Apple Retail Sales SQL Project

Sales, Warranty & Performance Analysis using SQL

PostgreSQL

Robust relational database management system for enterprise-grade data storage

SQL

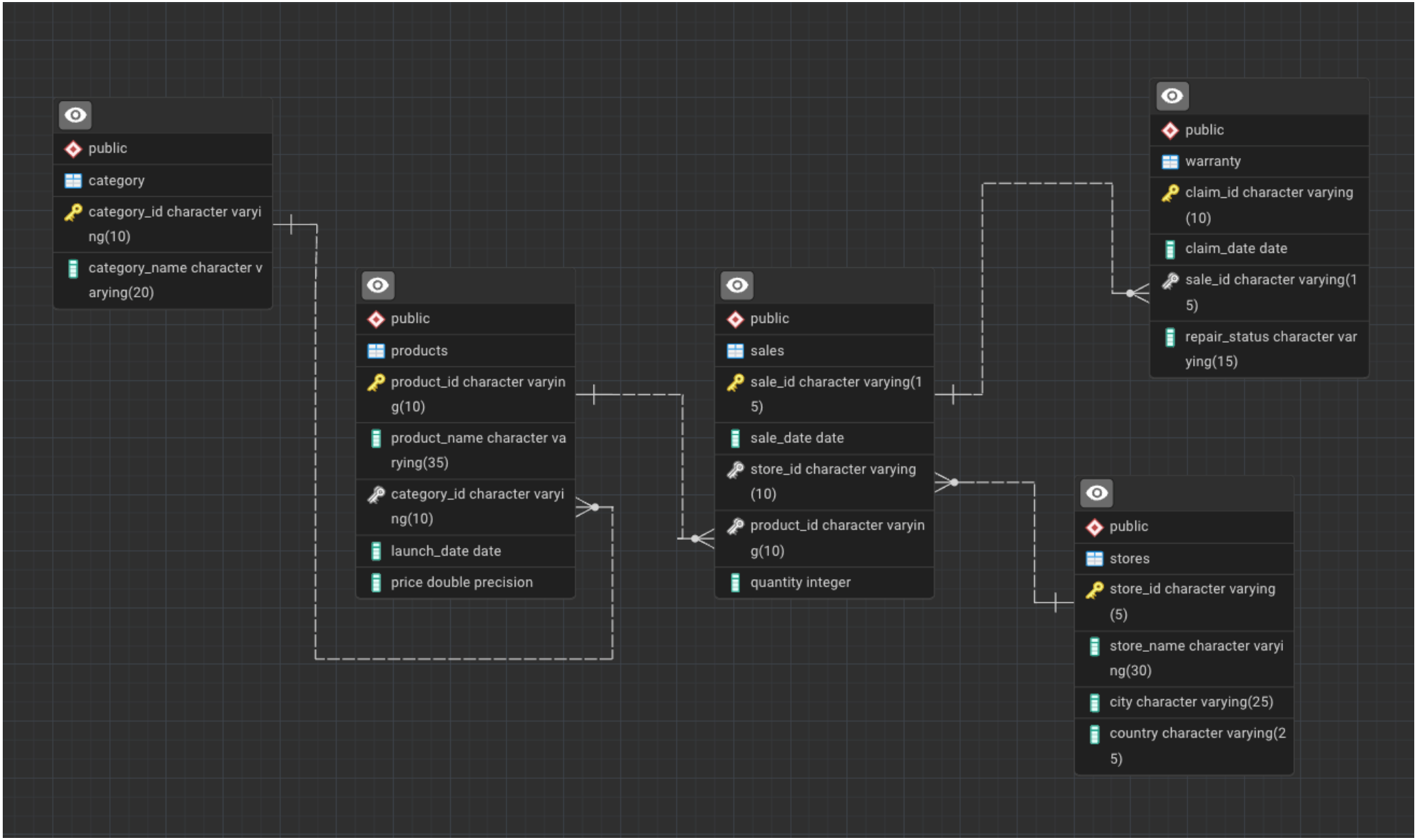
Structured query language for complex data analysis and reporting

