Superstore DataFlow: End-to-End ETL Project Using PostgreSQL & PowerBI

Project by: Rohan Verma
GitHub Repository: [Link to your project repo]

Abstract

This project demonstrates a complete data engineering pipeline built on PostgreSQL to ingest, transform, validate, and automate the processing of Superstore sales data. The pipeline enables real-time updates of sales data from raw to aggregated forms and serves as a robust foundation for supply chain analytics, sales forecasting, and inventory optimization. By leveraging Python (Pandas, SQLAlchemy), PostgreSQL, and Windows Task Scheduler, we simulate a production-ready ETL workflow applicable to real-world business intelligence environments.

---

1. Introduction

Efficient management of sales and inventory data is crucial in domains like retail, supply chain, and operations. This project showcases a working ETL (Extract, Transform, Load) pipeline using cleaned Superstore sales data. The ETL pipeline is developed to:

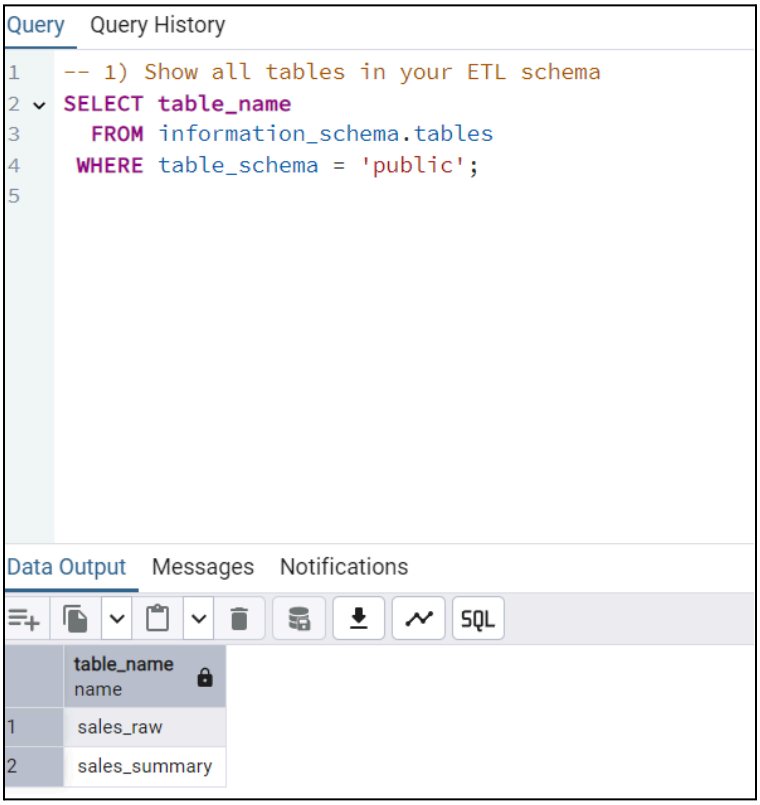
- Ingest raw CSV data into PostgreSQL.
- Transform it into a monthly summary.
- Validate data integrity with quality checks.
- Automate daily updates using Windows Task Scheduler.

Such a pipeline can aid business analysts, supply chain managers, and data teams to maintain clean, real-time dashboards and make timely decisions.

---

2. Database & Schema Setup

We created a PostgreSQL database named **etl_demo** under the default server (PostgreSQL 17). A user role **etl_user1** was granted permission to manage tables within the default schema **public.**



- **Fig 1:** pgAdmin query showing tables created under the **public** schema.

```
SELECT table_name
FROM information_schema.tables
WHERE table_schema = 'public';
```

The output confirmed the existence of our key tables:

- **sales_raw (raw ingestion layer)**
- **sales_summary (aggregated layer)**

---

3. Table Structure

3.1 sales_raw

This table mirrors the structure of the Superstore dataset with 21 columns.

	column_name name	data_type character varying
1	row_id	integer
2	order_date	date
3	ship_date	date
4	sales	numeric
5	quantity	integer
6	discount	numeric
7	profit	numeric
8	segment	text
9	country	text
10	city	text
11	state	text
12	postal_code	text
13	region	text
14	product_id	text
15	category	text
16	order_id	text
17	sub_category	text
18	product_name	text
19	ship_mode	text
20	customer_id	text
21	customer_name	text

