ETL on transaction data

Purpose and scope

The purpose of this assignment is to create a small but still realistic project to form the basis of a discussion on how you approach data engineering. We expect the project should not take more than a total 4-8 hours of work time to complete. If it takes much longer than that contact us to discuss reducing the scope to make it possible to complete in that time.

Scenario

Viktor maintains an ever-growing large database with transaction data used in a production deployment. His stakeholders want to answer questions like:

- How many transactions happened at ATG for a particular day?
- Which category had the most transactions for a particular day?
- Which hours during the day were the busiest in total?
- Which days of the week were the busiest in total? For a single merchant?

Attached is a subset of this data, most of which is from a single day.

Requirements

Your ETL solution should:

- Load data from the two provided CSV files into a tables with the same columns as in the CSV.
- Create a data warehouse table structure suitable for answering questions above, assuming the data will grow to very large sizes.
- Write an incremental ETL-loader from the tables you created in the first point to the data warehouse structure you created in the second point.
- Write queries (preferably with documentation) for the resulting data warehouse that answer the questions above.

Use tooling you are familiar with. We use Python, Airflow, Postgres and Redshift, but for this project, writing a simple program in whatever language you feel comfortable with is perfectly fine. It is not necessary to use a framework of any sort, we are interested in how you approach problems from a data engineering perspective. One suggestion is to use Python and SQLite without any frameworks.

The project should be available and runnable for us, by following instructions in the project. It should preferably be available on GitHub or similar site.

What we will be discussing

- Overall code structure
- Naming, commenting
- Documentation
- Load performance
- Correctness
- Data issues
- Scalability
- Query performance and tradeoffs
- Ease of querying