Data Analysis

Advanced analytical techniques to extract actionable business insights

Prepared by: **PRAVIN CHAVAN**

Database Schema :



```sql
--1. Find the number of stores in each country.
SELECT COUNTRY, COUNT(STORE_ID) AS TOTAL_STORES
FROM STORES
GROUP BY COUNTRY
ORDER BY TOTAL_STORES DESC;
```

Data Output | Messages | Notifications

	country character varying (25)	total_stores bigint
1	United States	15
2	Australia	7
3	China	7
4	Japan	6
5	Canada	5
6	UAE	5
7	United Kingdom	4
8	France	4
9	Germany	3
10	Mexico	3
11	Singapore	3
12	Thailand	3
13	South Korea	2
14	Colombia	2
15	Italy	2
16	Spain	1

Total rows: 19 | Query complete 00:00:00.251

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

SELECT S.STORE_ID, ST.STORE_NAME, SUM(S.QUANTITY) AS TOTAL_UNITS_SOLDS
FROM SALES AS S
JOIN STORES AS ST ON ST.STORE_ID = S.STORE_ID
GROUP BY S.STORE_ID, ST.STORE_ID
ORDER BY TOTAL_UNITS_SOLDS DESC;
```

Data Output | Messages | Notifications

	store_id character varying (10)	store_name character varying (30)	total_units_solds bigint
1	ST-56	Apple Southland	77795
2	ST-34	Apple Fukuoka	77787
3	ST-1	Apple Fifth Avenue	77689
4	ST-30	Apple Dubai Mall	77571
5	ST-24	Apple Kurfuerstendamm	77532
6	ST-39	Apple Taipei 101	77518
7	ST-75	Apple Beijing SKP	77482
8	ST-25	Apple Schildergasse	77385
9	ST-5	Apple SoHo	77186
10	ST-52	Apple Chadstone	77170
11	ST-12	Apple North Michigan Avenue	77152
12	ST-73	Apple Covent Garden	77115
13	ST-44	Apple Kyoto	77062
14	ST-36	Apple Nanjing East	77014
15	ST-70	Apple Gangnam	76987
16	ST-37	Apple Sanlitun	76870

Total rows: 75 | Query complete 00:00:00.448

```sql
-- 3. 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	18076

```sql

--3.1 Total quantity sold in December 2023.
SELECT SUM(QUANTITY) AS TOTAL_SALES
FROM SALES
WHERE TO_CHAR(SALE_DATE, 'MM-YYYY') = '12-2023';

```

Data Output | Messages | Notifications

	total_sales bigint
1	99631

```sql
-- 4. Determine how many stores have never had a warranty claim filed.
SELECT COUNT(*)
FROM STORES
WHERE STORE_ID NOT IN (
SELECT DISTINCT STORE_ID
FROM SALES AS S
RIGHT JOIN WARRANTY AS W ON S.SALE_ID = W.SALE_ID
);
```

Data Output | Messages | Notifications

	count bigint
1	0

```sql
-- 5. Calculate the percentage of warranty claims marked as "Completed".

SELECT (COUNT(*) * 100.0 / (SELECT COUNT(*) FROM WARRANTY))::NUMERIC
FROM WARRANTY
WHERE REPAIR_STATUS = 'Completed';

```

Data Output | Messages | Notifications

	numeric numeric
1	24.8866666666666667

```sql
-- 6. Identify which store had the highest total units sold in 2023.
SELECT ST.STORE_ID, ST.STORE_NAME, SUM(S.QUANTITY) AS TOTAL_UNITS_SOLD
FROM STORES AS ST
JOIN SALES AS S ON ST.STORE_ID = S.STORE_ID
WHERE EXTRACT(YEAR FROM S.SALE_DATE) = 2023
GROUP BY ST.STORE_ID, ST.STORE_NAME
ORDER BY TOTAL_UNITS_SOLD DESC
LIMIT 1;
```

