REPORT

DEVELOPING A DATA WAREHOUSE USING POSTGRESQL: AN END-TO-END RETAIL SALES DATA PIPELINE

Presented by:
Saiful Islam Rupom

Repository: https://github.com/saiful-islam-rupom/data-warehouse-sql-retail-sales.git

Project Overview

This project demonstrates the development of a modern data warehouse using PostgreSQL, encompassing the ETL process, Medallion Architecture (Bronze, Silver, Gold layers), dimensional data modeling, and preparation of clean, analytics-ready datasets. It focuses on transforming raw retail sales data into a structured format to support business insights and data-driven decision-making.

Objectives

Develop a modern and organized data warehouse using PostgreSQL to bring together sales data from different sources into a single, unified system. This structured approach allows for better data management, improved accuracy, and easier access to meaningful information. By turning raw data into well-prepared datasets, the project aims to support clear, reliable reporting and help businesses make informed, data-driven decisions.

Tech Stack

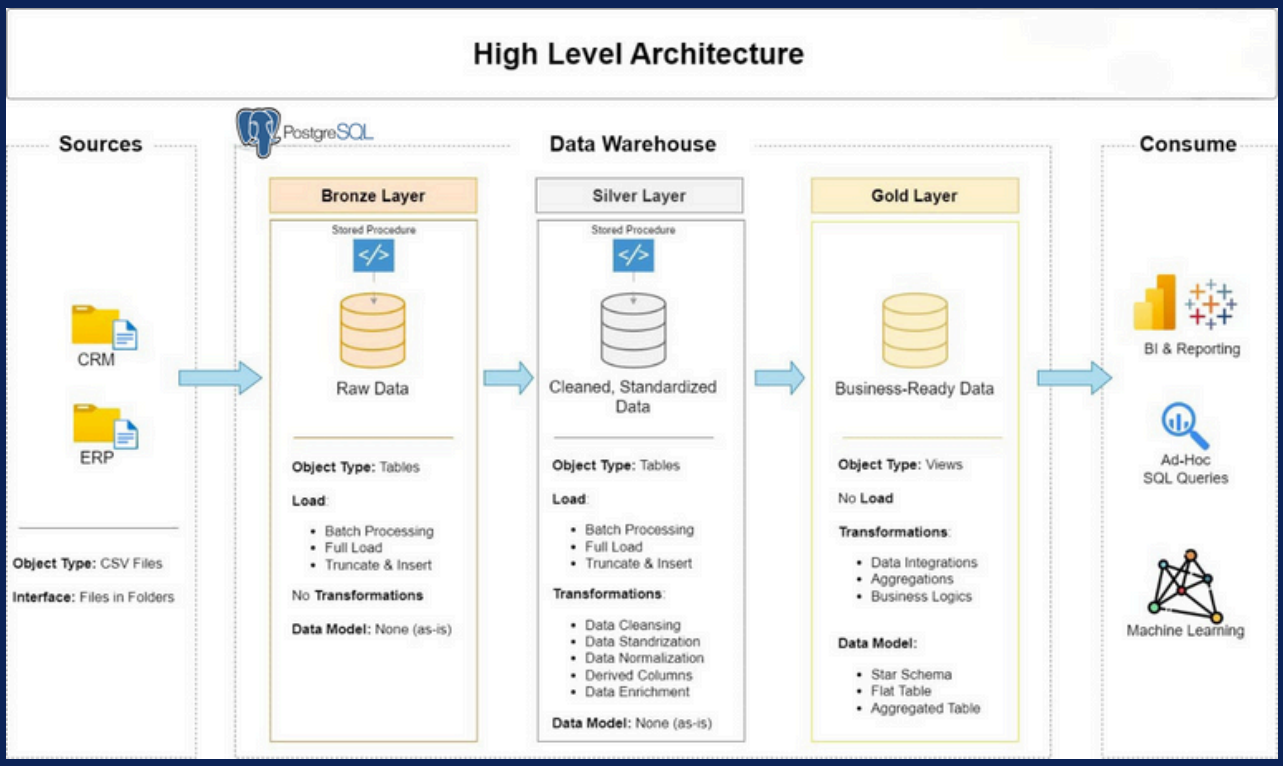
- Database: PostgreSQL
- Tool: pgAdmin 4
- ETL Workflow: Manual SQL scripting
- Architecture: Medallion Architecture (Bronze → Silver → Gold)
- Modeling: Star Schema (Fact and Dimension tables)
- Data Sources: CRM & ERP datasets (CSV files)
- Diagrams & Flows: Draw.io

ETL Workflow Overview

1. Data Ingestion: Import raw data from CRM and ERP Datasets into Bronze layer (staging tables).
2. Data Transformation: Clean and standardize data in Silver layer, merge sources, apply business logic.
3. Data Modeling: Design and implement a star schema in the Gold layer:
    - fact_sales
    - dim_customers
    - dim_products
4. Data Loading: Populate the Gold layer tables with fully transformed data.
5. Analysis & Reporting: Run analytical SQL queries to uncover trends and support decision-making.(Not part of this project)

Data Architecture

This project follows the Medallion Architecture approach. A high-level architectural overview showing the flow of data across layers and components from source ingestion to analytics-ready output.

