

# PostgreSQL Extract, Transform Load project

Created a data pipeline that transforms input sales and product datasets into a star schema ready for analysis and querying.

Project Includes the following

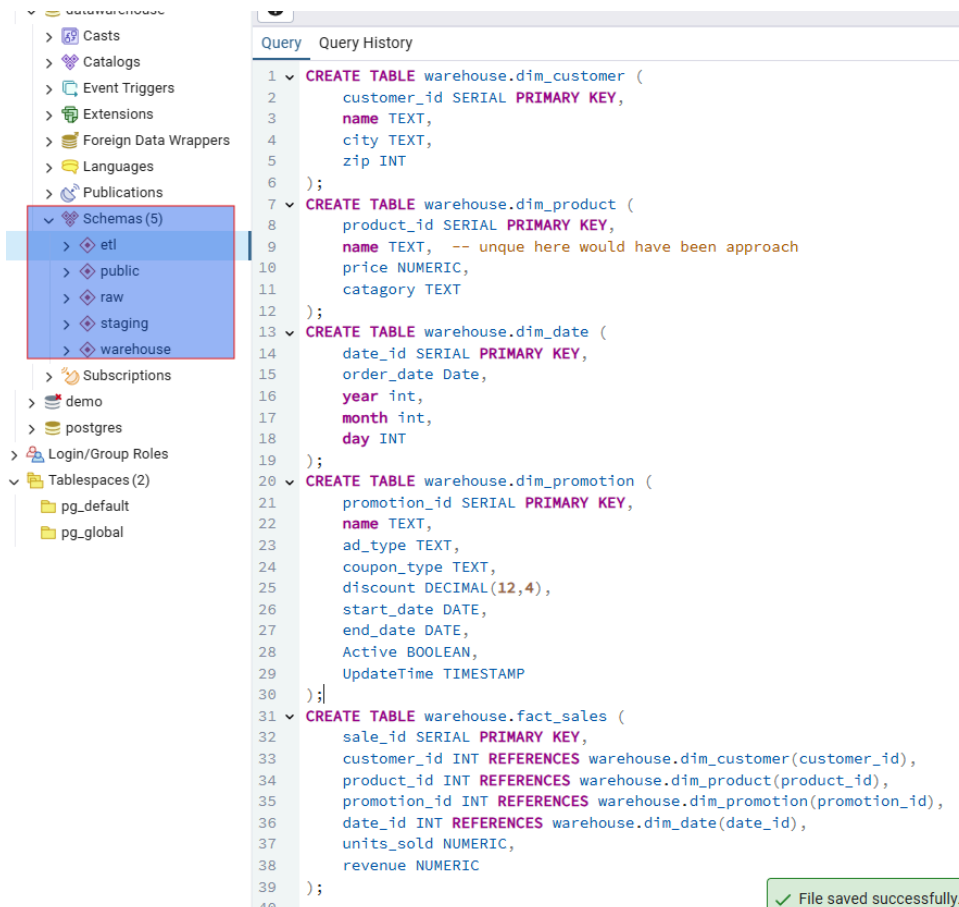
- Raw data loading
- Staging, transforming and validating data
- Warehouse, loading data into star scheme data model.
- ETL scripts, fully automated so can easily convert new raw data into ready to query tables
- Joins

Tools used:

- PostgreSQL Database
- pgAdmin 4
- Vscode (with many extensions)
- SQL
- Python
- Tableau

