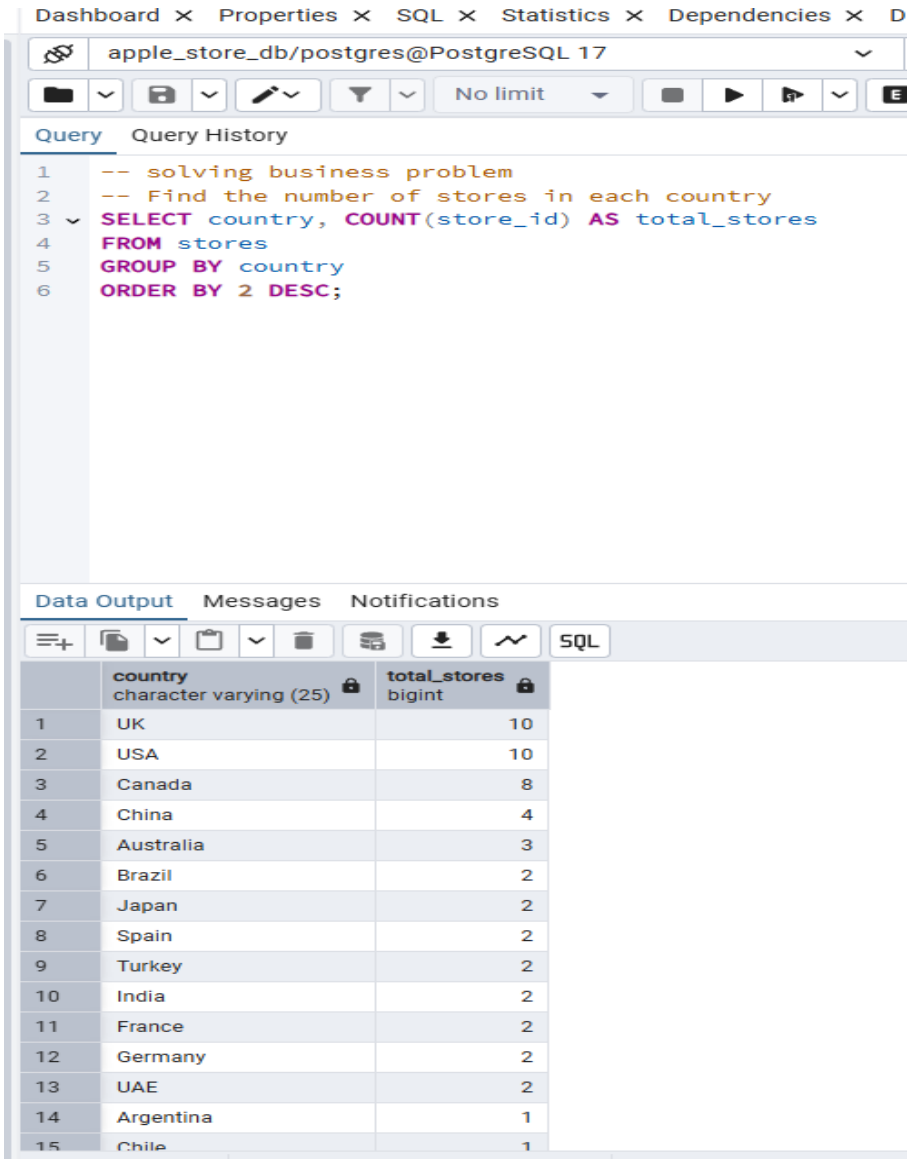
Used PostgreSQL database to solve 20+ critical problems

- Find the number of stores in each country.

SELECT country, COUNT(store_id) AS total_stores
FROM stores
GROUP BY country
ORDER BY 2 DESC;



	country character varying (25)	total_stores bigint
1	UK	10
2	USA	10
3	Canada	8
4	China	4
5	Australia	3
6	Brazil	2
7	Japan	2
8	Spain	2
9	Turkey	2
10	India	2
11	France	2
12	Germany	2
13	UAE	2
14	Argentina	1
15	Chile	1

- Calculate the total number of units sold by each store.

