Section 1
Data QA Assessment

https://docs.google.com/spreadsheets/d/1Hkj2mjP0ENsPXAmro8gWH3XCK2jrQ2G57esHINEzuhc/edit?usp=sharing

Some scenarios to manually test the ETL flow			
Scenario	**Sample Data**	**Expected Result**	**Automatable**
Test the successful loading of the data from the csv file into the **dim_country** table	sample csv data	All the data from the csv file is successfully loaded into the dim_stores table with correct country codes and names	Y
Test the successful loading of the data from the csv file into the **dim_stores** table	sample csv data	All the data from the csv file is successfully loaded into the dim_stores table with no errors.	Y
Test the lookup functionality for the **country_code** in the **dim_country** table	sample csv data	Correct country is retrived for each row form the csv file	Y
Test the lookup functionality of the **country_code** for invalid country name	Country name = "Mars" (an invalid country name)	Should not find corresponding country code and should not insert the store data into the dim_stores table.	Y
Test the case sensitivity of country lookup	different cases (eg. India, INDIA, india, inDIA)	The ETL job should correctly map each country name to the corresponding country code in the dim_country table, regardless of the case	Y
Test the ETL's handling of a missing country in the **dim_country** table	Add new row with inalid country code that is not present in dim_country table	The ETL job should fail and return an error message	Y
Test the ETL's handling of a incorrect country in the **dim_country** table	country_name = nepal	The ETL job should fail and return an error message	Y
Test **store_id** generation	CSV file containing a list of stores without a store_id	ETL should automatically generate a unique store_id to dim_stores table.	Y
Test the geographical hierarchy lookup	dim_lookup_geog	Verify that the ETL job is correctly looking up the geographical hierarchy for each row in the dim_stores table by checking that the correct lvl1_geog, lvl2_geog, and lvl3_geog values are inserted for	

		each row.	

Github link to assignment: https://github.com/Rononoa13/eyos-assessment

Application running instruction:

1. First (**uncomment create_table()** function line 51 from **database_3.py**) and run **database_3.py** file
2. Comment create **create_table()** function line 51) and run **application_3.py**

prove pseudo-code for the code.

1. Connect to database
2. Create all required tables.
3. Read the CSV file.
4. For row in “sample_csv_data”
   Lookup country code from dim_country table
   Insert row into dim_stores table with country_code
5. Check expected result against actual result

Open Question:

What are the possible data quality issues that could arise with the current ETL logic.	
Invalid or missing country_code	The ETL job relies on the country_code column to look up the corresponding country_name from the dim_country table. If the country_code value is missing or invalid, the ETL job will fail to insert the row into the dim_stores table, resulting in data loss or incomplete data.
Inconsistent data in csv file	Missing or incorrect values in store_name, street_name, pin_code
Case sensitivity issues:	If the ETL does not perform case-insensitive lookups, it may result in missing or incorrect data being inserted into the

	dim_stores table.
Performance Issues	If the volume of data in the CSV file is large or if the lookups and transformations required in the ETL process are complex, it could result in longer process time or even failure.
Inconsistent or missing data in the dim_lookup_geog table:	
Inconsistent or missing data in the source csv file	

Section 2
Functional QA Assessment

API Link:
Weather: https://rapidapi.com/weatherbit/api/weather
Google Translate: https://rapidapi.com/googlecloud/api/google-translate1/

Task 1:

Task 1
Positive Test Cases
API returns expected response: Verify the request to the API with valid input parameters gives the expected response
API correct status code: Verify that the API return correct status code for different request (200 - successfull response , 201 - successfully created.)
API returns appropriate error messages: Verify that API returns appropriate error messages when invalid input parameters are provided.
API returns valid data types: Verify that the API returns valid data types in the response such as integer, string, boolean, etc.

Negative Test Cases
Invalid Input Parameters: Send a request to the API with invalid input parameters and verify that the API returns the appropriate error message.
Incorrect Data Types: Send a request to the API with input parameters that are of incorrect data types and verify that the API returns the appropriate error message.
Missing Input Parameters: Send a request to the API with missing input parameters and verify that the API returns the appropriate error message.
Wrong Request: Send a request to the API with wront request (GET request instead of POST, DELETE instead of PUT)
Security Testing: Try to bypass authentication and authorization, verify that unauthorized access to API is denied and appropriate error message is returned.

Request Specific	**GET**
Verify that the API returns the expected 3-hourly weather forecast data for the requested location.	3Hourly
Verify that the API displays correct location for proficed longitude and latitude	
Verify that the API displays correct weather data for current location	Current Weather Data of a Location
Verify API returns without parameters	
Verify API returns with parameters (api_key , units=optional)	
Verify API returns a 16 day (daily) forcast	16 Day Forecast
Verify API returns a forecast for up to 120 hours in the future (default 48 hours)	120 Hour Forecast
Verify API returns weather alerts from local meteorological agencies (US, EU, Israel)	Severe Weather Alerts

Task 2

The written test case into automated tests using python and pytest. More on link
Github link to assignment: https://github.com/Rononoa13/eyos-assessment

Task 3

HTML Reports is used to generate the log.

```
pytest --html=report.html  pytest --html=report.html <filename>
```