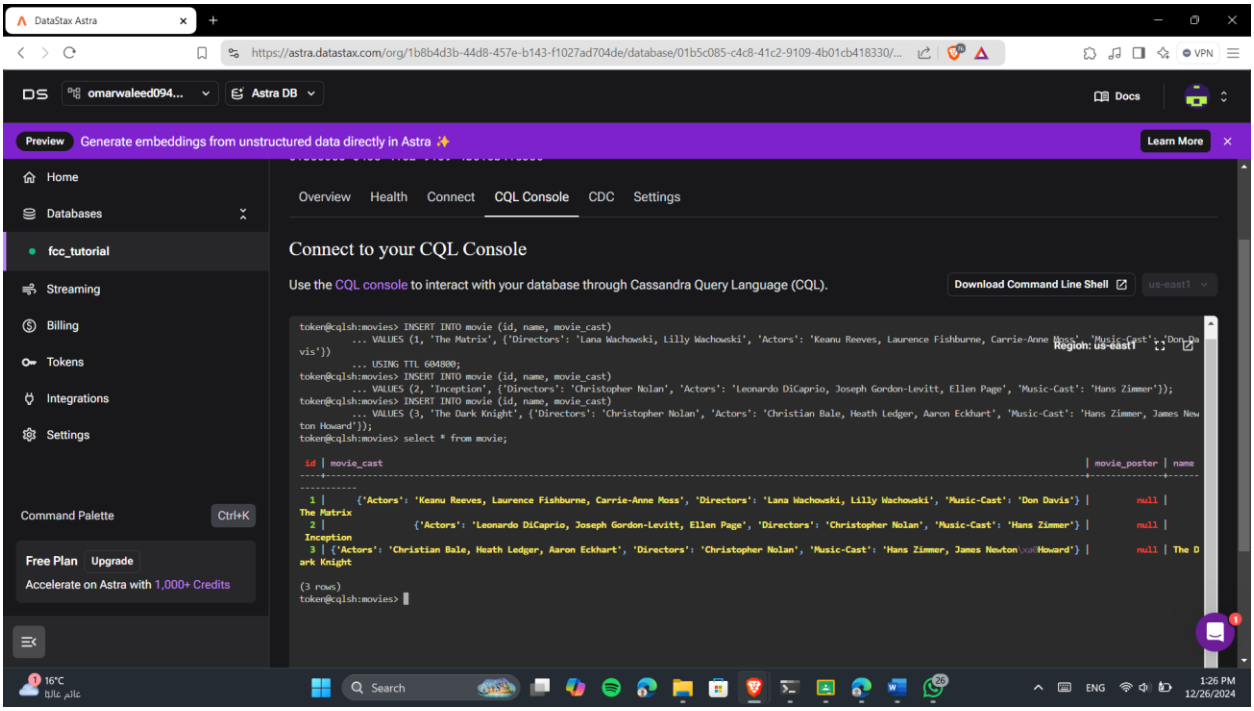
1) Creating a keyspace named “movies”:



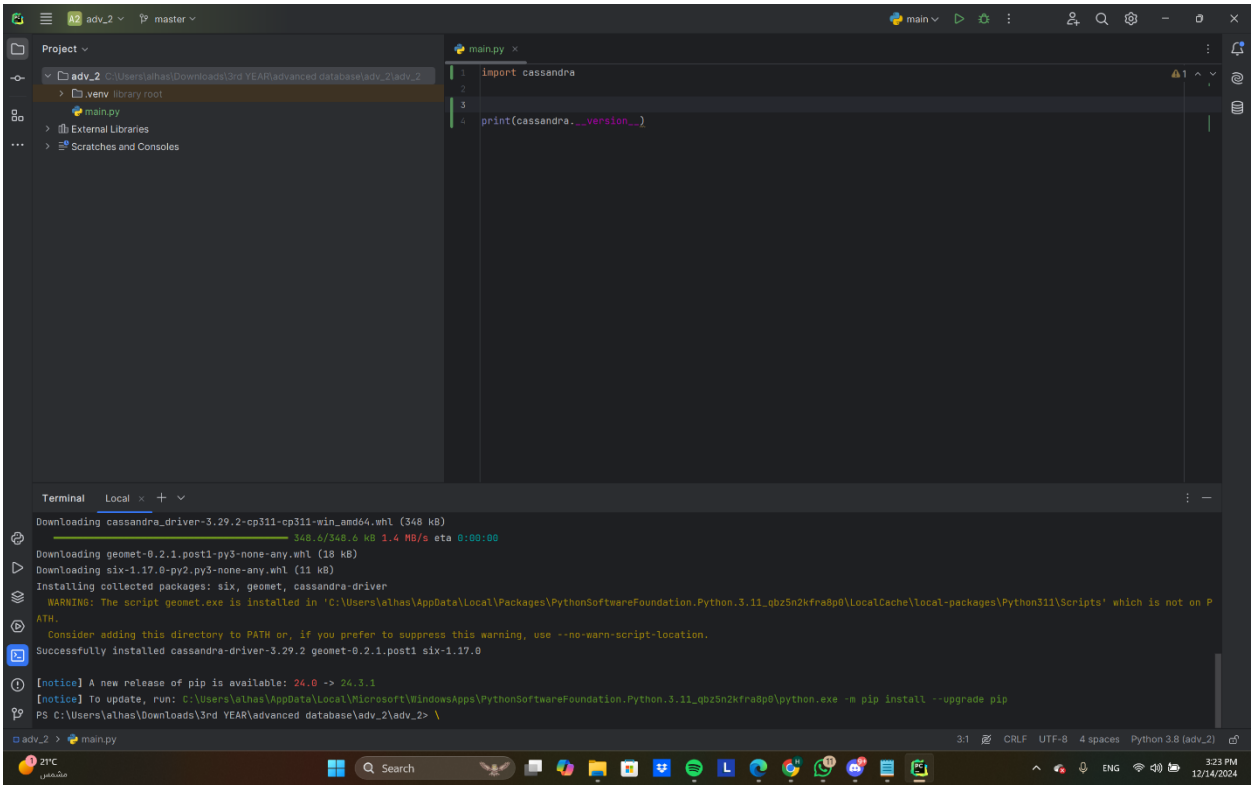
2) Creating a column-family “movie” with columns (Id int, name text, movie-cast map, movie-poster blob). The cast column is a map with (director(s) – actors(s) – music-cast-person(s)).
3) Check the schema of the “movie” column-family.

