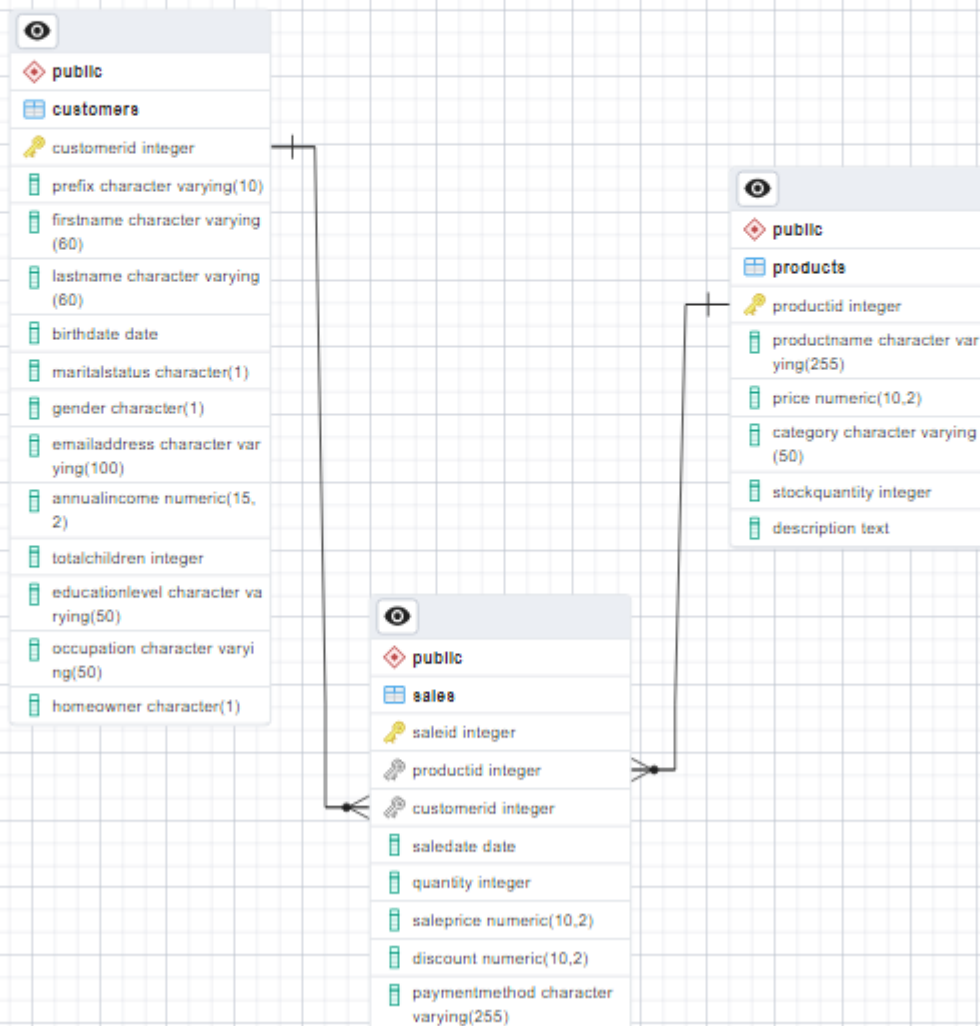
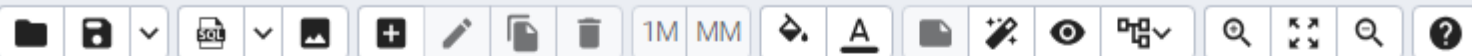


SQL CLASS/postgres@PostgreSQL 16



SQL CLASS/postgres@PostgreSQL 16



Query Query History

```
1 select * from products;|
```

Data Output Messages Notifications



	productid [PK] integer	productname character varying (255)	price numeric (10,2)	category character varying (50)	stockquantity integer	description text
1	8	Air Fryer	100.00	Appliances	35	Oil-less air fryer for healthier cooking
2	10	Robot Vacuum	250.00	Appliances	50	Smart robot vacuum with mapping technology
3	13	Office Chair	120.00	Furniture	25	Comfortable and adjustable office chair
4	15	Coffee Grinder	30.00	Appliances	15	Electric coffee grinder for fresh coffee beans
5	17	Standing Desk	200.00	Furniture	10	Adjustable standing desk for ergonomic work
6	18	Smart Thermostat	120.00	Home Improvement	30	Programmable smart thermostat for energy efficiency
7	20	Blender	40.00	Appliances	25	High-speed blender for smoothies and shakes
8	21	Digital Watch	50.00	Accessories	100	Water-resistant digital watch with multiple functions
9	23	Air Purifier	120.00	Home Improvement	20	HEPA air purifier for clean indoor air
10	24	Compact Refrigerator	150.00	Appliances	15	Space-saving compact refrigerator for dorms or offices
11	25	Smart Doorbell	120.00	Home Improvement	40	Video doorbell with motion detection and two-way commun
12	26	Yoga Mat	20.00	Fitness	60	Non-slip yoga mat for comfortable workou