- **Fig 2:** Column metadata extracted using:

```
SELECT column_name, data_type
FROM information_schema.columns
WHERE table_name = 'sales_raw';
```

3.2 sales_summary

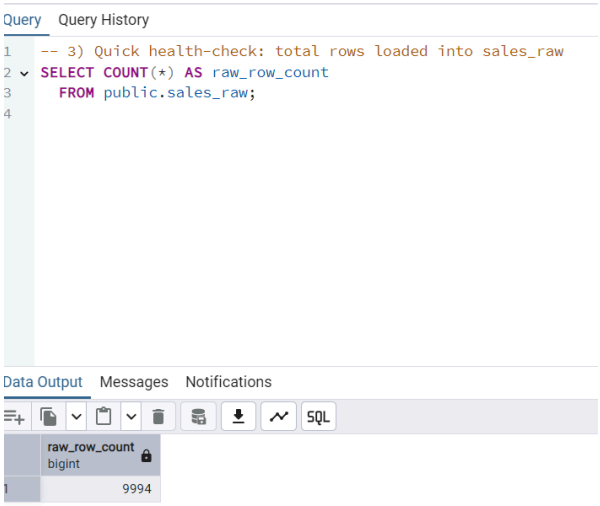
Aggregates data per **(year, month)** with calculated fields like total_sales, avg_discount, and **total_profit**.

---

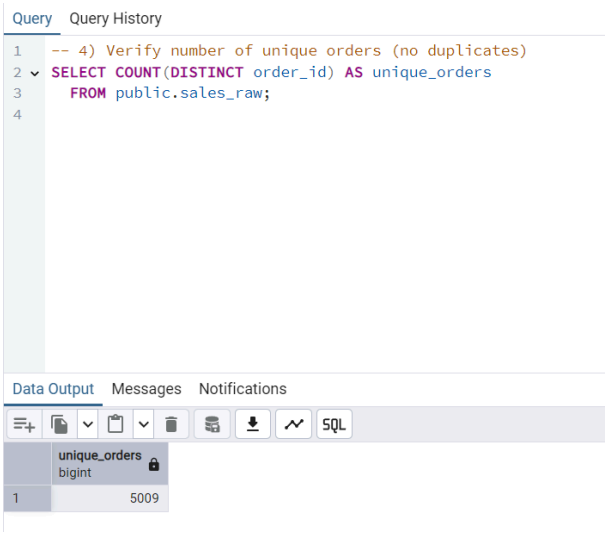
4. Data Ingestion using load_raw.py

The load_raw.py script handles the **Extract** and **Load** stages:

- Reads the cleaned dataset from disk.
- Truncates the existing sales_raw table.
- Loads all rows using df.to_sql(..., chunksize=1000).



- **Fig 3:** Output of SELECT COUNT(*) returned 9994 rows:



- **Fig 4:** Validation of unique orders:

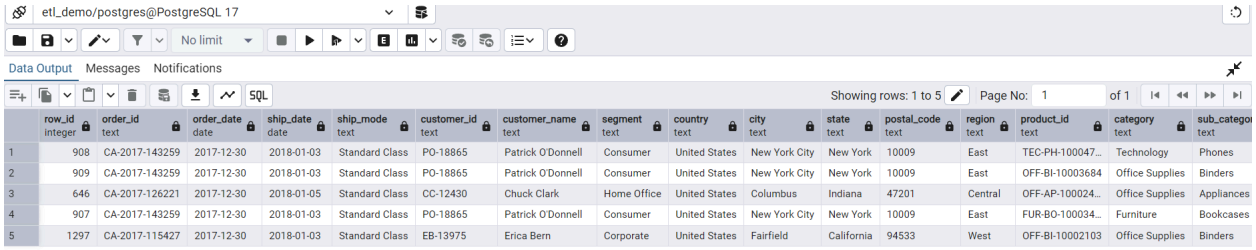
This confirms ingestion without duplication.