SELECT
str.store_id,
str.store_name,
SUM(quantity) as total_units
FROM sales sl
INNER JOIN stores str
ON str.store_id = sl.store_id
GROUP BY str.store_id,str.store_name -- or group by 1,2
ORDER BY total_units Desc; --order by 3

```sql
-- Calculate the total number of units sold by each store.
SELECT
str.store_id,
str.store_name,
SUM(quantity) as total_units
FROM sales sl
INNER JOIN stores str
ON str.store_id = sl.store_id
GROUP BY str.store_id,str.store_name -- or group by 1,2
ORDER BY total_units Desc; --order by 3
```

Data Output Messages Notifications

	store_id [PK] character varying (10)	store_name character varying (30)	total_units bigint
1	ST-5	Apple South Coast Plaza	47498
2	ST-3	Apple Michigan Avenue	44891
3	ST-4	Apple The Grove	44784
4	ST-2	Apple Union Square	44395
5	ST-1	Apple Fifth Avenue	44367
6	ST-12	Apple Yorkdale	29882
7	ST-14	Apple Pacific Centre	29595
8	ST-9	Apple Lennox Town Cent...	29536
9	ST-6	Apple Scottsdale	29504
10	ST-8	Apple Eastview Mall	29451
11	ST-13	Apple Square One	29357
12	ST-11	Apple Eaton Centre	29297
13	ST-10	Apple Tysons Corner	29215
14	ST-7	Apple Mall of America	29196

- Identify how many sales occurred in December 2023

SELECT
COUNT (sale_id) as total_sales
FROM sales
WHERE TO_CHAR(sale_date,'MM-YYYY') = '12-2023';

```
-- Identify how many sales occurred in December 2023.
SELECT
COUNT (sale_id) as total_sales
FROM sales
WHERE TO_CHAR(sale_date,'MM-YYYY') = '12-2023';
```

Data Output Messages Notifications

	total_sales bigint
1	32846

- Calculate the percentage of warranty claims marked as "Warranty Void".

SELECT
ROUND
(COUNT(claim_id)/(SELECT COUNT(*) FROM warranty)::numeric * 100,2) as warranty_void_percentage
FROM warranty
WHERE repair_status = 'Warranty Void'

