**Database Description**

Database for online gardening store which sells plants and gardening equipment.

It also contains data about clients, business partners with some important details for business relations (business contract numbers, warehouses, legal information etc.).

**Files description**

1.1. OLTP database schema and script for its creation.

1.2. All files with .csv extension.

1.3. ETL script for filling OLTP database.

2.1. OLAP database schema and script for its creation.

2.2. ETL script for filling OLAP database.

2.3. PowerBI report.

3.1. OLTP script for getting information about supplier's warehouses and total orders from each with contract number and responsible person from their side and information about the number of customers in particular country with total amounts spent.

3.2. OLAP  script for getting information about average profit for category and information about the total sales by category.

Schemas generated by postgreSQL for OLTP and OLAP databases as well added to the repository with corresponding file names.

**OLTP schema description**

Schema contains 24 tables in 3NF form.

**1.1. OLTP schema.png file contains schema’s visual representation with all relations.**

**Overall Description of the Schema**

The OLTP schema is designed to manage and store transactional data for an e-commerce platform. Below is an overview of the key components:

1. **Core Tables:**
   * **Products**: Stores common for all types of products information.
   * **Contacts**: Represents common information about individuals or businesses entities interacting with the store without any additional specification. For business partners (for example suppliers, manufacturers and delivery providers) fields “**contact\_name, contact\_surname, contact\_email, contact\_phone\_number**” contains information about business partner’s representative which is responsible for communication with our business” and field “**contact\_address\_id**” refers legal address of partner’s company.
   * **Orders**: Contains common information for customer and supply orders. In the case of supply\_order  customer\_id refers to information about our store. In this work I used contact *plantstore* (this contact contains data about our store only).
   * **Addresses:** Addresses information.

1. **Supporting Tables:**
   * **Product\_Categories and Product\_Subcategories**: Organizes products into hierarchical groups.
   * **Coupons and Coupon\_Usage**: Manages promotional discounts and their usage. The Usage table contains unique pairs (customer-coupon) to guarantee that the customer could have only one coupon of a particular type.
   * **Images**: Images for reviews and products, check condition ensures that the photo cannot be at the same time in product description and customer’s review.
   * **Wishlists**: Customers’ wishlists.
   * **Refunds**: Data about customers’ refunds.
   * **Warehouses**: Data about suppliers’ warehouses.
   * **Cart:** Customer’s cart Coupons could be applied for particular products.
   * **Reviews:** Customers’ product reviews.
   * **Brands:** Contains information about the brands of product manufacturers.
   * **Users:** Web\_app users’ table.

1. **Details Tables:**
   * **Product\_Details\_Fertilizers:** Specific details about specific products’ type.
   * **Product\_Details\_Containers:** Specific details about specific products’ type.
   * **Product\_Details\_Plants:** Specific details about specific products’ type.
   * **Product\_Details\_Tools:** Specific details about specific products’ type.
   * **Product\_Properties:** Purpose of that table is flexibility. If some products require some additional characteristics, they could be added using this table.
   * **Business\_Partner\_Details:** Contains detailed information about business partners.
   * **Customer\_Details:** Exists for storing customer card numbers.
   * **Order\_Details:** Information about every product in the order.

1. **Constraints and Relationships:**
   * Foreign keys are extensively used to enforce referential integrity (e.g., Products references Product\_Subcategories, Orders references Contacts and Addresses etc.).
   * Check constraints ensure data validity (e.g., rating BETWEEN 0 AND 5 for reviews, quantity > 0 in orders).
   * Composite keys and unique constraints ensure data uniqueness, such as in Coupon\_Usage and Product\_Properties.

1. **Enum Types:**
   * Custom enumerations (e.g., discount\_type\_enum, order\_status\_enum, user\_role\_enum) are used to standardize specific attributes.

**1.1. OLTP table creation.sql file contains the script for creating this OLTP database.**

All files with .csv extension correspond to 1.2. step and contain data for loading to the OLTP database in 1.3. step.

**1.3.1. ETL to load data from CSV to OLTP**

1.3.1. upload data for OLTP from CSV.sql file contains the script for creating temporary tables for further usage in ETL process and extracting the data from csv files to those temporary tables. Those tables contain the same columns as csv files, no data transformation in that step is performed, only data extracting from csv.

**1.3.2. ETL for OLTP .sql - ETL script for transforming and loading data to OLTP database**

Script fills following 15  tables:

Addresses,

Contacts,

Business\_Partner\_Details,

Customer\_Details,

Warehouses,

Coupons,

Product\_Categories,

Product\_Subcategories,

Products,

Product\_Details\_Plants,

Product\_Details\_Containers,

Product\_Details\_Fertilizers,

Product\_Details\_Tools,

Orders,

Order\_Details;

In the script all inserting procedures are placed in correct order to not cause conflicts in relations and all operations are rerunnable, they’re not adding duplicated data. Duplication is avoided by using “joins” and comparing all fields or using “on conflict” by one particular field, which indicates duplicated records.

