# SOLUTION

```
-- Apple Sales Project - 1M rows sales datasets


SELECT * FROM category;

SELECT * FROM products;

SELECT * FROM stores;

SELECT * FROM sales;

SELECT * FROM warranty;


-- EDA

SELECT DISTINCT repair_status FROM warranty;

SELECT COUNT(*) FROM sales;


-- Improving Query Performance


-- et - 64.ms

-- pt - 0.15ms

-- et after index 5-10 ms


EXPLAIN ANALYZE

SELECT * FROM sales

WHERE product_id ='P-44'


CREATE INDEX sales_product_id ON sales(product_id);

CREATE INDEX sales_store_id ON sales(store_id);

CREATE INDEX sales_sale_date ON sales(sale_date);
```

```
-- et - 58.ms
-- pt - 0.069
-- et after index 2 ms
EXPLAIN ANALYZE
SELECT * FROM sales
WHERE store_id ='ST-31'

-- Business Problems
-- Medium Problems
-- 1. Find the number of stores in each country.

SELECT
   country,
   COUNT(store_id) as total_stores
FROM stores
GROUP BY 1
ORDER BY 2 DESC

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

SELECT
   s.store_id,
   st.store_name,
   SUM(s.quantity) as total_unit_sold
FROM sales as s
JOIN
stores as st
```

```
ON st.store_id = s.store_id
GROUP BY 1, 2
ORDER BY 3 DESC

-- Q.3 Identify how many sales occurred in December 2023.

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

-- Q.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
                    );

-- Q.5 Calculate the percentage of warranty claims marked as "Warranty Void".
no claim that as wv/total claim * 100

SELECT
   ROUND
```

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

-- Q.6 Identify which store had the highest total units sold in the last year.

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

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

SELECT
  COUNT(DISTINCT product_id)
FROM sales
WHERE sale_date >= (CURRENT_DATE - INTERVAL '1 year')
```

```
-- Q.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 1, 2
ORDER BY 3 DESC

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

SELECT
    COUNT(*) as warranty_claim
FROM warranty
WHERE EXTRACT(YEAR FROM claim_date) = 2020

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

-- store_id, day_name, sum(qty)
--  window dense rank

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 1, 2
) as t1
WHERE rank = 1


-- Medium to Hard Questions
-- Q.11 Identify the least selling product in each country for each year 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


-- Q.12 Calculate how many warranty claims were filed within 180 days of a product sale.


SELECT
   COUNT(*)
FROM warranty as w
LEFT JOIN
sales as s
ON s.sale_id = w.sale_id
WHERE
   w.claim_date - sale_date <= 180


--Q.13  Determine how many warranty claims were filed for products launched in the last two years.
-- each prod
--  no claim
--  no sale
-- each must be launcnhed in last 2 year


SELECT
   p.product_name,
```

```
    COUNT(w.claim_id) as no_claim,
    COUNT(s.sale_id)
FROM warranty as w
RIGHT JOIN
sales as s
ON s.sale_id = w.sale_id
JOIN products as p
ON p.product_id = s.product_id
WHERE p.launch_date >= CURRENT_DATE - INTERVAL '2 years'
GROUP BY 1
HAVING COUNT(w.claim_id) > 0

-- Q.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 = 'USA'
    AND
    s.sale_date >= CURRENT_DATE - INTERVAL '3 year'
GROUP BY 1
HAVING SUM(s.quantity) > 5000
```

-- Q.15 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
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
JOIN
category as c
ON c.category_id = p.category_id
WHERE
   w.claim_date >= CURRENT_DATE - INTERVAL '2 year'
GROUP BY 1
```

-- Complex Problems

-- Q.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 1) t1
ORDER BY 4 DESC

-- 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 1, 2, 3
    ORDER BY 2, 3
),
growth_ratio
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_ratio
FROM growth_ratio
WHERE
```

```
    last_year_sale IS NOT NULL
    AND
    YEAR <> EXTRACT(YEAR FROM CURRENT_DATE)

-- 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 Expenses 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_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 1

-- Q.19 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 w.sale_id = s.sale_id
WHERE w.repair_status = 'Paid Repaired'
GROUP BY 1
),

total_repaired
AS
(SELECT
    s.store_id,
    COUNT(w.claim_id) as total_repaired
FROM sales as s
RIGHT JOIN warranty as w
ON w.sale_id = s.sale_id
GROUP BY 1)

SELECT
    tr.store_id,
    st.store_name,
    pr.paid_repaired,
    tr.total_repaired,
    ROUND(pr.paid_repaired::numeric/
```

```
        tr.total_repaired::numeric * 100
    ,2) as percentage_paid_repaired
FROM paid_repair as pr
JOIN
total_repaired tr
ON pr.store_id = tr.store_id
JOIN stores as st
ON tr.store_id = st.store_id


-- Q.20 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 1, 2, 3
ORDER BY 1, 2,3
)
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

-- Bonus Question
-- 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 1, 2
ORDER BY 1, 3 DESC
```