Swiss Transport ETL Project

This document serves as a short project overview to present some of it's procedures and final outcome.

1. Data source

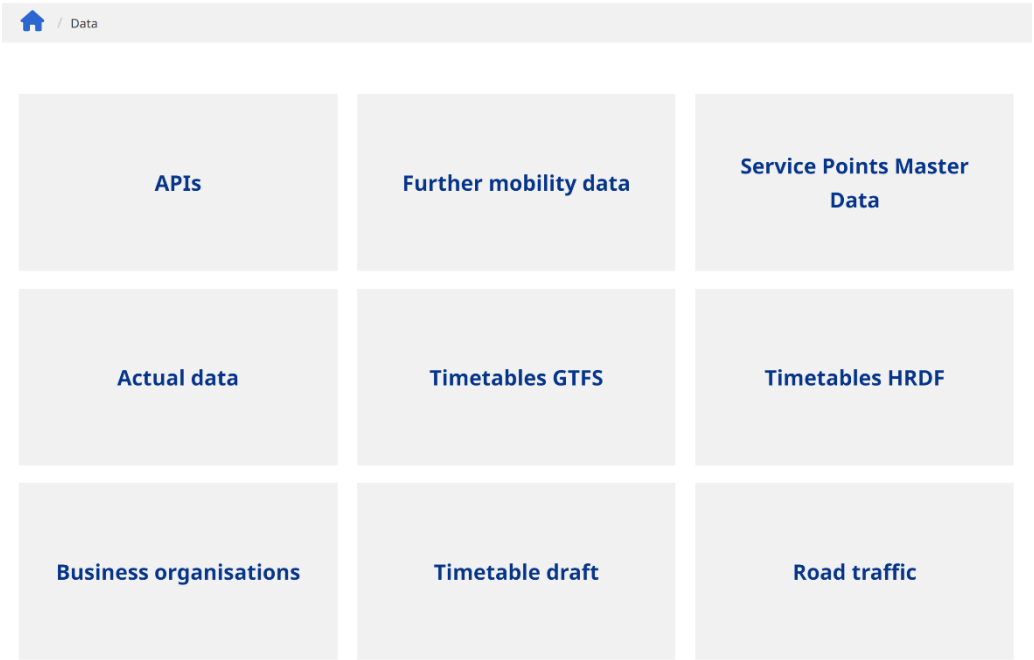
The data source for this project is the Swiss Transport Open Data Mobility Platform - https://opentransportdata.swiss/en/

The platform provides many files sorted in categories like Actual Data, Real-time data, Service Mobility Data, etc.

For this particular project, 12 files have been chosen to create a complex database for daily updates. The files are in different formats (.csv, .xlsx, .json) and some of them need additional processing before ingesting into the cloud.

The file downloads are fully automated with Selenium package running in headless mode on Docker container. A standard Chrome GUI mode is also available to run using files from *local_testing_files* folder.

Files are downloaded with different frequencies which are set intuitively and are fully customizable. However, the main file, which contains data on all trips that happened on the preceding day is updated daily.



Pic. 1.1 – Data platform categories

2. Data ingestion

Downloaded files are ingested into the Snowflake internal stage right after download – the python scripts obtains the file name and incorporates SnowSQL to do so.

Each download frequency has its own folder - both Selenium download directory & Snowflake internal stage. Furthermore, each file is being downloaded into separate subfolder. This way files can be downloaded in parallel mode without disrupting filename retrieval process.

An example of PUT command being successfully received in Snowflake is visible below.

Pic. 2.1 – SnowSQL PUT command sample

3. Data transformation & storage

Data from each file is copied into raw tables in the ‘raw’ schema right after ingestion process is done. In most cases only minor transformations are done during this step.

Files are deleted from the stage folders right after the COPY INTO command is successfully completed.

Further transformations are done while moving data to ‘curated’ & ‘consumption’ schemas – joins, string cleaning, formatting, removing useless data.

Copy processes and transformations are run directly from python scripts using Snowpark.

Query history for automated ‘consumption’ schema tables update is below.