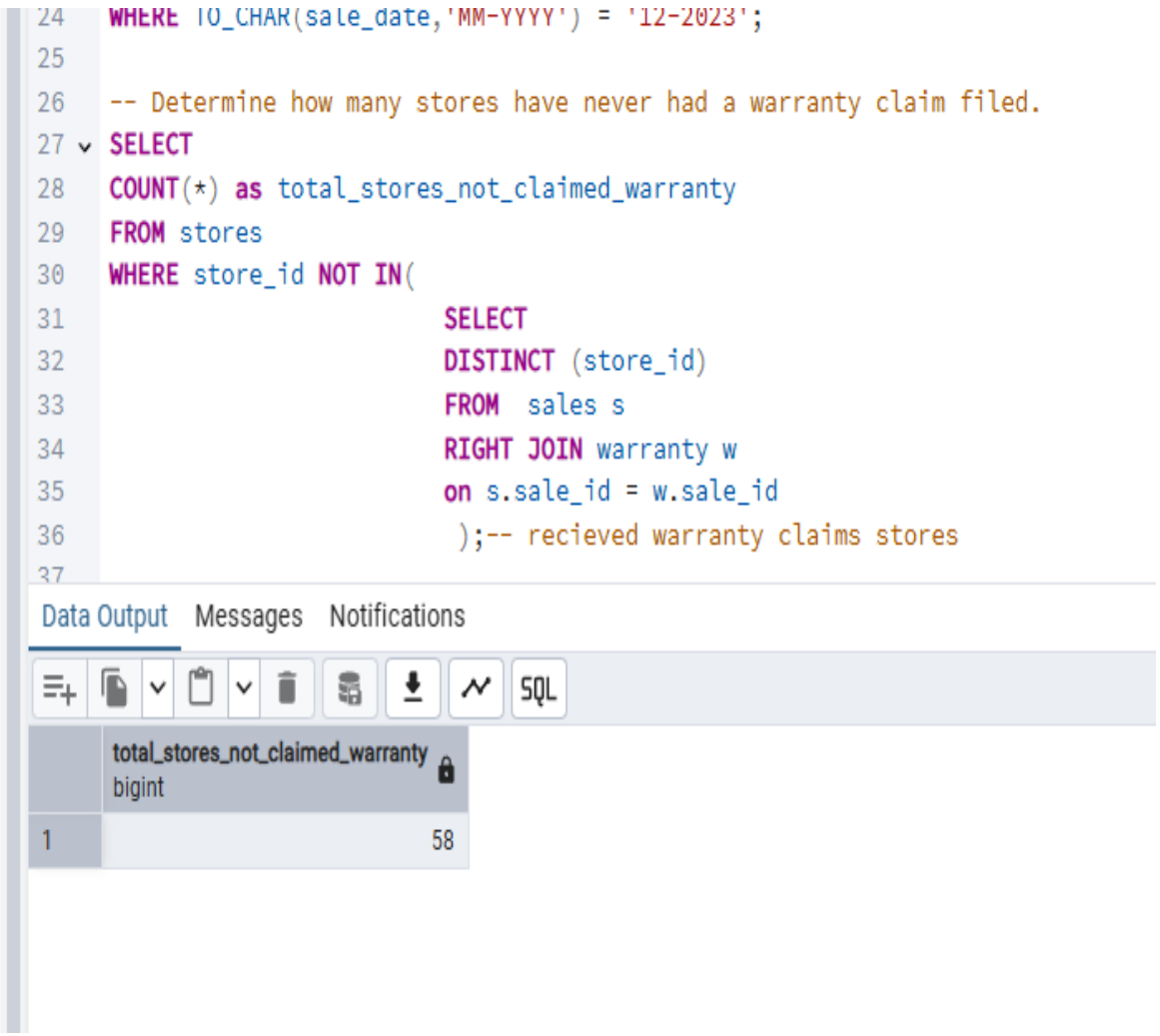
```
-- Calculate the percentage of warranty claims marked as "Warranty Void".
SELECT
ROUND
(COUNT(claim_id)/(SELECT COUNT(*) FROM warranty)::numeric * 100,2) as warranty_void_percentage
FROM warranty
WHERE repair_status = 'Warranty Void'
```

Data Output Messages Notifications

	warranty_void_percentage numeric
1	23.16

- Determine how many stores have never had a warranty claim filed.

```
SELECT
COUNT(*) as total_stores_not_claimed_warranty
FROM stores
WHERE store_id NOT IN(
                        SELECT
                        DISTINCT (store_id)
                        FROM  sales s
                        RIGHT JOIN warranty w
                        on s.sale_id = w.sale_id
                         );-- recieved warranty claims stores
```



- Identify which store had the highest total units sold in the last year.

```
SELECT st.store_id, st.store_name, SUM (sl.quantity) as highest_unit_sold
FROM sales as sl
JOIN stores as st
ON st.store_id = sl.store_id
WHERE sale_date >= (CURRENT_DATE - INTERVAL '1 year')
GROUP BY 1,2
ORDER BY 3 DESC
LIMIT 1
```

```
-- Identify which store had the highest total units sold in the last year.

SELECT st.store_id, st.store_name, SUM (sl.quantity) as highest_unit_sold
FROM sales as sl
JOIN stores as st
ON st.store_id = sl.store_id
WHERE sale_date >= (CURRENT_DATE - INTERVAL '1 year')
GROUP BY 1,2
ORDER BY 3 DESC
LIMIT 1
```

Data Output Messages Notifications

	store_id [PK] character varying (10)	store_name character varying (30)	highest_unit_sold bigint
1	ST-54	Apple Ankara	127

- Count the number of unique products sold in the2 year.