Data Output | Messages | Notifications

SQL

	store_id [PK] character varying (5)	store_name character varying (30)	total_units_sold bigint
1	ST-14	Apple The Americana at Bra...	16461

```sql
-- 7. Count the number of unique products sold in the 2023.
SELECT DISTINCT S.PRODUCT_ID, P.PRODUCT_NAME
FROM SALES AS S
JOIN PRODUCTS AS P ON P.PRODUCT_ID = S.PRODUCT_ID
WHERE EXTRACT(YEAR FROM SALE_DATE) = 2023;
```

ata Output | Messages | Notifications

SQL

	product_id character varying (10)	product_name character varying (35)
	P-1	MacBook
	P-10	MacBook (Early 2015)
	P-11	AirPods (2nd Generation)
	P-12	AirPods (3rd Generation)
	P-13	AirPods Pro
	P-14	AirPods Pro (2nd Generatio...
	P-15	AirPods Max
	P-16	Beats Studio Buds
	P-17	Beats Fit Pro
0	P-18	Beats Solo Pro
1	P-19	Beats Powerbeats Pro
2	P-2	MacBook Air (M1)
3	P-20	HomePod (2nd Generation)
4	P-21	HomePod mini
5	P-22	iPad (10th Generation)
6	P-23	iPad (9th Generation)

Total rows: 89 | Query complete 00:00:00.623

```sql
-- 8. Find the average price of products in each category.
SELECT
    P.CATEGORY_ID,
    C.CATEGORY_NAME,
    AVG(P.PRICE) AS AVG_PRICE
FROM
    PRODUCTS AS P
    JOIN CATEGORY AS C ON P.CATEGORY_ID = C.CATEGORY_ID
GROUP BY
    P.CATEGORY_ID,
    C.CATEGORY_NAME
ORDER BY
    AVG_PRICE DESC;
```

Data Output | Messages | Notifications

	category_id character varying (10)	category_name character varying (20)	avg_price double precision
1	CAT-3	Tablet	1479.5
2	CAT-1	Laptop	1194.1
3	CAT-5	Wearable	1146.888888888889
4	CAT-2	Audio	1124.8181818181818
5	CAT-4	Smartphone	1037.5384615384614
6	CAT-10	Accessories	1030.142857142857
7	CAT-6	Streaming Device	996.6666666666666
8	CAT-7	Desktop	838.7
9	CAT-8	Subscription Service	823.2857142857143
10	CAT-9	Smart Speaker	734

```sql
--10. For each store, identify the best-selling day based on highest quantity sold.

SELECT *
FROM (
SELECT STORE_ID,
TO_CHAR(SALE_DATE, 'Day') AS DAY_NAME,
SUM(QUANTITY) AS TOTAL_UNIT_SOLD,
RANK() OVER (PARTITION BY STORE_ID ORDER BY SUM(QUANTITY) DESC) AS RANK
FROM SALES
GROUP BY STORE_ID, DAY_NAME
) AS T1
WHERE RANK = 1;
```

Data Output | Messages | Notifications

	store_id character varying (10)	day_name text	total_unit_sold bigint	rank bigint
1	ST-1	Tuesday	11233	1
2	ST-10	Thursday	11243	1
3	ST-11	Thursday	11943	1
4	ST-12	Saturday	11416	1
5	ST-13	Sunday	11320	1
6	ST-14	Monday	11139	1
7	ST-15	Saturday	10913	1
8	ST-16	Tuesday	11296	1
9	ST-17	Saturday	11088	1
10	ST-18	Wednesday	11167	1
11	ST-19	Sunday	11425	1
12	ST-2	Tuesday	11266	1
13	ST-20	Friday	11236	1
14	ST-21	Monday	11023	1
15	ST-22	Monday	11019	1
16	ST-23	Sunday	11408	1
17	ST-24	Thursday	11376	1