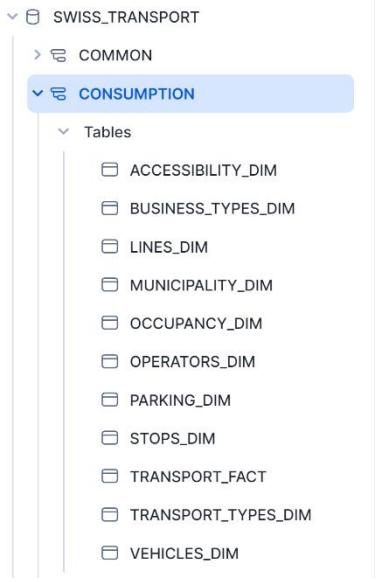
Query History

Status All | User USER_01 | Last 14 days | Filters | 1000+ Queries | Columns

SQL TEXT	QUERY ID	STATUS	USER	WAREHOUSE	DURATI...	STARTED
INSERT OVERWRITE INTO consumption.t…	01b4b436-0000-b983-0000-a3a10…	Succe…	USER_…	TRANSPORT_…	5.8s	5/31/2024, 11:10:27 PM
INSERT OVERWRITE INTO consumption.v…	01b4b436-0000-b983-0000-a3a10…	Succe…	USER_…	TRANSPORT_…	644ms	5/31/2024, 11:10:28 P…
INSERT OVERWRITE INTO consumption.o…	01b4b436-0000-b983-0000-a3a10…	Succe…	USER_…	TRANSPORT_…	513ms	5/31/2024, 11:10:28 P…
INSERT OVERWRITE INTO consumption.l…	01b4b436-0000-b98d-0000-a3a10…	Succe…	USER_…	TRANSPORT_…	591ms	5/31/2024, 11:10:27 PM
INSERT OVERWRITE INTO consumption.a…	01b4b436-0000-b983-0000-a3a10…	Succe…	USER_…	TRANSPORT_…	636ms	5/31/2024, 11:10:27 PM
INSERT OVERWRITE INTO consumption.s…	01b4b436-0000-b98d-0000-a3a10…	Succe…	USER_…	TRANSPORT_…	589ms	5/31/2024, 11:10:27 PM
INSERT OVERWRITE INTO consumption.t…	01b4b436-0000-b983-0000-a3a10…	Succe…	USER_…	TRANSPORT_…	415ms	5/31/2024, 11:10:27 PM
INSERT OVERWRITE INTO consumption.b…	01b4b436-0000-b98d-0000-a3a10…	Succe…	USER_…	TRANSPORT_…	413ms	5/31/2024, 11:10:27 PM

Pic. 3.1 – Query history for consumption schema tables update

The final database structure is a snowflake data model made of 1 fact table and 10 dimension tables, which names, sample code and links you can see below.