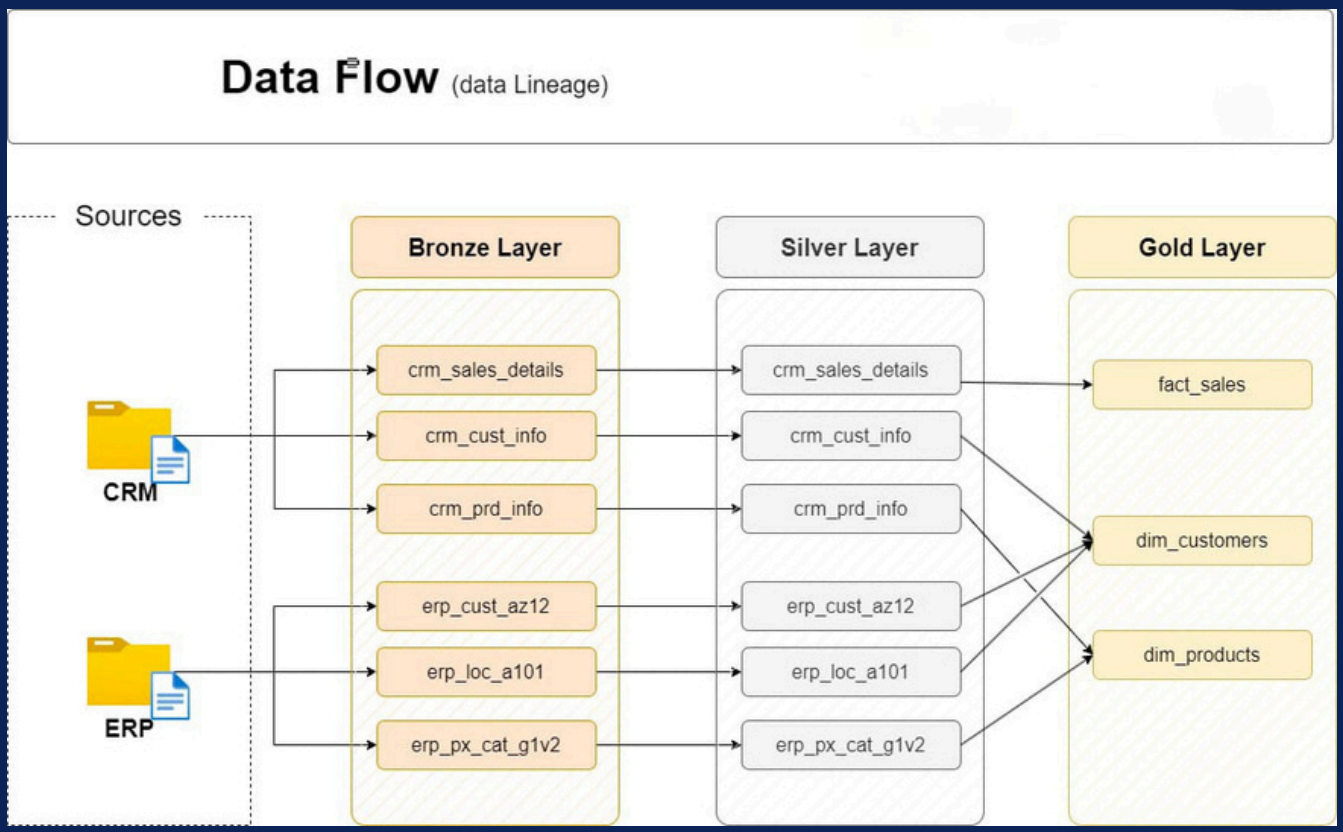


	Bronze Layer	Silver Layer	Gold Layer
Definition	Raw, unprocessed data as-is from sources	Clean & standardized data	Business-Ready data
Objective	Traceability & Debugging	(Intermediate Layer) Prepare Data for Analysis	Provide data to be consumed for reporting & Analytics
Object Type	Tables	Tables	Views
Load Method	Full Load (Truncate & Insert)	Full Load (Truncate & Insert)	None
Data Transformation	None (as-is)	- Data **Cleaning** - Data **Standardization** - Data **Normalization** - **Derived Columns** - Data **Enrichment**	- Data **Integration** - Data **Aggregation** - **Business Logic & Rules**
Data Modeling	None (as-is)	None (as-is)	- **Start Schema** - **Aggregated Objects** - **Flat Tables**
Target Audience	- Data Engineers	- Data Analysts - Data Engineers	- Data Analysts - Business Users

1. Bronze Layer (Raw Stage)
    - Loads raw CRM and ERP data into staging tables.
    - No transformations applied yet.
    - Represents the "source of truth" from external systems.
2. Silver Layer (Cleansed Stage)
    - Cleans and transforms data from the Bronze layer.
    - Handles nulls, formats, deduplication, and joins CRM/ERP data.
    - Stores the refined and validated data for downstream use.
3. Gold Layer (Business Layer)
    - Builds analytical fact and dimension tables using a star schema.
    - Used directly for reporting, dashboards, and business decision-making.

