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

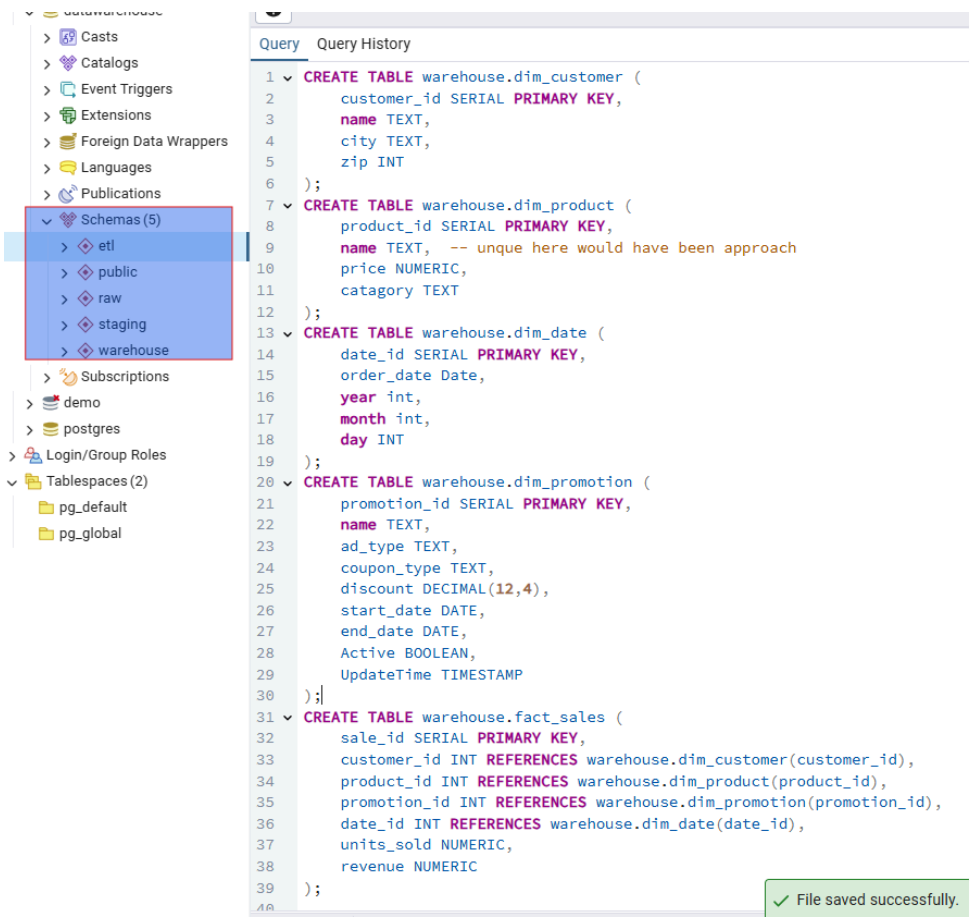
- Raw data loading
- Staging, transforming and validating data
- Warehouse, loading data into star scheme data model.
- ETL scripts, fully automated so can easily convert new raw data into ready to query tables
- Joins

Tools used:

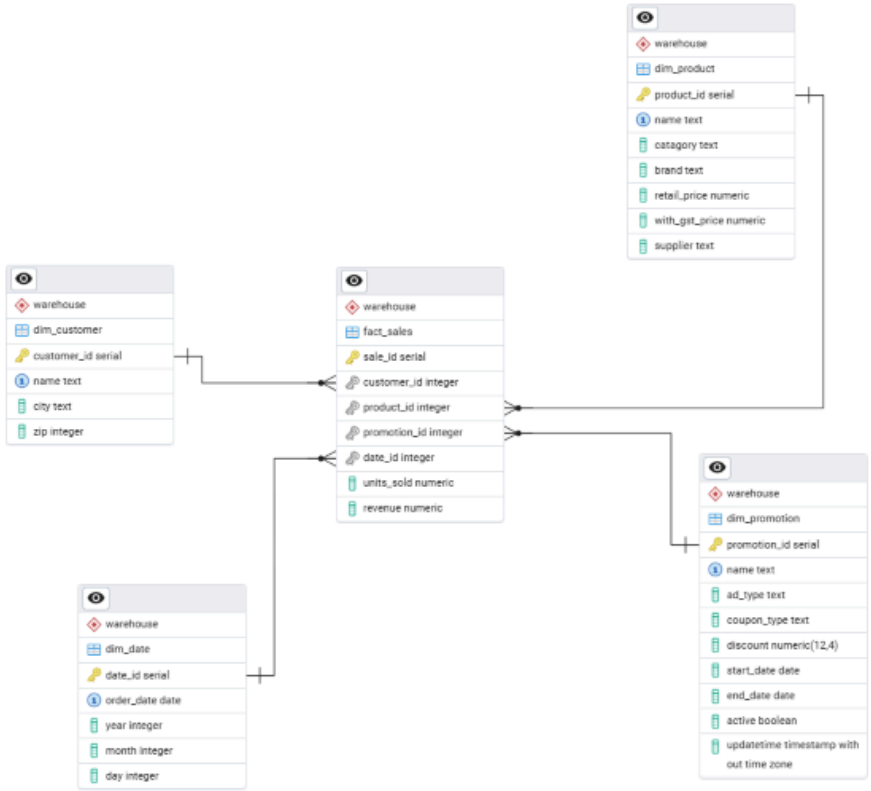
- PostgreSQL Database
- pgAdmin 4
- Vscode (with man y extensions)
- SQL
- Python
- Tableau

Stage 1, create tables and schemas

Schema creation



Entity Relationship Diagram below after creating tables.



