

Assignment

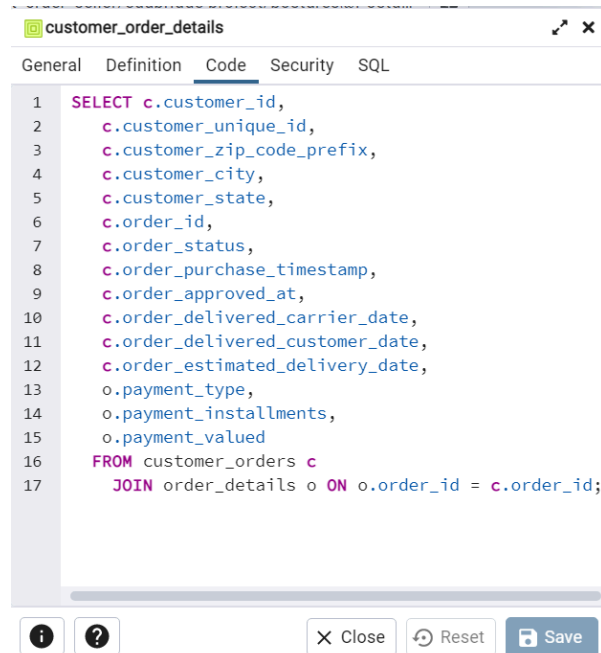
I have taken the Brazilian e-commerce dataset. The provided datasets contain information related to various aspects of an e-commerce site. Here is a description of each dataset:

1. `order.csv`: This dataset, named "olist_orders_dataset.csv," contains information about the orders made on the e-commerce site. It likely includes details such as order ID, customer ID, order status, purchase timestamp, and other relevant order-related information.
2. `customer.csv`: This dataset, named "olist_customers_dataset.csv," contains information about the customers who have made purchases on the e-commerce site. It likely includes customer ID, customer name, customer location, and other related customer information.
3. `payment.csv`: This dataset, named "olist_order_payments_dataset.csv," contains information about the payments made for the orders. It likely includes order ID, payment ID, payment type, payment value, and other relevant information.
4. `product.csv`: This dataset, named "olist_products_dataset.csv," contains information about the products available for sale on the e-commerce site. It likely includes product ID, product name, product category, product price, and other relevant product-related information.
5. `geo.csv`: This dataset, named "olist_geolocation_dataset.csv," contains geolocation information related to Brazilian zip codes. It likely includes information such as zip code, latitude, longitude, and other relevant geographic details.
6. `sellers.csv`: This dataset, named "olist_sellers_dataset.csv," contains information about the sellers associated with the e-commerce platform. It likely includes seller ID, name, location, and other relevant seller-related information.

Each of these datasets provides data from different perspectives of the e-commerce platform, including orders, customers, payments, products, geolocation, and sellers. These datasets can be used together to gain insights into the sales performance, customer behavior, product analysis, payment patterns, and geographic distribution of the e-commerce site's operations.

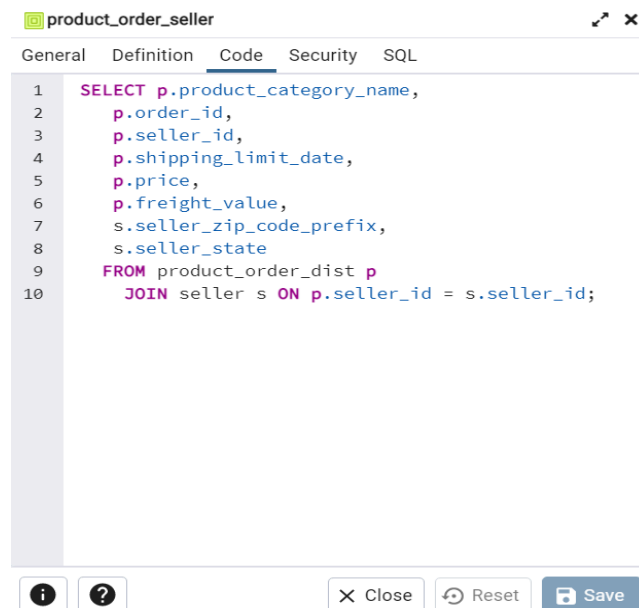
Views:

There were 6 datasets so I have used the view to join the tables and store is locally. Below the image are the datasets customer, orders, and payment were joined together and termed as customer_order_details_dist.