Data Flow

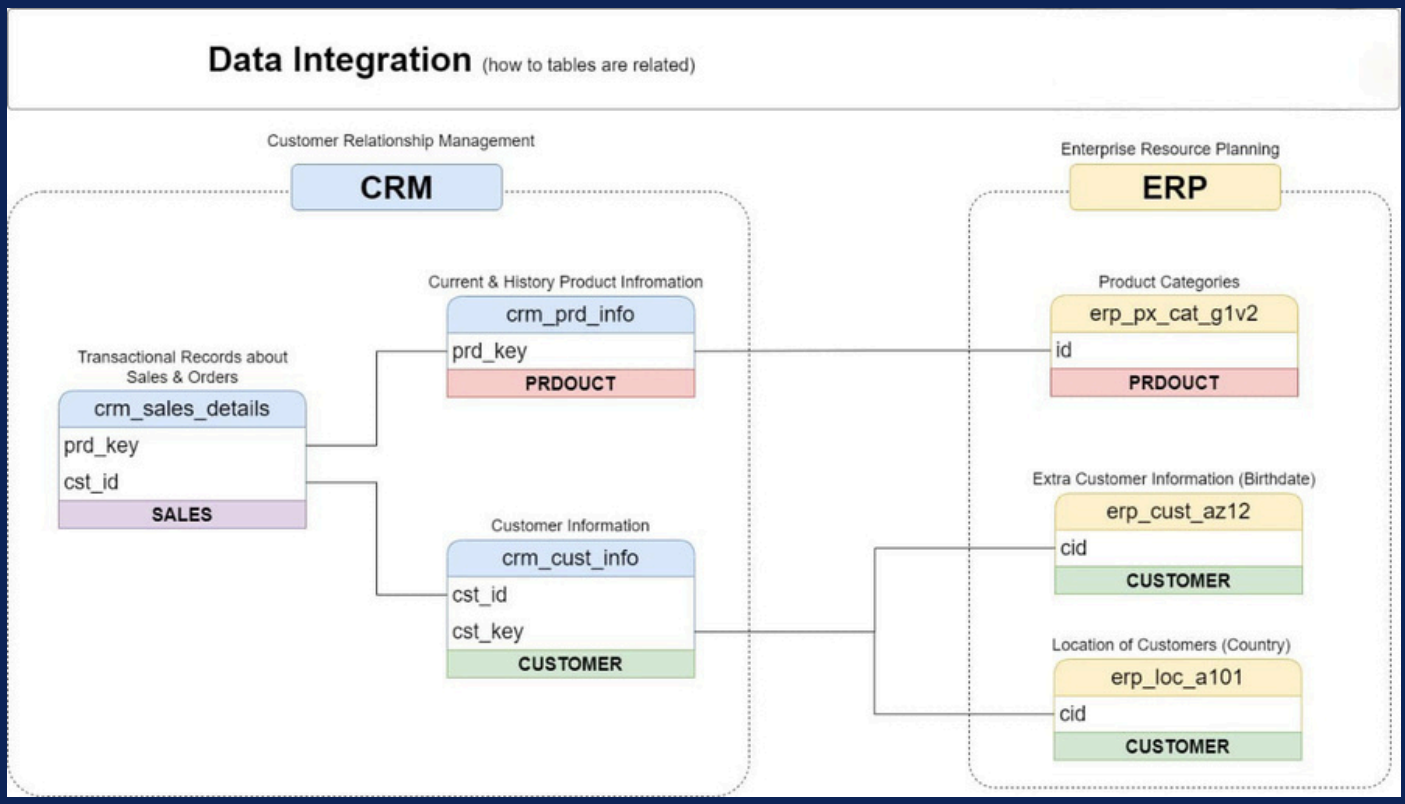
Data lineage in the Medallion Architecture, showing how raw data from CRM and ERP sources flows through Bronze (raw), Silver (cleaned), and Gold (business-ready) layers. Each layer progressively transforms and integrates the data into structured tables used for analytics and reporting.



Data Integration

This project integrates raw data from two different business domains (Each contains three distinct .csv files):