Total rows: 75 | Query complete 00:00:01.683

```sql
-- 11. Identify the least selling product in each country based on total units sold.

WITH PRODUCT_RANK AS (
SELECT ST.COUNTRY, P.PRODUCT_NAME, SUM(S.QUANTITY) AS TOTAL_QTY_SOLD,
RANK() OVER (PARTITION BY ST.COUNTRY ORDER BY SUM(S.QUANTITY)) AS RANK
FROM SALES AS S
JOIN STORES AS ST ON S.STORE_ID = ST.STORE_ID
JOIN PRODUCTS AS P ON S.PRODUCT_ID = P.PRODUCT_ID
GROUP BY 1,2
)
SELECT * FROM PRODUCT_RANK WHERE RANK = 1;
```

Data Output | Messages | Notifications

SQL

	country character varying (25)	product_name character varying (35)	total_qty_sold bigint	rank bigint
1	Australia	MacBook Air (M1)	5516	1
2	Austria	Apple Music	663	1
3	Canada	Apple Pencil (1st Generati...	3798	1
4	China	iPhone 12	5343	1
5	Colombia	Beats Studio Buds	1428	1
6	France	iMac Pro	3056	1
7	Germany	MacBook Pro (Touch Bar)	2246	1
8	Italy	Beats Solo Pro	1463	1
9	Japan	Apple Watch Series 5	4772	1
10	Mexico	Apple Watch Hermès	2275	1
11	Netherlands	iMac Pro	680	1
12	Singapore	Smart Cover for iPad	2241	1
13	South Korea	Silicone Case for iPhone	1511	1
14	Spain	Beats Fit Pro	665	1
15	Taiwan	iPhone 14 Pro	732	1
16	Thailand	Lightning to USB Cable	2195	1
17	UAE	AirPods (2nd Generation)	3953	1

Total rows: 19 | Query complete 00:00:00.550

```sql
-- 12. Calculate how many warranty claims were filed within 180 days of a product sale.
SELECT
    COUNT(*)
FROM
    SALES AS S
    JOIN WARRANTY AS W ON S.SALE_ID = W.SALE_ID
WHERE
    W.CLAIM_DATE - S.SALE_DATE <= 180;
```

Data Output | Messages | Notifications

	count bigint
1	5733

```sql
-- 13. Find sales info where warranty claims are filed within 180 days of a product sale.

SELECT * FROM
    SALES AS S
    JOIN WARRANTY AS W ON S.SALE_ID = W.SALE_ID
WHERE
    W.CLAIM_DATE - S.SALE_DATE <= 180;
```

Data Output | Messages | Notifications

	sale_id character varying (15)	sale_date date	store_id character varying (10)	product_id character varying (10)	quantity integer	claim_id character varying (10)	claim_date date	sale_id character varying (15)	repair_status character varying (15)
1	II-3954	2024-04-19	ST-28	P-45	10	CL-30514	2024-09-05	II-3954	Pending
2	JB-46303	2024-08-31	ST-14	P-29	5	CL-75561	2024-01-24	JB-46303	Completed
3	UN-941274	2023-10-27	ST-10	P-87	4	CL-7124	2024-01-04	UN-941274	Completed
4	JM-75607	2024-10-27	ST-69	P-12	1	CL-78615	2024-11-09	JM-75607	Completed
5	UJ-07947	2024-05-08	ST-20	P-4	6	CL-29490	2024-03-14	UJ-07947	Completed
6	NY-82428	2024-10-04	ST-8	P-64	10	CL-12189	2024-11-12	NY-82428	Completed
7	VT-954957	2024-03-26	ST-60	P-9	7	CL-3992	2024-05-17	VT-954957	Pending
8	GN-099683	2024-07-22	ST-25	P-9	1	CL-23879	2024-05-04	GN-099683	In Progress
9	CA-4116	2024-02-19	ST-72	P-70	4	CL-8833	2024-06-25	CA-4116	Completed
10	YD-9562	2024-03-24	ST-66	P-25	5	CL-81539	2024-07-05	YD-9562	Rejected
11	XR-468203	2023-12-21	ST-55	P-29	7	CL-63006	2024-02-21	XR-468203	Pending
12	NQ-18124	2024-01-18	ST-14	P-77	9	CL-16284	2024-03-03	NQ-18124	Completed
13	XZ-59032	2024-07-07	ST-5	P-81	5	CL-31307	2024-01-02	XZ-59032	Pending
14	AJ-06146	2024-05-09	ST-23	P-34	9	CL-46503	2024-04-05	AJ-06146	Pending
15	FL-9609	2024-05-14	ST-70	P-52	4	CL-94775	2024-01-31	FL-9609	Pending
16	JC-31529	2024-03-15	ST-1	P-31	2	CL-49125	2024-05-29	JC-31529	Completed
17	NH-656788	2024-01-10	ST-73	P-16	7	CL-8853	2024-05-08	NH-656788	Rejected

