**SQL PROJECT**

**DATA ANALYTICS PORTFOLIO WITH PYTHON**

**Basic Queries**

1. List all unique cities where customers are located.

Query –

select distinct customer\_city from customers;



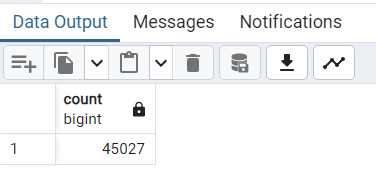
2. Count the number of orders placed in 2017.

Query –

Select count(\*)

from orders

where order\_purchase\_timestamp between '2017-01-01' and '2017-12-31';



3. Find the total sales per category.

Query –

select products.product\_category category,

sum(payments.payment\_value) sales

from products join order\_items

on products.product\_id = order\_items.product\_id

join payments

on payments.order\_id = order\_items.order\_id

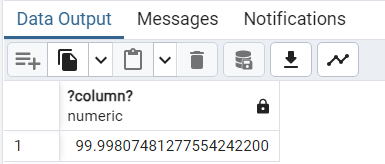
group by category



4. Calculate the percentage of orders that were paid in installments.

Query –

select (cast (sum(case when payment\_installments >= 1 then 1 else 0 end) AS decimal) / count(\*)) \* 100 as percentage\_with\_installments from payments;

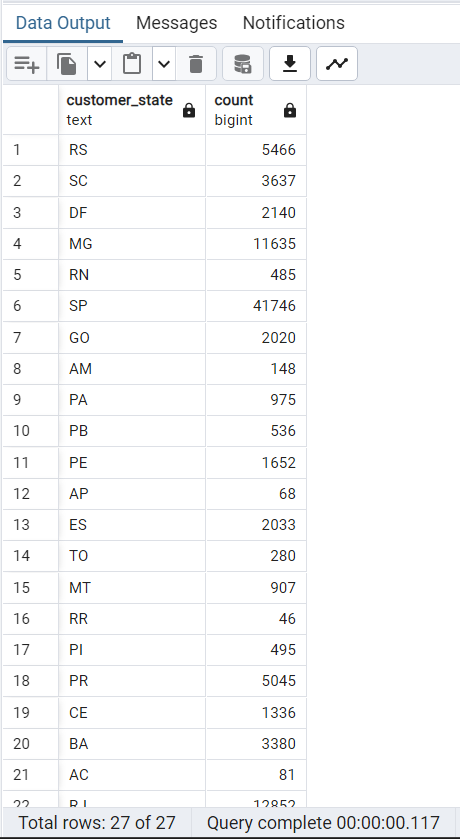


5. Count the number of customers from each state.

Query –

select customer\_state , count(customer\_id)

from customers group by customer\_state



**Intermediate Queries**

1. Calculate the number of orders per month in 2018.

Query –

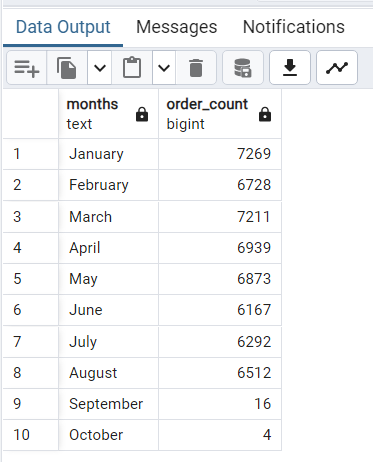
select to\_char(order\_purchase\_timestamp::timestamp, 'Month') as months, count(order\_id) as order\_count

from orders

where extract(YEAR from order\_purchase\_timestamp::timestamp) = 2018

group by to\_char(order\_purchase\_timestamp::timestamp, 'Month')

order by min(order\_purchase\_timestamp::timestamp);



2. Find the average number of products per order, grouped by customer city.

Query –

with count\_per\_order as

(select orders.order\_id, orders.customer\_id, count(order\_items.order\_id) as oc

from orders join order\_items

on orders.order\_id = order\_items.order\_id

group by orders.order\_id, orders.customer\_id)

select customers.customer\_city, round(avg(count\_per\_order.oc),2) average\_orders

from customers join count\_per\_order

on customers.customer\_id = count\_per\_order.customer\_id

group by customers.customer\_city order by average\_orders desc;



3. Calculate the percentage of total revenue contributed by each product category.

Query –

Select upper(products.product\_category) AS category,

round(cast((sum(payments.payment\_value) / (select sum(payment\_value) from payments)) \* 100 as numeric), 2) as sales\_percentage