Stage 2, load raw data into tables

Query Query History

```
CREATE OR REPLACE FUNCTION etl.load_raw_sales()
    RETURNS void
    LANGUAGE 'sql'
    COST 100
    VOLATILE PARALLEL UNSAFE
AS $BODY$

DROP TABLE IF EXISTS raw.sales_raw;

CREATE TABLE raw.sales_raw (
    sale_id INT,
    customer_name TEXT,
    city TEXT,
    product_name TEXT,
    promotion_name TEXT,
    units_sold INT,
    price NUMERIC,
    order_date DATE
);

COPY raw.sales_raw
FROM 'C:\projects\data_warehouse_project\data\sale
DELIMITER ','
CSV HEADER;

$BODY$;
ALTER FUNCTION etl.load_raw_sales()
    OWNER TO postgres;
```

Query Query History

```
CREATE OR REPLACE FUNCTION etl.load_raw_products(
    RETURNS void
    LANGUAGE 'sql'
    COST 100
    VOLATILE PARALLEL UNSAFE
AS $BODY$

DROP TABLE IF EXISTS raw.products_raw;

CREATE TABLE raw.products_raw (
    product_Id INT,
    product_name TEXT,
    catagory TEXT,
    brand TEXT,
    retail_price NUMERIC,
    supplier TEXT
);

COPY raw.products_raw
FROM 'C:\projects\data_warehouse_project\data\pro
DELIMITER ','
CSV HEADER;

$BODY$;
ALTER FUNCTION etl.load_raw_products()
    OWNER TO postgres;
```

Stage 2, Very basic data cleaning and transformation

Script for transforming the raw data in clean staging tables

```sql

CREATE OR REPLACE FUNCTION etl.transform_products()
    RETURNS void
    LANGUAGE 'sql'
    COST 100
    VOLATILE PARALLEL UNSAFE
AS $BODY$



DROP TABLE IF EXISTS staging.products_cleaned;
CREATE TABLE staging.products_cleaned AS
SELECT DISTINCT ON (product_id)
    product_id,
    INITCAP(TRIM(product_name)) as product_name,
    INITCAP(TRIM(catagory)) as catagory,
    INITCAP(TRIM(brand)) as brand,
    retail_price,
    retail_price * 1.15 AS with_gst_price,
    INITCAP(supplier) as supplier
FROM raw.products_raw
WHERE retail_price > 0;
AND product_name != '' AND product_name IS NOT NULL;


$BODY$;
ALTER FUNCTION etl.transform_products()
    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

```
JOIN staging.products_cleaned p ON s.product_name = p.product_name
ON CONFLICT (name) DO NOTHING;

INSERT INTO warehouse.dim_promotion (name,ad_type,coupon_type,
    discount,start_date, end_date, active, updatetime)
SELECT DISTINCT  -- default values
    promotion_name,
    'Unknown' AS ad_type,
    'None' AS coupon_type,
    0 AS discount,
    CURRENT_DATE AS start_date,
    CURRENT_DATE + INTERVAL '3 years' AS end_date,
    TRUE AS active,
    CURRENT_TIMESTAMP AS updatetime
FROM staging.sales_cleaned
WHERE promotion_name IS NOT NULL
ON CONFLICT (name) DO NOTHING;

INSERT INTO warehouse.fact_sales (customer_id, product_id,
promotion_id, date_id, units_sold, revenue)
SELECT
c.customer_id,
pr.product_id,
pro.promotion_id,
d.date_id,
s.units_sold,
s.total_revenue AS revenue

FROM staging.sales_cleaned s
JOIN warehouse.dim_customer c ON s.customer_name = c.name
JOIN warehouse.dim_product pr ON s.product_name = pr.name
--left join as many sales have may have no promotion.
LEFT JOIN warehouse.dim_promotion pro ON s.promotion_name = pro.name
JOIN warehouse.dim_date d ON s.order_date = d.order_date;

$BODY$;
ALTER FUNCTION etl.load_warehouse()
    OWNER TO postgres;
```

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

```sql
SELECT etl.delete_sales_tables();
SELECT etl.clear_warehouse();
CALL etl.etl();

SELECT * FROM warehouse.fact_sales
ORDER BY revenue DESC
LIMIT 10;
```

Data Output Messages Notifications

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