SELECT
COUNT(DISTINCT(product_id))as unique_products
FROM sales
WHERE sale_date >= (Current_date - Interval '2 year');

```
LIMIT 1

-- Count the number of unique products sold in the last 2 year.
SELECT
COUNT(DISTINCT(product_id) )as unique_products
FROM sales
WHERE sale_date >= (Current_date - Interval '2 year');
```

Data Output Messages Notifications

	unique_products bigint
1	50

- Find the average price of products in each category.

```
SELECT c.category_id, c.category_name, AVG(p.price) as avg_price
  FROM category as c
  JOIN products as p
  ON c.category_id = p.category_id
  GROUP BY 1,2
  ORDER BY 3 DESC
```

```
-- Find the average price of products in each category.
SELECT c.category_id, c.category_name, AVG(p.price) as avg_price

  FROM category as c
  JOIN products as p
  ON c.category_id = p.category_id
  GROUP BY 1,2
  ORDER BY 3 DESC
```

Data Output Messages Notifications

	category_id [PK] character varying (10)	category_name character varying (20)	avg_price double precision
1	CAT-1	Laptop	1511.5
2	CAT-7	Desktop	1199
3	CAT-4	Smartphone	889.4347826086956
4	CAT-3	Tablet	656.5
5	CAT-5	Wearable	362.3333333333333
6	CAT-2	Audio	259
7	CAT-6	Streaming Device	179
8	CAT-8	Subscription Service	149.5
9	CAT-9	Smart Speaker	99
10	CAT-10	Accessory	29

- How many warranty claims were filed in 2020?
  SELECT COUNT (claim_id)
  FROM warranty
  WHERE TO_CHAR (claim_date,'YYYY')= '2020';
  -- or we can write WHERE EXTRACT(YEAR FROM claim_date)= 2020

```sql
-- Calculate how many warranty claims were filed within 180 days of a product sale.
SELECT COUNT (w.*)
FROM warranty as w
LEFT JOIN sales as s
ON s.sale_id = w.sale_id
WHERE w.claim_date - s.sale_date <= 180
```

Data Output Messages Notifications

	count bigint
1	19907

- For each store, identify the best-selling day based on highest quantity sold.

using cte and window function

```
WITH my_cte AS(
        SELECT store_id, SUM (quantity) AS total_unit_sold,
TO_CHAR(sale_date, 'Day') AS day_name,
                RANK() OVER(PARTITION BY store_id ORDER BY
SUM(quantity) DESC) AS rank
                FROM sales
                GROUP BY 1, 3
    )
SELECT  mc.store_id, st.store_name, mc.day_name, mc.total_unit_sold
FROM my_cte as mc
JOIN stores as st
ON mc.store_id = st.store_id
WHERE rank = 1
```

```
-- For each store, identify the best-selling day based on highest quantity sold.
-- using cte and window function
WITH my_cte AS(
                SELECT store_id, SUM (quantity) AS total_unit_sold, TO_CHAR(sale_date, 'Day') AS day_name,
                RANK() OVER(PARTITION BY store_id ORDER BY SUM(quantity) DESC) AS rank
                FROM sales
                GROUP BY 1, 3
              )
SELECT  mc.store_id, st.store_name, mc.day_name, mc.total_unit_sold
FROM my_cte as mc
JOIN stores as st
ON mc.store_id = st.store_id
WHERE rank = 1
```

Data Output Messages Notifications

	store_id character varying (10)	store_name character varying (30)	day_name text	total_unit_sold bigint
1	ST-1	Apple Fifth Avenue	Thursday	6830
2	ST-10	Apple Tysons Corner	Sunday	4432
3	ST-11	Apple Eaton Centre	Monday	4314
4	ST-12	Apple Yorkdale	Sunday	4539
5	ST-13	Apple Square One	Monday	4400
6	ST-14	Apple Pacific Centre	Thursday	4366
7	ST-15	Apple Chinook Centre	Monday	2879
8	ST-16	Apple Rideau Centre	Monday	2940
9	ST-17	Apple West Edmonton M...	Thursday	2837
10	ST-18	Apple CF Sherway Garde...	Sunday	2865
11	ST-19	Apple Marunouchi	Monday	2985
12	ST-2	Apple Union Square	Thursday	6614
13	ST-20	Apple Shibuya	Sunday	2864
14	ST-21	Apple Orchard Road	Monday	3522
15	ST-22	Apple Regent Street	Monday	3388
16	ST-23	Apple Covent Garden	Monday	3416