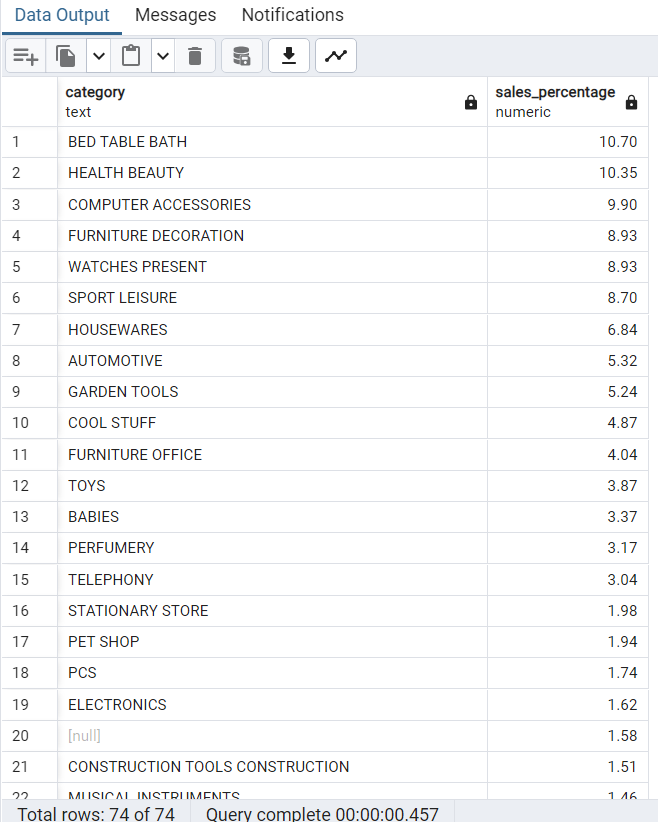
from products

join order\_items on products.product\_id = order\_items.product\_id

join payments on payments.order\_id = order\_items.order\_id

group by category

order by sales\_percentage desc;



4. Identify the correlation between product price and the number of times a product has been purchased.

Query –

select products.product\_category,

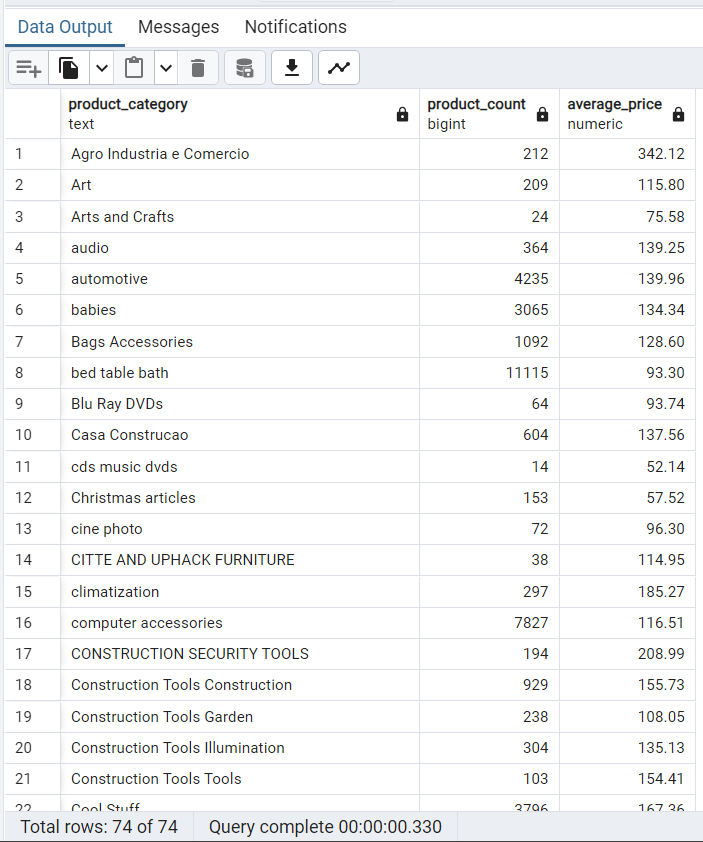
count(order\_items.product\_id) as product\_count,

round(avg(order\_items.price)::numeric, 2) as average\_price

from products

join order\_items on products.product\_id = order\_items.product\_id

group by products.product\_category;



5. Calculate the total revenue generated by each seller, and rank them by revenue.

Query –

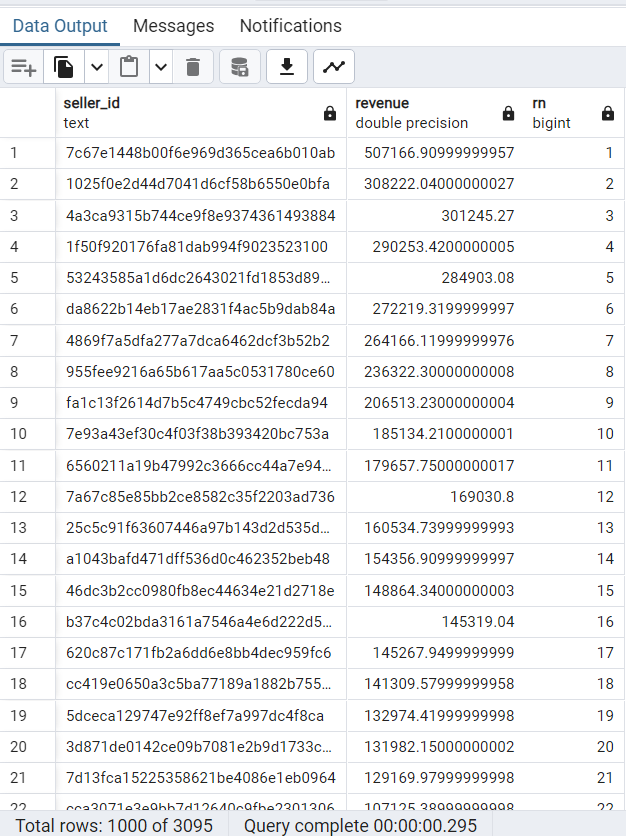
select \*, dense\_rank() over(order by revenue desc) as rn from

(select order\_items.seller\_id, sum(payments.payment\_value)

revenue from order\_items join payments

on order\_items.order\_id = payments.order\_id

group by order\_items.seller\_id) as a



**Advanced Queries**

1. Calculate the moving average of order values for each customer over their order history.

Query –

select customer\_id, order\_purchase\_timestamp, payment,

avg(payment) over(partition by customer\_id order by order\_purchase\_timestamp

rows between 2 preceding and current row) as mov\_avg

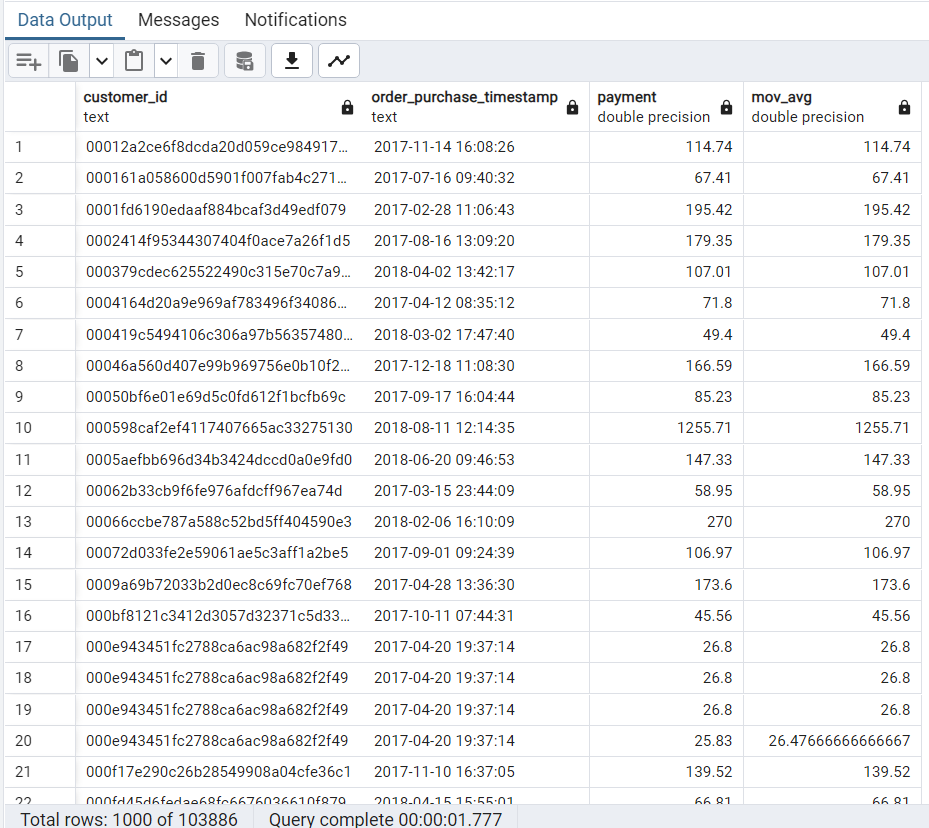
from

(select orders.customer\_id, orders.order\_purchase\_timestamp,

payments.payment\_value as payment

from payments join orders

on payments.order\_id = orders.order\_id) as a



2. Calculate the cumulative sales per month for each year.

Query –

select years, months, payment,

sum(payment) over (order by years, months) as cumulative\_sales

from (

select

extract(year from orders.order\_purchase\_timestamp::timestamp) as years,

extract(month from orders.order\_purchase\_timestamp::timestamp) as months,

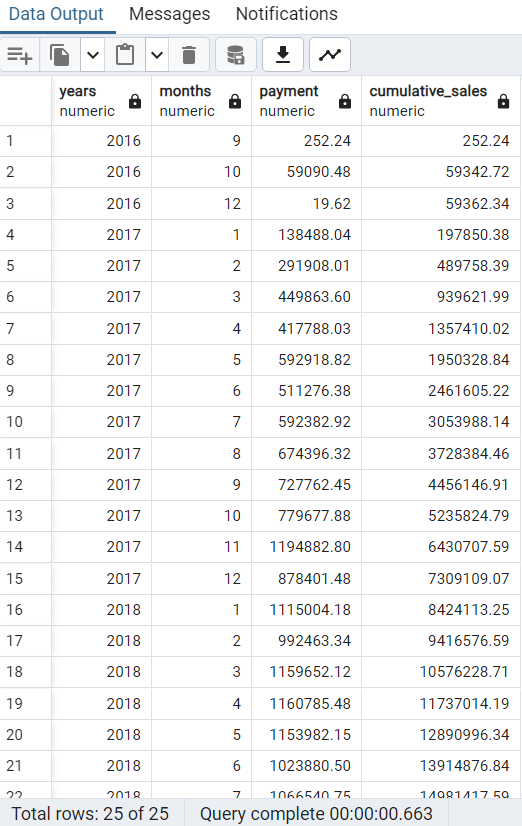
round(sum(payments.payment\_value)::numeric, 2) as payment

from orders

join payments on orders.order\_id = payments.order\_id

group by years, months

order by years, months) as a;



3. Calculate the year-over-year growth rate of total sales.

Query –

with a as (select

extract(year from orders.order\_purchase\_timestamp::timestamp) as years,

round(sum(payments.payment\_value)::numeric, 2) as payment

from orders

join payments on orders.order\_id = payments.order\_id

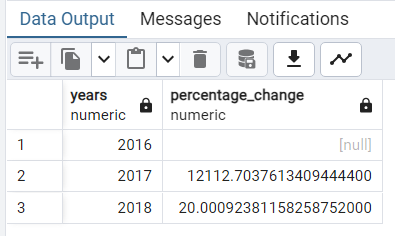
group by years

order by year)

select years, ((payment - lag(payment, 1) over (order by years)) /

nullif(lag(payment, 1) over (order by years), 0)) \* 100 as percentage\_change

from a;



4. Calculate the retention rate of customers, defined as the percentage of customers who make another purchase within 6 months of their first purchase.

Query –

with a as (select

customers.customer\_id,

min(orders.order\_purchase\_timestamp::timestamp) as first\_order

from customers

join orders on customers.customer\_id = orders.customer\_id

group by customers.customer\_id),

b as (select a.customer\_id,

count(distinct orders.order\_purchase\_timestamp) as next\_order

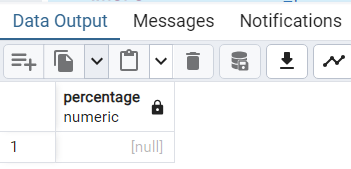
from a join orders on orders.customer\_id = a.customer\_id

where orders.order\_purchase\_timestamp::timestamp > a.first\_order

and orders.order\_purchase\_timestamp::timestamp < a.first\_order + interval '6 months'

group by a.customer\_id)

select 100.0 \* count(distinct a.customer\_id) / nullif(count(distinct b.customer\_id), 0) as percentage from a left join b on a.customer\_id = b.customer\_id;



5. Identify the top 3 customers who spent the most money in each year.

Query –

select years, customer\_id, payment, d\_rank

from (select extract(year from orders.order\_purchase\_timestamp::timestamp) as years,

orders.customer\_id, sum(payments.payment\_value) as payment,

dense\_rank() over (

partition by extract(year from orders.order\_purchase\_timestamp::timestamp)

order by sum(payments.payment\_value) desc) as d\_rank

from orders

join payments on payments.order\_id = orders.order\_id

group by extract(year from orders.order\_purchase\_timestamp::timestamp),

orders.customer\_id) as a where d\_rank <= 3;