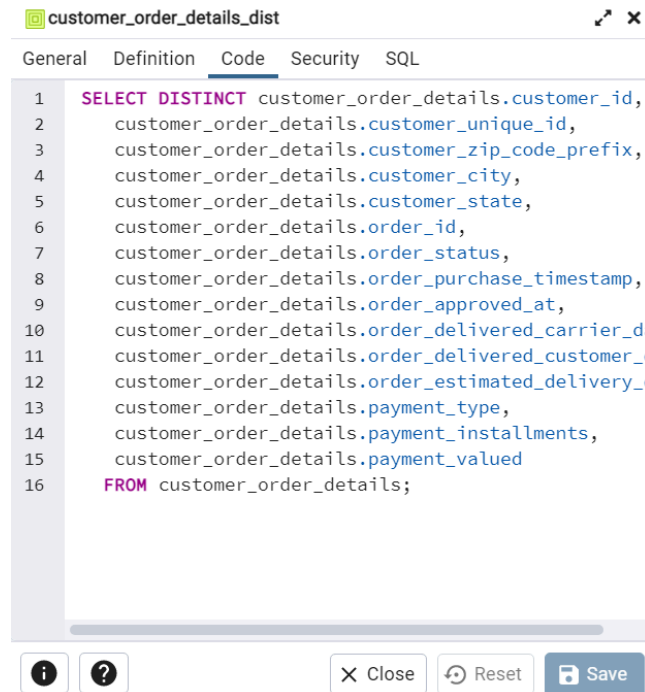
```
customer_order_details
General Definition Code Security SQL
1 SELECT c.customer_id,
2       c.customer_unique_id,
3       c.customer_zip_code_prefix,
4       c.customer_city,
5       c.customer_state,
6       c.order_id,
7       c.order_status,
8       c.order_purchase_timestamp,
9       c.order_approved_at,
10      c.order_delivered_carrier_date,
11      c.order_delivered_customer_date,
12      c.order_estimated_delivery_date,
13      o.payment_type,
14      o.payment_installments,
15      o.payment_valued
16 FROM customer_orders c
17      JOIN order_details o ON o.order_id = c.order_id;
```

order_item, products, and seller were joined together and termed as product_order_seller.



```
product_order_seller
General Definition Code Security SQL
1 SELECT p.product_category_name,
2       p.order_id,
3       p.seller_id,
4       p.shipping_limit_date,
5       p.price,
6       p.freight_value,
7       s.seller_zip_code_prefix,
8       s.seller_state
9 FROM product_order_dist p
10      JOIN seller s ON p.seller_id = s.seller_id;
```

During the joining of the tables repetition of the same records was happening, to overcome this issue, distinct records were used.



```
1 SELECT DISTINCT customer_order_details.customer_id,
2   customer_order_details.customer_unique_id,
3   customer_order_details.customer_zip_code_prefix,
4   customer_order_details.customer_city,
5   customer_order_details.customer_state,
6   customer_order_details.order_id,
7   customer_order_details.order_status,
8   customer_order_details.order_purchase_timestamp,
9   customer_order_details.order_approved_at,
10  customer_order_details.order_delivered_carrier_da
11  customer_order_details.order_delivered_customer_d
12  customer_order_details.order_estimated_delivery_d
13  customer_order_details.payment_type,
14  customer_order_details.payment_installments,
15  customer_order_details.payment_valued
16 FROM customer_order_details;
```



Using this process, records that are duplicated a few times were removed by doing this we can analyze the data accurately by ignoring outliers.

