

1. SELECT: View sample rows

SELECT *

FROM retail_data

LIMIT 20;

The screenshot shows the pgAdmin 4 interface. The left sidebar displays a tree view of the database structure, including 'PostgreSQL 17', 'Databases (14)', and 'public'. The main pane shows a SQL query editor with the following query:

```
1 SELECT *
2 FROM retail_data
3 LIMIT 20;
```

Below the query editor, the 'Data Output' tab shows the results of the query. The table has 11 columns: invoice, character varying (20), stock_code, character varying (20), description, text, quantity, integer, invoice_date, timestamp without time zone, price, numeric (10,2), customer_id, character varying (20), and country, character varying (50). The results show 20 rows of data, including items like 'WHITE HANGING HEART T-LIGHT HOLDER', 'CREAM CUPID HEARTS COAT HANGER', and 'RED WOOLLY HOTTIE WHITE HEART'.

invoice	character varying (20)	stock_code	character varying (20)	description	text	quantity	integer	invoice_date	timestamp without time zone	price	numeric (10,2)	customer_id	character varying (20)	country	character varying (50)
1	536365	85123A		WHITE HANGING HEART T-LIGHT HOLDER		6		2010-12-01 08:26:00		2.55		17850		United Kingdom	
2	536365	71053		WHITE METAL LANTERN		6		2010-12-01 08:26:00		3.39		17850		United Kingdom	
3	536365	84406B		CREAM CUPID HEARTS COAT HANGER		8		2010-12-01 08:26:00		2.75		17850		United Kingdom	
4	536365	84029G		KNITTED UNION FLAG HOT WATER BOTTLE		6		2010-12-01 08:26:00		3.39		17850		United Kingdom	
5	536365	84029E		RED WOOLLY HOTTIE WHITE HEART.		6		2010-12-01 08:26:00		3.39		17850		United Kingdom	
6	536365	22752		SET 7 BAIUSHKA NESTING BOXES		2		2010-12-01 08:26:00		7.65		17850		United Kingdom	
7	536365	21730		GLASS STAR FROSTED T-LIGHT HOLDER		6		2010-12-01 08:26:00		4.25		17850		United Kingdom	
8	536366	22633		HAND WARMER UNION JACK		6		2010-12-01 08:28:00		1.85		17850		United Kingdom	
9	536366	22632		HAND WARMER RED POLKA DOT		6		2010-12-01 08:28:00		1.85		17850		United Kingdom	
10	536368	22960		JAM MAKING SET WITH JARS		6		2010-12-01 08:34:00		4.25		13047		United Kingdom	
11	536368	22913		RED COAT RACK PARS FASHION		3		2010-12-01 08:34:00		4.95		13047		United Kingdom	

2. WHERE: Filter rows based on conditions

SELECT *

FROM retail_data

WHERE country = 'United Kingdom'

AND quantity > 10

LIMIT 20;

The screenshot shows the pgAdmin 4 interface. The left sidebar displays a tree view of the database structure, including 'PostgreSQL 17', 'Databases (14)', and 'public'. The main pane shows a SQL query editor with the following query:

```
1 SELECT *
2 FROM retail_data
3 WHERE country = 'United Kingdom'
4 AND quantity > 10
5 LIMIT 20;
```

Below the query editor, the 'Data Output' tab shows the results of the query. The table has 11 columns: invoice, character varying (20), stock_code, character varying (20), description, text, quantity, integer, invoice_date, timestamp without time zone, price, numeric (10,2), customer_id, character varying (20), and country, character varying (50). The results show 20 rows of data, including items like 'ASSORTED COLOUR BIRD ORNAMENT', 'PAPER CHAIN KIT 50'S CHRISTMAS', and 'VICTORIAN SEWING BOX LARGE'.