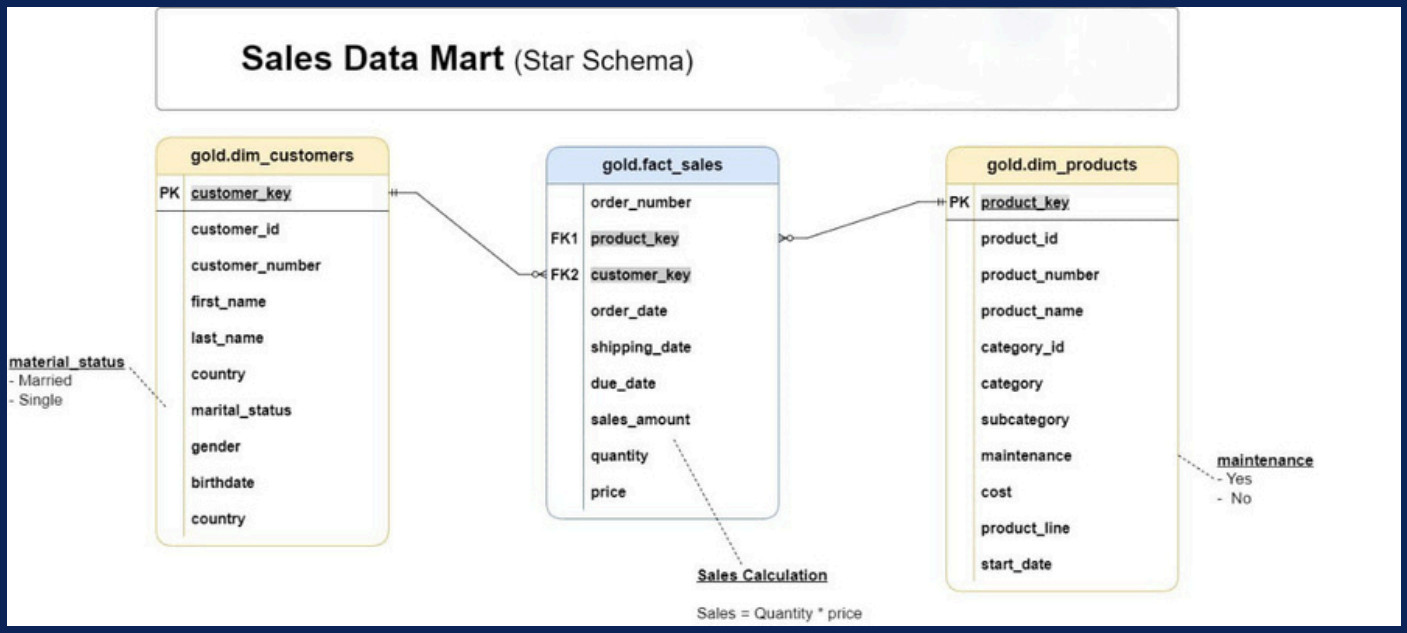
1. CRM: Customer relationship data like customers info, transactional records about sales & orders, and product info.
    - cust_info.csv
    - prd_info.csv
    - sales_details.csv
2. ERP: Sales transactions, customer's extra info, product info etc.
    - CUST_AZ12.csv
    - LOC_A101.csv
    - PX_CAT_G1V2.csv



Data Modeling — Star Schema

A star schema was designed consisting of:

- One central Fact Table:
  - fact_sales
- Supporting two Dimension Tables:
  - dim_customers
  - dim_products