Total rows: 5733 | Query complete 00:00:00.356

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

SELECT TO_CHAR(SALE_DATE, 'MM-YYYY') AS MONTH,
SUM(S.QUANTITY) AS TOTAL_UNIT_SOLD
FROM SALES AS S
JOIN STORES AS ST ON S.STORE_ID = ST.STORE_ID
WHERE ST.COUNTRY = 'United States'
AND S.SALE_DATE >= CURRENT_DATE - INTERVAL '3 year'
GROUP BY MONTH
HAVING SUM(S.QUANTITY) > 5000;
```

Data Output | Messages | Notifications

	month text	total_unit_sold bigint
1	01-2024	19814
2	02-2023	17489
3	02-2024	19412
4	03-2023	19641
5	03-2024	20051
6	04-2023	19803
7	04-2024	19513
8	05-2023	19492
9	05-2024	19865
10	06-2023	19146
11	06-2024	19205
12	07-2023	19924
13	07-2024	19996
14	08-2023	19955
15	08-2024	19657
16	09-2023	19280
17	09-2024	19203

Total rows: 22 | Query complete 00:00:00.383

```sql
-- 15. Identify the product category with the most warranty claims filed in year 2024.

SELECT CATEGORY_NAME, COUNT(W.CLAIM_ID) AS TOTAL_CLAIM
FROM WARRANTY W
JOIN SALES S ON W.SALE_ID = S.SALE_ID
JOIN PRODUCTS P ON S.PRODUCT_ID = P.PRODUCT_ID
JOIN CATEGORY C ON C.CATEGORY_ID = P.CATEGORY_ID
WHERE EXTRACT(YEAR FROM W.CLAIM_DATE) = 2024
GROUP BY CATEGORY_NAME
ORDER BY TOTAL_CLAIM DESC;
```

Data Output | Messages | Notifications

	category_name character varying (20)	total_claim bigint
1	Accessories	4650
2	Smartphone	4434
3	Audio	3768
4	Tablet	3397
5	Laptop	3384
6	Desktop	3372
7	Wearable	2986
8	Subscription Service	2374
9	Streaming Device	940
10	Smart Speaker	695

```sql
--16. Determine the percentage chance of receiving warranty claims after each purchase for each country.