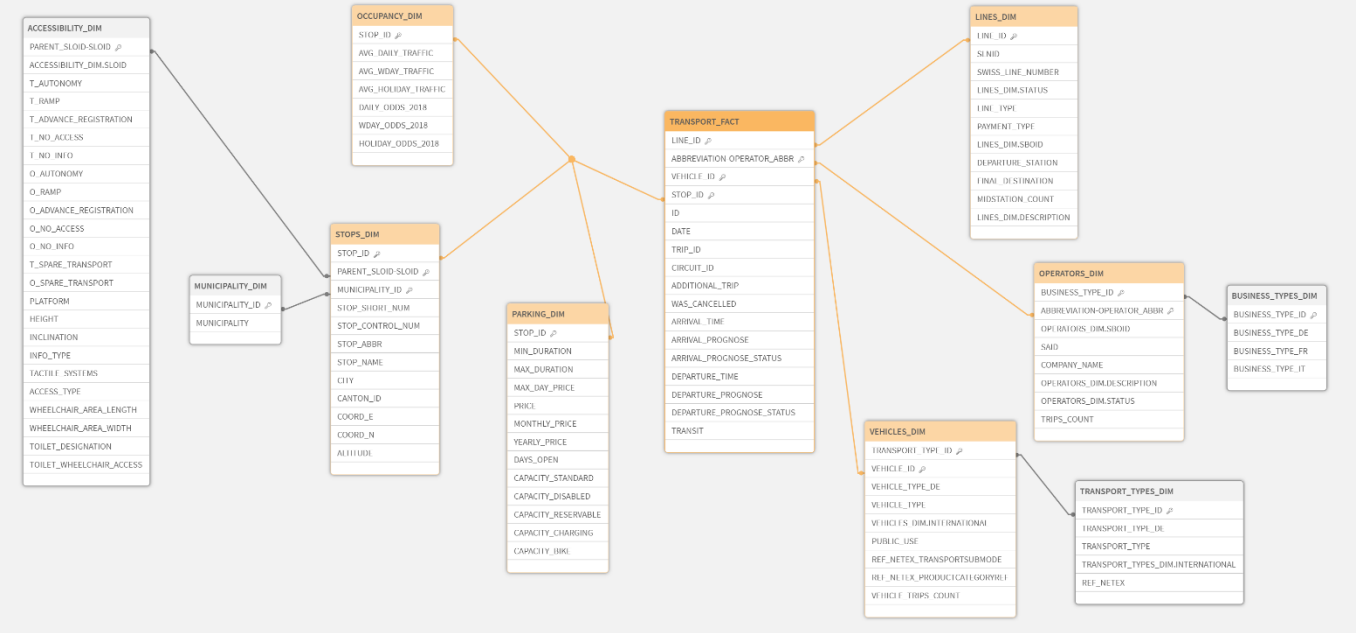
Pic. 3.2 – List of final tables

Table definition

```
create or replace TABLE SWISS_TRANSPORT.CONSUMPTION.TRANSPORT_FACT (
    ID NUMBER(38,0) NOT NULL autoincrement start 1 increment 1 noorder,
    DATE DATE,
    TRIP_ID VARCHAR(16777216),
    OPERATOR_ABBR VARCHAR(16777216),
    LINE_ID VARCHAR(16777216),
    CIRCUIT_ID VARCHAR(16777216),
    VEHICLE_ID VARCHAR(16777216),
    ADDITIONAL_TRIP BOOLEAN,
    WAS_CANCELLED BOOLEAN,
    STOP_ID NUMBER(38,0),
    ARRIVAL_TIME TIMESTAMP_NTZ(9),
    ARRIVAL_PROGNOSE TIMESTAMP_NTZ(9),
    ARRIVAL_PROGNOSE_STATUS VARCHAR(16777216),
    DEPARTURE_TIME TIMESTAMP_NTZ(9),
    DEPARTURE_PROGNOSE TIMESTAMP_NTZ(9),
    DEPARTURE_PROGNOSE_STATUS VARCHAR(16777216),
    TRANSIT BOOLEAN,
    primary key (ID)
);
```

Show less

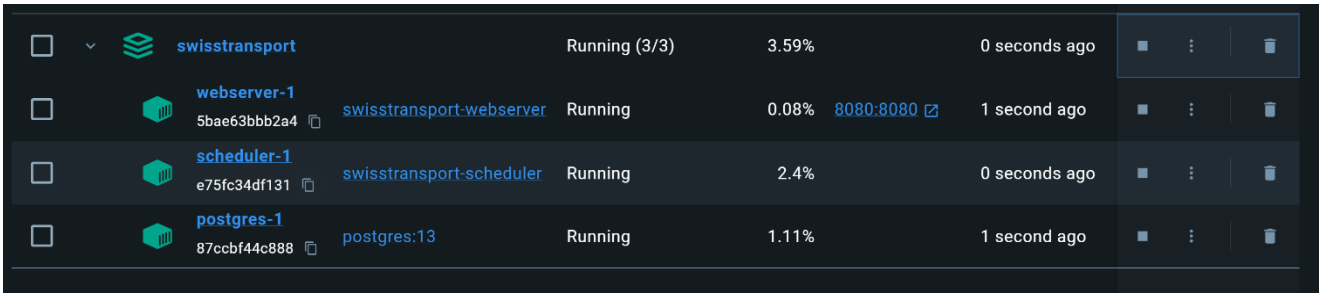
Pic. 3.3 – Sample SQL code – TRANSPORT_FACT table