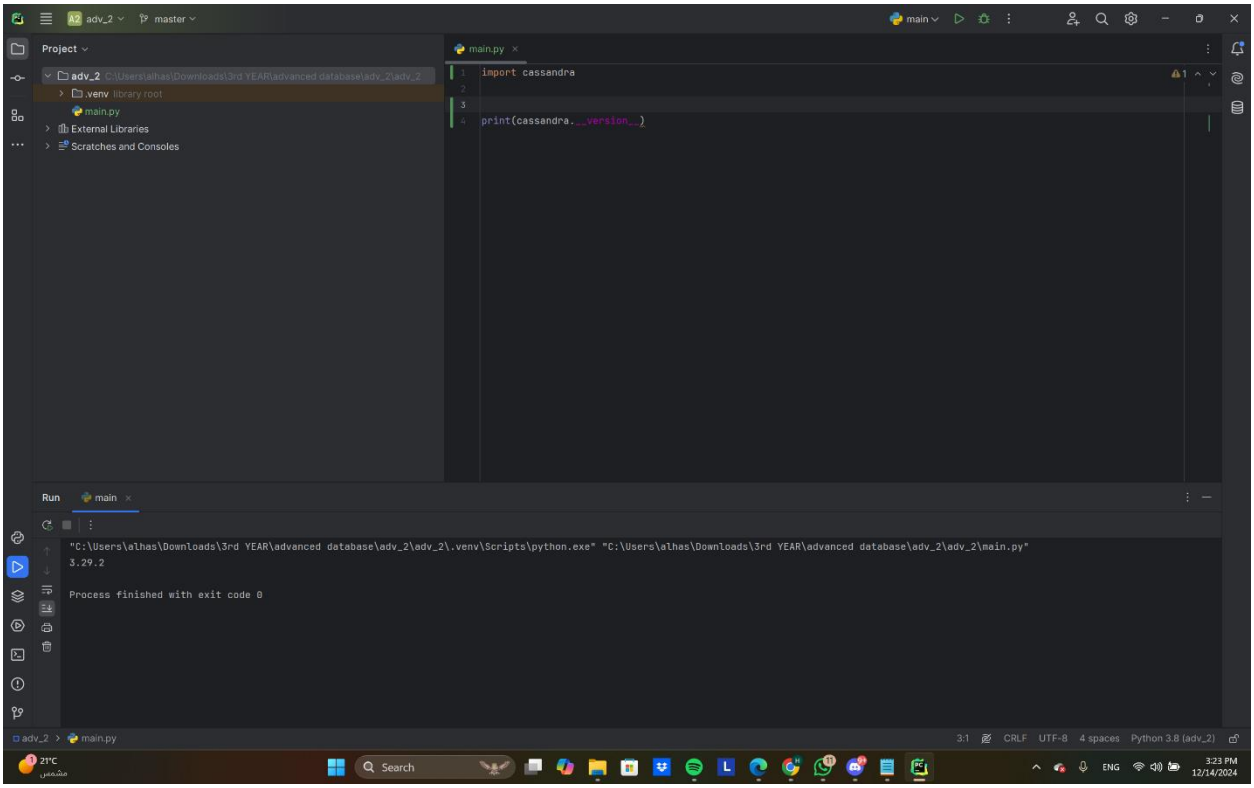


4) Populate the movie table with 3 real movies without populating the movie poster:

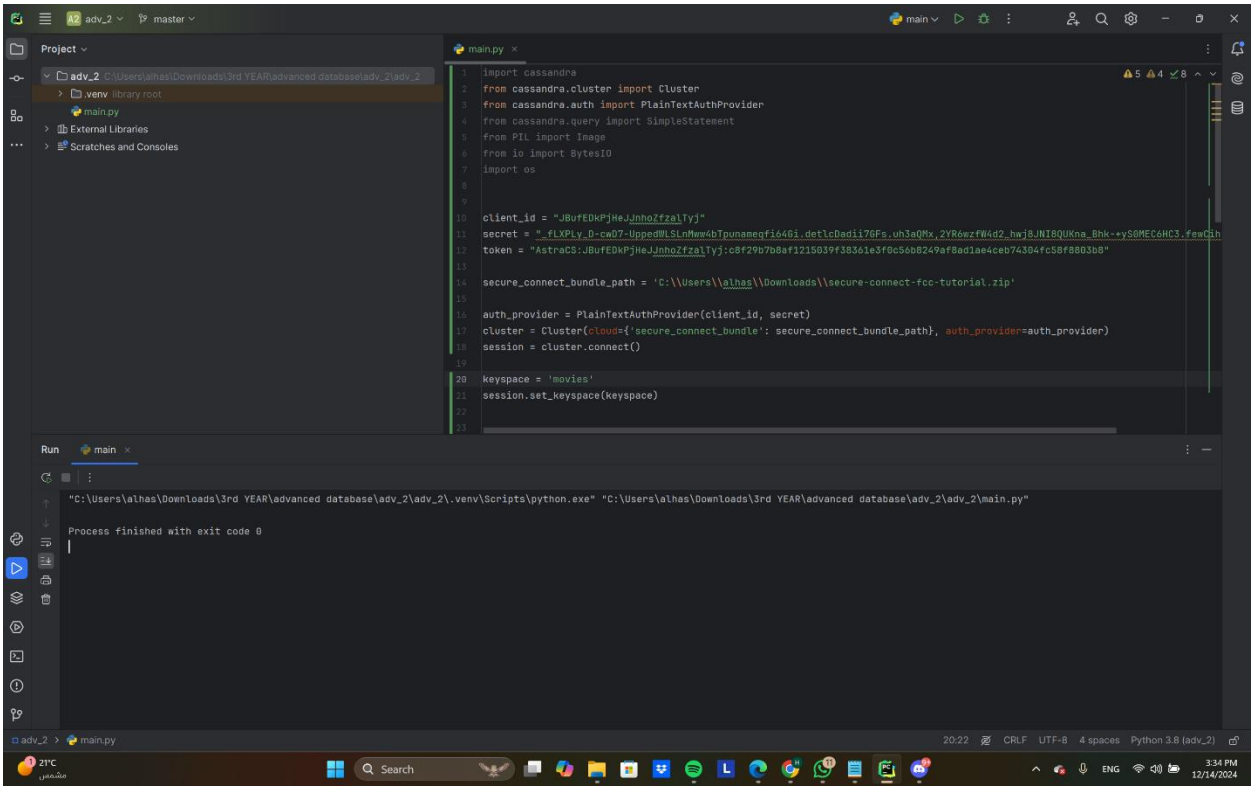


5) Writing python function to transform the movie poster image to from IMDB to blob datatype and update the movie-poster column in the three rows with the blob datatype:
   - First connecting to our keyspace:



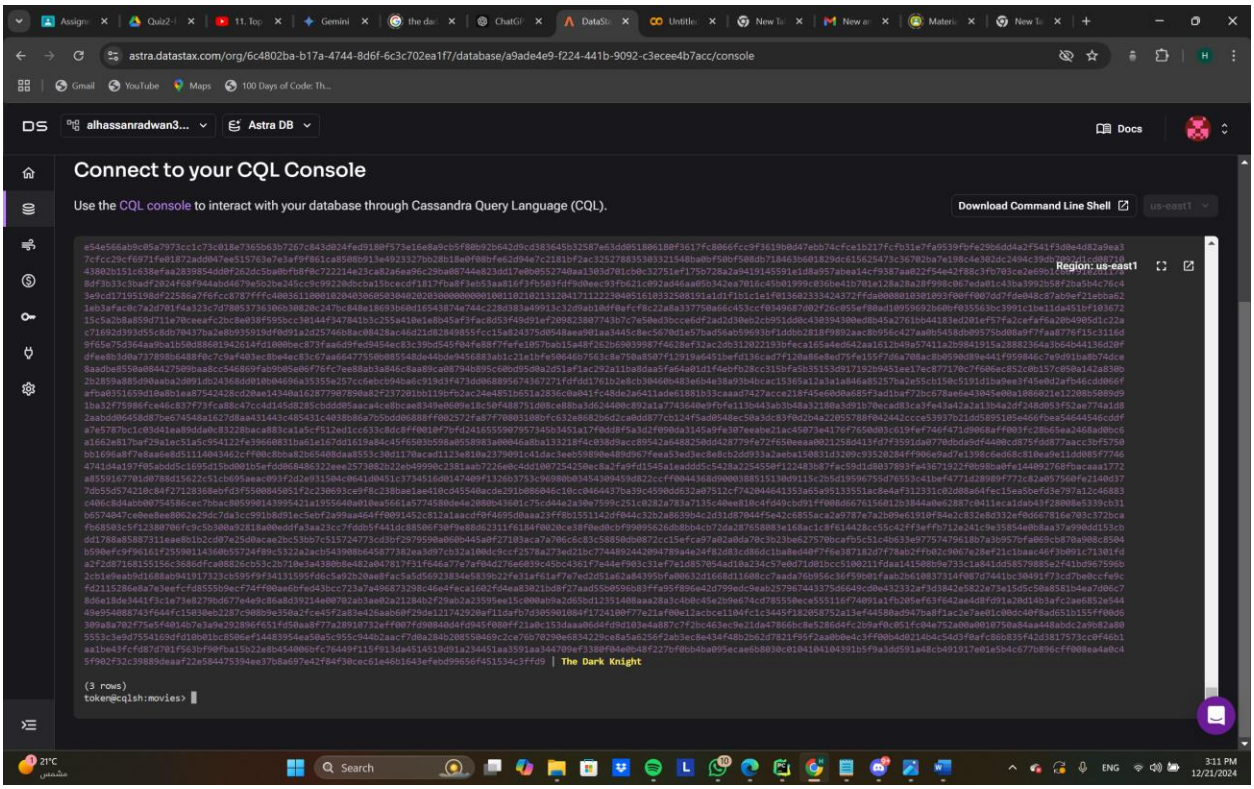


- The function “image_to_blob()” that converting image to blob datatype and then populate the movie-poster column “update_movie_poster()” :




```python
# Function to convert an image to a blob
1 usage  new *
def image_to_blob(image_path):
    with open(image_path, 'rb') as file:
        return file.read()

# Function to update the movie poster
1 usage  new *
def update_movie_poster(session, movie_id, image_path):
    try:
        poster_blob = image_to_blob(image_path)
        query = "UPDATE movie SET movie_poster = %s WHERE id = %s"
        session.execute(query, (poster_blob, movie_id))
        print(f"Poster for movie ID {movie_id} has been updated.")
    except Exception as e:
        print(f"Failed to update poster for movie ID {movie_id}: {e}")
```

- Making sure that the functions works correctly :



6) Writing python function “query_movies_by_person()” to query the movies given certain director or actor and display the row results including the image

- Key Difference:

• LIKE works for partial string matching, useful when you store a single string (like the director's name).

• CONTAINS is more efficient for checking if a collection (like a list of actors) contains a specific element.

-**Conclusion**:

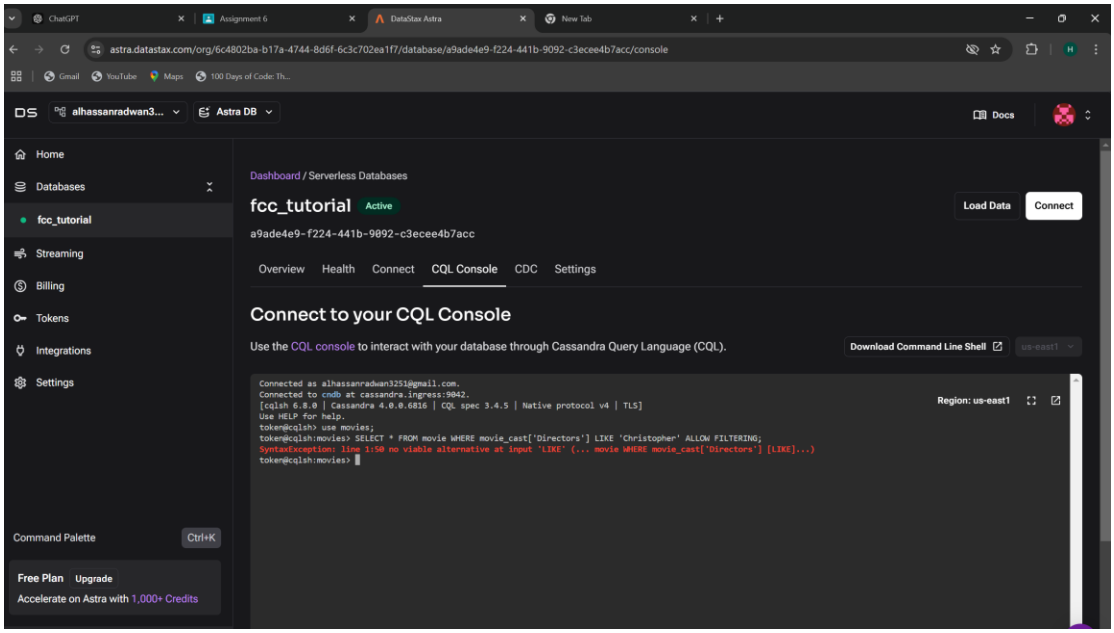
• Use LIKE when searching within a text field where you're looking for a part of a string (e.g., finding all movies with "DiCaprio" in the actors list).

