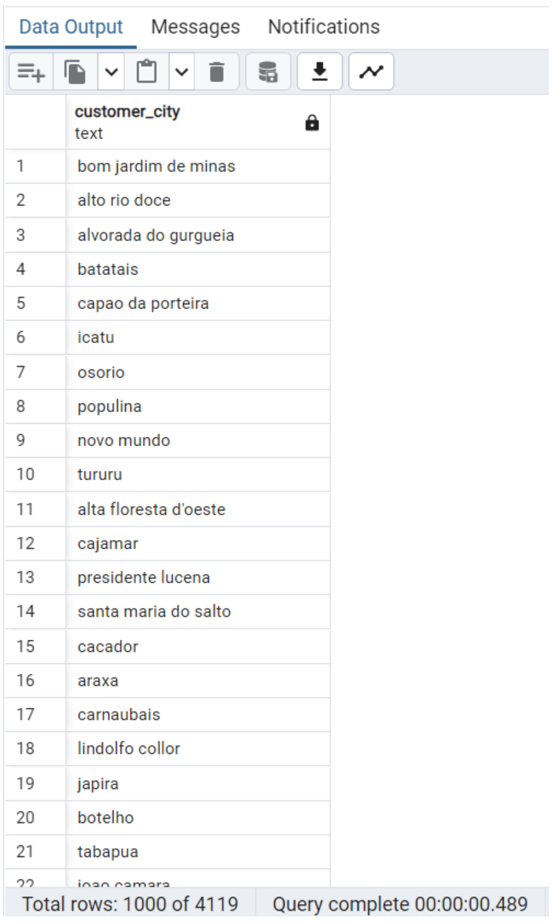
SQL PROJECT
DATA ANALYTICS PORTFOLIO WITH PYTHON

Basic Queries

1. List all unique cities where customers are located.
Query –

```
select distinct customer_city from customers;
```

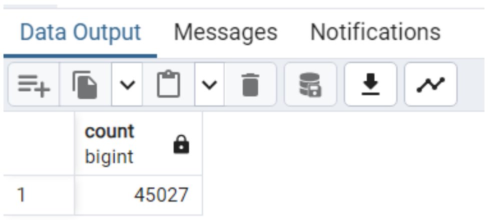


	customer_city text
1	bom jardim de minas
2	alto rio doce
3	alvorada do gurgueia
4	batatais
5	capao da porteira
6	icatu
7	osorio
8	populina
9	novo mundo
10	tururu
11	alta floresta d'oeste
12	cajamar
13	presidente lucena
14	santa maria do salto
15	cacador
16	araxa
17	carnaubais
18	lindolfo collor
19	japira
20	botelho
21	tabapua
22	joao camara

Total rows: 1000 of 4119 Query complete 00:00:00.489

2. Count the number of orders placed in 2017.
Query –

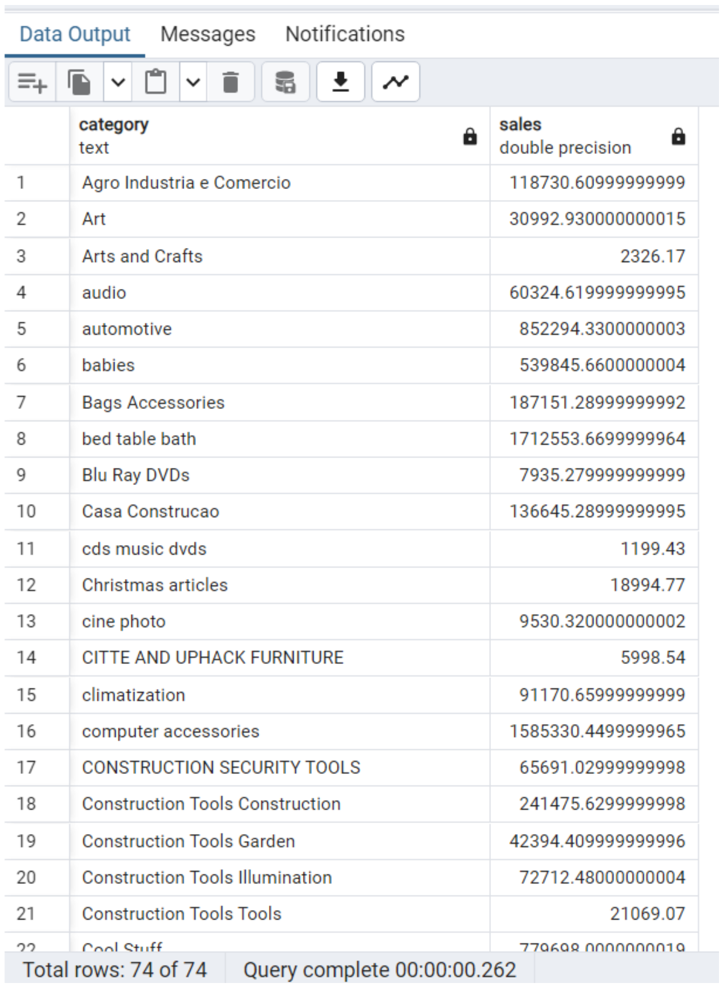
```
Select count(*)
from orders
where order_purchase_timestamp between '2017-01-01' and '2017-12-31';
```



	count bigint
1	45027

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
```

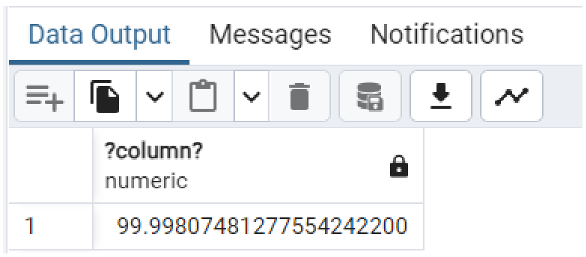


	category text	sales double precision
1	Agro Industria e Comercio	118730.60999999999
2	Art	30992.930000000015
3	Arts and Crafts	2326.17
4	audio	60324.619999999995
5	automotive	852294.3300000003
6	babies	539845.6600000004
7	Bags Accessories	187151.28999999992
8	bed table bath	1712553.6699999964
9	Blu Ray DVDs	7935.279999999999
10	Casa Construcao	136645.28999999995
11	cds music dvds	1199.43
12	Christmas articles	18994.77
13	cine photo	9530.320000000002
14	CITTE AND UPHACK FURNITURE	5998.54
15	climatization	91170.65999999999
16	computer accessories	1585330.4499999965
17	CONSTRUCTION SECURITY TOOLS	65691.02999999998
18	Construction Tools Construction	241475.6299999998
19	Construction Tools Garden	42394.409999999996
20	Construction Tools Illumination	72712.48000000004
21	Construction Tools Tools	21069.07
22	Cool Stuff	779698.0000000019

Total rows: 74 of 74 Query complete 00:00:00.262

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

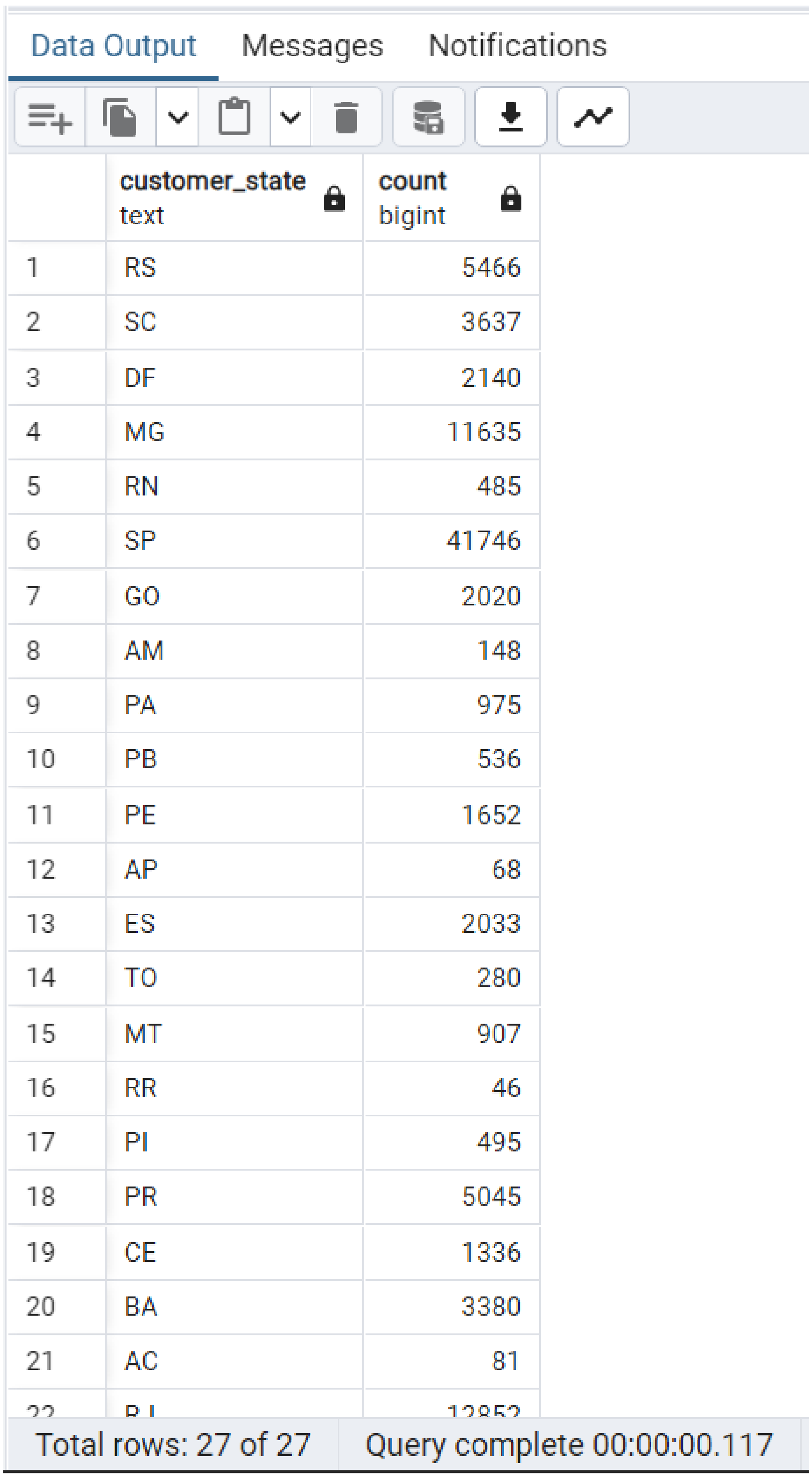
```
select (cast (sum(case when payment_installments >= 1 then 1 else 0 end) AS decimal) /
count(*)) * 100 as percentage_with_installments from payments;
```



	?column? numeric
1	99.9980748127755424200

5. Count the number of customers from each state.
Query –

```
select customer_state , count(customer_id)
from customers group by customer_state
```



Data Output | Messages | Notifications

	customer_state text	count bigint
1	RS	5466
2	SC	3637
3	DF	2140
4	MG	11635
5	RN	485
6	SP	41746
7	GO	2020
8	AM	148
9	PA	975
10	PB	536
11	PE	1652
12	AP	68
13	ES	2033
14	TO	280
15	MT	907
16	RR	46
17	PI	495
18	PR	5045
19	CE	1336
20	BA	3380
21	AC	81
22	RJ	12852

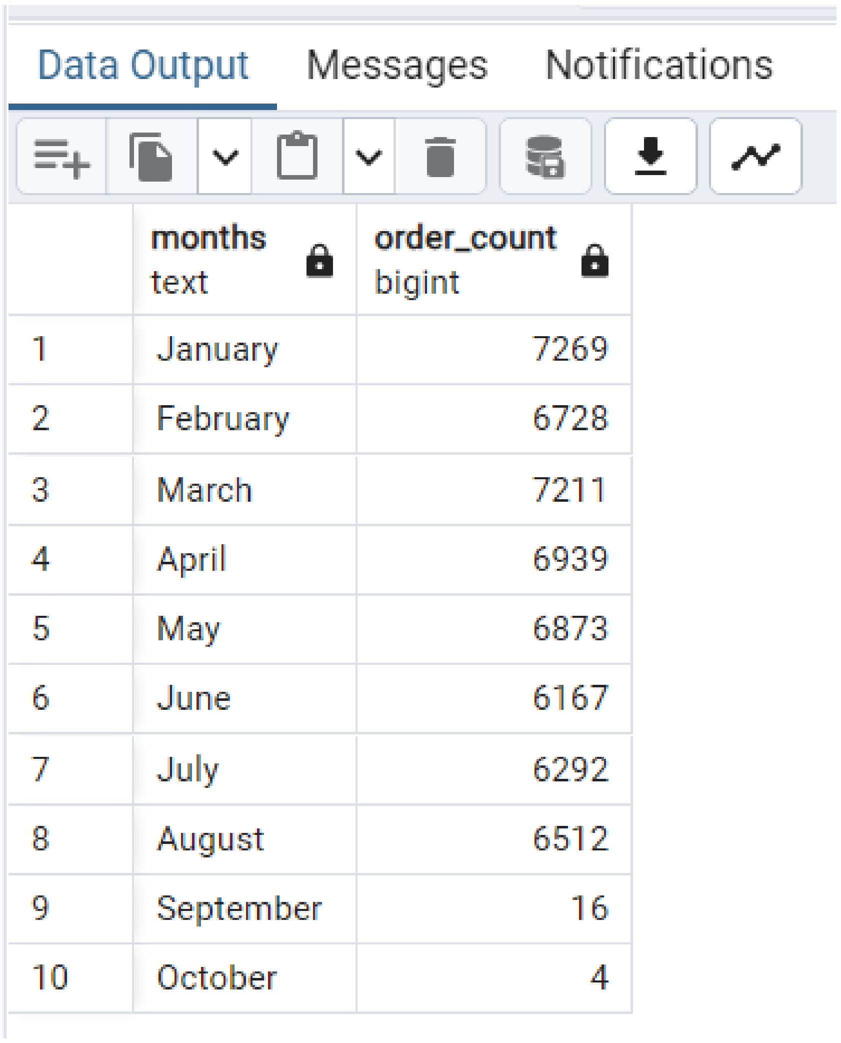
Total rows: 27 of 27 Query complete 00:00:00.117

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);
```

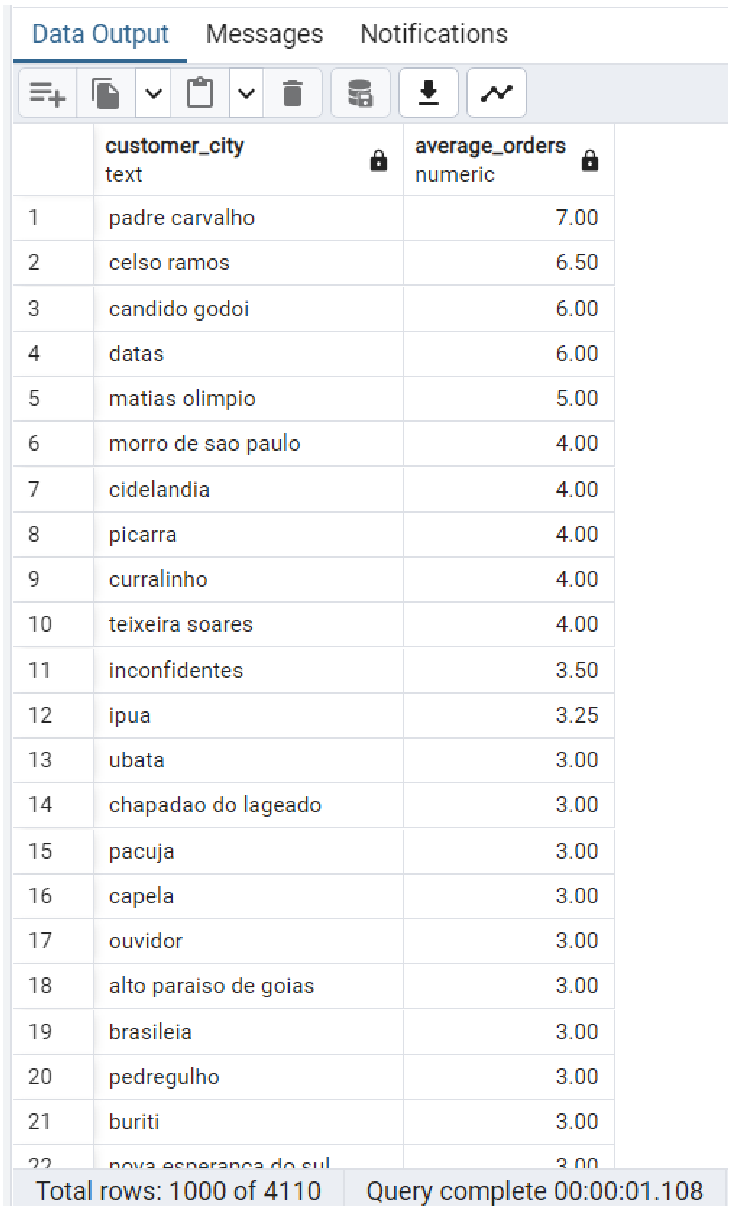


	months text	order_count bigint
1	January	7269
2	February	6728
3	March	7211
4	April	6939
5	May	6873
6	June	6167
7	July	6292
8	August	6512
9	September	16
10	October	4

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;
```



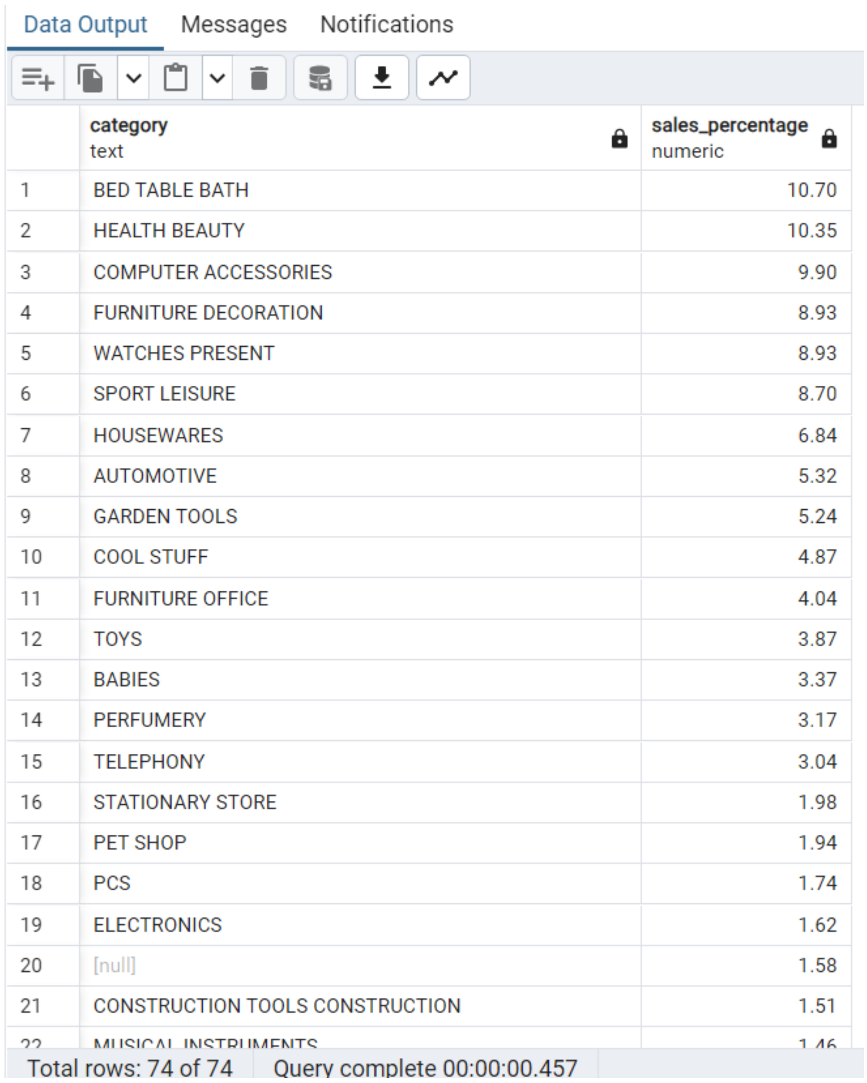
	customer_city text	average_orders numeric
1	padre carvalho	7.00
2	celso ramos	6.50
3	candido godoi	6.00
4	datas	6.00
5	matias olimpio	5.00
6	morro de sao paulo	4.00
7	cidelandia	4.00
8	picarra	4.00
9	curralinho	4.00
10	teixeira soares	4.00
11	inconfidentes	3.50
12	ipua	3.25
13	ubata	3.00
14	chapadao do lageado	3.00
15	pacuja	3.00
16	capela	3.00
17	ouvidor	3.00
18	alto paraiso de goias	3.00
19	brasileia	3.00
20	pedregulho	3.00
21	buriti	3.00
22	nova esperanca do sul	3.00

Total rows: 1000 of 4110   Query complete 00:00:01.108

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;
```



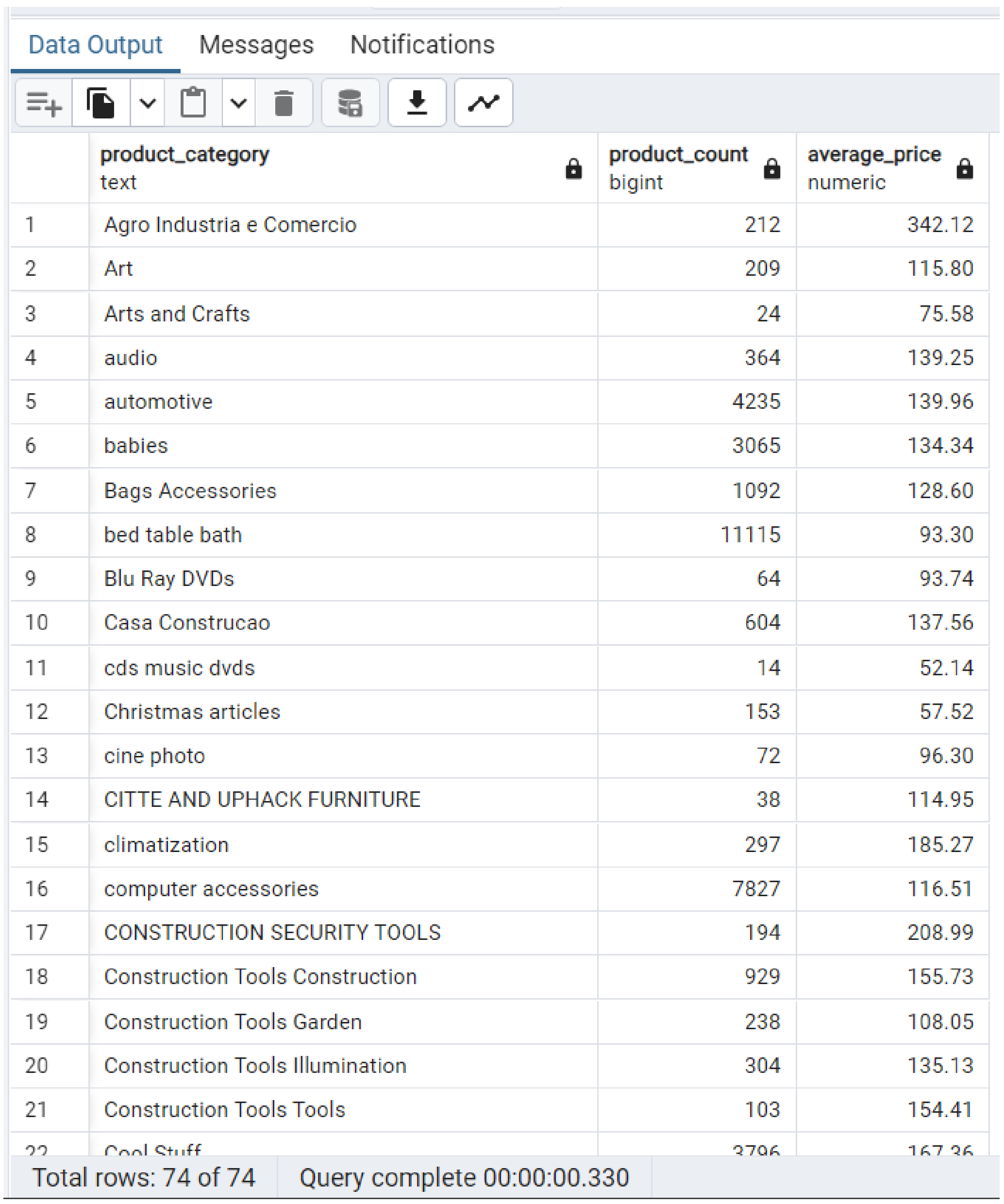
	category text	sales_percentage numeric
1	BED TABLE BATH	10.70
2	HEALTH BEAUTY	10.35
3	COMPUTER ACCESSORIES	9.90
4	FURNITURE DECORATION	8.93
5	WATCHES PRESENT	8.93
6	SPORT LEISURE	8.70
7	HOUSEWARES	6.84
8	AUTOMOTIVE	5.32
9	GARDEN TOOLS	5.24
10	COOL STUFF	4.87
11	FURNITURE OFFICE	4.04
12	TOYS	3.87
13	BABIES	3.37
14	PERFUMERY	3.17
15	TELEPHONY	3.04
16	STATIONARY STORE	1.98
17	PET SHOP	1.94
18	PCS	1.74
19	ELECTRONICS	1.62
20	[null]	1.58
21	CONSTRUCTION TOOLS CONSTRUCTION	1.51
22	MUSICAL INSTRUMENTS	1.46

Total rows: 74 of 74   Query complete 00:00:00.457

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;
```

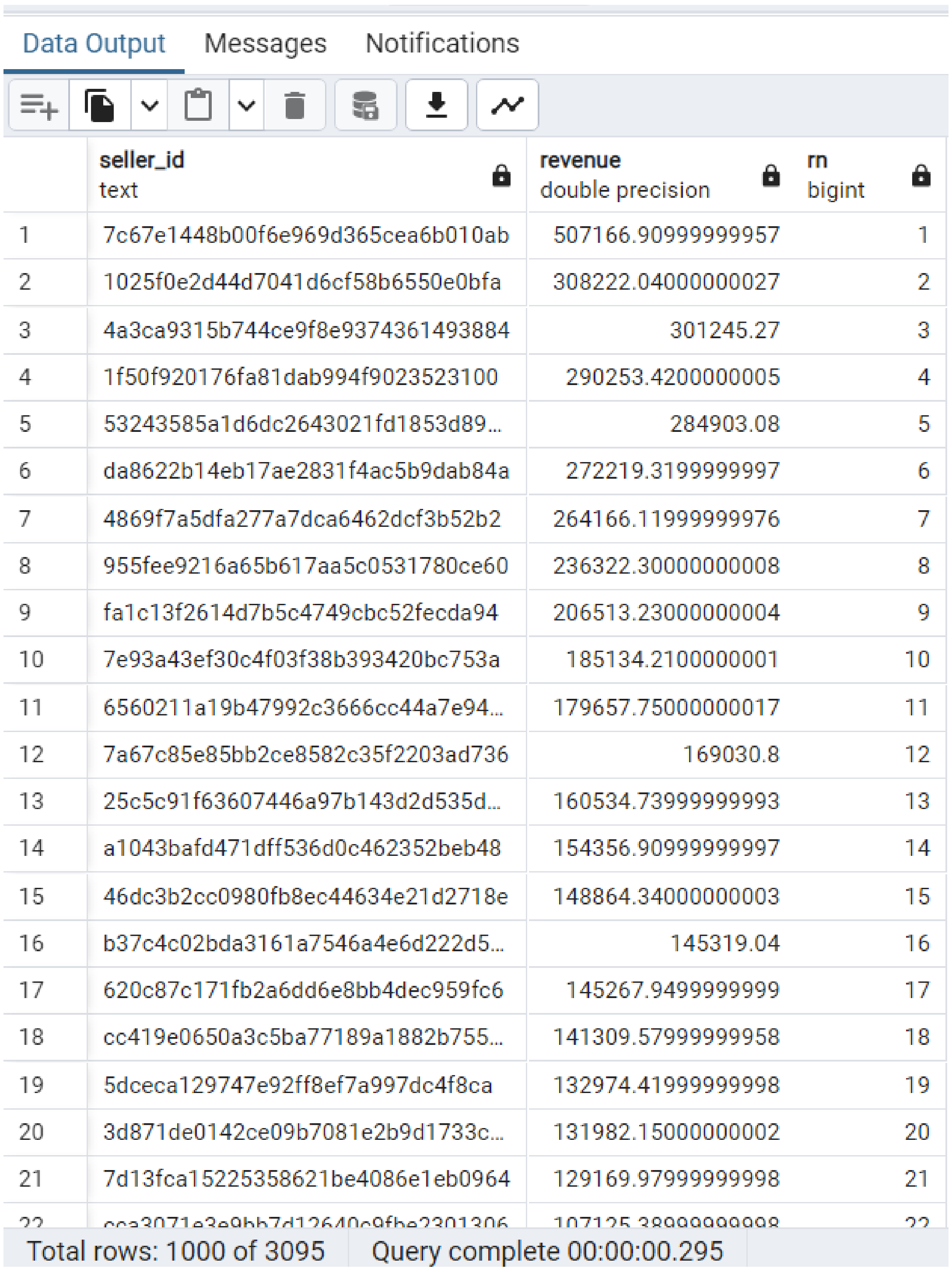


	product_category text	product_count bigint	average_price numeric
1	Agro Industria e Comercio	212	342.12
2	Art	209	115.80
3	Arts and Crafts	24	75.58
4	audio	364	139.25
5	automotive	4235	139.96
6	babies	3065	134.34
7	Bags Accessories	1092	128.60
8	bed table bath	11115	93.30
9	Blu Ray DVDs	64	93.74
10	Casa Construcao	604	137.56
11	cds music dvds	14	52.14
12	Christmas articles	153	57.52
13	cine photo	72	96.30
14	CITTE AND UPHACK FURNITURE	38	114.95
15	climatization	297	185.27
16	computer accessories	7827	116.51
17	CONSTRUCTION SECURITY TOOLS	194	208.99
18	Construction Tools Construction	929	155.73
19	Construction Tools Garden	238	108.05
20	Construction Tools Illumination	304	135.13
21	Construction Tools Tools	103	154.41
22	Cool Stuff	3796	167.36

Total rows: 74 of 74 Query complete 00:00:00.330

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
```



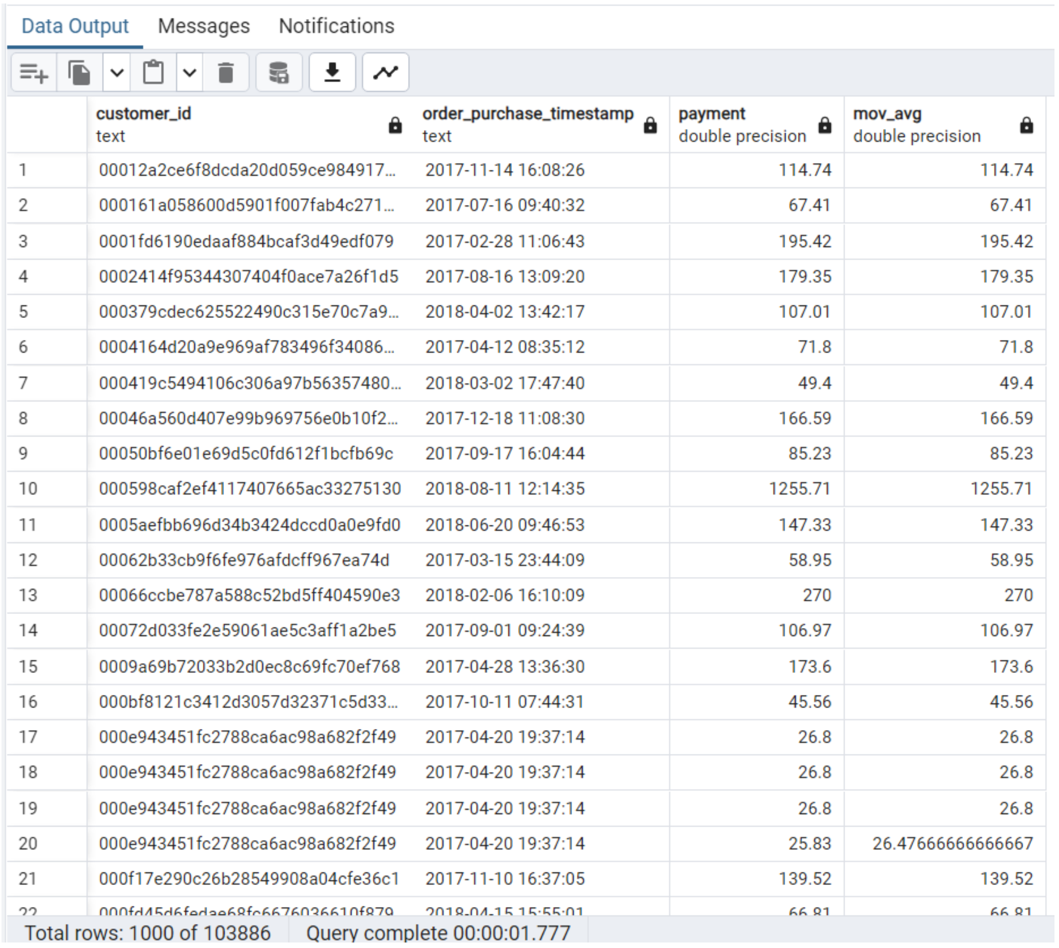
	seller_id text	revenue double precision	rn bigint
1	7c67e1448b00f6e969d365cea6b010ab	507166.90999999957	1
2	1025f0e2d44d7041d6cf58b6550e0bfa	308222.04000000027	2
3	4a3ca9315b744ce9f8e9374361493884	301245.27	3
4	1f50f920176fa81dab994f9023523100	290253.4200000005	4
5	53243585a1d6dc2643021fd1853d89...	284903.08	5
6	da8622b14eb17ae2831f4ac5b9dab84a	272219.3199999997	6
7	4869f7a5dfa277a7dca6462dcf3b52b2	264166.11999999976	7
8	955fee9216a65b617aa5c0531780ce60	236322.30000000008	8
9	fa1c13f2614d7b5c4749cbc52fecda94	206513.23000000004	9
10	7e93a43ef30c4f03f38b393420bc753a	185134.2100000001	10
11	6560211a19b47992c3666cc44a7e94...	179657.75000000017	11
12	7a67c85e85bb2ce8582c35f2203ad736	169030.8	12
13	25c5c91f63607446a97b143d2d535d...	160534.73999999993	13
14	a1043bafd471dff536d0c462352beb48	154356.90999999997	14
15	46dc3b2cc0980fb8ec44634e21d2718e	148864.34000000003	15
16	b37c4c02bda3161a7546a4e6d222d5...	145319.04	16
17	620c87c171fb2a6dd6e8bb4dec959fc6	145267.9499999999	17
18	cc419e0650a3c5ba77189a1882b755...	141309.57999999958	18
19	5dceca129747e92ff8ef7a997dc4f8ca	132974.41999999998	19
20	3d871de0142ce09b7081e2b9d1733c...	131982.15000000002	20
21	7d13fca15225358621be4086e1eb0964	129169.97999999998	21
22	cca3071e3e9bb7d12640c9fbe2301306	107125.38999999998	22

Total rows: 1000 of 3095 Query complete 00:00:00.295

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
```



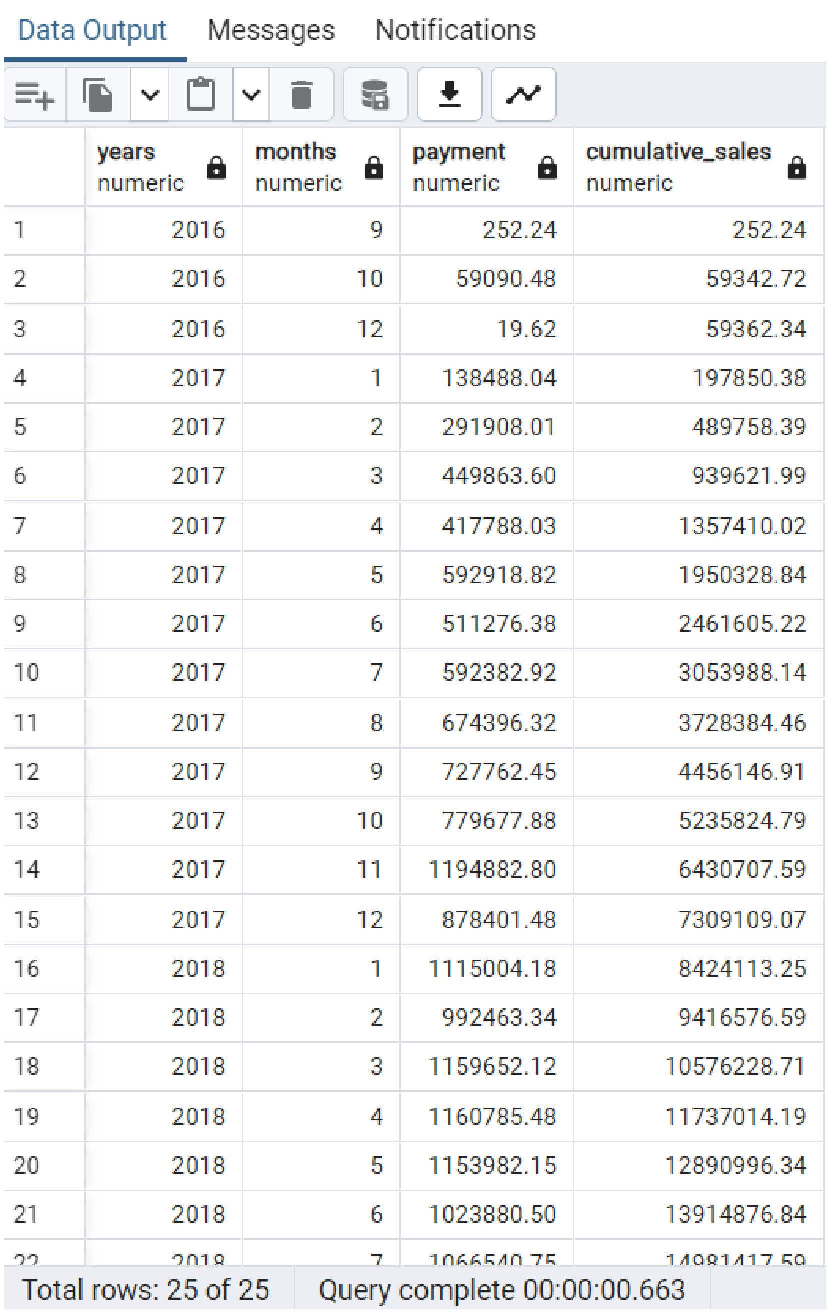
	customer_id text	order_purchase_timestamp text	payment double precision	mov_avg double precision
1	00012a2ce6f8dcda20d059ce984917...	2017-11-14 16:08:26	114.74	114.74
2	000161a058600d5901f007fab4c271...	2017-07-16 09:40:32	67.41	67.41
3	0001fd6190edaaf884bcaf3d49edf079	2017-02-28 11:06:43	195.42	195.42
4	0002414f95344307404f0ace7a26f1d5	2017-08-16 13:09:20	179.35	179.35
5	000379cdec625522490c315e70c7a9...	2018-04-02 13:42:17	107.01	107.01
6	0004164d20a9e969af783496f34086...	2017-04-12 08:35:12	71.8	71.8
7	000419c5494106c306a97b56357480...	2018-03-02 17:47:40	49.4	49.4
8	00046a560d407e99b969756e0b10f2...	2017-12-18 11:08:30	166.59	166.59
9	00050bf6e01e69d5c0fd612f1bcfb69c	2017-09-17 16:04:44	85.23	85.23
10	000598caf2ef4117407665ac33275130	2018-08-11 12:14:35	1255.71	1255.71
11	0005aefbb696d34b3424dccd0a0e9fd0	2018-06-20 09:46:53	147.33	147.33
12	00062b33cb9f6fe976afdcff967ea74d	2017-03-15 23:44:09	58.95	58.95
13	00066ccbe787a588c52bd5ff404590e3	2018-02-06 16:10:09	270	270
14	00072d033fe2e59061ae5c3aff1a2be5	2017-09-01 09:24:39	106.97	106.97
15	0009a69b72033b2d0ec8c69fc70ef768	2017-04-28 13:36:30	173.6	173.6
16	000bf8121c3412d3057d32371c5d33...	2017-10-11 07:44:31	45.56	45.56
17	000e943451fc2788ca6ac98a682f2f49	2017-04-20 19:37:14	26.8	26.8
18	000e943451fc2788ca6ac98a682f2f49	2017-04-20 19:37:14	26.8	26.8
19	000e943451fc2788ca6ac98a682f2f49	2017-04-20 19:37:14	26.8	26.8
20	000e943451fc2788ca6ac98a682f2f49	2017-04-20 19:37:14	25.83	26.476666666666667
21	000f17e290c26b28549908a04cfe36c1	2017-11-10 16:37:05	139.52	139.52
22	000fd45d6fedae68fc6676036610f879	2018-04-15 15:55:01	66.81	66.81

Total rows: 1000 of 103886 Query complete 00:00:01.777

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;
```

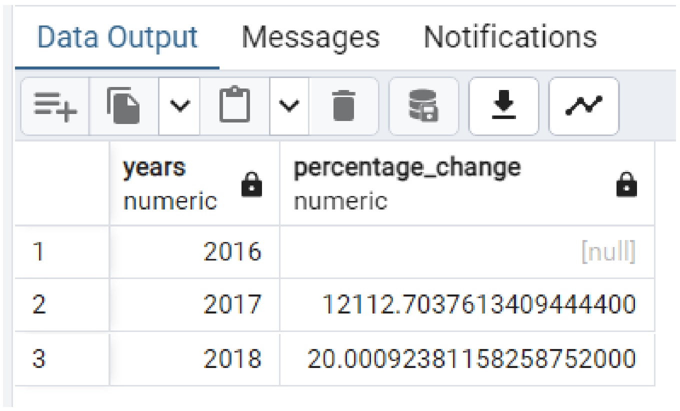


	years numeric	months numeric	payment numeric	cumulative_sales numeric
1	2016	9	252.24	252.24
2	2016	10	59090.48	59342.72
3	2016	12	19.62	59362.34
4	2017	1	138488.04	197850.38
5	2017	2	291908.01	489758.39
6	2017	3	449863.60	939621.99
7	2017	4	417788.03	1357410.02
8	2017	5	592918.82	1950328.84
9	2017	6	511276.38	2461605.22
10	2017	7	592382.92	3053988.14
11	2017	8	674396.32	3728384.46
12	2017	9	727762.45	4456146.91
13	2017	10	779677.88	5235824.79
14	2017	11	1194882.80	6430707.59
15	2017	12	878401.48	7309109.07
16	2018	1	1115004.18	8424113.25
17	2018	2	992463.34	9416576.59
18	2018	3	1159652.12	10576228.71
19	2018	4	1160785.48	11737014.19
20	2018	5	1153982.15	12890996.34
21	2018	6	1023880.50	13914876.84
22	2018	7	1066540.75	14981417.59

Total rows: 25 of 25   Query complete 00:00:00.663

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

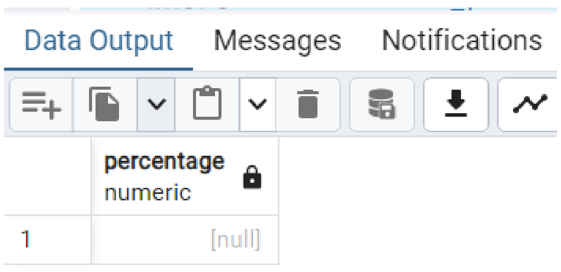
```
with a as (select
extract(year from orders.order_purchase_timestamp::timestamp) as years,
round(sum(payments.payment_value)::numeric, 2) as payment
from orders
join payments on orders.order_id = payments.order_id
group by years
order by year)
select years, ((payment - lag(payment, 1) over (order by years)) /
nullif(lag(payment, 1) over (order by years), 0)) * 100 as percentage_change
from a;
```



	years numeric	percentage_change numeric
1	2016	[null]
2	2017	12112.7037613409444400
3	2018	20.0009238115825875200

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 –

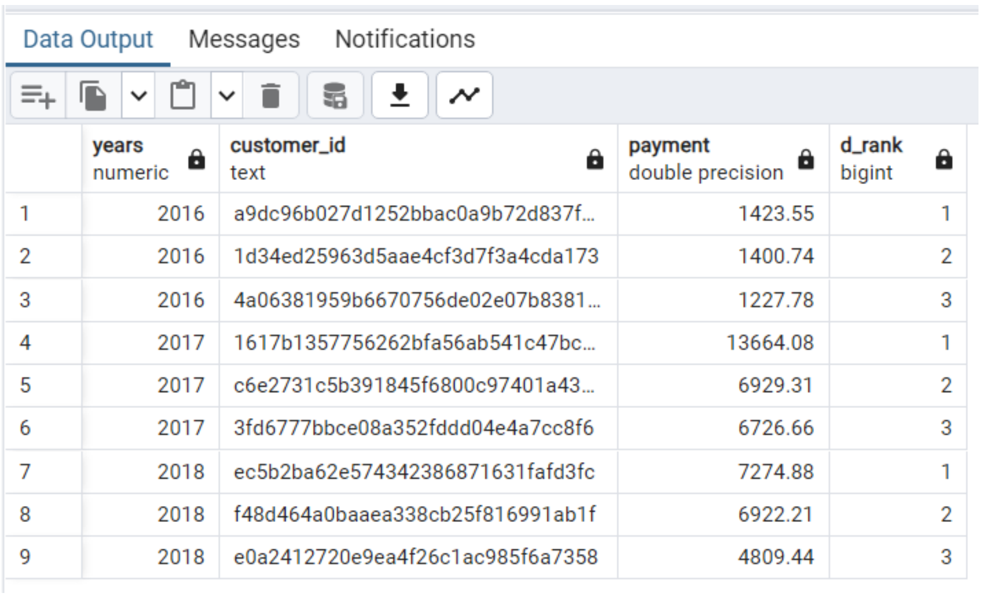
```
with a as (select
customers.customer_id,
min(orders.order_purchase_timestamp::timestamp) as first_order
from customers
join orders on customers.customer_id = orders.customer_id
group by customers.customer_id),
b as (select a.customer_id,
count(distinct orders.order_purchase_timestamp) as next_order
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;
```



	percentage numeric
1	[null]

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;
```



	years numeric	customer_id text	payment double precision	d_rank bigint
1	2016	a9dc96b027d1252bbac0a9b72d837f...	1423.55	1
2	2016	1d34ed25963d5aae4cf3d7f3a4cda173	1400.74	2
3	2016	4a06381959b6670756de02e07b8381...	1227.78	3
4	2017	1617b1357756262bfa56ab541c47bc...	13664.08	1
5	2017	c6e2731c5b391845f6800c97401a43...	6929.31	2
6	2017	3fd6777bbce08a352fddd04e4a7cc8f6	6726.66	3
7	2018	ec5b2ba62e574342386871631fafd3fc	7274.88	1
8	2018	f48d464a0baaea338cb25f816991ab1f	6922.21	2
9	2018	e0a2412720e9ea4f26c1ac985f6a7358	4809.44	3