Overview

This project is designed to showcase advanced SQL querying techniques through the analysis of over 1 million rows of Apple retail sales data. The dataset includes information about products, stores, sales transactions, and warranty claims across various Apple retail locations globally. By tackling a variety of questions, from basic to complex, you'll demonstrate your ability to write sophisticated SQL queries that extract valuable insights from large datasets.

The project is ideal for data analysts looking to enhance their SQL skills by working with a large-scale dataset and solving real-world business questions.

* Database Schema

The project uses five main tables:

1. \*\*stores\*\*: Contains information about Apple retail stores.

- `store\_id`: Unique identifier for each store.

- `store\_name`: Name of the store.

- `city`: City where the store is located.

- `country`: Country of the store.

2. \*\*category\*\*: Holds product category information.

- `category\_id`: Unique identifier for each product category.

- `category\_name`: Name of the category.

3. \*\*products\*\*: Details about Apple products.

- `product\_id`: Unique identifier for each product.

- `product\_name`: Name of the product.

- `category\_id`: References the category table.

- `launch\_date`: Date when the product was launched.

- `price`: Price of the product.

4. \*\*sales\*\*: Stores sales transactions.

- `sale\_id`: Unique identifier for each sale.

- `sale\_date`: Date of the sale.

- `store\_id`: References the store table.

- `product\_id`: References the product table.

- `quantity`: Number of units sold.

5. \*\*warranty\*\*: Contains information about warranty claims.

- `claim\_id`: Unique identifier for each warranty claim.

- `claim\_date`: Date the claim was made.

- `sale\_id`: References the sales table.

- `repair\_status`: Status of the warranty claim (e.g., Paid Repaired, Warranty Void).

Easy

-- 1.Find each country and number of stores

select country, count(distinct store\_id) as no\_of\_stores

from `automatekpis.roughwork.stores` group by 1

-- 2. What is the total number of units sold by each store?

select distinct store\_id,sum(quantity) as total\_unit

from `automatekpis.roughwork.sales` group by 1 order by 1

-- 3. How many sales occurred in December 2023?

select count(distinct sale\_id) as sale from `automatekpis.roughwork.sales`

where sale\_date between '2023-12-01' and '2023-12-31'

-- 4. How many stores have never had a warranty claim filed against any of their products?

select distinct store\_id from (select distinct s.store\_id,count(w.claim\_id) as no\_of\_claim\_id

from `automatekpis.roughwork.sales` s left join `automatekpis.roughwork.warranty` w

on s.sale\_id = w.sale\_id group by 1) where no\_of\_claim\_id = 0

-- 5. What percentage of warranty claims are marked as "Warranty Void"?

with t as (select \* from `automatekpis.roughwork.warranty` )

,t1 as

(select count(distinct claim\_id) as warranty\_void from t where repair\_status = 'Warranty Void')

,t2 as

(select count(distinct claim\_id) as no\_of\_claim from t)

select round((warranty\_void/no\_of\_claim)\*100.0,2) from t1, t2

-- 6. Which store had the highest total units sold in the last year?

select store\_id,sum(quantity) as unit

from `automatekpis.roughwork.sales` where EXTRACT(YEAR FROM sale\_date) = 2023

group by 1,2 order by unit desc limit 1

-- 7. Count the number of unique products sold in the last year.

select distinct product\_id from `automatekpis.roughwork.sales`

where EXTRACT(YEAR FROM sale\_date) = 2023

-- 8. What is the average price of products in each category?

select category\_id,round(avg(price)) as avg\_price

from `automatekpis.roughwork.products` group by 1

-- 9. How many warranty claims were filed in 2020?

select count(distinct claim\_id) as claim\_count

from `automatekpis.roughwork.warranty`

where EXTRACT(YEAR FROM claim\_date) = 2020

-- 10. Identify each store and best selling day based on highest qty sold

with t as

(select store\_id,sale\_date,sum(quantity) as qty\_sold

from `automatekpis.roughwork.sales` group by 1,2)

,t2 as (select \* , rank() over(partition by store\_id order by qty\_sold desc) as rank\_ from t )

select \* from t2 where rank\_ = 1

Intermediate

-- 11. Identify least selling product of each country for each year based on total unit sold

with t as

(SELECT extract(Year from s.sale\_date) as year,st.country,s.product\_id,

round(sum(s.quantity),2) as total,

FROM `automatekpis.roughwork.sales` s join `automatekpis.roughwork.stores` st

on s.store\_id = st.store\_id group by 1,2,3)

,t2 as (select \*, rank() over(partition by year,country order by total asc ) as rank\_ from t)

select t2.year,t2.country,t2.product\_id,p.product\_name,total

from t2 join `automatekpis.roughwork.products` p

on t2.product\_id = p.product\_id

where t2.rank\_ = 1 order by 1,2,3

-- 12. How many warranty claims were filed within 180 days of a product sale?

Select count(distinct claim\_id) as Claim\_id from

(SELECT claim\_id,s.sale\_date,w.claim\_date,

datetime\_diff(w.claim\_date,s.sale\_date,Day) as datediff FROM `automatekpis.roughwork.sales` s

join `automatekpis.roughwork.warranty` w

on w.sale\_id = s.sale\_id

where datetime\_diff(w.claim\_date,s.sale\_date,Day) < 180)