Total rows: 70 Query complete 00:00:00.567

- Identify the least selling product in each country for each year based on total units sold.

```
WITH my_cte as (

        SELECT sl.product_id, p.product_name,st.country, EXTRACT (YEAR FROM
sl.sale_date) AS each_year, SUM(sl.quantity) AS total_sold_products,
                RANK () OVER(PARTITION BY st.country,EXTRACT (YEAR FROM
sl.sale_date) ORDER BY SUM (sl.quantity)ASC) AS rank
                FROM stores as st
                JOIN sales as sl
                ON st.store_id = sl.store_id
                JOIN products as p
                ON sl.product_id = p.product_id
                GROUP BY 1,2,3,4
)

SELECT product_name, country, each_year,total_sold_products
FROM my_cte
WHERE rank = 1
```

Data Output Messages Notifications

	product_name character varying (35)	country character varying (25)	each_year numeric	total_sold_products bigint
1	iPhone 13 Mini	Argentina	2021	28
2	Apple TV 4K	Argentina	2022	5
3	iPad Pro (M1, 11-inch)	Argentina	2022	5
4	iPhone 15 Pro Max	Argentina	2023	139
5	iPhone 15	Argentina	2024	60
6	Apple Fitness+	Australia	2020	37
7	iPhone 12 Mini	Australia	2021	28
8	Mac mini (M1)	Australia	2021	28
9	iMac (24-inch, M1)	Australia	2022	22
10	iPad Pro (M1, 12.9-inch)	Australia	2022	22
11	Apple TV 4K	Australia	2022	22
12	iPhone 15	Australia	2023	400

Total rows: 188 Query complete 00:00:00.676

- Calculate how many warranty claims were filed within 180 days of a product sale.

SELECT COUNT (w.*)

FROM warranty as w

LEFT JOIN sales as s

ON s.sale_id = w.sale_id

WHERE w.claim_date - s.sale_date <= 180

```sql
-- Calculate how many warranty claims were filed within 180 days of a product sale.
SELECT COUNT (w.*)
FROM warranty as w
LEFT JOIN sales as s
ON s.sale_id = w.sale_id
WHERE w.claim_date - s.sale_date <= 180
```

Data Output Messages Notifications

	count bigint
1	19907

- Determine how many warranty claims were filed for products launched in the last three years.

```
SELECT p.product_name, COUNT(w.claim_id),COUNT(s.sale_id)
FROM warranty as w
RIGHT JOIN
sales as s
ON w.sale_id = s.sale_id
JOIN
products as p
ON s.product_id = p.product_id
WHERE p.launch_date >= CURRENT_DATE - INTERVAL'3 years'
GROUP BY 1
HAVING COUNT (w.claim_id)> 0
```

```
-- Determine how many warranty claims were filed for products launched in the last three years.

SELECT p.product_name, COUNT(w.claim_id),COUNT(s.sale_id)
 FROM warranty as w
 RIGHT JOIN
 sales as s
 ON w.sale_id = s.sale_id
 JOIN
 products as p
 ON s.product_id = p.product_id
 WHERE p.launch_date >= CURRENT_DATE - INTERVAL'3 years'
 GROUP BY 1
 HAVING COUNT (w.claim_id)> 0
```

Data Output Messages Notifications

	product_name character varying (35)	count bigint	count bigint
1	AirPods Pro (2nd Gen)	913	26742
2	iPad (10th Gen)	685	26648
3	iPad Pro (M2, 11-inch)	716	26723
4	iPhone 14	1321	41131
5	iPhone 14 Pro	1260	40866
6	iPhone 15	321	21547
7	iPhone 15 Pro	290	21861
8	iPhone 15 Pro Max	320	21661

- List the months in the last three years where sales exceeded 5,000 units in the USA.