SELECT COUNTRY, TOTAL_UNIT_SOLD, TOTAL_CLAIM,
COALESCE(TOTAL_CLAIM::NUMERIC / TOTAL_UNIT_SOLD::NUMERIC * 100, 0) AS RISK
FROM (
SELECT ST.COUNTRY, SUM(S.QUANTITY) AS TOTAL_UNIT_SOLD, COUNT(W.CLAIM_ID) AS TOTAL_CLAIM
FROM SALES AS S
JOIN STORES AS ST ON S.STORE_ID = ST.STORE_ID
LEFT JOIN WARRANTY AS W ON W.SALE_ID = S.SALE_ID
GROUP BY COUNTRY
) T1
ORDER BY RISK DESC;
```

Data Output | Messages | Notifications

	country character varying (25)	total_unit_sold bigint	total_claim bigint	risk numeric
1	Austria	75965	448	0.58974527743039557700
2	Netherlands	76764	428	0.55755301964462508500
3	Taiwan	77518	419	0.54051962124925823700
4	UAE	381972	2046	0.53564135591090446400
5	Canada	381212	2035	0.53382369914902993600
6	United Kingdom	304861	1624	0.53270178868402321100
7	United States	1144783	6086	0.53162913844807269200
8	Australia	535623	2827	0.52779660320785328500
9	Singapore	227828	1198	0.52583527924574679100
10	Japan	460325	2417	0.52506381360994949200
11	France	302785	1585	0.52347375200224581800
12	Germany	230567	1201	0.52088980643370473700
13	Mexico	227917	1186	0.52036486966746666500
14	Thailand	227858	1159	0.50865012420015974900
15	South Korea	153849	782	0.50829059662396245700
16	China	534345	2713	0.50772441025928941000
17	Colombia	150569	752	0.49943879550239425100

Total rows: 19 | Query complete 00:00:00.555

```sql
-- Q.17 Analyze the year-by-year growth ratio for each store each store and their yearly sale

WITH YEARLY_SALES AS (
SELECT S.STORE_ID, ST.STORE_NAME,
EXTRACT(YEAR FROM SALE_DATE) AS YEAR,
SUM(S.QUANTITY * P.PRICE) AS TOTAL_SALE
FROM SALES AS S
JOIN PRODUCTS AS P ON S.PRODUCT_ID = P.PRODUCT_ID
JOIN STORES AS ST ON ST.STORE_ID = S.STORE_ID
GROUP BY S.STORE_ID, ST.STORE_NAME, YEAR
),
GROWTH_RATE AS (
SELECT STORE_NAME, YEAR,
LAG(TOTAL_SALE,1) OVER (PARTITION BY STORE_NAME ORDER BY YEAR) AS LAST_YEAR_SALE,
TOTAL_SALE AS CURRENT_YEAR_SALE
FROM YEARLY_SALES
)
SELECT STORE_NAME, YEAR, LAST_YEAR_SALE, CURRENT_YEAR_SALE,
ROUND((CURRENT_YEAR_SALE - LAST_YEAR_SALE)::NUMERIC / LAST_YEAR_SALE::NUMERIC * 100, 3) AS GROWTH_RATE
FROM GROWTH_RATE
WHERE LAST_YEAR_SALE IS NOT NULL
AND YEAR <> EXTRACT(YEAR FROM CURRENT_DATE);
```

Data Output | Messages | Notifications

	store_name character varying (30)	year numeric	last_year_sale double precision	current_year_sale double precision	growth_rate numeric
1	Apple Ala Moana	2021	16736679	16351838	-2.299
2	Apple Ala Moana	2022	16351838	16679728	2.005
3	Apple Ala Moana	2023	16679728	17286752	3.639
4	Apple Ala Moana	2024	17286752	15215747	-11.980
5	Apple Andino	2021	15873867	16657275	4.935
6	Apple Andino	2022	16657275	16807255	0.900
7	Apple Andino	2023	16807255	16452478	-2.111
8	Apple Andino	2024	16452478	14591539	-11.311
9	Apple Antara	2021	16614977	16516324	-0.594
10	Apple Antara	2022	16516324	16625607	0.662
11	Apple Antara	2023	16625607	15789600	-5.028
12	Apple Antara	2024	15789600	14795003	-6.299
13	Apple Beijing SKP	2021	17004025	16967537	-0.215
14	Apple Beijing SKP	2022	16967537	17293828	1.923
15	Apple Beijing SKP	2023	17293828	16793173	-2.895
16	Apple Beijing SKP	2024	16793173	14943387	-11.015
17	Apple Beverly Center	2021	16876768	16086769	-4.681

Total rows: 306 | Query complete 00:00:03.786

