What do I want?  
  
1. There are 2 files from Kaggle offline and online sales of sport gear  
2. I want an application which – when I run a container – reads from both of the files 1 row at a time (probably like a while loop), places them in the raw\_data schema of sport\_gear\_store database  
3. Then ETL runs, jsons from raw\_data are taken and after ETL placed in the sales table (fact table)  
  
When I up the docker and run some command – tables are created. Records are taken from raw data and sent via API and stored in raw\_data table with all of the events, then ETL, Sales table with customers table get updated (as the rest of the tables will have data)  
  
**The actual plan of actions:**

1. New project is open with virual environment

The design of the project is : **(Update the schema when all is done)**  
SPORT\_GEAR\_ONLINE\_SHOP\_APP

│

├── docker/

│ └── docker-compose.yml

│

├── db/

│ ├── ddl/

│ │ └── create\_schemas\_and\_tables.sql ✅ (✅ DONE)

│ └── connection.py

│

├── api\_server/

│ ├── app.py

│ └── upload\_logic.py

│

├── etl/

│ ├── etl\_online.py

│ ├── etl\_offline.py

│ └── star\_schema\_uploader.py

│

├── data/

│ ├── raw/

│ ├── processed/

│ └── star\_schema\_ready/

│

├── .env

└── README.md

1. I need to upload the files, to imitate like S3 or something

Create .env file with login data for postgres:  
DB\_HOST=localhost

DB\_PORT=5432

DB\_NAME=gear\_sales

DB\_USER=admin

DB\_PASSWORD=admin

1. Docker is connected, image of postgress is pulled, **docker-compose up -d** and it runs 😊
2. I need to create a connection and write a script that creates all of the tables I want with all of the data types (from snowflake\_schema) in the right schemas (**create\_schemas\_and\_tables.sql**)  
   Then I need to run it and make sure all is created and we are good to continue!!!!!!  
     
   2 scripts were created: 1 drops all schemas and tables, 1 creates all schemas and tables with the right columns. Path root/db/ddl  
   2 python files were created to run the scripts: drop\_schemas\_tables.py & create\_schemas\_tables.py, I run them via the terminal

Eventually create\_schemas\_tables\_db.py was modified to run in a loop to create tables in prod and playground environments simultaneously  
  
there is a trick to keep script files in a folder ‘scripts’  
# a trick to run scripts from another folder

import sys

import os

sys.path.append(os.path.abspath(os.path.join(os.path.dirname(\_\_file\_\_), "..")))

4.1 So, probably, I can fill the tables with data from the respective files after processing:  
 a) employees  
 b) products  
 c) product\_categories  
 d) product\_subcategories  
 e) stores  
 f) payment\_methods  
 g) shipping\_methods  
  
I guess these tables can be filled after only creating them as this is the info we have.

**Write what exactly was done here!**

**AT the moment there are 4 scripts under scripts/ directory:  
1) drop\_schemas\_tables.py runs drop\_schemas\_tables.sql  
2) create\_schemas\_tables.py runs create\_schemas\_tables.sql  
3) truncate\_dim\_tables.py runs truncate\_dim\_tables.sql  
4) load\_dim\_tables.py runs and loads data from dim csv files under data/start\_schema\_ready/  
5) truncate\_all\_tables.py runs truncate\_all\_tables.csv  
6) read\_send\_insert\_raw.py reads raw files (2), starts an API server, sends 2 rows (offline + online) at a time through API, opens a connection to the db, and inserts the record to raw\_data.sales\_raw**

1. Build the API part that reads raw data and sends one by one online and offline transactions as jsons, then they are received and placed in the raw\_data.raw\_data with a respectful id
2. Build the ETL part that takes care of all of the things  
   Should ETL be written in SQL or retrieve data from raw and convert to pandas and do all the stuff in pandas??? It is a very good question to address. (because raw\_data it is a relational database already, so I think there is no point in returning it and converting to pandas really…)  
     
   not clear yet, but first of all, 2 tables will have to be updated (customers and sales)  
   Why? Because this is the info that we don’t have  
     
   So payments are coming – then we call ETL, it looks at all of the ids in raw\_data which don’t exist in sales table, this is the first like action, retrieve all of the payment which are not yet ETLed  
     
   2 action – perform a joins with all of the relevant tables to get the final kind of structure of the data.  
     
   Then update table customers with new customers  
     
   Let’s start to build it!  
     
   All the data from raw\_json was retrieved in the query and prepared for joining it with other tables. So it can be done tomorrow on 6.07.2025  
   I need to think over logic on how to take that data, insert it into sales table and then update only the records that have to be updated. Then update customers table with new customers  
     
   Then do some checks and tests and so on and so forth

Take care of customer id in the etl part  
maybe first of all I need to insert a new customer and then do all of the joins and stuff…

The NaN in dfs problem was solved with fixing loading data to the dim tables  
with this line:  
**cleaned\_row = tuple(None if pd.isna(val) else val for val in row)**

**06.07.2025.**

**Today, I need to do the following:  
  
1. Finish ETl, add Customers with the respective joins**

**2. make it update the table customers then table sales**

**3. Make sure I can stop the run and it will continue from where it started**

**4. Try Streamlit and see what you can do with it.**

**It seems like all works, in order to turn it on:  
1. Up docker  
2. drop\_schemas\_tables.py in scripts 🡪 run  
3. create\_schemas\_tables.py in scripts 🡪 run  
4. load\_dim\_tables.py in scripts 🡪 run  
5. read\_send\_insert\_raw.py 🡪 run, it will start reading files from raw csv files and move it to raw\_data  
6. etl\_customers.py 🡪 run, it will get new customers to customers table, 2 schemas  
7. etl\_sales.py 🡪 run, it will get all sales to the sales table in both schemas**

**08.07.2025 After talking to the lecturer – a completely new idea with new logic**

1. **First of all, add a new container MinIO – done, imitation of s3**
2. **Conditional check if raw data in raw – data folder, if not – download data, sort it, creat a folder raw\_data, put it there and then copy to the bucket of raw-data-kaggle**
3. **Then all of the runs of creating tables, schemas, loading dimention tables – work logically. There is a check of what exists and what doesn’t and then you can see what you should do with it.**
4. **So up the containers in docker🡪download raw data🡪put it in raw\_data folder & s3🡪create schemas and tables🡪load dimention tables🡪run reading and creating files daily and saving them in unprocessed-bucket bucket with this hierarchy: yyyy/mm/day/online.csv + offline.csv. Keep track of what was read, processed and passed to s3.**
5. **Create an ETL process that checks what was not processed, takes the files, maybe joins them, does etl to them, unites then upserts to sql for customers**
6. **Step 5 but final one for sales, upsert, check for new records**
7. **Streamlit application envelope to make**