SELECT TO_CHAR (sl.sale_date, 'Month') AS listing_months, SUM(sl.quantity) AS total_units_sold
FROM stores as st
JOIN
sales as sl
ON st.store_id = sl.store_id
WHERE sl.sale_date >= CURRENT_DATE - INTERVAL '3 years' AND st.country = 'USA'
GROUP BY 1
HAVING SUM(sl.quantity)>5000

```
SELECT  TO_CHAR (sl.sale_date, 'Month') AS listing_months,  SUM(sl.quantity) AS total_units_sold
FROM stores as st
JOIN
sales as sl
ON st.store_id = sl.store_id
WHERE sl.sale_date >= CURRENT_DATE - INTERVAL '3 years' AND st.country = 'USA'
GROUP BY 1
HAVING SUM(sl.quantity)>5000
```

Data Output Messages Notifications

	listing_months text	total_units_sold bigint
1	December	12069
2	March	5036
3	November	12593
4	October	12316
5	September	10965

- Identify the product category with the most warranty claims filed in the last two years.

```
SELECT c.category_name, COUNT (w.claim_id) as total_claims_filed
FROM warranty as w
LEFT JOIN sales as s
ON w.sale_id = s.sale_id
JOIN products as p
ON s.product_id = p.product_id
JOIN category as c
ON p.category_id = c.category_id
 WHERE w.claim_date >= CURRENT_DATE - INTERVAL '3 years'
 GROUP BY 1
 ORDER BY 2 DESC
```

```
-- Identify the product category with the most warranty claims filed in the last three years.

SELECT c.category_name, COUNT (w.claim_id) as total_claims_filed
FROM warranty as w
LEFT JOIN sales as s
ON w.sale_id = s.sale_id
JOIN products as p
ON s.product_id = p.product_id
JOIN category as c
ON p.category_id = c.category_id
 WHERE w.claim_date >= CURRENT_DATE - INTERVAL '3 years'
 GROUP BY 1
 ORDER BY 2 DESC
```

Data Output Messages Notifications

	category_name character varying (20)	total_claims_filed bigint
1	Smartphone	9364
2	Tablet	3457
3	Accessory	1212
4	Wearable	1008
5	Audio	953
6	Subscription Service	945
7	Laptop	595
8	Streaming Device	82
9	Desktop	79

- Determine the percentage chance of receiving warranty claims after each purchase for each country.

```
WITH my_cte AS (
        SELECT
                st.country AS country,
                COUNT(w.claim_id) AS total_claims,
        SUM (sl.quantity) AS total_sales
                FROM stores as st
                JOIN sales as sl
                ON st.store_id = sl.store_id
                LEFT JOIN
                warranty as w
                ON sl.sale_id = w.sale_id
                GROUP BY 1
                )

SELECT country,total_claims,total_sales,
ROUND(COALESCE(total_claims::numeric/total_sales::numeric*100,0),2) AS
percentage_warrenty_claims
FROM my_cte
ORDER BY 4 DESC
```

173

Data Output Messages Notifications

	country character varying (25)	total_claims bigint	total_sales bigint	percentage_warrenty_claims numeric
1	UAE	11816	17939	65.87
2	Spain	6052	21978	27.54
3	Italy	1691	14631	11.56
4	Turkey	6157	54255	11.35
5	India	2241	54919	4.08
6	UK	1740	71720	2.43
7	Germany	641	37573	1.71
8	France	335	29218	1.15
9	Netherlands	163	14863	1.10
10	Indonesia	0	8173	0.00
11	Japan	0	38531	0.00
12	Malaysia	0	8232	0.00

- Analyze the year-by-year growth ratio for each store.

```
WITH yearly_sales AS(        SELECT   st.store_id AS store_id,
                                      st.store_name AS store_name,
                                      EXTRACT( YEAR FROM sl.sale_date) AS years,
                                      SUM (p.price * sl.quantity) AS total_sales
                                      FROM stores as st
                                      JOIN
                                      sales as sl
                                      ON st.store_id = sl.store_id
                                      JOIN
                                      products as p
                                      ON sl.product_id = p.product_id
                                      GROUP BY 1,2,3
                                      ORDER BY 1,2 ),
                              growth_ratio AS    (
      SELECT  store_name,  years,LAG(total_sales, 1) OVER (PARTITION BY store_name ORDER
BY years ASC ) AS last_year_sales,
                 total_sales AS current_year_sales
                 FROM yearly_sales
                 )  SELECT *,
                ROUND ((current_year_sales - last_year_sales ):: numeric/
                    last_year_sales:: numeric* 100 ,3) AS YOY_growth_ratio
               FROM growth_ratio
               WHERE last_year_sales IS NOT NULL
               AND years <>EXTRACT (YEAR FROM CURRENT_DATE)
```