SQL CLASS/postgres@PostgreSQL 16

No limit

Query Query History

```
6
7 select * from sales;
8
9
10
```

Data Output Messages Notifications

	saleid [PK] integer	productid integer	customerid integer	saledate date	quantity integer	saleprice numeric (10,2)	discount numeric (10,2)	paymentmethod character varying (255)
1	1179	72	11183	2023-12-28	2	300.00	12.33	Debit Card
2	1180	45	11093	2023-11-09	7	25.00	2.18	Cash
3	1181	94	11194	2023-08-17	1	500.00	55.42	Credit Card
4	1182	65	11081	2023-02-23	10	600.00	74.56	Debit Card
5	1183	68	11118	2023-08-30	8	900.00	79.61	Cash
6	1184	76	11037	2023-09-03	10	700.00	127.81	Credit Card
7	1185	86	11008	2024-02-13	3	700.00	2.09	Debit Card
8	1186	93	11008	2023-07-21	3	400.00	69.32	Cash
9	1187	63	11066	2023-07-18	8	400.00	48.16	Credit Card
10	1188	69	11182	2024-01-24	10	1000.00	14.05	Cash
11	1189	82	11004	2023-08-24	9	300.00	32.83	Debit Card
12	1190	64	11007	2023-08-21	9	500.00	21.48	Credit Card
13	1191	66	11004	2023-04-14	4	700.00	2.51	Credit Card

Total rows: 1000 of 1000

Query complete 00:00:00.114

Ln 7, Col 21

SQL CLASS/postgres@PostgreSQL 16



Query Query History

```

8
9 select * from customers;
10
11
12

```

Data Output Messages Notifications



	customerid [PK] integer	prefix character varying (10)	firstname character varying (60)	lastname character varying (60)	birthdate date	maritalstatus character	gender character	emailaddress character varying (100)
1	11000	MR.	JON	YANG	1980-01-27	M	M	jon24@adventure-works.com
2	11001	MR.	EUGENE	HUANG	1975-11-27	S	M	eugene10@adventure-works.com
3	11002	MR.	RUBEN	TORRES	1976-02-13	M	M	ruben35@adventure-works.com
4	11003	MS.	CHRISTY	ZHU	1991-09-06	S	F	christy12@adventure-works.com
5	11004	MRS.	ELIZABETH	JOHNSON	1996-06-04	S	F	elizabeth5@adventure-works.com
6	11005	MR.	JULIO	RUIZ	1975-07-25	S	M	julio1@adventure-works.com
7	11006	MR.	JULIUS	SALVADOR	1976-07-25	S	M	julius1@adventure-works.com
8	11007	MR.	MARCO	MEHTA	1960-02-28	M	M	marco14@adventure-works.com
9	11008	MRS.	ROBIN	VERHOFF	1965-06-08	S	F	rob4@adventure-works.com
10	11009	MR.	SHANNON	CARLSON	1964-01-28	S	M	shannon38@adventure-works.com
11	11010	MS.	JACQUELYN	SUAREZ	1988-03-26	S	F	jacquelyn20@adventure-works.co...
12	11011	MR.	CURTIS	LU	1953-07-12	M	M	curtis9@adventure-works.com

SQL CLASS/postgres@PostgreSQL 16



Query Query History

```
15
16 -- a.Use SQL queries to determine the total revenue generated by each product category. Additionally, identify
17 select category, sum (quantity * saleprice) as totalrevenue
18 from sales
19 join products
20 on sales.productid = products.productid
21 group by category
22
```

Data Output Messages Notifications



	category character varying (50)	totalrevenue numeric
1	Appliances	453050.00
2	Home Improvement	95000.00
3	Fitness	34900.00
4	Tools	55625.00
5	Accessories	150200.00
6	Furniture	43000.00
7	Electronics	903695.00
8	Office Supplies	217555.00
9	Kitchenware	209600.00

SQL CLASS/postgres@PostgreSQL 16






Query Query History

```
23 -- b.the top 3 customers by spending within each category
24
25 select firstname, lastname, sum (quantity * saleprice) as totalspent
26 from sales s
27 join customers cu
28 on s.customerid= cu.customerid
29 group by firstname, lastname
30 order by totalspent desc
31 limit 3
32
```

