Setup Guide

This file contains all steps necessary to perform the setup for the sales_dw data warehouse project

Prerequisites

The following installments are required for the setup:

- Docker Desktop
- Power BI

Folder structure

The project contains the following structure:

```
data_warehouse
    │   automated_copy_to_models.py                # copies staging, dimension,
and fact tables into their dedicated dbt subfolders
    │   docker-compose.yml                         # initial docker setup for
data warehouse project
    │   drop_tables.sql                            # script for dropping all
tables
    │   source.yml                                 # raw table setup for dbt
    │   Use_Case_salesdb.pbix                      # power bi use case
visualization
    │   Visualization_Salesdw_PowerBI.pbix         # power bi dashboard
visualization
    │
    ├───dim_sql_files                              # dimension tables sql files
    │       dim_customers.sql
    │       dim_datetime.sql
    │       dim_event_types.sql
    │       dim_location.sql
    │       dim_products.sql
    │       dim_suppliers.sql
    │
    ├───documentation                              # project documentation files
    │       multidimensional_schema.md
    │       setup_guide.md
    │       visualization_of_usecase.md
    │
    ├───fact_sql_files                             # fact table sql files
    │       fact_inventory.sql
    │       fact_orders.sql
    │
    ├───import_data                                # raw input csv files
    │       customers.csv
    │       event_types.csv
    │       inventory.csv
    │       locations.csv
```

```
        │       orders.csv
        │       products.csv
        │       suppliers.csv
        │
        └───stag_sql_files                              # staging table sql files
                stg_customers.sql
                stg_event_types.sql
                stg_inventory.sql
                stg_locations.sql
                stg_orders.sql
                stg_products.sql
                stg_suppliers.sql
```

Technology stack

For this project, the following technology stack was used:

- PostgreSQL – A popular open-source relational database
- pgAdmin – A web-based administration tool for managing PostgreSQL databases.
- dbt – SQL-based data transformation and modeling tool.
- Power BI – Visualization tool for building interactive dashboards.

Initial setup

Initiate the project environment using the provided `docker-compose.yml` file:

```
docker compose up -d
```

This command initializes and starts all required containers.

At the end, you should see the following:

```
[+] Running 4/4
 ✔ Network data_warehouse_default  Created
 ✔ Container dw_postgres           Started
 ✔ Container dw_dbt                Started
 ✔ Container dw_pgadmin            Started
```

Verify if PostgreSQL database exists

After starting the containers, you can verify that the database is by accessing it via PgAdmin @ localhost:8080. Use the following credentials to login:

- Username: **admin@pgadmin.com**
- Password: **admin**

The PostgreSQL container automatically creates a database named `sales_dw` on startup. You can verify its existence by clicking `Add New Server`. Enter the following:

- **Name:** `PostgreSQL` (or any name you like)
- Go to the **Connection** tab and enter:
  - **Host name/address:** `postgres`
  - **Port:** `5432`
  - **Username:** `dbt`
  - **Password:** `dbt`

Now you should be able to find the empty `sales_dw` database in the Object Explorer tab.

Creating a new dbt project

A `dbt` project has to be created for to manage the tables of this database. Enter the dbt container using the following command:

```
docker exec -it dw_dbt bash
```

If not already in */usr/app*, move into this folder and type the following command to intiate the project

```
dbt init sales_project
```

When asked, enter the following information:

- Database: **[1] postgres**
- Host: **postgres**
- Port: **5432**
- User: **dbt**
- Password: **dbt**
- DBname: **sales_dw**
- Schema: **public**
- Threads: **1**

After this is done, the project should be successfully created. In order to verify this, navigate into the sales_project folder like this:

```
cd /usr/app/sales_project
```

Then, test if the seutp was successful by typing in the following command:

```
dbt debug
```

This should result in the following message:

```
**All checks passed!**
```

This confirms the dbt project was created successfully.

Creating a multidimensional model

From inside of your data_warehouse folder, move to the following folder location:

```
dbt/sales_project/seeds
```

Store the raw input data located in the `import_data` folder from the project folder structure inside of this `seeds`. After taht, from inside the dbt container, use the following command

```
dbt seed
```

After completion, the something like following result should be seen in the console:

```
Finished running 7 seeds in 0 hours 0 minutes and 0.64 seconds (0.64s).

Completed successfully
```

Now within the data warehouse project, execute following script to load the multidimensional model into the dbt container:

`automated_copy_to_models.py`

After that, all sql scripts should be copied into the folder `dbt/models/`.

Finally, execute the loaded scripts using the following command:

`dbt run`

After completion the terminal should show this report:

```
Finished running 8 table models, 7 view models in 0 hours 0 minutes and 1.27
seconds (1.27s).

Completed successfully
```

This concludes the setup of the data warehouse.