ANALYTICS ENGINEERING ASSIGNMENT

TASK 1

Parquet File Conversion to CSV

- Download the `.parquet` file and save it locally.

AWS Free Tier Account Setup

- Create an Amazon Web Services (AWS) account if you don't already have one.
- Use an IAM user account instead of the root user account for security best practices.
- Assign appropriate **permissions** and **roles** to the IAM user.
- Log in using the IAM user credentials.

S3 Bucket Creation

- Navigate to the S3 service in the AWS Management Console.
- Create two S3 buckets within the **same AWS region**.
- Ensure both bucket names follow AWS's global uniqueness and naming conventions.

Lambda Function Creation

- Search for **Lambda** in the AWS Console and create a new Lambda function.
- Select the **Author from scratch** option to create the function.
- Provide a meaningful name for the Lambda function.

- Choose the runtime environment (e.g., **Python 3.13**) and architecture (e.g., **x86_64**).
- Under permissions, select a role with **basic Lambda execution permissions**.
- Create the function.

Lambda Function Development

- Develop a script that:
  - Triggers when a `.parquet` file is uploaded to the source bucket.
  - Converts the `.parquet` file to `.csv`.
  - Uploads the resulting `.csv` file to the destination bucket.
- Add a trigger for the Lambda function:
  - Choose **S3** as the trigger source.
  - Select the source bucket (e.g., `source-bucket-analytics-v1`).
  - Set the event type to **All object create events**.
  - Add a suffix filter for `.parquet` files.
- Specify the destination bucket (e.g., `processed-bucket-analytics-v1`) and provide its ARN.
- Add necessary Lambda layers, such as **pandas**, to handle data processing.
- Attach an IAM policy to the Lambda execution role granting required permissions to access both S3 buckets.

Testing and Deployment

- Upload a `.parquet` file to the source bucket (`source-bucket-analytics-v1`).
- Test the Lambda function to verify it converts the file correctly.
- Deploy the Lambda function.
- Monitor CloudWatch metrics:
  - Confirm **Invocations** count reflects the test.
  - Verify **Error count** is zero.

    - Check **Success rate (%)** is 100% (green status).
- Review CloudWatch logs for detailed execution information, ensuring no errors occurred.
- Verify that the converted `.csv` file appears in the destination bucket (`processed-bucket-analytics-v1`).
- Download the converted file for further use.

TASK 2

TXT File Conversion

- Download the `.txt` file and save it locally.
- Choose and install an IDE of your preference; in this case, **PyCharm** is used.
- Navigate to the directory containing the `.txt` file using basic Linux commands such as `cd`, `ls`, etc.
- Create a new Python script named `txt_file_conversion.py` and save it in your working directory.
- Open the `.txt` file to review its structure and identify the delimiter used (e.g., comma, tab, pipe).
- Install the required Python libraries, primarily **pandas** and **os**, if not already installed.
- Import the necessary libraries in your script.
- Read the `.txt` file using `pandas.read_csv()`, specifying the correct delimiter.
- Convert the data to `.csv` format and save it to your desired location.
- Implement error handling using `try` and `except` blocks to catch and display any errors during the conversion process.
- Verify that the converted `.csv` file is created successfully in the specified location.

TASK 3

DBeaver Schema and Table Creation

- Download and install **DBeaver** from https://dbeaver.io/download.
- Ensure **PostgreSQL** is installed on your machine. If not, download and install the latest stable version compatible with your OS from https://www.postgresql.org/download/.
- Launch DBeaver, then add a new connection by selecting **PostgreSQL**.
- Configure the connection using your PostgreSQL instance details: Host, Port, Username, Password, and Database name.
- Test the connection; if successful, you will receive a confirmation message.
- Create a new database using the SQL DDL command:

```
CREATE DATABASE your_database_name;
```

- Create a schema within the database as needed.
- Import CSV files into your database by right-clicking on the target **Tables** node and selecting **Import Data**.
- Verify successful ingestion by inspecting the table structure (columns) and running a simple query such as:

```
SELECT * FROM table_name LIMIT 10;
```

Data Cleaning

- Perform a quick Exploratory Data Analysis (EDA) to ensure data quality:
  1. Verify that the unique identifier column contains no duplicates.

2. Identify any NULL or missing values and determine an appropriate strategy to handle them.
3. Standardize categorical data to maintain consistency (e.g., unify representations such as “M” and “Male” to a single standard value).
4. Confirm that each column’s data type is appropriate for its contents.
5. Assess overall data quality and integrity.
6. Apply necessary changes using SQL DDL commands such as `ALTER TABLE` to clean and adjust the schema or data as required.

TASK 4

Data Integration via SQL

- Perform SQL joins on tables using common columns to create a unified, consolidated dataset.
- Use the integrated data to answer the required business questions; the corresponding SQL queries are provided in a separate document.
- For question 3, the justification for using **average imputation** to fill null values is that the question specifically focuses on average metrics.

  Replacing nulls with the average preserves the integrity of the analysis and yields a more accurate and meaningful result compared to using zeros or removing records entirely.

TASK 5

Dashboard Creation

- Export the unified dataset into a BI tool of your choice—**Power BI** or **Amazon QuickSight**.
- In this case, **Power BI** was used. If not already installed, download it from:

  https://www.microsoft.com/en-us/power-platform/products/power-bi/downloads

- Develop a comprehensive dashboard or report by applying appropriate visualizations that best communicate the data insights.
- Choose chart types based on the nature of the data (e.g., bar charts for comparisons, line charts for trends, pie/donut charts for proportions, etc.).
- Ensure the dashboard is clean, interactive, and provides a clear narrative for decision-makers.

DBT ASSESMENT

- Install Docker if not already installed: Windows download link
- Install Git if not already installed: macOS download link
- Download or clone the project files to your local machine
- Launch the Docker daemon and ensure it is running
- Open the project folder using Visual Studio Code
- Run `docker build -t dbt_project .` to build the Docker image
- Run `docker run -it -v $(pwd):/app dbt_project` to start the container and mount the project directory
- Run `dbt --version` to verify dbt is installed and working
- Run `dbt debug` to confirm the `profiles.yml` is correctly configured and the database connection works
- Run `dbt seed` to load the `.csv` seed files into the database
- Navigate to the `/models/dim/` folder and edit `dim_patients.sql` to define the patient dimension model
- Navigate to the `/models/fct/` folder and edit `fct_patient_claims.sql` to define the fact model for patient claims
- Add tests and documentation in `schema.yml`:

    - Include column- and model-level tests such as `not_null`, `unique`, etc.
    - Write clear documentation for each model and its columns
- Run `dbt test` to execute the defined tests
- Run `dbt run` to build all dbt models
- Zip the entire project folder
- Push the zipped (or unzipped) folder to a Git repository
- Create a `README.md` file containing:
    - Assumptions made during the project
    - Steps to build and run the dbt models
    - List of tools and technologies used