Pic. 3.4 – ERD diagram of SWISS_TRANSPORT database

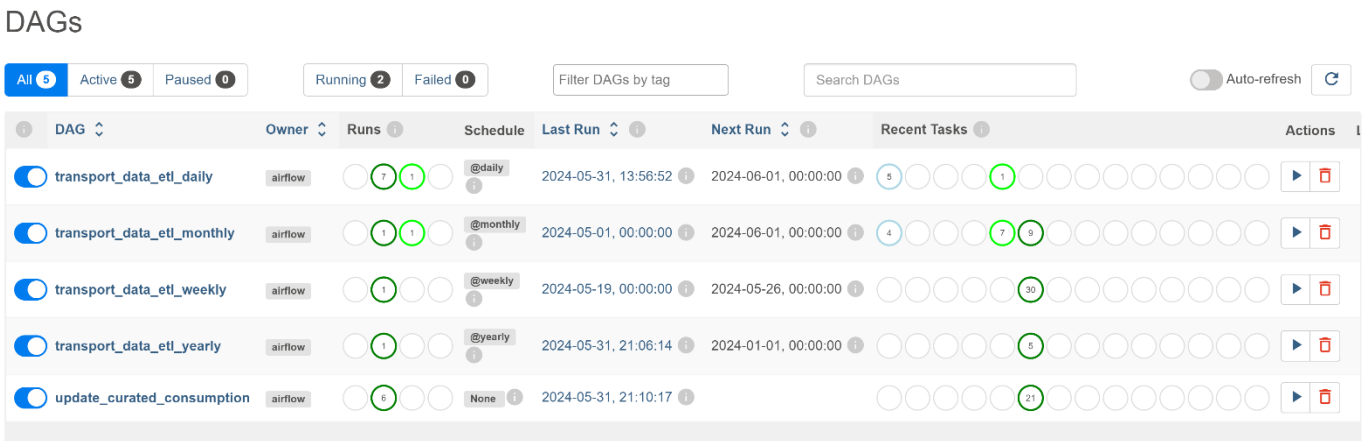
4. Orchestration

All above processes are orchestrated using Airflow. To make this project easy to run and more versatile, Airflow does not run locally – Docker container is used instead.

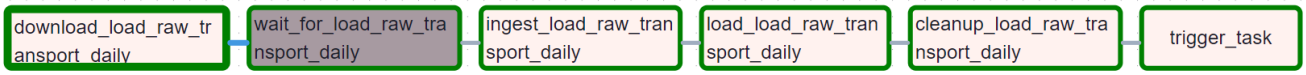


Pic. 4.1 – Docker container – Airflow instances

The orchestration is done using 5 DAGs – which of 1 is externally triggered by the successful run of main, daily DAG – which tasks are visible in pic. 4.3.