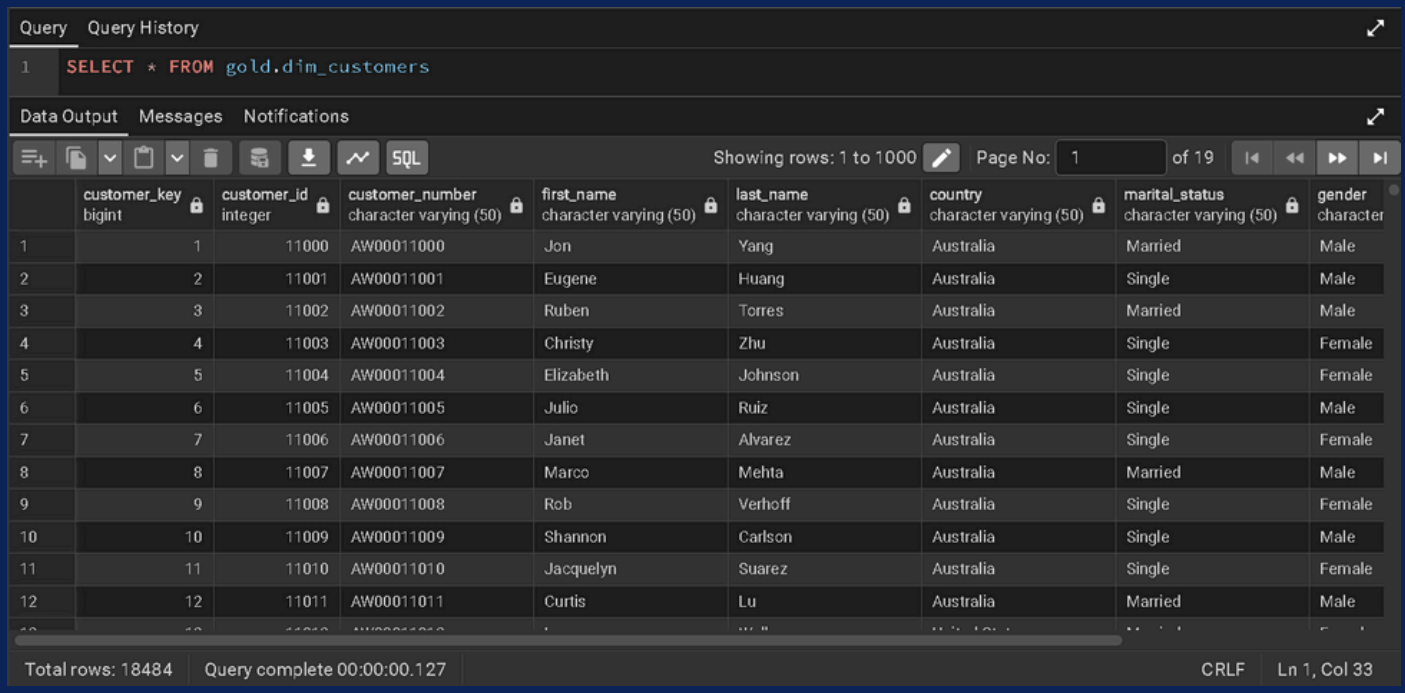


Gold Layer Table Snapshots

The Gold Layer is the business-level data representation, well-suited for analytical and reporting use cases. It consists of dimension tables and fact tables for specific business metrics.

1. gold.dim_customers

Contains enriched customer information including unique customer IDs, names, and regions used for customer-based analysis.



Query | Query History

```
SELECT * FROM gold.dim_customers
```

Data Output | Messages | Notifications

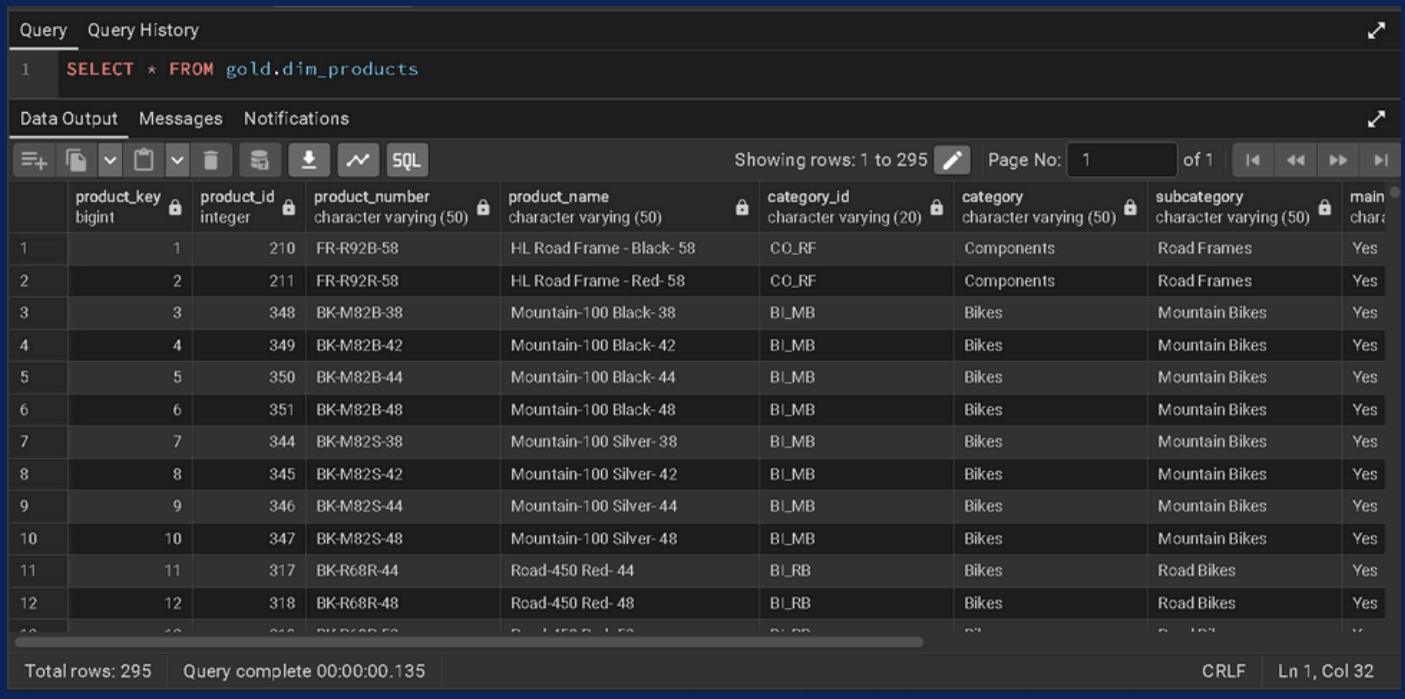
Showing rows: 1 to 1000 | Page No: 1 of 19

	customer_key bigint	customer_id integer	customer_number character varying (50)	first_name character varying (50)	last_name character varying (50)	country character varying (50)	marital_status character varying (50)	gender character
1	1	11000	AW00011000	Jon	Yang	Australia	Married	Male
2	2	11001	AW00011001	Eugene	Huang	Australia	Single	Male
3	3	11002	AW00011002	Ruben	Torres	Australia	Married	Male
4	4	11003	AW00011003	Christy	Zhu	Australia	Single	Female
5	5	11004	AW00011004	Elizabeth	Johnson	Australia	Single	Female
6	6	11005	AW00011005	Julio	Ruiz	Australia	Single	Male
7	7	11006	AW00011006	Janet	Alvarez	Australia	Single	Female
8	8	11007	AW00011007	Marco	Mehta	Australia	Married	Male
9	9	11008	AW00011008	Rob	Verhoff	Australia	Single	Female
10	10	11009	AW00011009	Shannon	Carlson	Australia	Single	Male
11	11	11010	AW00011010	Jacquelyn	Suarez	Australia	Single	Female
12	12	11011	AW00011011	Curtis	Lu	Australia	Married	Male