Data Output Messages Notifications



	firstname character varying (60)	lastname character varying (60)	totalspent numeric
1	EMILY	JOHNSON	36860.00
2	ALAN	ZHENG	28690.00
3	CAROL	HOWARD	27160.00

 SQL CLASS/postgres@PostgreSQL 16  








 No limit 












Query Query History

```

42 ---- 2.Customer Purchasing Patterns
43 ---- a.Perform an analysis on customers whose names start with 'J' ' and have made purchases totaling over
44 $1,000.
45 -- This task involves joining sales data with customer information, which can help in understanding high
46 -- value customer behaviors and preferences.
47
48 select firstname, lastname , sum(quantity * saleprice) as totalspent
49 from customers cu
50 join sales s
51 on cu.customerid = s.customerid
52 where firstname like 'J%'
53 group by firstname, lastname
54 having sum(quantity * saleprice) > 1000
55 order by totalspent desc
56

```

Data Output Messages Notifications














	firstname character varying (60) 	lastname character varying (60) 	totalspent numeric 
1	JILL	JIMENEZ	26000.00
2	JEREMY	POWELL	21320.00
3	JAVIER	ALVAREZ	18500.00
4	JON	YANG	17820.00
5	JENNIFER	RUSSELL	17400.00

SQL CLASS/postgres@PostgreSQL 16



Query Query History

```
56
57 -- 3. Inventory Management:
58 -- Identify products that have never been sold by using subqueries.
59 -- This analysis is crucial for inventory management,helping to decide on discontinuing certain products or initiating
60
61 select distinct productid
62 from sales
63
64 -- using sub query
65 select productid,productname, category, price
66 from products
67 where productid not in (select distinct productid
68 from sales
69 );
70
```

Data Output Messages Notifications



	productid [PK] integer	productname character varying (255)	category character varying (50)	price numeric (10,2)
1	101	10Alytics	home improvement	350.00
2	102	saturday	home improvement	60.00

SQL CLASS/postgres@PostgreSQL 16



Query Query History

```
78 -- Analyze sales made in the first quarter and December, using UNION operations
79 -- This task will help in understanding seasonal trends and planning inventory
80 -- and marketing efforts accordingly.
81
82 select saleid, productid, customerid, saledate, quantity, saleprice
83 from sales
84 where extract(year from saledate) = 2023
85 and extract(month from saledate) between 1 and 3
86 Union
87 -- sales in december
88 select saleid, productid, customerid, saledate, quantity, saleprice
89 from sales
90 -- where extract(year from saledate) = 2023
91 where extract(month from saledate) = 12
92
```




Data Output Messages Notifications



	saleid integer	productid integer	customerid integer	saledate date	quantity integer	saleprice numeric (10,2)
1	1294	63	11191	2023-12-16	2	400.00
2	2086	98	11198	2023-03-11	5	900.00
3	1789	34	11186	2023-02-18	3	15.00
4	1255	83	11190	2023-12-02	1	400.00
5	2056	94	11099	2023-02-21	7	500.00

Total rows: 210 of 210 Query complete 00:00:00.099

Ln 1


 SQL CLASS/postgres@PostgreSQL 16
 









 No limit
 













[Query](#)
[Query History](#)

```

98
99 -- 5.Payment Method Insights:
100 -- Evaluate the average, minimum, and maximum sale amounts for each payment method
101 -- excluding sales below $50.
102 -- These insights can inform payment process improvements and promotional offers
103
104 select paymentmethod, avg (quantity * saleprice) as averagesaleamount,
105 min(quantity * saleprice) as minimumsaleamount,
106 max(quantity * saleprice) as maximumsaleamount
107 from sales
108 where quantity * saleprice >= 50
109 group by paymentmethod
110 order by paymentmethod
111
    
```

[Data Output](#)
[Messages](#)
[Notifications](#)















	paymentmethod  character varying (255)	averagesaleamount  numeric	minimumsaleamount  numeric	maximumsaleamount  numeric
1	Cash	2111.2537313432835821	50.00	10800.00
2	Credit Card	2313.0246913580246914	50.00	10000.00
3	Debit Card	2325.7590759075907591	50.00	15000.00

SQL CLASS/postgres@PostgreSQL 16



Query Query History

```
111
112 -- 6. Product Catalog Optimization:
113 -- Determine the number of unique product categories and
114 -- identify products with the word 'smart' in their names.
115 -- This task can guide product development and marketing strategies, focusing on trending products and categories
116
117 select count(distinct category) as uniquecategories
118 from products
119
120 select productname
121 from products
122 where productname ilike '%smart%'
123
```