-- 13. How many warranty claims have been filed for products launched in the last two years

with t as

(SELECT distinct s.product\_id,p.launch\_date,w.claim\_id,

extract(Year from launch\_date) as year,Current\_date() as currentDate

FROM `automatekpis.roughwork.sales` s

join `automatekpis.roughwork.products` p on s.product\_id = p.product\_id

join `automatekpis.roughwork.warranty` w on s.sale\_id = w.sale\_id)

select count(distinct claim\_id) as claim\_id

from t where (Extract(Year from currentDate) - year) <= 2

-- 14. List the months in the last 3 years where sales exceeded 5000 Units from usa

with t as

(SELECT st.country,

extract(Year from current\_date()) as CurrentDate,

extract(Year from sale\_date) as year,

extract(Month from sale\_date) as Months,

round(sum(quantity),0) as Quantity

FROM `automatekpis.roughwork.sales` s

join `automatekpis.roughwork.stores` st

on s.store\_id = st.store\_id

where st.country = 'USA'

group by 1,2,3,4 order by 1,2,3,4)

select year,Months from t where (CurrentDate - year) <= 3

and Quantity > 5000

-- 15. Which product category had the most warranty claims filed in the last 2 years?

with t as

(SELECT \*, extract(year from claim\_date) as year

FROM `automatekpis.roughwork.sales` s

join `automatekpis.roughwork.products` p on s.product\_id = p.product\_id

join `automatekpis.roughwork.warranty` w on s.sale\_id = w.sale\_id

join `automatekpis.roughwork.category` c on c.category\_id = p.category\_id)

select category\_name, count(distinct claim\_id) as claim\_id from t

where (extract(year from current\_date())- year) <= 3

group by 1 order by 2 desc limit 1

Complex

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

select country,round((claim\_id/total\_quantity)\*100.0,2) as percentage

from (

SELECT st.country,sum(quantity) as total\_quantity,count(distinct w.claim\_id) as claim\_id

FROM `automatekpis.roughwork.sales` s

join `automatekpis.roughwork.stores` st on s.store\_id = st.store\_id

left join `automatekpis.roughwork.warranty` w on s.sale\_id = w.sale\_id

group by 1)

order by 2 desc

-- 17. Analyze each stores year by year growth ratio

with t as

(SELECT distinct store\_id,extract(year from sale\_date) as year,

round(sum(quantity),2) as total\_quantity

FROM `automatekpis.roughwork.sales` s

group by 1,2 order by 1,2)

,t2 as

(select \*, lag(total\_quantity) over(partition by store\_id order by year)

as last\_year\_revenue from t)

select store\_id,year, round(coalesce(((total\_quantity - last\_year\_revenue)/last\_year\_revenue)\*100.0,100),2) as ratio

from t2 order by 1,2

-- 18. What is the correlation between product price and warranty claims for products sold in the last five years? (Segment based on diff price)

SELECT

case when p.price < 500 then 'Less Expensive'

when p.price between 500 and 1000 then 'Moderate Product'

when p.price > 1000 then 'Expensive Product'

else '0' end as Price\_segment,

count(w.claim\_id) as Total\_claim

FROM `automatekpis.roughwork.sales` s

join `automatekpis.roughwork.products` p on s.product\_id = p.product\_id

left join `automatekpis.roughwork.warranty` w on s.sale\_id = w.sale\_id

where w.claim\_date >= DATE\_SUB(CURRENT\_DATE(), INTERVAL 5 YEAR)

group by 1

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

with t1 as

(SELECT s.store\_id, count(claim\_id) as total\_claim

FROM `automatekpis.roughwork.sales` s

left join `automatekpis.roughwork.warranty` w on s.sale\_id = w.sale\_id

group by 1 order by 1 )

,t2 as

(SELECT s.store\_id, count(claim\_id) as paid\_repaired\_claim

FROM `automatekpis.roughwork.sales` s

left join `automatekpis.roughwork.warranty` w on s.sale\_id = w.sale\_id

where w.repair\_status = 'Paid Repaired'

group by 1 order by 1 )

select t1.store\_id,round((paid\_repaired\_claim/total\_claim)\*100.0,2) as Percentage

from t1 join t2 on t1.store\_id = t2.store\_id order by 2 desc limit 1

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

select \* ,sum(total\_quantity)

over(partition by store\_id order by year, month ) as running\_total\_quantity

from (SELECT s.store\_id,

extract(year from s.sale\_date) as year,

extract(month from s.sale\_date) as month,

round(sum(quantity)) as total\_quantity

FROM `automatekpis.roughwork.sales` s

left join `automatekpis.roughwork.warranty` w on s.sale\_id = w.sale\_id

group by 1,2,3 order by 1,2,3 )

Bonus

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

with t as

(select

p.product\_id,s.sale\_date,s.quantity,p.price,p.launch\_date

FROM `automatekpis.roughwork.sales` s

join `automatekpis.roughwork.products` p on s.product\_id = p.product\_id)

,t1 as

(select \*,

CASE

WHEN DATE\_DIFF(sale\_date, launch\_date,Day) <= 180 THEN '0-6 months'

WHEN DATE\_DIFF(sale\_date, launch\_date,Day) > 180 AND DATE\_DIFF(sale\_date, launch\_date,Day) <= 365 THEN '6-12 months'

WHEN DATE\_DIFF(sale\_date, launch\_date,Day) > 365 AND DATE\_DIFF(sale\_date, launch\_date,Day) <= 540 THEN '12-18 months'

ELSE '>18 months' END AS time\_segment from t )

select time\_segment,

Round(SUM(price),2) AS total\_sales,

Round(SUM(quantity),2) AS total\_quantity,

COUNT(\*) AS total\_transactions,

round(AVG(price),2) AS avg\_sales\_amount,

round(AVG(quantity),2) AS avg\_quantity\_sold

from t1 group by 1

Output

Query Output -

[Sql Query Results](https://docs.google.com/spreadsheets/d/1j4emuGAxAdiAwhbkNqRrhwMVYIw6iNCNXjjar6PIoVk/edit?usp=sharing)

By completing this project, I have developed advanced SQL querying skills, improved my ability to handle large datasets, and gain practical experience in solving complex data analysis problems that are crucial for business decision-making.