Total rows: 18484 | Query complete 00:00:00.127 | CRLF | Ln 1, Col 33

2. gold.dim_products

Holds detailed product data such as product names, categories, and prices to support product-level performance reporting.



Query | Query History

```
SELECT * FROM gold.dim_products
```

Data Output | Messages | Notifications

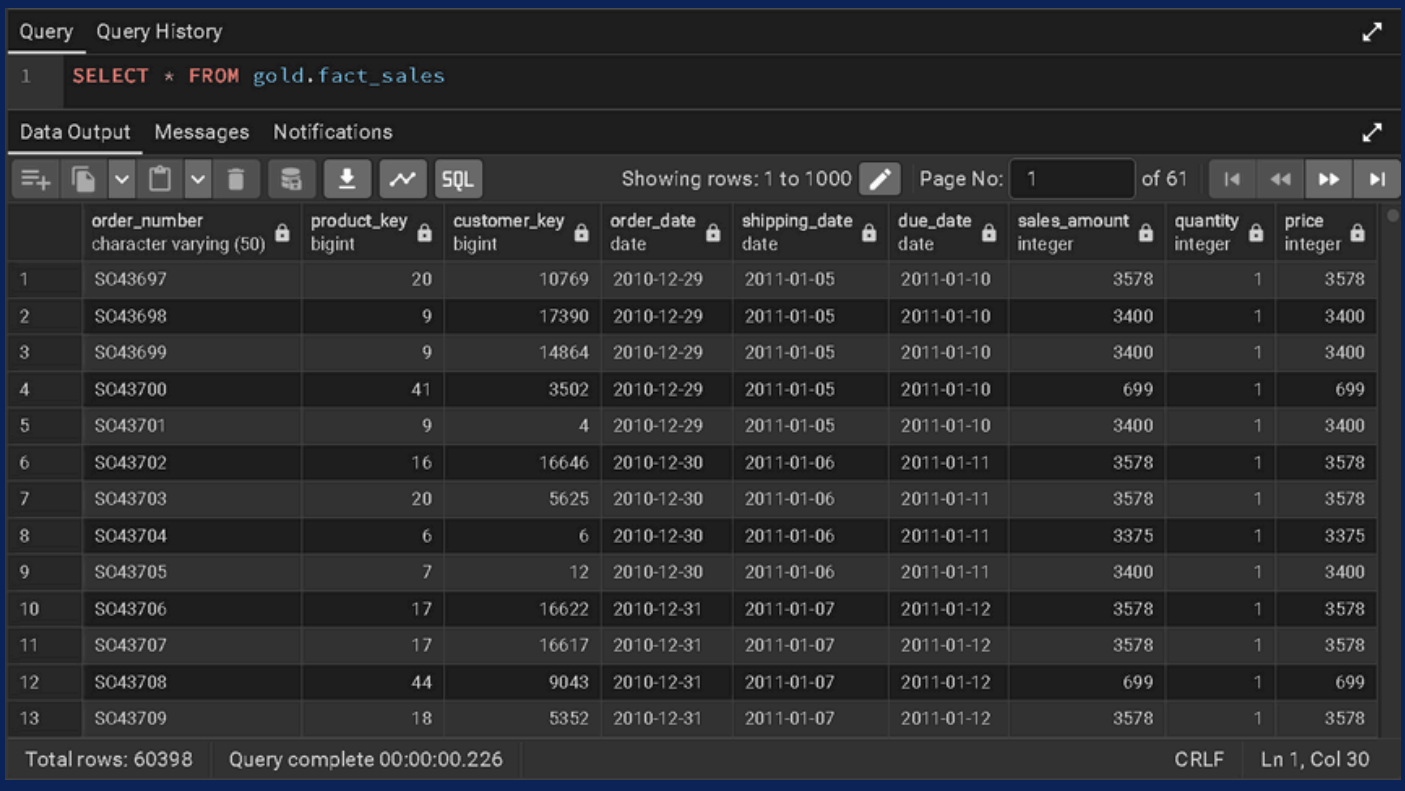
Showing rows: 1 to 295 | Page No: 1 of 1

