ToursGuide Docs

First of all there is a diagram “ER diagram.pdf” which illustrates the basic outline of the database and its main purposes. Then I would recommend reading the guide below to execute the whole database and see how it works. After that there are some explanations of some of the scripts that are also useful to read about as they give the insight of how the database works.

How to use the scripts.

Step 1:

Open pgadmin 4 and open a file named “tables_creation.sql”. Execute the query. This will create all the necessary tables for the ToursGuide App.

Step 2:

Open the files named: “INDEX.sql”, “functions.sql”, “queries for main screens.sql” and “roles.sql”. In roles sql you will need to specify a username that is already known by the database. Execute these queries.

Step 3:

The above files were to create a database on the server without any input info in them. Let's fix that by filling the table “users” and “rooms” with some data. To do so you will need to download the two csv files that have data_set in their name. After that, open the psql console and copy-paste the script from the file “PSQL script to load data.sql”. Follow the instructions in the file because copy-pasting the whole script won’t necessarily work due to copy-paste incorrect formatting.

Step 4:

Now let's create the dwh. Open the script named “dwh_creation.sql” and execute it.

Step 5:

This step is to transform data from our database to data warehouse. Open the script “ETL.sql” which includes basic procedures of transferring data from the database to dwh.

Step: 6

After completing all the above steps we can now go to Power BI and open the “tour_guide_app_analysis.pbix” which will show the analysis of the data in two tables: dimuser and dimroom, with some graphs and statics to cover it.

“Tables_creation.sql” - This script sets up a database called travelling_tours and a schema named GuideToursApp. Within this schema, it creates several tables to manage users, chat rooms, votes, and information about countries, cities, landmarks, hotels, and restaurants.

“INDEX.sql”, “functions.sql”, “queries for main screens.sql” and “roles.sql” - Indexes are created to speed up queries on frequently searched columns. Roles are created to manage user permissions. Queries retrieve specific sets of data from the database, useful for application functionality. And functions encapsulate logic for common operations, making it easier to manage and reuse code within the database.

“PSQL script to load data.sql” - This script describes a series of steps to import data from CSV files into a PostgreSQL database using temporary tables. Temporary tables are created to hold the data from the CSV files temporarily before merging them into the actual tables in the database. Data from the specified CSV files is copied into the temporary tables. The data from the temporary tables is inserted into the actual tables in the database, ensuring no duplicates based on **unique** constraints.

“dwh_creation.sql” - This script defines the creation of dimension and fact tables for a star schema in a data warehouse, including slowly changing dimensions (SCD Type 2) for tracking changes over time.

Dimension Tables: These tables store descriptive information about the dimensions of the data, such as users, countries, cities, rooms, and dates. The DimUser table uses SCD Type 2 to maintain historical records of user data changes.

Fact Tables: These tables store transactional data. The BookingFact table stores booking details, and the VotesFacttable stores voting details. They reference dimension tables using foreign keys to provide context to the transactions.
SCD Type 2: In the DimUser table, the ValidFrom and ValidTo columns allow tracking changes to user data over time. New rows are added with updated values instead of updating the existing rows.

This schema design supports efficient querying for analytical purposes, enabling tracking and analyzing historical data changes and transactions.

“ETL.sql” - This script creates an ETL (Extract, Transform, Load) process for maintaining a data warehouse. It involves creating log tables, defining ETL procedures, and scheduling these procedures to run automatically.

- **ETL Log Table:** Tracks the last processed ID for each ETL process.
- **ETL Procedures:** Handle incremental data loading for users, countries, cities, rooms, and dates.
- **Scheduling:** Automates the ETL process to run daily.

This setup ensures the data warehouse is regularly updated with new data while maintaining historical records and supporting incremental loads.