Data Output Messages Notifications

	store_name character varying (30)	years numeric	last_year_sales double precision	current_year_sales double precision	yoy_growth_ratio numeric
1	Apple Amsterdam	2020	1938674	2057521	6.130
2	Apple Amsterdam	2021	2057521	3011823	46.381
3	Apple Amsterdam	2022	3011823	3580638	18.886
4	Apple Amsterdam	2023	3580638	1454241	-59.386
5	Apple Amsterdam	2024	1454241	186817	-87.154
6	Apple Ankara	2021	1898618	2451685	29.130
7	Apple Ankara	2022	2451685	2355457	-3.925
8	Apple Ankara	2023	2355457	15821752	571.706
9	Apple Ankara	2024	15821752	287913	-98.180
10	Apple Bangkok	2021	1843576	2427853	31.693
11	Apple Bangkok	2022	2427853	2301700	-5.196
12	Apple Bangkok	2023	2301700	450951	-80.408
13	Apple Bangkok	2024	450951	273128	-39.433
14	Apple Barcelona	2021	1888830	1622698	-14.090
15	Apple Barcelona	2022	1622698	1740833	7.280

- Calculate the correlation between product price and warranty claims for products sold in the last five years, segmented by price range

```
SELECT
CASE
 WHEN p.price < 500 THEN 'Less Expensive product'
 WHEN p.price  BETWEEN 500 AND 1000 THEN 'Mid Range Product'
 ELSE 'Expensive product'
END AS price_segment,
COUNT(w.claim_id) AS total_claims
FROM warranty as w
LEFT JOIN sales as s
ON w.sale_id = s.sale_id
JOIN products as p
ON s.product_id = p.product_id
WHERE w.claim_date >= CURRENT_DATE - INTERVAL '5 year'
GROUP BY 1
```

Data Output Messages Notifications

	price_segment text	total_claims bigint
1	EXPENSIVE PRODUCT	4395
2	LESS EXPENSIVE PRODUCT	14298
3	MID RANGE PRODUCT	11783

- Identify the store with the highest percentage of "Paid Repaired" claims relative to total claims filed.

```
WITH paid_repair
AS
(SELECT
    s.store_id,
    COUNT(w.claim_id) AS paid_repaired
    FROM sales as s
    RIGHT JOIN
    warranty as w
    ON s.sale_id = w.sale_id
    WHERE w.repair_status = 'Paid Repaired'
    GROUP BY 1
    ),
total_repair_status AS
(
SELECT s.store_id,
    COUNT(w.claim_id) AS total_repair_status
      FROM sales as s
      RIGHT JOIN
      warranty as w
      ON s.sale_id = w.sale_id
      GROUP BY 1

)
SELECT ts.store_id,
    st.store_name,
      pr.paid_repaired,
      ts.total_repair_status,
      ROUND( pr.paid_repaired :: numeric / ts.total_repair_status*100,2) AS
percentage_of_paid_repair

FROM        paid_repair as pr
      JOIN
      total_repair_status as ts
            ON pr.store_id = ts.store_id
            JOIN
            stores as st
            ON st.store_id = pr.store_id
            ORDER BY percentage_of_paid_repair DESC
```

ANSWER :

apple_stores_db/postgres@PostgreSQL 17

No limit

Query Query History

```sql
-- Identify the store with the highest percentage of "Paid Repaired" claims relative to total claims filed.
WITH paid_repair
AS
(SELECT
    s.store_id,
    COUNT(w.claim_id) AS paid_repaired
    FROM sales as s
    RIGHT JOIN
    warranty as w
    ON s.sale_id = w.sale_id
    WHERE w.repair_status = 'Paid Repaired'
    GROUP BY 1
    ),
total_repair_status AS
(
SELECT s.store_id,
       COUNT(w.claim_id) AS total_repair_status
       FROM sales as s
       RIGHT JOIN
```