	product_key bigint	product_id integer	product_number character varying (50)	product_name character varying (50)	category_id character varying (20)	category character varying (50)	subcategory character varying (50)	main
1	1	210	FR-R92B-58	HL Road Frame - Black- 58	CO_RF	Components	Road Frames	Yes
2	2	211	FR-R92R-58	HL Road Frame - Red- 58	CO_RF	Components	Road Frames	Yes
3	3	348	BK-M82B-38	Mountain-100 Black- 38	BI_MB	Bikes	Mountain Bikes	Yes
4	4	349	BK-M82B-42	Mountain-100 Black- 42	BI_MB	Bikes	Mountain Bikes	Yes
5	5	350	BK-M82B-44	Mountain-100 Black- 44	BI_MB	Bikes	Mountain Bikes	Yes
6	6	351	BK-M82B-48	Mountain-100 Black- 48	BI_MB	Bikes	Mountain Bikes	Yes
7	7	344	BK-M82S-38	Mountain-100 Silver- 38	BI_MB	Bikes	Mountain Bikes	Yes
8	8	345	BK-M82S-42	Mountain-100 Silver- 42	BI_MB	Bikes	Mountain Bikes	Yes
9	9	346	BK-M82S-44	Mountain-100 Silver- 44	BI_MB	Bikes	Mountain Bikes	Yes
10	10	347	BK-M82S-48	Mountain-100 Silver- 48	BI_MB	Bikes	Mountain Bikes	Yes
11	11	317	BK-R68R-44	Road-450 Red- 44	BI_RB	Bikes	Road Bikes	Yes
12	12	318	BK-R68R-48	Road-450 Red- 48	BI_RB	Bikes	Road Bikes	Yes

Total rows: 295 | Query complete 00:00:00.135 | CRLF | Ln 1, Col 32

3. gold.fact_sales

Captures transactional sales records linked to customers and products, with measures like quantity sold, total sales value, and sales dates.



Query | Query History

```
SELECT * FROM gold.fact_sales
```

Data Output | Messages | Notifications

Showing rows: 1 to 1000 | Page No: 1 of 61

	order_number character varying (50)	product_key bigint	customer_key bigint	order_date date	shipping_date date	due_date date	sales_amount integer	quantity integer	price integer
1	SO43697	20	10769	2010-12-29	2011-01-05	2011-01-10	3578	1	3578
2	SO43698	9	17390	2010-12-29	2011-01-05	2011-01-10	3400	1	3400
3	SO43699	9	14864	2010-12-29	2011-01-05	2011-01-10	3400	1	3400
4	SO43700	41	3502	2010-12-29	2011-01-05	2011-01-10	699	1	699
5	SO43701	9	4	2010-12-29	2011-01-05	2011-01-10	3400	1	3400
6	SO43702	16	16646	2010-12-30	2011-01-06	2011-01-11	3578	1	3578
7	SO43703	20	5625	2010-12-30	2011-01-06	2011-01-11	3578	1	3578
8	SO43704	6	6	2010-12-30	2011-01-06	2011-01-11	3375	1	3375
9	SO43705	7	12	2010-12-30	2011-01-06	2011-01-11	3400	1	3400
10	SO43706	17	16622	2010-12-31	2011-01-07	2011-01-12	3578	1	3578
11	SO43707	17	16617	2010-12-31	2011-01-07	2011-01-12	3578	1	3578
12	SO43708	44	9043	2010-12-31	2011-01-07	2011-01-12	699	1	699
13	SO43709	18	5352	2010-12-31	2011-01-07	2011-01-12	3578	1	3578

Total rows: 60398 | Query complete 00:00:00.226 | CRLF | Ln 1, Col 30

Data Catalog for Gold Layer

The Gold Layer is the business-level data representation, structured to support analytical and reporting use cases. It consists of dimension tables and fact tables for specific business metrics.

1. gold.dim_customers

- Purpose: Stores customer details enriched with demographic and geographic data.
- Columns:

Column Name	Data Type	Description
customer_key	INT	Surrogate key uniquely identifying each customer record in the dimension table.
customer_id	INT	Unique numerical identifier assigned to each customer.
customer_number	VARCHAR(50)	Alphanumeric identifier representing the customer, used for tracking and referencing.
first_name	VARCHAR(50)	The customer's first name, as recorded in the system.
last_name	VARCHAR(50)	The customer's last name or family name.
country	VARCHAR(50)	The country of residence for the customer (e.g., 'Australia').
marital_status	VARCHAR(50)	The marital status of the customer (e.g., 'Married', 'Single').
gender	VARCHAR(50)	The gender of the customer (e.g., 'Male', 'Female', 'n/a').
birthdate	DATE	The date of birth of the customer, formatted as YYYY-MM-DD (e.g., 1971-10-06).
create_date	DATE	The date and time when the customer record was created in the system

2. gold.dim_products

- Purpose: Provides information about the products and their attributes.
- Columns:

Column Name	Data Type	Description
product_key	INT	Surrogate key uniquely identifying each product record in the product dimension table.
product_id	INT	A unique identifier assigned to the product for internal tracking and referencing.
product_number	VARCHAR(50)	A structured alphanumeric code representing the product, often used for categorization or inventory.
product_name	VARCHAR(50)	Descriptive name of the product, including key details such as type, color, and size.
category_id	VARCHAR(50)	A unique identifier for the product's category, linking to its high-level classification.
category	VARCHAR(50)	The broader classification of the product (e.g., Bikes, Components) to group related items.
subcategory	VARCHAR(50)	A more detailed classification of the product within the category, such as product type.
maintenance_required	VARCHAR(50)	Indicates whether the product requires maintenance (e.g., 'Yes', 'No').
cost	INT	The cost or base price of the product, measured in monetary units.
product_line	VARCHAR(50)	The specific product line or series to which the product belongs (e.g., Road, Mountain).
start_date	DATE	The date when the product became available for sale or use, stored in

3. gold.fact_sales

- Purpose: Stores transactional sales data for analytical purposes.
- Columns:

Column Name	Data Type	Description
order_number	VARCHAR(50)	A unique alphanumeric identifier for each sales order (e.g., 'SO54496').
product_key	INT	Surrogate key linking the order to the product dimension table.
customer_key	INT	Surrogate key linking the order to the customer dimension table.
order_date	DATE	The date when the order was placed.
shipping_date	DATE	The date when the order was shipped to the customer.
due_date	DATE	The date when the order payment was due.
sales_amount	INT	The total monetary value of the sale for the line item, in whole currency units (e.g., 25).
quantity	INT	The number of units of the product ordered for the line item (e.g., 1).
price	INT	The price per unit of the product for the line item, in whole currency units (e.g., 25).

Naming Conventions

General Principles

- Naming Conventions: Used snake_case, with lowercase letters and underscores (_) to separate words.
- Language: Used English for all names.
- Avoid Reserved Words: Did not use SQL reserved words as object names.

Table Naming Conventions

Bronze Rules

- All names must start with the source system name, and table names must match their original names without renaming.
- <sourcesystem>_<entity>
  - <sourcesystem>: Name of the source system (e.g., crm, erp).
  - <entity>: Exact table name from the source system.
  - Example: crm_customer_info → Customer information from the CRM system.

Silver Rules

- All names must start with the source system name, and table names must match their original names without renaming.
- <sourcesystem>_<entity>
  - <sourcesystem>: Name of the source system (e.g., crm, erp).
  - <entity>: Exact table name from the source system.
  - Example: crm_customer_info → Customer information from the CRM system.

Gold Rules

- All names must use meaningful, business-aligned names for tables, starting with the category prefix.
- <category>_<entity>
  - <category>: Describes the role of the table, such as dim (dimension) or fact (fact table).
  - <entity>: Descriptive name of the table, aligned with the business domain (e.g., customers, products, sales).
  - Examples:
    - dim_customers → Dimension table for customer data.
    - fact_sales → Fact table containing sales transactions.

Glossary of Category Patterns

Pattern	Meaning	Example(s)
dim_	Dimension table	dim_customer, dim_product
fact_	Fact table	fact_sales
report_	Report table	report_customers, report_sales_monthly

Column Naming Conventions

Surrogate Keys

- All primary keys in dimension tables must use the suffix _key.
- <table_name>_key
  - <table_name>: Refers to the name of the table or entity the key belongs to.
  - _key: A suffix indicating that this column is a surrogate key.
  - Example: customer_key → Surrogate key in the dim_customers table.

Technical Columns

- All technical columns must start with the prefix dwh_, followed by a descriptive name indicating the column's purpose.
- dwh_<column_name>
  - dwh: Prefix exclusively for system-generated metadata.
  - <column_name>: Descriptive name indicating the column's purpose.
  - Example: dwh_load_date → System-generated column used to store the date when the record was loaded.

Stored Procedure

- All stored procedures used for loading data must follow the naming pattern:
- load_<layer>.
  - <layer>: Represents the layer being loaded, such as bronze, silver, or gold.
  - Example:
    - load_bronze → Stored procedure for loading data into the Bronze layer.
    - load_silver → Stored procedure for loading data into the Silver layer.

Features of this Project

- End-to-End Data Pipeline: Covers the complete lifecycle from raw data ingestion to creating structured, analysis-ready data.
- Data Warehouse Development: Built using PostgreSQL with a strong focus on performance, scalability, and clarity.
- ETL Process Implementation: Manual SQL scripting to extract, transform, and load data from CRM and ERP sources.
- Medallion Architecture: Structured using Bronze (raw), Silver (cleaned), and Gold (business-ready) data layers.
- Star Schema Design: Incorporates dimension and fact tables to support fast, analytical queries.
- Business-Ready Tables: Gold layer includes tables like dim_customers, dim_products, and fact_sales for analysis, reporting and decision-making.
- Source System Integration: Combines datasets from multiple systems (CRM and ERP) into a unified model.
- Clear Naming Conventions: Consistent and meaningful table and column names for easier understanding and maintenance.
- Visual Documentation: Includes diagrams for data flow, data architecture, data integration, and data model.

Acknowledgements

This project is inspired by the excellent guided tutorial created by **Baraa Khatib Salkini**. Special thanks to him for sharing his knowledge and providing a comprehensive walkthrough of building a data warehouse from scratch.