Data Output Messages Notifications



	productname character varying (255)
1	Smart Thermostat
2	Smart Doorbell
3	Smart Thermostat
4	Smart LED Light Bulbs
5	ProWrite Smart Notepad
6	FitTrack Smart Exercise Bike

SQL CLASS/postgres@PostgreSQL 16



Query Query History

```
124 -- 7.Revenue Generation Analysis:
125 -- Create a view showing total sales and revenue generated by each product
126 -- then select products that have generated significant revenue.This analysis helps in identifying star products and optimiz
127
128 create view productsalessummary as
129 select s.productid,productname, count(saleid) as totalsales,
130 sum(quantity * saleprice) as totalrevenue
131 from sales s
132 join products pr
133 on s.productid = pr.productid
134 group by s.productid, productname;
135
136 select productid, productname , totalsales, totalrevenue
137 from productsalessummary
138 where totalrevenue > 10000
139
```

Data Output Messages Notifications



	productid integer	productname character varying (255)	totalsales bigint	totalrevenue numeric
1	84	Smartech Wireless Earbuds	15	40000.00
2	65	FitTrack Smart Exercise Bike	4	15000.00
3	82	Smartech Wireless Earbuds	11	22500.00
4	67	Smartech Wireless Earbuds	15	60000.00

Total rows: 45 of 45 Query complete 00:00:00.200



Ln 130, Col 42

SQL CLASS/postgres@PostgreSQL 16



Query Query History

```
142 -- 8.Stock Level Adjustment
143 -- Develop a stored procedure for updating stock levels by product category
144 -- This automated task will improve stock management efficiency, ensuring optimal inventory levels based on sales data
145
146 create or replace procedure update_stock_level(category_name VARCHAR, stock_increment INT)
147 Language plpgsql
148 as $$
149 begin update products
150 set stockquantity = stockquantity + stock_increment
151 where category = category_name;
152 end;
153 $$
154 call update_stock_level ('Electronics',10);
155 select * from products
156 where category = 'Electronics';
157
```

Data Output Messages Notifications



	productid [PK] integer	productname character varying (255)	price numeric (10,2)	category character varying (50)	stockquantity integer	description text
1	64	Smartech Wireless Earbuds	500.00	Electronics	155	High-quality wireless earbuds with noise-cancellation technology
2	67	Smartech Wireless Earbuds	800.00	Electronics	105	High-quality wireless earbuds with noise-cancellation technology
3	69	Smartech Wireless Earbuds	1000.00	Electronics	155	High-quality wireless earbuds with noise-cancellation technology

SQL CLASS/postgres@PostgreSQL 16

No limit

Query Query History

```
174 -- 9.Customer Spending Ranking:
175 -- Use window functions to rank customers based on their total spending and
176 -- calculate cumulative revenue per product category over time.
177 -- These tasks are vital for customer segmentation and targeted marketing campaigns
178
179 select s.customerid,firstname,lastname , sum(quantity * saleprice) as totalspending,
180 rank() over(order by sum(quantity *saleprice) desc) as spendingrank
181 from customers cu
182 join sales s
183 on cu.customerid = s.customerid
184 group by s.customerid, firstname, lastname
185 order by totalspending desc;
186
187
```

Data Output Messages Notifications

	customerid integer	firstname character varying (60)	lastname character varying (60)	totalspending numeric	spendingrank bigint
1	11085	EMILY	JOHNSON	36860.00	1
2	11050	ALAN	ZHENG	28690.00	2
3	11170	CAROL	HOWARD	27160.00	3
4	11101	ABBY	SAI	26820.00	4
5	11021	DESTINY	WILSON	26560.00	5

SQL CLASS/postgres@PostgreSQL 16



Query Query History

```
187
188 -- 10.Sales Performance Categorization:
189 -- Categorize each sale into 'Low', 'Medium', and 'High' intensity
190 -- based on the total sale amount using CASE statements.
191 -- This analysis will help in identifying sales patterns and adjusting sales strategies accordingly.
192
193 select category , saledate,
194 sum (quantity * saleprice) over(partition by category order by saledate) as cumulativerevenue
195 from sales s
196 join products pr
197 on s.productid = pr.productid
198 order by category, saledate desc
199
```

Data Output Messages Notifications



	category character varying (50)	saledate date	cumulativerevenue numeric
1	Accessories	2023-02-16	450.00
2	Accessories	2023-02-17	585.00
3	Accessories	2023-02-20	945.00
4	Accessories	2023-02-28	1395.00
5	Accessories	2023-03-06	4095.00

Total rows: 1000 of 1000 Query complete 00:00:00.096



Ln 198, Col 33