Questions:

1. Find the order placed by product category-wise

```
select product_category_name,
count(distinct order_id) as orders_placed
from product_order_seller
group by 1
order by 2 desc
limit 10
```

Output:

	product_category_name 	orders_placed 
	text	bigint
1	cama_mesa_banho	9417
2	beleza_saude	8836
3	esporte_lazer	7720
4	informatica_acessorios	6689
5	moveis_decoracao	6449
6	utilidades_domesticas	5884
7	relorios_presentes	5624
8	telefonica	4199
9	automotivo	3897
10	brinquedos	3886

2. Which payment type was used by most of the users and what is the total amount spent in each payment type?

```
select payment_type,  
count(distinct customer_id),  
round(sum(payment_valued)::numeric,2) as amount_paid  
from customer_order_details_dist  
group by 1  
order by 2 desc
```

Output:

	payment_type text	count bigint	amount_paid numeric
1	credit_card	76505	12540058.04
2	boleto	19784	2869361.27
3	voucher	3866	363793.38
4	debit_card	1528	217989.79
5	not_defined	3	0.00

3. What is the date range of orders made by customers?

```
select min(order_purchase_timestamp),  
max(order_purchase_timestamp)  
from customer_order_details_dist
```

Output:

	min timestamp without time zone	max timestamp without time zone
1	2016-09-04 21:15:19	2018-10-17 17:30:18

4. What are weightage of order status?

```
select order_status,  
count(distinct customer_id) as customer_count  
from customer_order_details_dist  
group by 1  
order by 2 desc
```

Output:

	order_status text	customer_count bigint
1	delivered	96477
2	shipped	1107
3	canceled	625
4	unavailable	609
5	invoiced	314
6	processing	301
7	created	5
8	approved	2

5. Check which product was canceled many times and from which seller.

```
select p.product_category_name,  
p.seller_id,  
count(distinct c.customer_id) as order_count  
from product_orderSeller p  
join customer_order_details c  
on c.order_id = p.order_id  
where c.order_status = 'canceled'  
and product_category_name != 'null'  
group by p.product_category_name,p.seller_id  
order by order_count desc
```

Output:

	product_category_name text	seller_id text	order_count bigint
1	beleza_saude	cc419e0650a3c5ba77189a1882b7556a	8
2	relogios_presentes	6560211a19b47992c3666cc44a7e94c0	7
3	beleza_saude	a416b6a846a11724393025641d4edd...	5
4	esporte_lazer	c3867b4666c7d76867627c2f7fb22e21	5
5	papelaria	3d871de0142ce09b7081e2b9d1733c...	4
6	ferramentas_jardim	1127b7f2594683f2510f1c2c834a486b	4
7	brinquedos	81783131d2a97c8d44d406a4be81b5...	4
8	beleza_saude	620c87c171fb2a6dd6e8bb4dec959fc6	4
9	relogios_presentes	7e93a43ef30c4f03f38b393420bc753a	4
10	papelaria	2a84855fd20af891be03bc5924d2b453	3

6. Which ten products is being canceled often?

```
select p.product_category_name,  
count(distinct c.customer_id) as order_count  
from product_order_seller p  
join customer_order_details c  
on c.order_id = p.order_id  
where c.order_status = 'canceled'  
and product_category_name != 'null'  
group by 1  
order by 2 desc
```

Output:

	product_category_name text	order_count bigint
1	esporte_lazer	47
2	utilidades_domesticas	37
3	beleza_saude	36
4	informatica_acessorios	35
5	brinquedos	31
6	automotivo	24
7	moveis_decoracao	24
8	relogios_presentes	20
9	cama_mesa_banho	18
10	bebes	16

7. Creating stored procedure

```
CREATE OR REPLACE FUNCTION customer()  
RETURNS INTEGER AS $$  
DECLARE  
    total INTEGER := 0;  
BEGIN  
    SELECT COUNT(customer_id)  
    INTO total  
    FROM customer_order_details;  
  
    RETURN total;  
END;  
$$ LANGUAGE plpgsql;  
  
select customer()
```

Output:

	customer integer
1	103886