## Stage 1, create tables and schemas

Schema creation

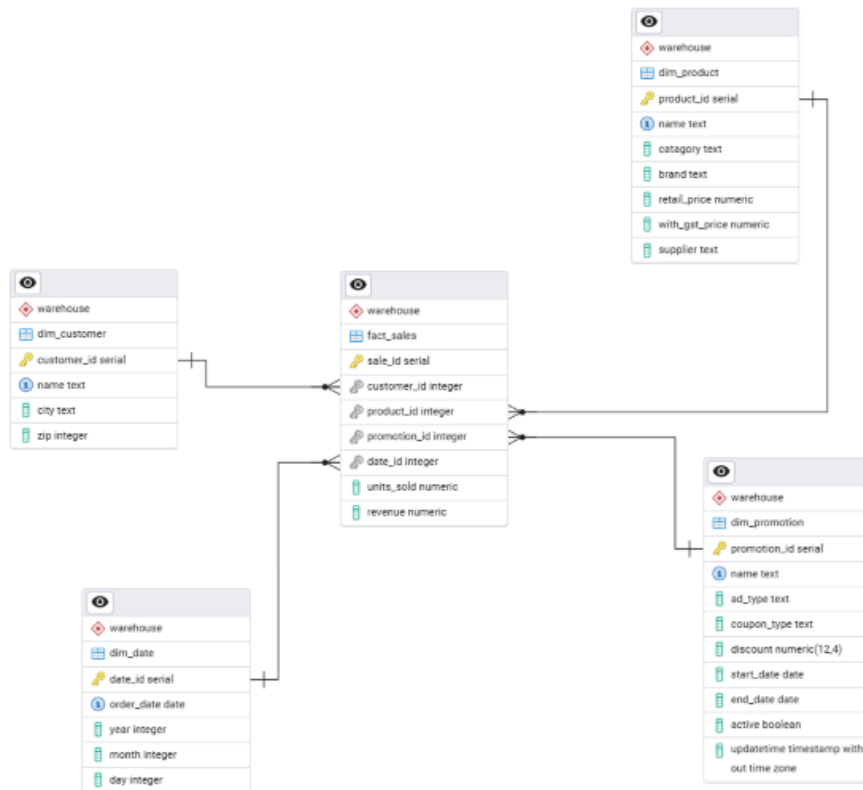


The screenshot displays the pgAdmin 4 interface. On the left, the 'Schemas' pane is expanded, showing a list of schemas: etl, public, raw, staging, and warehouse. The 'warehouse' schema is selected. The main pane shows the SQL query editor with the following code:

```
1 CREATE TABLE warehouse.dim_customer (  
2     customer_id SERIAL PRIMARY KEY,  
3     name TEXT,  
4     city TEXT,  
5     zip INT  
6 );  
7 CREATE TABLE warehouse.dim_product (  
8     product_id SERIAL PRIMARY KEY,  
9     name TEXT, -- unique here would have been approach  
10    price NUMERIC,  
11    catagory TEXT  
12 );  
13 CREATE TABLE warehouse.dim_date (  
14     date_id SERIAL PRIMARY KEY,  
15     order_date Date,  
16     year int,  
17     month int,  
18     day INT  
19 );  
20 CREATE TABLE warehouse.dim_promotion (  
21     promotion_id SERIAL PRIMARY KEY,  
22     name TEXT,  
23     ad_type TEXT,  
24     coupon_type TEXT,  
25     discount DECIMAL(12,4),  
26     start_date DATE,  
27     end_date DATE,  
28     Active BOOLEAN,  
29     UpdateTime TIMESTAMP  
30 );  
31 CREATE TABLE warehouse.fact_sales (  
32     sale_id SERIAL PRIMARY KEY,  
33     customer_id INT REFERENCES warehouse.dim_customer(customer_id),  
34     product_id INT REFERENCES warehouse.dim_product(product_id),  
35     promotion_id INT REFERENCES warehouse.dim_promotion(promotion_id),  
36     date_id INT REFERENCES warehouse.dim_date(date_id),  
37     units_sold NUMERIC,  
38     revenue NUMERIC  
39 );
```

A green notification bar at the bottom right indicates: "File saved successfully."

Entity Relationship Diagram below after creating tables.



## Stage 2, load raw data into tables

Query	Query History	Query	Query History
1		1	
2	CREATE OR REPLACE FUNCTION etl.load_raw_sales()	2	CREATE OR REPLACE FUNCTION etl.load_raw_products()
3	RETURNS void	3	RETURNS void
4	LANGUAGE 'sql'	4	LANGUAGE 'sql'
5	COST 100	5	COST 100
6	VOLATILE PARALLEL UNSAFE	6	VOLATILE PARALLEL UNSAFE
7	AS \$BODY\$	7	AS \$BODY\$
8		8	
9	DROP TABLE IF EXISTS raw.sales_raw;	9	DROP TABLE IF EXISTS raw.products_raw;
10		10	
11	CREATE TABLE raw.sales_raw (	11	CREATE TABLE raw.products_raw (
12	sale_id INT,	12	product_id INT,
13	customer_name TEXT,	13	product_name TEXT,
14	city TEXT,	14	category TEXT,
15	product_name TEXT,	15	brand TEXT,
16	promotion_name TEXT,	16	retail_price NUMERIC,
17	units_sold INT,	17	supplier TEXT
18	price NUMERIC,	18	);
19	order_date DATE	19	
20	);	20	COPY raw.products_raw
21		21	FROM 'C:\projects\data_warehouse_project\data\pro
22	COPY raw.sales_raw	22	DELIMITER ','
23	FROM 'C:\projects\data_warehouse_project\data\sale	23	CSV HEADER;
24	DELIMITER ','	24	
25	CSV HEADER;	25	\$BODY\$;
26		26	ALTER FUNCTION etl.load_raw_products()
27	\$BODY\$;	27	OWNER TO postgres;
28	ALTER FUNCTION etl.load_raw_sales()	28	
29	OWNER TO postgres;		
30			