During filling Addresses, Contacts, Products, Categories and Subcategories there are repeating parts with only difference in the source table. It’s because the initial data in csv files spreaded across different tables. For example addresses information contains files with partners and customers, and moreover there are warehouse addresses in partners table, but with different column names (like “warehouse\_country” etc.).

So it’s important to pay attention to the source table when executing the script partially. Better to execute it all, because it will not add any data that is already added, it will only add new data.

**Correct order for script execution**

Order of adding tables within one step is not important

The most important part about the script is that firstly filling commands for tables should be executed, which are not referring to other tables.

Order of adding tables within one step is not important, there is only need to follow the order of steps (For example step 2 cannot be executed before step 1).

**Step 1.** Tables that could be filled in the first place (they’re not referring to anything):

* Addresses
* Coupons
* Product\_Categories

**Step 2.** Tables refer to only one already added table in the last step (referred table specified in the brackets):

* Contacts(Addresses tables)
* Product\_Subcategories (Product\_Categories)

**Step 3.** Tables refer to one or more already added tables in the last steps (referred tables specified in the brackets):

* Business\_Partner\_Details(Contacts)
* Customer\_Details(Contacts)
* Products(Contacts, Product\_Subcategories)
* Warehouses(Contacts, Addresses)

**Step 4.** Tables refer to one or more already added tables in the last steps (referred tables specified in the brackets):

* Product\_Details\_Plants(Products)
* Product\_Details\_Containers(Products)
* Product\_Details\_Fertilizers(Products)
* Product\_Details\_Tools(Products)
* Orders(Contacts, Addresses, Coupons)

**Step 5:**

* Order\_Details (Orders, Products)

**2.1. OLAP schema.png file contains schema’s visual representation with all relations.**

**2.1. create\_OLAP.sql file contains the script for creating this OLAP database.**

**OLAP schema description**

**Contains  10 tables:**

fact\_orders

fact\_supply\_orders

dim\_suppliers

dim\_warehouses

dim\_products

dim\_delivery\_providers

dim\_locations

dim\_customers

dim\_dates

agg\_sales\_by\_category

This schema is designed for analytical processing, enabling detailed exploration of supply chain and sales data across various dimensions like time, location, suppliers, and products.

**Dimension Tables**

* **dim\_locations:** Stores detailed information about locations.
* **dim\_suppliers:** Stores information about suppliers.
* **dim\_warehouses:** Stores information about warehouses.
* **dim\_products:** Stores information about products.
* **dim\_delivery\_providers:** Stores information about delivery providers.
* **dim\_dates:** Stores date-related attributes for time-based analysis.
* **dim\_customers:** Stores information about customers.

**Fact Tables**

* **fact\_supply\_orders:** Stores data related to supply orders.
* **fact\_orders:** Stores data related to customer orders.

**Aggregate Table**

* **agg\_sales\_by\_category:** Stores aggregated sales data by product category and subcategory.

**SCD Type 2(History added as a new row):** There are several tables containing historical data (historical records distinguished by values in the columns effective\_start\_date, effective\_end\_date and ic\_current boolean flag): dim\_suppliers, dim\_products, dim\_delivery\_providers and dim\_customers

**2.2. ETL for OLAP.sql**

All steps with short description marked as comments inside the script.

It’s rerunnable and does not add duplicate data. This logic (avoiding duplicates) is implemented in the “Load staging tables” phase. And only after that on “Insert validated and deduplicated data into target table” phase correct data inserted in the destination tables.

All data transformation and validation also performed during the first phase.

So basically every step is divided into two phases: rerunnable and not  allowing duplicates filling the staging tables and filling that data into final tables.

**Important Note:** All steps should be executed in the same order as in the script and the second phase for inserting in OLAP tables should be executed for every staging table only once (while filling staging tables has no such limits and could be done without any limits).

If there would be a need to load new data into the staging table after performing the second phase, firstly sequence drop table-create table from step 0 should be executed.

Before Step 0 it configures a PostgreSQL database to access data from another PostgreSQL instance using the postgres\_fdw extension. It creates a foreign server named foreign\_db\_server\_2, specifying connection details such as the host and database name. A user mapping is established to authenticate the current user with the remote server using specific credentials. A local schema is created to organize imported foreign tables. Finally, it imports all tables from the public schema of the remote database into the local\_schema of the current database, enabling seamless access to the remote data.