---

5. Transformation Logic with transform_summary.py

This script performs the **Transform** step. It:

- Reads sales_raw into a Pandas DataFrame.
- Extracts year and month from order_date.
- Groups and aggregates metrics.
- Replaces data in sales_summary table.
-



	row_id integer	order_id text	order_date date	ship_date date	ship_mode text	customer_id text	customer_name text	segment text	country text	city text	state text	postal_code text	region text	product_id text	category text	sub_categor text
1	908	CA-2017-143259	2017-12-30	2018-01-03	Standard Class	PO-18865	Patrick O'Donnell	Consumer	United States	New York City	New York	10009	East	TEC-PH-100047…	Technology	Phones
2	909	CA-2017-143259	2017-12-30	2018-01-03	Standard Class	PO-18865	Patrick O'Donnell	Consumer	United States	New York City	New York	10009	East	OFF-BI-10003684	Office Supplies	Binders
3	646	CA-2017-126221	2017-12-30	2018-01-05	Standard Class	CC-12430	Chuck Clark	Home Office	United States	Columbus	Indiana	47201	Central	OFF-AP-100024…	Office Supplies	Appliances
4	907	CA-2017-143259	2017-12-30	2018-01-03	Standard Class	PO-18865	Patrick O'Donnell	Consumer	United States	New York City	New York	10009	East	FUR-BO-100034…	Furniture	Bookcases
5	1297	CA-2017-115427	2017-12-30	2018-01-03	Standard Class	EB-13975	Erica Bern	Corporate	United States	Fairfield	California	94533	West	OFF-BI-10002103	Office Supplies	Binders

- **Fig 5:** Preview of the **sales_summary** table in pgAdmin.

---

6. Data Quality Checks: quality_checks.py

This script ensures:

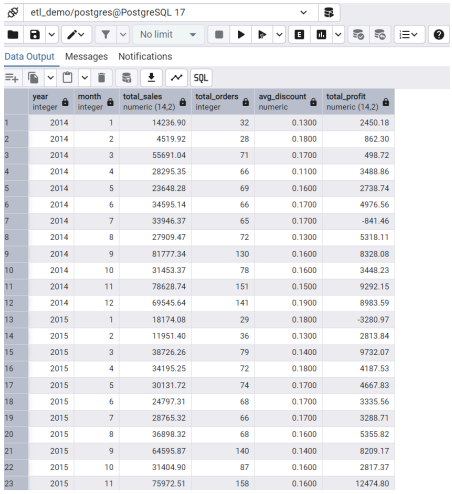
- No null or duplicate **order_ids**.
- **sales** ≥ 0.
- 0 ≤ **discount** ≤ 1.
- **sales_summary** is not empty.

Upon success, it prints: **All data quality checks passed.**

---

7. Advanced SQL Queries

We wrote SQL queries to verify integrity and extract insights:



etl_demo/postgres@PostgreSQL 17

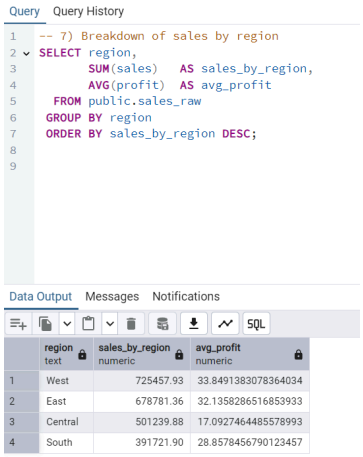
No limit

Data Output | Messages | Notifications

SQL

	year integer	month integer	total_sales numeric (14,2)	total_orders integer	avg_discount numeric	total_profit numeric (14,2)
1	2014	1	14236.90	32	0.1300	2450.18
2	2014	2	4519.92	28	0.1800	862.30
3	2014	3	55691.04	71	0.1700	498.72
4	2014	4	28295.35	66	0.1100	3488.86
5	2014	5	23648.28	69	0.1600	2738.74
6	2014	6	34595.14	66	0.1700	4976.56
7	2014	7	33946.37	65	0.1700	-841.46
8	2014	8	27909.47	72	0.1300	5318.11
9	2014	9	81777.34	130	0.1600	8328.08
10	2014	10	31453.37	78	0.1600	3448.23
11	2014	11	78628.74	151	0.1500	9292.15
12	2014	12	69545.64	141	0.1900	8983.59
13	2015	1	18174.08	29	0.1800	-3280.97
14	2015	2	11951.40	36	0.1300	2813.84
15	2015	3	38726.26	79	0.1400	9732.07
16	2015	4	34195.25	72	0.1800	4187.53
17	2015	5	30131.72	74	0.1700	4667.83
18	2015	6	24797.31	68	0.1700	3335.56
19	2015	7	28765.32	66	0.1700	3288.71
20	2015	8	36898.32	68	0.1600	5355.82
21	2015	9	64595.87	140	0.1400	8209.17
22	2015	10	31404.90	87	0.1600	2817.37
23	2015	11	75972.51	158	0.1600	12474.80

- **Fig 6:** Total and average profit by region:



Query | Query History

```
1  -- 7) Breakdown of sales by region
2  SELECT region,
3         SUM(sales)   AS sales_by_region,
4         AVG(profit)  AS avg_profit
5    FROM public.sales_raw
6   GROUP BY region
7   ORDER BY sales_by_region DESC;
8
9
```

Data Output | Messages | Notifications

SQL

	region text	sales_by_region numeric	avg_profit numeric
1	West	725457.93	33.8491383078364034
2	East	678781.36	32.1358286516853933
3	Central	501239.88	17.0927464485578993
4	South	391721.90	28.8578456790123457

- **Fig 7:** Sales performance by shipping mode:

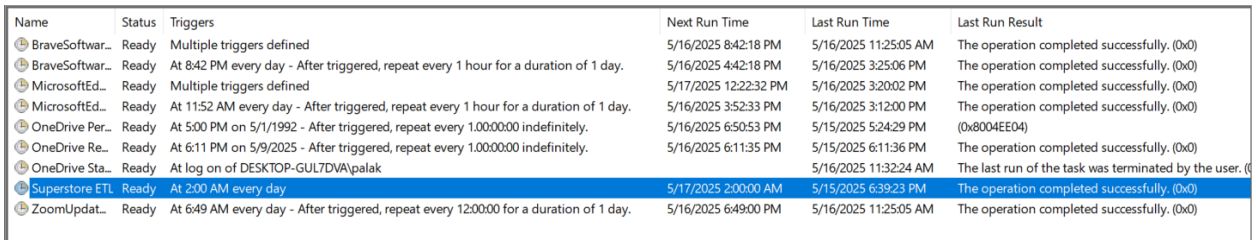
These analyses could power dashboards for marketing, supply chain and logistics teams.

---

8. Automation using Windows Task Scheduler

To ensure the ETL runs daily without manual intervention, we:

- Created a task called **Superstore ETL**.
- Configure it to run **python run_etl.py** at 2:00 AM every day.



Name	Status	Triggers	Next Run Time	Last Run Time	Last Run Result
BraveSoftwar...	Ready	Multiple triggers defined	5/16/2025 8:42:18 PM	5/16/2025 11:25:05 AM	The operation completed successfully. (0x0)
BraveSoftwar...	Ready	At 8:42 PM every day - After triggered, repeat every 1 hour for a duration of 1 day.	5/16/2025 4:42:18 PM	5/16/2025 3:25:06 PM	The operation completed successfully. (0x0)
MicrosoftEd...	Ready	Multiple triggers defined	5/17/2025 12:22:32 PM	5/16/2025 3:20:02 PM	The operation completed successfully. (0x0)
MicrosoftEd...	Ready	At 11:52 AM every day - After triggered, repeat every 1 hour for a duration of 1 day.	5/16/2025 3:52:33 PM	5/16/2025 3:12:00 PM	The operation completed successfully. (0x0)
OneDrive Per...	Ready	At 5:00 PM on 5/1/1992 - After triggered, repeat every 1.00:00:00 indefinitely.	5/16/2025 6:50:53 PM	5/15/2025 5:24:29 PM	(0x8004EE04)
OneDrive Re...	Ready	At 6:11 PM on 5/9/2025 - After triggered, repeat every 1.00:00:00 indefinitely.	5/16/2025 6:11:35 PM	5/15/2025 6:11:36 PM	The operation completed successfully. (0x0)
OneDrive Sta...	Ready	At log on of DESKTOP-GUL7DVA\palak		5/16/2025 11:32:24 AM	The last run of the task was terminated by the user. (
Superstore ETL	Ready	At 2:00 AM every day	5/17/2025 2:00:00 AM	5/15/2025 6:39:23 PM	The operation completed successfully. (0x0)
ZoomUpdat...	Ready	At 6:49 AM every day - After triggered, repeat every 12:00:00 for a duration of 1 day.	5/16/2025 6:49:00 PM	5/16/2025 11:25:05 AM	The operation completed successfully. (0x0)