## Stage 2, Very basic data cleaning and transformation

Script for transforming the raw data in clean staging tables

```
1
2  ✓ CREATE OR REPLACE FUNCTION etl.transform_products()
3      RETURNS void
4      LANGUAGE 'sql'
5      COST 100
6      VOLATILE PARALLEL UNSAFE
7  AS $BODY$
8
9
10
11 DROP TABLE IF EXISTS staging.products_cleaned;
12  ✓ CREATE TABLE staging.products_cleaned AS
13  SELECT DISTINCT ON (product_id)
14      product_id,
15      INITCAP(TRIM(product_name)) as product_name,
16      INITCAP(TRIM(catagory)) as catagory,
17      INITCAP(TRIM(brand)) as brand,
18      retail_price,
19      retail_price * 1.15 AS with_gst_price,
20      INITCAP(supplier) as supplier
21  FROM raw.products_raw
22  WHERE retail_price > 0;
23  AND product_name != '' AND product_name IS NOT NULL;
24
25
26  $BODY$;
27  ✓ ALTER FUNCTION etl.transform_products()
28      OWNER TO postgres;
```

## Stage 3, Script to load warehouse (loading 5 tables in star schema

Script of loading the promotion diminution table and fact table below.

Query	Query History
40	<code>JOIN staging.products_cleaned p ON s.product_name = p.product_name</code>
41	<code>ON CONFLICT (name) DO NOTHING;</code>
42	
43	<code>INSERT INTO warehouse.dim_promotion (name,ad_type,coupon_type,</code>
44	<code>discount,start_date, end_date, active, updatetime)</code>
45	<code>SELECT DISTINCT -- default values</code>
46	<code>promotion_name,</code>
47	<code>'Unknown' AS ad_type,</code>
48	<code>'None' AS coupon_type,</code>
49	<code>0 AS discount,</code>
50	<code>CURRENT_DATE AS start_date,</code>
51	<code>CURRENT_DATE + INTERVAL '3 years' AS end_date,</code>
52	<code>TRUE AS active,</code>
53	<code>CURRENT_TIMESTAMP AS updatetime</code>
54	<code>FROM staging.sales_cleaned</code>
55	<code>WHERE promotion_name IS NOT NULL</code>
56	<code>ON CONFLICT (name) DO NOTHING;</code>
57	
58	<code>INSERT INTO warehouse.fact_sales (customer_id, product_id,</code>
59	<code>promotion_id, date_id, units_sold, revenue)</code>
60	<code>SELECT</code>
61	<code>c.customer_id,</code>
62	<code>pr.product_id,</code>
63	<code>pro.promotion_id,</code>
64	<code>d.date_id,</code>
65	<code>s.units_sold,</code>
66	<code>s.total_revenue AS revenue</code>
67	
68	<code>FROM staging.sales_cleaned s</code>
69	<code>JOIN warehouse.dim_customer c ON s.customer_name = c.name</code>
70	<code>JOIN warehouse.dim_product pr ON s.product_name = pr.name</code>
71	<code>--left join as many sales have may have no promotion.</code>
72	<code>LEFT JOIN warehouse.dim_promotion pro ON s.promotion_name = pro.name</code>
73	<code>JOIN warehouse.dim_date d ON s.order_date = d.order_date;</code>
74	
75	<code>\$BODY\$;</code>
76	<code>ALTER FUNCTION etl.load_warehouse()</code>
77	<code>OWNER TO postgres;</code>
78	

### Stage 3, Create automated pipeline procedure to automate etl process

Created a script etl() that calls functions and different scripts including, load\_raw\_sales(), transform\_sales() and load\_warehouse().

In the image below I reset the raw and staging tables before running fully automated etl with 100,000 rows of data. This effectively works in 17s loading the fact table with clean.

Query

Query History

1

SELECT etl.delete\_sales\_tables();

2

SELECT etl.clear\_warehouse();

3

CALL etl.etl();

4

5

SELECT \* FROM warehouse.fact\_sales

6

ORDER BY revenue DESC

7

LIMIT 10;

Data Output

Messages

Notifications

+

SQL

Showing rows: 1 to 10

Page No: 1 of 1

	sale_id [PK] integer	customer_id integer	product_id integer	promotion_id integer	date_id integer	units_sold numeric	revenue numeric
1	50649	16651	3	4	344	20	19986.60
2	19809	27270	6	3	157	20	19986.20
3	29273	67018	7	5	52	20	19985.60
4	10126	10906	7	2	329	20	19982.20
5	86414	80045	9	2	294	20	19981.00
6	46940	37700	3	2	194	20	19977.00
7	66128	38933	2	4	351	20	19970.0
8	67094	46798	7	4	140	20	19959.60
9	1500	21525	9	2	111	20	19959.20
10	37149	51175	10	4	118	20	19952.0

From Bradley Erskine