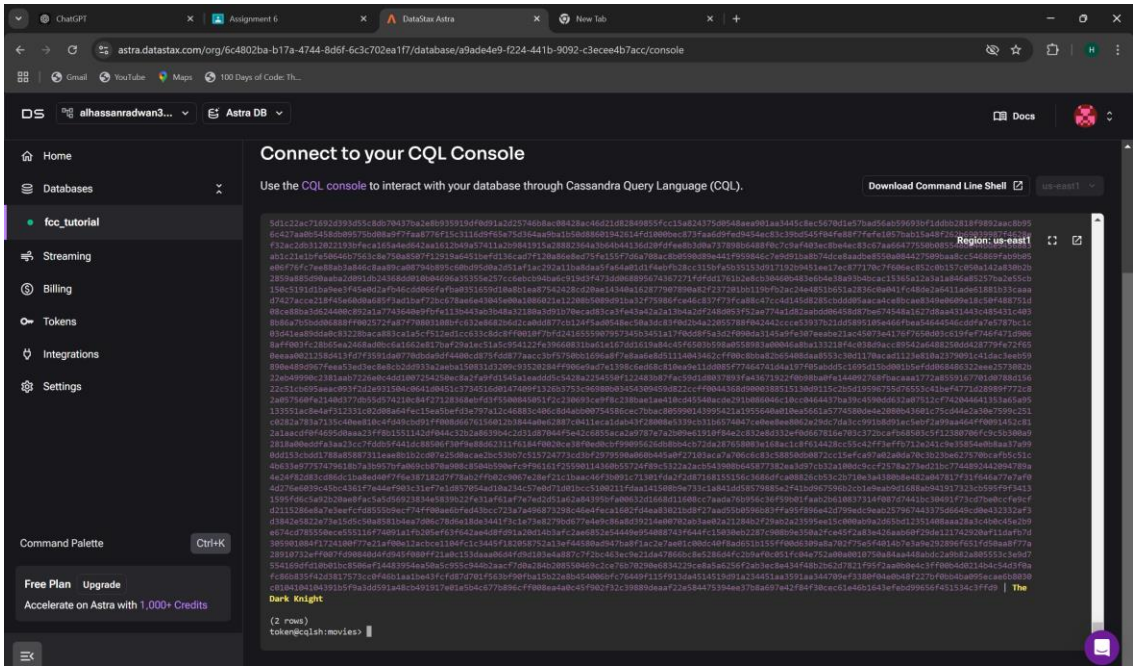
• Use CONTAINS when querying against a collection field (e.g., a list of actors) to check if a specific item exists in the list

LIKE: Works for text columns (string pattern matching).
CONTAINS: Works for collections (checks if a value exists).

- **LIKE in CQL:**
  Cassandra does not natively support the LIKE operator for wildcard searches (e.g., % or _) due to its distributed architecture.
  However, you can use LIKE with the ALLOW FILTERING clause if your query is on an indexed column. This is not recommended for production but works for demonstration.

- **Contain in CQL:**
  SELECT * FROM movie WHERE movie_cast CONTAINS 'Christopher Nolan' ALLOW FILTERING;
  This is working