```sql
-- Q.18 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'
        WHEN P.PRICE BETWEEN 1000 AND 1500  THEN ' Expensive Product'
        ELSE ' Very Expensive Product'
    END AS PRICE_SEGMENT,
    COUNT(W.CLAIM_ID) AS TOTAL_CLAIM
FROM
    WARRANTY AS W
    LEFT JOIN SALES AS S ON W.SALE_ID = S.SALE_ID
    JOIN PRODUCTS AS P ON P.PRODUCT_ID = S.PRODUCT_ID
WHERE
    CLAIM_DATE >= CURRENT_DATE - INTERVAL '5 year'
GROUP BY
    PRICE_SEGMENT;
```

Data Output | Messages | Notifications

Showing rows: 1

	price_segment text	total_claim bigint
1	Expensive Product	9808
2	Very Expensive Prod...	6982
3	Less Expensive Produ...	5089
4	Mid Range Product	8121

```sql
-- Q.19 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 STORE_ID,
EXTRACT(YEAR FROM SALE_DATE) AS YEAR,
EXTRACT(MONTH FROM SALE_DATE) AS MONTH,
SUM(P.PRICE * S.QUANTITY) AS TOTAL_REVENUE
FROM SALES AS S
JOIN PRODUCTS AS P ON S.PRODUCT_ID = P.PRODUCT_ID
GROUP BY 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

Showing rows: 1 to 1000

	store_id character varying (10)	month numeric	year numeric	total_revenue double precision	running_total double precision
1	ST-1	1	2020	1519710	1519710
2	ST-1	2	2020	1513077	3032787
3	ST-1	3	2020	1456142	4488929
4	ST-1	4	2020	1432050	5920979
5	ST-1	5	2020	1408946	7329925
6	ST-1	6	2020	1398598	8728523
7	ST-1	7	2020	1488995	10217518
8	ST-1	8	2020	1568175	11785693
9	ST-1	9	2020	1344362	13130055
10	ST-1	10	2020	1291883	14421938
11	ST-1	11	2020	1524397	15946335
12	ST-1	12	2020	1463945	17410280
13	ST-1	1	2021	1638264	19048544
14	ST-1	2	2021	1330913	20379457
15	ST-1	3	2021	1230568	21610025
16	ST-1	4	2021	1498796	23108821
17	ST-1	5	2021	1472878	24581699

Total rows: 4425 | Query complete 00:00:02.234

```sql
--Q.20 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'
        WHEN S.SALE_DATE BETWEEN P.LAUNCH_DATE + INTERVAL '12 month' AND P.LAUNCH_DATE  + INTERVAL '18 month' THEN '6-12'
        ELSE '18+'
    END AS PLC,
    SUM(S.QUANTITY) AS TOTAL_QTY_SALE
FROM
    SALES AS S
    JOIN PRODUCTS AS P ON S.PRODUCT_ID = P.PRODUCT_ID
GROUP BY
    P.PRODUCT_NAME,
    PLC
ORDER BY
    P.PRODUCT_NAME,
    TOTAL_QTY_SALE DESC;
```

Data Output | Messages | Notifications

Showing rows: 1 to 254

	product_name character varying (35)	plc text	total_qty_sale bigint
1	AirPods (2nd Generation)	18+	44681
2	AirPods (2nd Generation)	6-12	13250
3	AirPods (2nd Generation)	0-6 mon...	6650
4	AirPods (3rd Generation)	18+	44036
5	AirPods (3rd Generation)	6-12	13530
6	AirPods (3rd Generation)	0-6 mon...	6536
7	AirPods Max	18+	44846
8	AirPods Max	6-12	13392
9	AirPods Max	0-6 mon...	6606
10	AirPods Pro	18+	63349
11	AirPods Pro	0-6 mon...	1394
12	AirPods Pro (2nd Generatio...	18+	45051
13	AirPods Pro (2nd Generatio...	6-12	12995
14	AirPods Pro (2nd Generatio...	0-6 mon...	6788
15	AirTag	18+	45251
16	AirTag	6-12	13077
17	AirTag	0-6 mon	6408

Total rows: 254 | Query complete 00:00:01.772