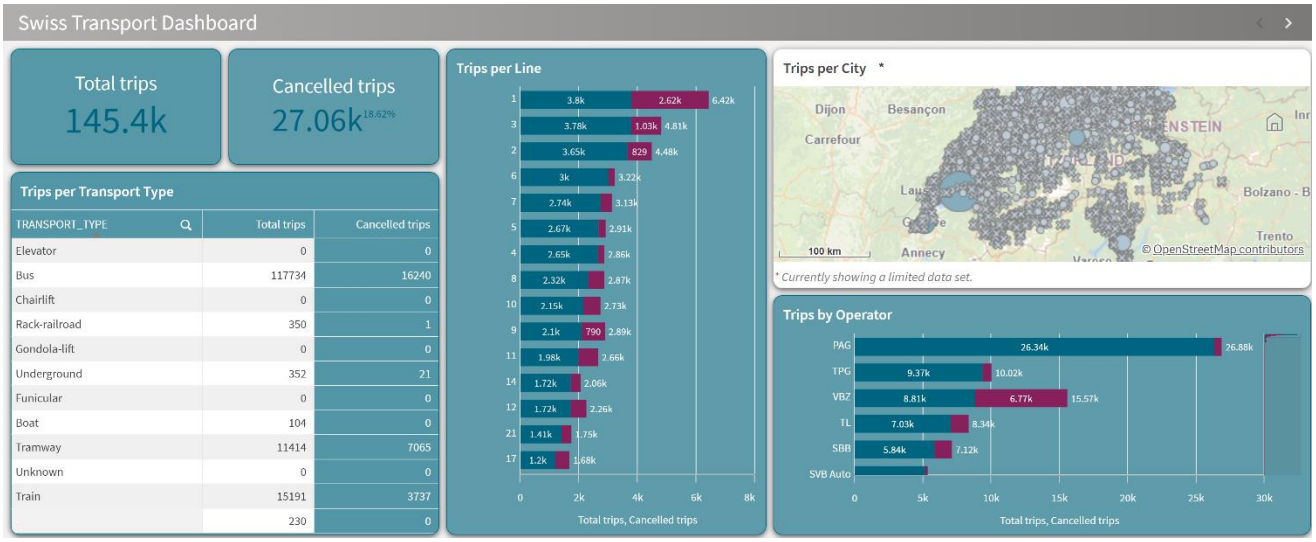
Pic. 4.2 – Airflow web UI – DAG list



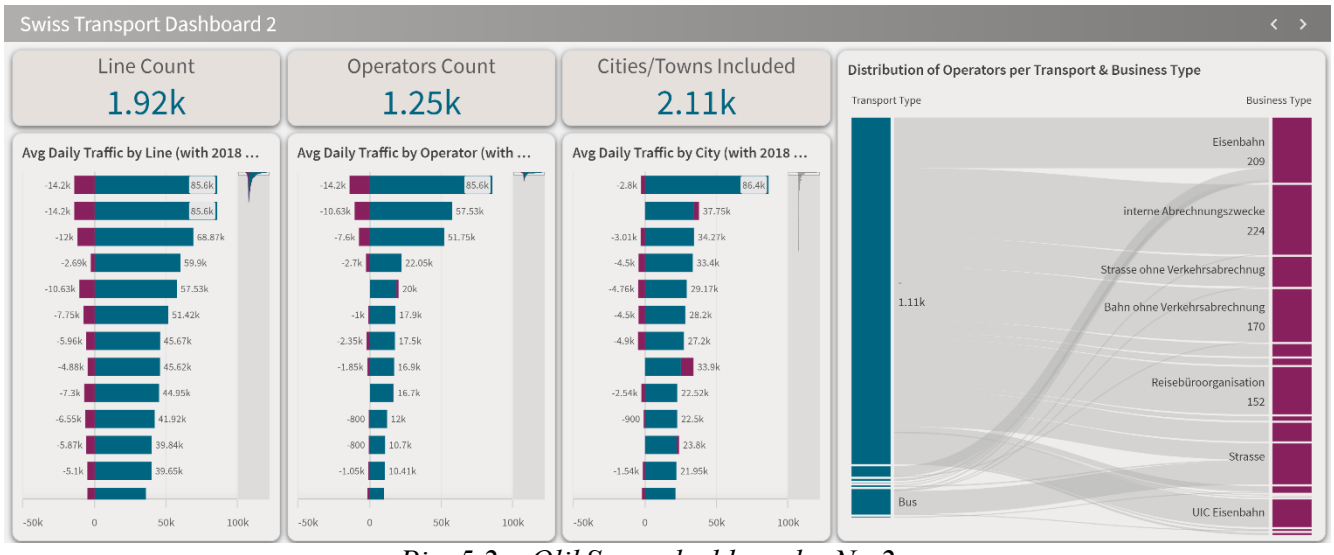
Pic. 4.3 – Sample DAG task dependencies – transport_data_etl_daily DAG

5. Visualization

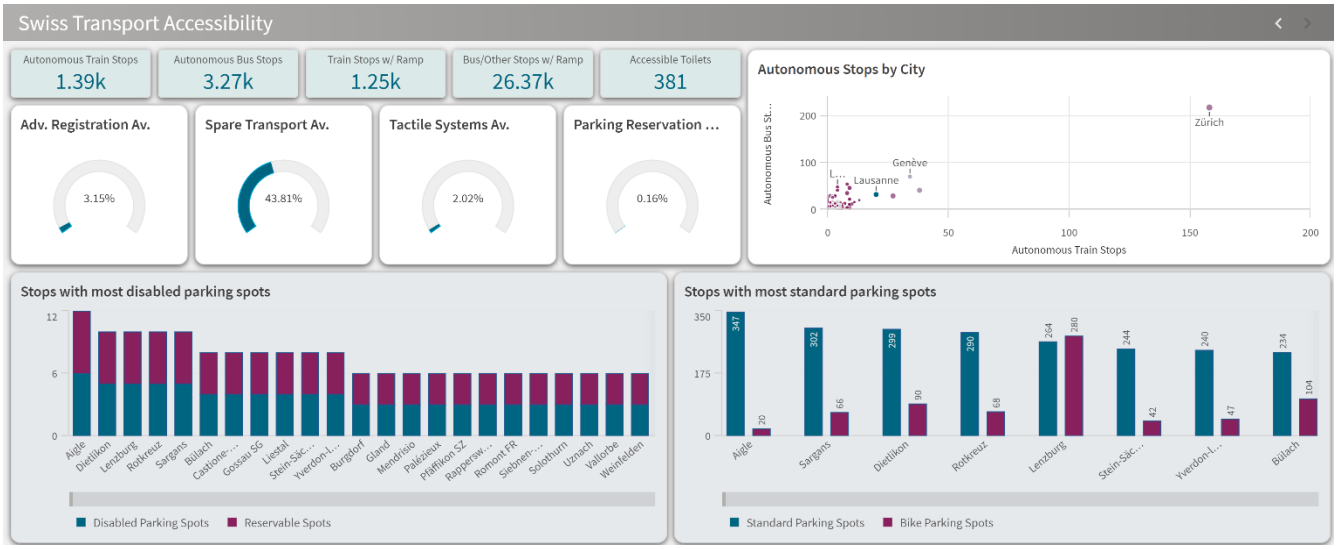
The final data is linked to Qlik Sense, in order to create interactive dashboards. Three different dashboards are created, to show some interesting dependencies, KPIs and more.



Pic. 5.1 – QlikSense dashboard – No 1



Pic. 5.2 – QlikSense dashboard – No 2



Pic. 5.3 – QlikSense dashboard – No 3

6. Summary

A lot of new things were learned during realization of this project in all it's aspects – automation, Snowflake tools, BI visualization.

The final outcome could obviously be expanded by more and more data, possibly including real-time traffic as well.