Data Output Messages Notifications

	store_id character varying (10)	store_name character varying (30)	paid_repaired bigint	total_repair_status bigint	percentage_of_paid_repair numeric
1	ST-26	Apple Kurfürstendamm	60	160	37.50
2	ST-28	Apple Champs-Élysées	53	151	35.10
3	ST-24	Apple Bluewater	86	259	33.20
4	ST-23	Apple Covent Garden	86	267	32.21
5	ST-27	Apple Amsterdam	52	163	31.90
6	ST-22	Apple Regent Street	377	1214	31.05
7	ST-25	Apple Munich	147	481	30.56
8	ST-29	Apple Lyon	50	184	27.17
9	ST-30	Apple Milan	417	1691	24.66
10	ST-31	Apple Dubai Mall	1427	5838	24.44
11	ST-32	Apple Mall of the Emirates	771	5978	12.90

- Write a query to calculate the monthly running total of sales for each store over the past four years and compare trends during this period.

```
WITH monthly_sales
AS
(
SELECT
    s.store_id,
    EXTRACT(YEAR from s.sale_date) as year,
    EXTRACT(MONTH from s.sale_date) as month,
    SUM(p.price * s.quantity) as total_revenue
FROM sales as s
JOIN products as p
ON p.product_id = s.product_id
GROUP BY s.store_id,year,month
ORDER BY s.store_id,year,month
)
SELECT
    store_id,
    month,
    year,
    total_revenue,
    SUM(total_revenue) OVER(PARTITION BY store_id ORDER BY year, month) as
running_total
FROM monthly_sales
```

Data Output Messages Notifications

	store_id character varying (10)	month numeric	year numeric	total_revenue double precision	running_total double precision
1	ST-1	1	2019	471480	471480
2	ST-1	2	2019	746364	1217844
3	ST-1	3	2019	611615	1829459
4	ST-1	4	2019	479549	2309008
5	ST-1	5	2019	738211	3047219
6	ST-1	6	2019	661938	3709157
7	ST-1	7	2019	597045	4306202
8	ST-1	8	2019	766416	5072618
9	ST-1	9	2019	599236	5671854

- Analyze product sales trends over time, segmented into key periods: from launch to 6 months, 6-12 months, 12-18 months, and beyond 18 months.

```
SELECT p.product_name,
    CASE
      WHEN s.sale_date BETWEEN p.launch_date AND p.launch_date + INTERVAL '6 month' THEN '0-6 month'
      WHEN s.sale_date BETWEEN p.launch_date + INTERVAL '6 month' AND p.launch_date + INTERVAL '12 month' THEN '6-12 month'
      WHEN s.sale_date BETWEEN p.launch_date + INTERVAL '12 month' AND p.launch_date + INTERVAL '18 months' THEN '12-18 moths'
      ELSE '18+'
      END AS key_period,
      SUM (s.quantity) AS total_sales
      FROM sales as s
      JOIN
      products as p
      ON s.product_id = p.product_id
      GROUP BY 1,2
      ORDER BY 3 DESC
```

Data Output Messages Notifications

	product_name character varying (35)	key_period text	total_sales bigint
1	AirTag	0-6 month	82549
2	iPhone X	18+	61561
3	AirTag	6-12 month	44222
4	MacBook Pro (M1 Max, 16-inch)	0-6 month	41844
5	MacBook Pro (M1 Pro, 14-inch)	0-6 month	41710
6	iPhone 14	0-6 month	32936
7	iPhone 14 Pro	0-6 month	32780
8	Mac mini (2018)	18+	30114
9	Mac mini (2018)	12-18 moths	28384
10	iPad (6th Gen)	18+	26454
11	iPhone XR	12-18 moths	23369
12	AirPods Pro (2nd Gen)	0-6 month	20823
13	iPad Pro (M2, 11-inch)	0-6 month	20340

Total rows: 201 Query complete 00:00:00.678