There are a challenge to select ETL or ELT!

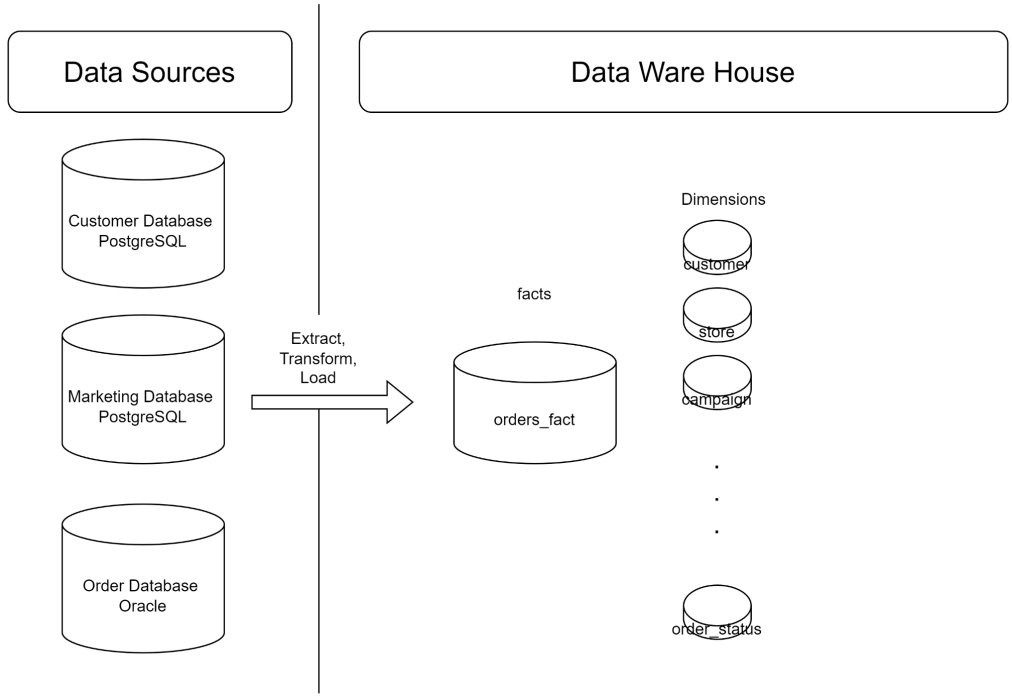
ETL (Extract, Transform, Load):

The ETL process involves extracting data from various source systems, transforming and enriching it according to predefined business rules and requirements, and then loading it into the data warehouse in a structured and optimized format. Extraction involves retrieving data from diverse sources such as PostgreSQL and Oracle databases, ensuring a comprehensive data collection. Transformation incorporates complex logic to clean, standardize, and model the data, preparing it for analytical purposes. Finally, loading the transformed data into the star schema, a well-designed data model for efficient querying and analysis, guarantees the data warehouse is up-to-date and ready for advanced analytics, providing valuable insights for business decision-making.

ELT (Extract, Load, Transform):

The ELT process begins with extracting data from multiple source systems, including PostgreSQL and Oracle databases, ensuring a comprehensive data collection. The extracted data is then loaded into the data warehouse in its raw form. Subsequently, the loaded data is transformed within the data warehouse, utilizing powerful processing capabilities, to meet the required business logic and analytical needs. This approach leverages the capabilities and efficiencies of the data warehouse infrastructure for the transformation phase, making it easier to scale and optimize data processing. The transformed data is organized into a star schema, facilitating efficient querying and analysis, ultimately providing valuable insights to drive informed business decisions.

ETL Loading of the Star Schema



Loading data into a star schema involves the ETL (Extract, Transform, Load) process. Let's break down the loading steps for each dimension and the fact table in the star schema.

Loading Dimensions:

1. Customer Dimension:

- Follow a similar Extract, Transform, Load process for customers data.

2. Items Dimension:

- Follow a similar Extract, Transform, Load process for items data.

3. Campaign Dimension:

- Follow a similar Extract, Transform, Load process for campaign data.

4. Store Dimension:

- Follow a similar Extract, Transform, Load process for store data.

5. Customer Preferences Dimension:

- Follow a similar Extract, Transform, Load process for customer preferences data.

6. Order List Dimension:

- Follow a similar Extract, Transform, Load process for order list data.

7. Order Status Dimension:

- Follow a similar Extract, Transform, Load process for order status data.

Loading the Fact Table:

1. Orders Fact:

- Extract:
    - Extract order-related data such as order_id, customer_id, campaign_id, store_id, order_list_id, item_id, order_status_id, total_gross, etc.
- Transform:
    - Apply transformations like aggregations (e.g., total_gross per order), derive calculated measures (e.g., gross_per_sale), and ensure data consistency.
- Load:
    - Load the transformed data into the Orders Fact table.

Example SQL Loading Statements (Simplified):

- Loading Customer Dimension:
-

```
INSERT INTO customer_dimension (customer_id, full_name, membership_type_id, ...)
SELECT customer_id,first_name +” ”+last_name, membership_type_id, ...
FROM customer;
```

- Loading Orders Fact:

```
INSERT INTO orders_fact (
   order_id,
   customer_id,
   campaign_id,
   store_id,
   order_list_id,
   item_id,
   order_status_id,
   customer_preference_id,
   total_gross,
   since_create_till_order,
   gross_per_sale,
   customer_preferences,
   delivery_time_avg,
   since_create_till_add_preference,
   since_improve_membership,
   campaign_per_store
)
SELECT
   o.order_id,
   cu.customer_id,
   c.campaign_id,
   s.store_id,
   ol.order_list_id,
   i.item_id,
   os.order_status_id,
   cp.customer_preference_id,
   (o.total + (o.tax_total_1+o.tax_total_1) - (o.discount_total_1+o.discount_total_2)) as .total_gross,
   DATEDIFF(day, o.created_date, cu.created_date) as since_create_till_order,
   (o.total_gross / count(*) AS gross_per_sale,
   (o.update_date - o.created_date) as delivery_time_avg,
   o.since_create_till_add_preference,
   (cu.updated_date - cu.created_date) as since_improve_membership,
   sum(c.store_id) as campaign_per_store
FROM
   orders o
JOIN
   customer_dimension c ON o.customer_id = c.customer_id
```

```
JOIN
    campaign_dimension cmp ON o.campaign_id = cmp.campaign_id
JOIN
    store_dimension s ON o.store_id = s.store_id
JOIN
    order_list_dimension ol ON o.order_list_id = ol.order_list_id
JOIN
    item_dimension i ON o.item_id = i.item_id
JOIN
    order_status_dimension os ON o.order_status_id = os.order_status_id
JOIN
    customer_preferences_dimension cp ON o.customer_preference_id = cp.customer_preferences_id;
```