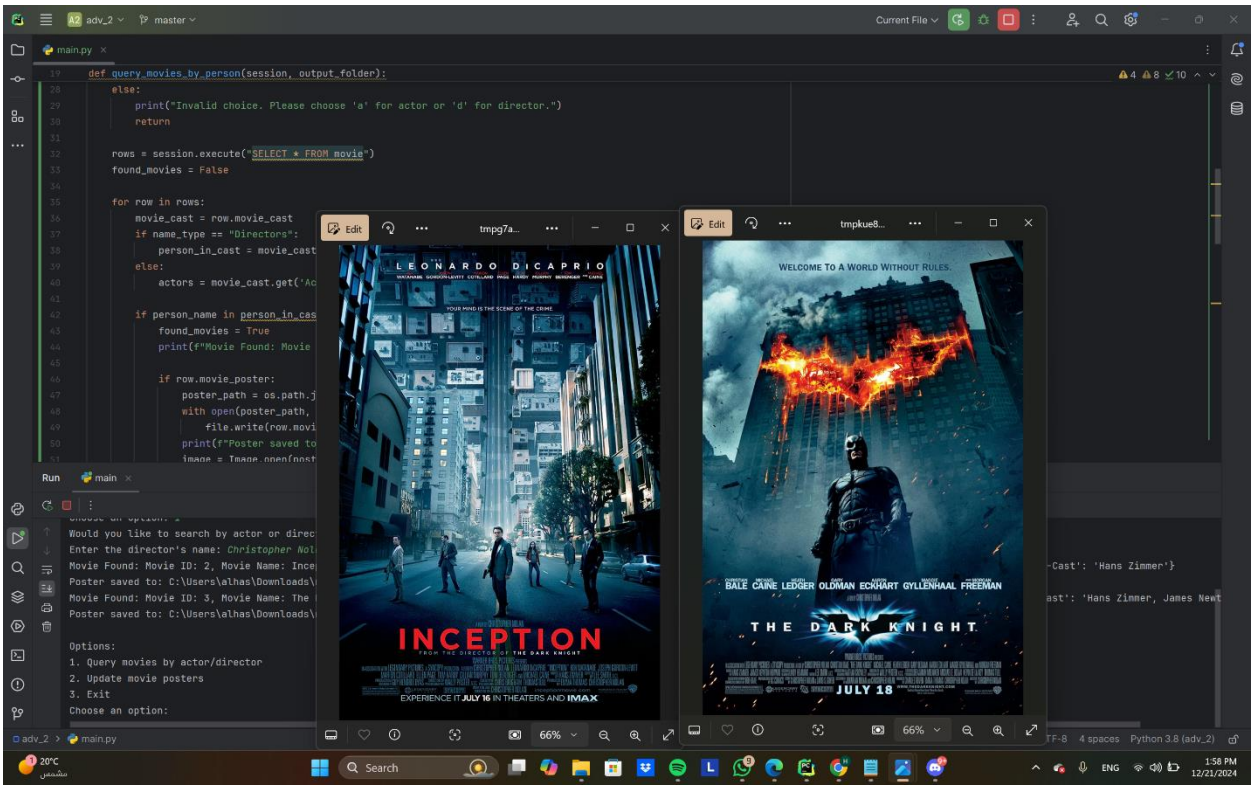




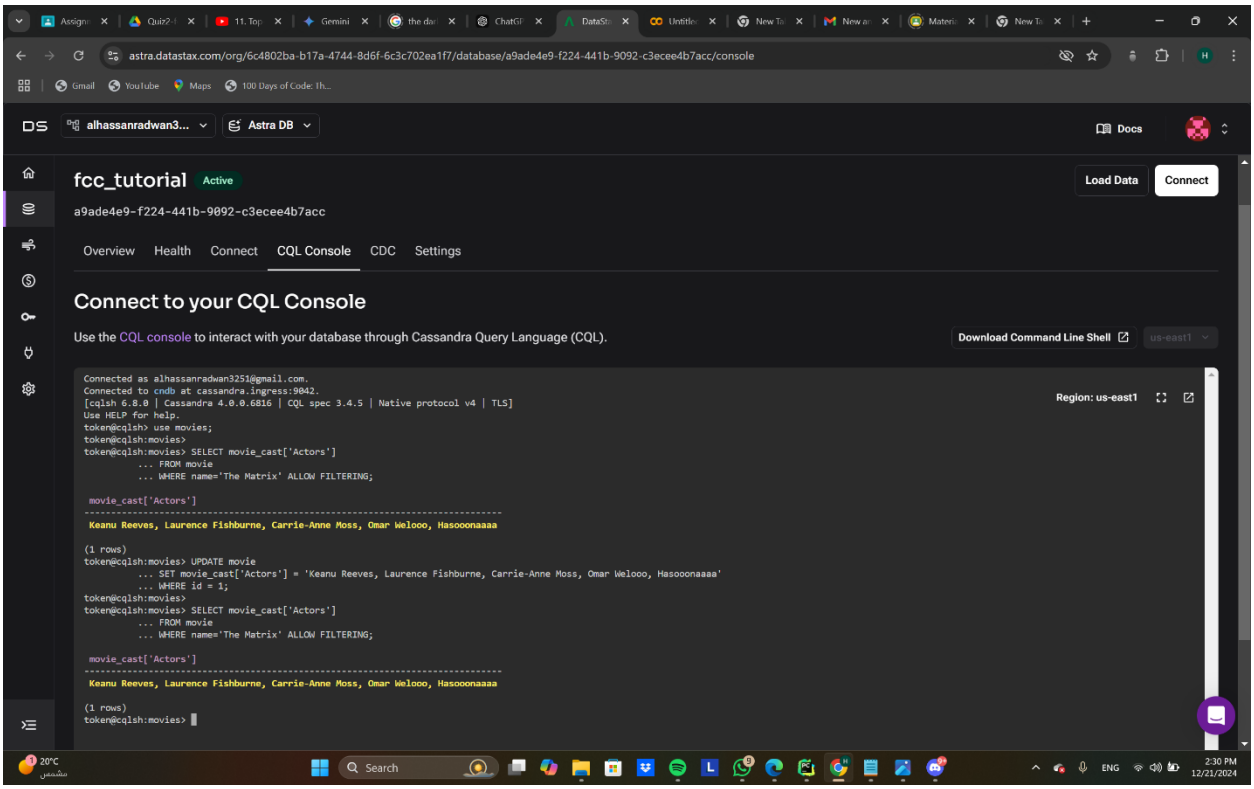
```python
def query_movies_by_person(session, output_folder):
    else:
        print("Invalid choice. Please choose 'a' for actor or 'd' for director.")
        return

    rows = session.execute("SELECT * FROM movie")
    found_movies = False

    for row in rows:
        movie_cast = row.movie_cast
        if name_type == "Directors":
            person_in_cast = movie_cast.get('Directors', '')
        else:
            actors = movie_cast.get('Actors', '')

        if person_name in person_in_cast:
            found_movies = True
            print(f"Movie Found: Movie ID: {row.id}, Movie Name: {row.name}, Movie Cast: {row.movie_cast}")

            if row.movie_poster:
                poster_path = os.path.join(output_folder, f"{row.name.replace(' ', '_')}.jpg")
                with open(poster_path, "wb") as file:
                    file.write(row.movie_poster)
                print(f"Poster saved to: {poster_path}")
                image = Image.open(poster_path)
```

```
Would you like to search by actor or director? Choose (a) for actor, (d) for director: d
Enter the director's name: Christopher Nolan
Movie Found: Movie ID: 2, Movie Name: Inception, Movie Cast: {'Actors': 'Leonardo DiCaprio, Joseph Gordon-Levitt, Ellen Page', 'Directors': 'Christopher Nolan', 'Music-Cast': 'Hans Zimmer'}
Poster saved to: C:\Users\alhas\Downloads\movie_posters\Inception.jpg
Movie Found: Movie ID: 3, Movie Name: The Dark Knight, Movie Cast: {'Actors': 'Christian Bale, Heath Ledger, Aaron Eckhart', 'Directors': 'Christopher Nolan', 'Music-Cast': 'Hans Zimmer, James Newt
Poster saved to: C:\Users\alhas\Downloads\movie_posters\The_Dark_Knight.jpg

Options:
1. Query movies by actor/director
2. Update movie posters