invoice	character varying (20)	stock_code	character varying (20)	description	text	quantity	integer	invoice_date	timestamp without time zone	price	numeric (10,2)	customer_id	character varying (20)	country	character varying (50)
1	536367	84879		ASSORTED COLOUR BIRD ORNAMENT		32		2010-12-01 08:34:00		1.69		13047		United Kingdom	
2	536371	22086		PAPER CHAIN KIT 50'S CHRISTMAS		80		2010-12-01 09:00:00		2.55		13748		United Kingdom	
3	536374	21258		VICTORIAN SEWING BOX LARGE		32		2010-12-01 09:09:00		10.95		15100		United Kingdom	
4	536376	22114		HOT WATER BOTTLE TEA AND SYMPATHY		48		2010-12-01 09:32:00		3.45		15291		United Kingdom	
5	536376	21733		RED HANGING HEART T-LIGHT HOLDER		64		2010-12-01 09:32:00		2.55		15291		United Kingdom	
6	536378	84997B		RED 3 PIECE RETROSPOT CUTLERY SET		12		2010-12-01 09:37:00		3.75		14688		United Kingdom	
7	536378	21094		SET/6 RED SPOTTY PAPER PLATES		12		2010-12-01 09:37:00		0.85		14688		United Kingdom	
8	536378	21212		PACK OF 72 RETROSPOT CAKE CASES		120		2010-12-01 09:37:00		0.42		14688		United Kingdom	
9	536378	21975		PACK OF 60 DINOSAUR CAKE CASES		24		2010-12-01 09:37:00		0.55		14688		United Kingdom	
10	536378	21977		PACK OF 60 PINK PANSLEY CAKE CASES		24		2010-12-01 09:37:00		0.55		14688		United Kingdom	
11	536378	84991		60 TEATIME FAIRY CAKE CASES		24		2010-12-01 09:37:00		0.55		14688		United Kingdom	

-- 3. Aggregate Functions: Data Quality Summary

SELECT

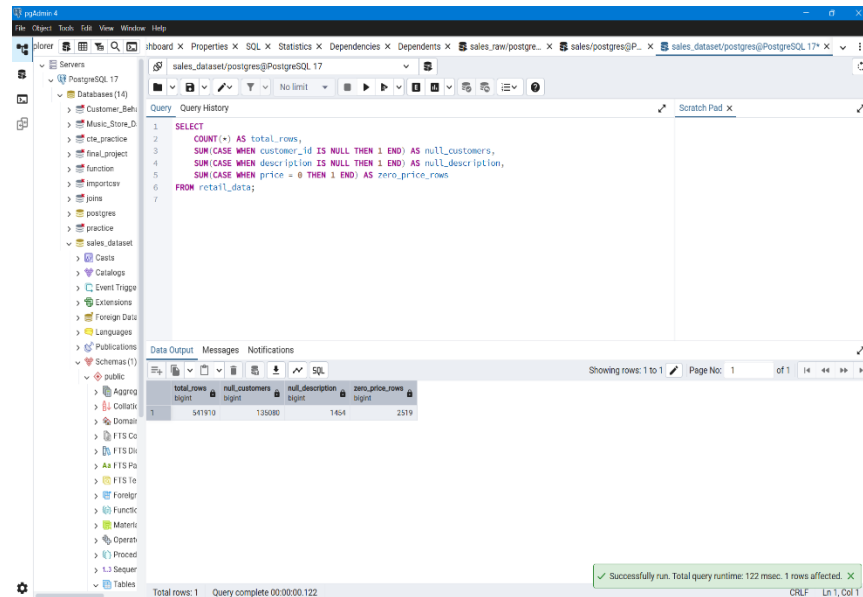
COUNT(*) AS total_rows,

SUM(CASE WHEN customer_id IS NULL THEN 1 END) AS null_customers,

SUM(CASE WHEN description IS NULL THEN 1 END) AS null_description,

SUM(CASE WHEN price = 0 THEN 1 END) AS zero_price_rows

FROM retail_data;



The screenshot shows the pgAdmin 4 interface. The SQL query editor contains the following query:

```
1 SELECT
2   COUNT(*) AS total_rows,
3   SUM(CASE WHEN customer_id IS NULL THEN 1 END) AS null_customers,
4   SUM(CASE WHEN description IS NULL THEN 1 END) AS null_description,
5   SUM(CASE WHEN price = 0 THEN 1 END) AS zero_price_rows
6 FROM retail_data;
```

The Data Output pane shows the results of the query:

	total_rows bigint	null_customers bigint	null_description bigint	zero_price_rows bigint
1	541910	130080	1464	2519

At the bottom, a status bar indicates: "Total rows: 1 Query complete 00:00:00.122" and a green message box says "Successfully run. Total query runtime: 122 msec. 1 rows affected."

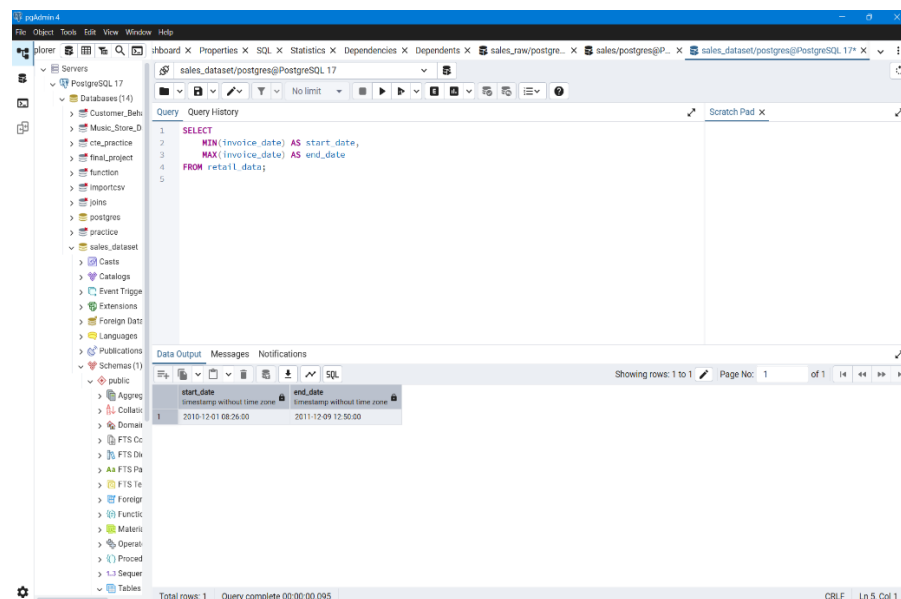
-- Date Range Using MIN and MAX

SELECT

MIN(invoice_date) AS start_date,

MAX(invoice_date) AS end_date

FROM retail_data;



The screenshot shows the pgAdmin 4 interface. The SQL query editor contains the following query:

```
1 SELECT
2   MIN(invoice_date) AS start_date,
3   MAX(invoice_date) AS end_date
4 FROM retail_data;
```

The Data Output pane shows the results of the query:

	start_date timestamp without time zone	end_date timestamp without time zone
1	2010-12-01 08:26:00	2011-12-09 12:50:00

At the bottom, a status bar indicates: "Total rows: 1 Query complete 00:00:00.095" and a status bar on the right shows "CRLF Ln 5, Col 1".

-- 4. GROUP BY: Revenue per Country

```
SELECT

    country,

    ROUND(SUM(quantity * price), 2) AS revenue

FROM retail_data

GROUP BY country

ORDER BY revenue DESC;
```

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database structure, including the 'sales_dataset' database and its 'public' schema. The central pane shows a SQL query being executed. The bottom pane displays the query results in a table format. A status bar at the bottom indicates the query was successful and affected 38 rows.

	country	revenue
1	United Kingdom	8187806.36
2	Netherlands	284661.54
3	EIRE	263276.82
4	Germany	221698.21
5	France	197421.90
6	Australia	137077.27
7	Switzerland	56385.35
8	Spain	54774.58
9	Belgium	40910.96
10	Sweden	36595.91
11	Japan	35340.62

Success/ully run. Total query runtime: 190 msec. 38 rows affected.

-- 5. Subquery: Top 5 Customers by Total Spending

```
SELECT customer_id, total_spent

FROM (

    SELECT

        customer_id,

        SUM(quantity * price) AS total_spent

    FROM retail_data

    WHERE customer_id IS NOT NULL

    GROUP BY customer_id

) AS t

ORDER BY total_spent DESC

LIMIT 5;
```

The screenshot shows the pgAdmin 4 interface with a SQL query executed in the 'Query' tab. The query is as follows:

```

1 SELECT customer_id, total_spent
2 FROM (
3     SELECT
4         customer_id,
5         SUM(quantity * price) AS total_spent
6     FROM retail_data
7     WHERE customer_id IS NOT NULL
8     GROUP BY customer_id
9 ) AS t
10 ORDER BY total_spent DESC
11 LIMIT 5;

```

The 'Data Output' tab shows the results of the query:

	customer_id	total_spent
1	14646	279489.02
2	18102	256438.49
3	17450	187482.17
4	14911	132572.62
5	12415	123725.45

The status bar at the bottom indicates: 'Total rows: 5 Query complete 00:00:00.164' and a green message box says 'Successfully run. Total query runtime: 164 msec. 5 rows affected.'

-- 6.1 View: Monthly Revenue

CREATE VIEW monthly_revenue AS

SELECT

DATE_TRUNC('month', invoice_date) AS month,

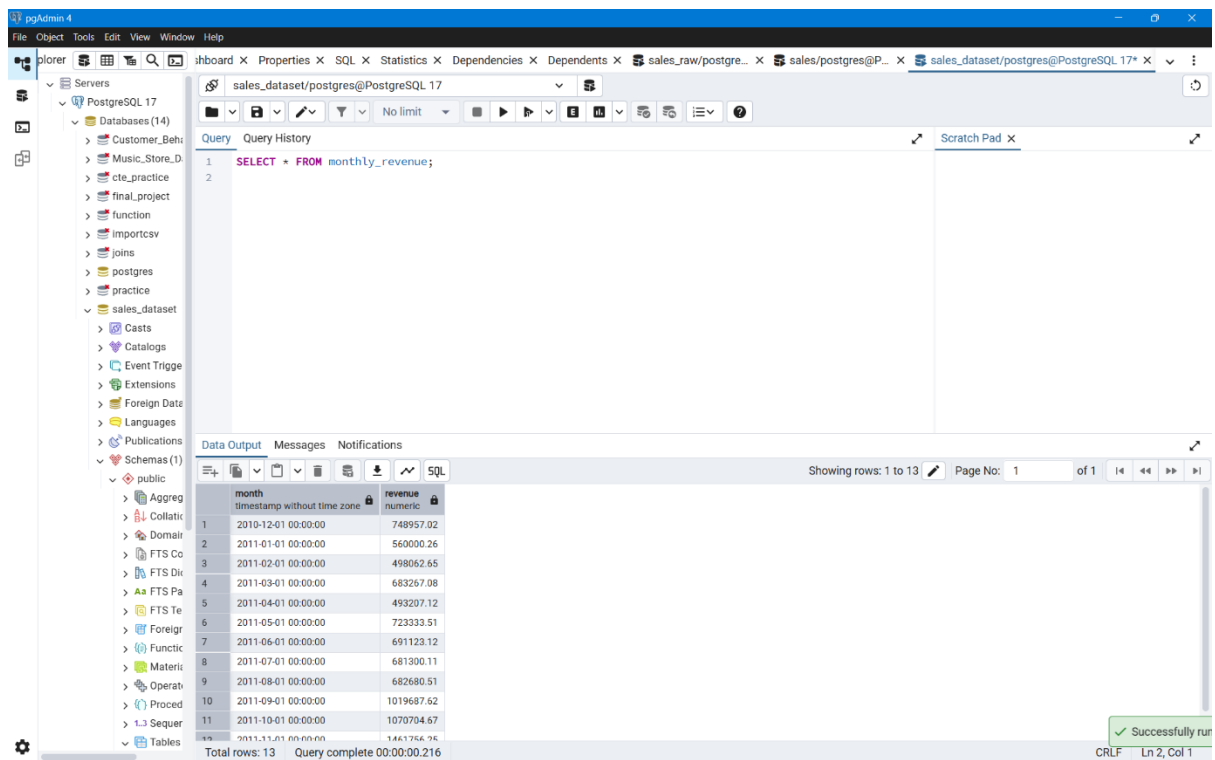
SUM(quantity * price) AS revenue

FROM retail_data

GROUP BY month

ORDER BY month;

SELECT * FROM monthly_revenue;



-- 1. INNER JOIN: Show products that have a category

SELECT

 r.stock_code,

 r.description,

 r.quantity,

 r.price,

 p.category

FROM retail_data r

INNER JOIN product_category p

 ON r.stock_code = p.stock_code

LIMIT 20;

The screenshot shows the pgAdmin 4 interface with a PostgreSQL 17 database. The left sidebar displays the database structure, including schemas and tables. The main window shows a SQL query in the 'Query' tab, which is a SELECT statement joining 'retail_data' and 'product_category' tables. The 'Data Output' tab shows the results of the query, displaying 20 rows of product information. The status bar at the bottom indicates that the query was successfully run, affecting 20 rows in 64 milliseconds.

```

SELECT
  r.stock_code,
  r.description,
  r.quantity,
  r.price,
  p.category
FROM retail_data r
INNER JOIN product_category p
  ON r.stock_code = p.stock_code
LIMIT 20;

```

stock_code	description	quantity	price	category
84029E	RED WOOLLY HOTTIE WHITE HEART.	6	3.39	Toys
84879	ASSORTED COLOUR BIRD ORNAME...	32	1.69	Cosmetics
84029E	RED WOOLLY HOTTIE WHITE HEART.	6	3.39	Toys
84029E	RED WOOLLY HOTTIE WHITE HEART.	6	3.39	Toys
22386	JUMBO BAG PINK POLKADOT	10	1.95	Kitchen
85099B	JUMBO BAG RED RETROSPOT	100	1.65	Home Decor
85099B	JUMBO BAG RED RETROSPOT	100	1.65	Home Decor
84879	ASSORTED COLOUR BIRD ORNAME...	16	1.69	Cosmetics
84879	ASSORTED COLOUR BIRD ORNAME...	32	1.69	Cosmetics
84029E	RED WOOLLY HOTTIE WHITE HEART.	6	3.39	Toys
84029E	RED WOOLLY HOTTIE WHITE HEART.	6	3.39	Toys

Successfully run. Total query runtime: 64 msec. 20 rows affected.

-- 2. LEFT JOIN: Keep all products, add category if available

SELECT

r.stock_code,

r.description,

r.quantity,

r.price,

p.category

FROM retail_data r

LEFT JOIN product_category p

ON r.stock_code = p.stock_code

LIMIT 20;

The screenshot shows the pgAdmin 4 interface with a SQL query executed in the 'Query History' tab. The query is a LEFT JOIN between 'retail_data' and 'product_category' on 'stock_code'. The results are displayed in the 'Data Output' tab, showing 20 rows of data.

Query:

```

SELECT
  r.stock_code,
  r.description,
  r.quantity,
  r.price,
  p.category
FROM retail_data r
LEFT JOIN product_category p
  ON r.stock_code = p.stock_code
LIMIT 20;

```

Data Output:

stock_code	description	quantity	price	category
22243	5 HOOK HANGER RED MAGIC TOADSTOOL	12	1.65	[null]
47421	ASSORTED COLOUR LIZARD SUCTION HOOK	24	0.42	[null]
20712	JUMBO BAG WOODLAND ANIMALS	10	1.95	[null]
20713	JUMBO BAG OWLS	10	1.95	[null]
22837	HOT WATER BOTTLE BABUSHKA	4	4.65	[null]
22969	HOMEMADE JAM SCENTED CANDLES	12	1.45	[null]
22973	CHILDREN'S CIRCUS PARADE MUG	12	1.65	[null]
84569B	PACK 3 FIRE ENGINE/CAR PATCHES	12	1.25	[null]
22549	PICTURE DOMINOES	12	1.45	[null]
POST	POSTAGE	1	18.00	[null]
22815	CARD PSYCHEDELIC APPLES	12	0.42	[null]
22865	MAINT. WARMED OVER DESIGN	1	2.10	[null]

Successfully run. Total query runtime: 90 msec. 20 rows affected.

-- 3. RIGHT JOIN: Show all categories, even if no sales exist

SELECT

r.stock_code,

r.description,

r.quantity,

r.price,

p.category

FROM retail_data r

RIGHT JOIN product_category p

ON r.stock_code = p.stock_code

LIMIT 20;

The screenshot shows the pgAdmin 4 interface with a PostgreSQL 17 database. The left sidebar displays the database structure, including schemas like 'public'. The central pane shows a SQL query in the 'Query' editor:

```

1  SELECT
2      r.stock_code,
3      r.description,
4      r.quantity,
5      r.price,
6      p.category
7  FROM retail_data r
8  RIGHT JOIN product_category p
9      ON r.stock_code = p.stock_code
10 LIMIT 20;
11

```

The 'Data Output' pane at the bottom displays the results of the query as a table with 20 rows. The columns are: stock_code, description, quantity, price, and category. The status bar at the bottom indicates 'Total rows: 20' and 'Query complete 00:00:00.056'. A green message box at the bottom right states: 'Successfully run. Total query runtime: 56 msec. 20 rows affected.'

stock_code	description	quantity	price	category
84879	ASSORTED COLOUR BIRD ORNAME...	80	1.69	Cosmetics
22386	JUMBO BAG PINK POLKADOT	30	1.95	Kitchen
85099B	JUMBO BAG RED RETROSPOT	40	1.95	Home Decor
84029E	RED WOOLLY HOTTIE WHITE HEART.	1	7.62	Toys
85099B	JUMBO BAG RED RETROSPOT	1	4.21	Home Decor
84029E	RED WOOLLY HOTTIE WHITE HEART.	1	3.75	Toys
85099B	JUMBO BAG RED RETROSPOT	70	1.65	Home Decor
84029E	RED WOOLLY HOTTIE WHITE HEART.	96	2.95	Toys
85099B	JUMBO BAG RED RETROSPOT	30	1.65	Home Decor
84029E	RED WOOLLY HOTTIE WHITE HEART.	200	1.65	Home Decor
84029E	RED WOOLLY HOTTIE WHITE HEART.	384	2.95	Toys
84029E	RED WOOLLY HOTTIE WHITE HEART.	1	1.05	Kitchen

-- 4. FULL OUTER JOIN: All rows from both tables

SELECT

r.stock_code,

r.description,

r.quantity,

r.price,

p.category

FROM retail_data r

FULL OUTER JOIN product_category p

ON r.stock_code = p.stock_code;

The screenshot displays the pgAdmin 4 web interface. On the left, a tree view shows the database structure, including a 'sales_dataset' database. The main pane shows a SQL query window with the following query:

```

1 SELECT
2     r.stock_code,
3     r.description,
4     r.quantity,
5     r.price,
6     p.category
7 FROM retail_data r
8 FULL OUTER JOIN product_category p
9     ON r.stock_code = p.stock_code;
10

```

Below the query window, the 'Data Output' tab shows the results of the query. The results are displayed in a table with the following columns: stock_code, description, quantity, price, and category. The table shows 11 rows of data, including items like 'BLUE WHITE PLASTIC RINGS LAMPSHADE' and '72 SWEETHEART FAIRY CAKE CASES'.

	stock_code	description	quantity	price	category
1	440928	BLUE WHITE PLASTIC RINGS LAMPSHADE	3	0.85	[null]
2	440928	BLUE WHITE PLASTIC RINGS LAMPSHADE	3	0.85	[null]
3	440928	BLUE WHITE PLASTIC RINGS LAMPSHADE	1	0.85	[null]
4	440928	BLUE WHITE PLASTIC RINGS LAMPSHADE	1	0.85	[null]
5	440928	BLUE WHITE PLASTIC RINGS LAMPSHADE	6	0.85	[null]
6	84992	72 SWEETHEART FAIRY CAKE CASES	1	1.28	[null]
7	84992	72 SWEETHEART FAIRY CAKE CASES	1	1.28	[null]
8	84992	72 SWEETHEART FAIRY CAKE CASES	24	0.55	[null]
9	84992	72 SWEETHEART FAIRY CAKE CASES	24	0.55	[null]
10	84992	72 SWEETHEART FAIRY CAKE CASES	24	0.55	[null]
11	84992	72 SWEETHEART FAIRY CAKE CASES	2	0.55	[null]

The bottom status bar indicates 'Total rows: 541910' and 'Query complete 00:00:00.556'.

Conclusion

SQL analysis was performed successfully across filtering, grouping, joins, subqueries, view creation, and optimization.