- **Fig 8:** Task Scheduler confirmation screenshot showing task status and result.

---

10. Project Structure

```
week1/
├── data/
│   └── superstore_clean.csv
├── scripts/
│   ├── load_raw.py
│   ├── transform_summary.py
│   ├── quality_checks.py
│   ├── notify.py
│   └── run_etl.py
├── screenshots/
└── README.md
```

---

9. Dashboard Visualization with Power BI

To complete the end-to-end pipeline, we built an interactive Power BI dashboard on top of the aggregated **sales_summary** table. This serves business users—such as supply chain managers or retail analysts—by delivering key insights at a glance.

9.1 Connecting Power BI to PostgreSQL

1. **Get Data → PostgreSQL database**
2. Server: **127.0.0.1** Database: **etl_demo** Authentication: **etl_user1 / 3******@**
3. Loaded both the tables.

9.2 Key Visuals

Below are the four principal visuals we built in Power BI—each tied directly to our **sales_summary** and **sales_raw** tables—along with the business insights they enable.

---

1. Monthly Sales Trend

Type: Line chart

- **X-axis:** Month (across all years)
- **Y-axis:** Sum of **total_sales**
- **Insight:** Shows how overall revenue evolves month-over-month. Seasonal peaks (e.g., holidays) and troughs (e.g., early year) become immediately apparent, helping supply-chain planners anticipate inventory needs.

---

2. Regional Performance Comparison

Type: Clustered bar chart

- **Axis (categorical):** Region (**Central, East, South, West**)
- **Values (series):**

    - Sum of **total_sales**
    - Sum of **total_orders**
    - Sum of **total_profit**

- **Insight:** Enables executives to compare how each region contributes to revenue, order volume, and profitability in a single view. Regions with high sales but low profit can be flagged for discount strategy review.

---

3. Profit Heatmap by State

Type: Filled map

- **Location:** State
- **Color intensity:** Sum of `total_profit`
- **Insight:** Visualizes geographic profit distribution, highlighting states where margins are strongest or weakest—critical for optimizing logistics routes and promotional campaigns.

---

4. Category & Discount Distribution

Type: Donut / pie chart with percentage labels

- **Category slices:** Product **Category** (Office Supplies, Furniture, Technology)
- **Slice size:** Count of orders
- **Color shading (inner ring):** Discount brackets (e.g. 0–5%, 5–10%, >10%)
- **Insight:** Quickly shows which product categories dominate order volume and how different discount levels are applied across those categories—helping merchandising teams balance discount depth against sales uplift.

These visuals together form a **dynamic analytics dashboard** that empowers stakeholders to track sales trends, regional performance, profitability hotspots, and discount effectiveness—all updated daily by your automated ETL pipeline.

9.3 Theming & Layout

- Applied a **corporate blue** theme and set a **dark** report page background.
- Added **date** and **region** slicers to enable ad-hoc filtering.
- Arranged visuals on a single 16:9 canvas for executive review.

9.4 Publish & Share

- **Saved** the report as **Superstore_Dashboard.pbit** to include in the GitHub repo.

---

10. Conclusion & Next Steps

This Power BI dashboard completes the Superstore DataFlow project, delivering:

- **Operational Metrics:** Automated ETL keeps data current.
- **Analytical Insights:** Interactive visuals for revenue, profit, and product performance.
- **Business Value:** Enables supply chain and sales teams to monitor KPIs and drive data-led decisions.

Future enhancements could include real-time data streaming, advanced forecasting visuals, or integration with cloud data warehouses like Snowflake.