3. Exit
Choose an option:
```

```python
def query_movies_by_person(session, output_folder):
    search_by = input("Would you like to search by actor or director? Choose (a) for actor, (d) for director: ").strip().lower()

    if search_by == 'd':
        name_type = "Directors"
        person_name = input("Enter the director's name: ")
    elif search_by == 'a':
        name_type = "Actors"
        person_name = input("Enter the actor's name: ")
    else:
        print("Invalid choice. Please choose 'a' for actor or 'd' for director.")
        return

    rows = session.execute("SELECT * FROM movie")
    found_movies = False

    for row in rows:
        movie_cast = row.movie_cast
        if name_type == "Directors":
            person_in_cast = movie_cast.get('Directors', '')
        else:
            person_in_cast = movie_cast.get('Actors', '')


        if person_name in person_in_cast:
            found_movies = True
            print(f"Movie Found: Movie ID: {row.id}, Movie Name: {row.name}, Movie Cast: {row.movie_cast}")

            if row.movie_poster:
                poster_path = os.path.join(output_folder, f"{row.name.replace(' ', '_')}.jpg")
                with open(poster_path, "wb") as file:
                    file.write(row.movie_poster)
                print(f"Poster saved to: {poster_path}")
                image = Image.open(poster_path)
                image.show()

    if not found_movies:
        print(f"No movies found for {name_type} '{person_name}'.")
```



7) Update a certain movie actors list to add another actor to the list:



8) Updating the first row TTL to 3 seconds and after that re-querying the table:
   - We set TTL to 15 then we checked the first row, We waited 15 seconds, and re-queried to confirm if the result remained or was deleted: