```sql
-- 📊 Sales & Revenue Analysis

--What is the total revenue generated per product?
SELECT
  p.product_name,
  sum(s.total_price) as total_revenue
FROM products p
JOIN sales s ON p.product_id = s.product_id
GROUP BY p.product_name
ORDER BY total_revenue DESC;

--Which store generated the highest revenue in 2025?
SELECT
  st.store_name,
  sum(s.total_price) AS total_revenue,
  strftime('%Y', s.sale_date) AS year
FROM sales s
JOIN stores st ON s.store_id = st.store_id
WHERE strftime('%Y', s.sale_date) = '2025'
GROUP BY st.store_name
ORDER BY total_revenue DESC
LIMIT 1;

--Which month had the highest total sales across all stores?
SELECT
  st.store_name,
  sum(s.total_price) AS total_revenue,
  strftime('%Y-%m', s.sale_date) AS sale_month
FROM stores st
JOIN sales s ON st.store_id = s.store_id
GROUP BY sale_month
```

```
ORDER BY total_revenue DESC
LIMIT 1;

--What is the average order value per store?
SELECT
 st.store_name,
 count(s.sale_id) AS total_sales,
 round(sum(s.total_price), 2) AS total_revenue,
 round(sum(s.total_price) *1.0/count(s.sale_id),2) AS avg_order_value
FROM stores st
JOIN sales s ON st.store_id = s.store_id
GROUP BY st.store_name
ORDER BY avg_order_value DESC;

-- 🛒 Product Insights

--Which are the top 5 best-selling products by quantity sold?
SELECT
  p.product_name,
  sum(s.quantity_sold) AS total_quantity
FROM products p
JOIN sales s ON p.product_id = s.product_id
GROUP BY p.product_name
ORDER BY total_quantity DESC
LIMIT 5;

--List products that were never sold.
SELECT
  p.product_name
FROM products p
LEFT JOIN sales s ON p.product_id = s.product_id
```

```
WHERE s.sale_id IS NULL;


--Which products have a high price but low sales volume?
WITH highest_price AS (
  SELECT product_id, product_name, price
  FROM products
  GROUP BY product_name
  ORDER BY price DESC
 LIMIT 10)

 SELECT
  hp.product_name,
  hp.price,
  sum(s.quantity_sold) AS total_quantity_sold
FROM highest_price hp
JOIN sales s ON hp.product_id = s.product_id
GROUP BY hp.product_name
ORDER BY s.total_price ASC
LIMIT 10;

-- 🧍 Customer Behavior

--Which customer made the highest number of purchases?
SELECT
  c.name,
  count(s.sale_id) AS total_orders,
  sum(s.total_price) AS total_amount
FROM customers c
JOIN sales s ON c.customer_id = s.customer_id
GROUP BY c.name
ORDER BY total_orders DESC
```

```
LIMIT 1;


--Which customer spent the most in total?
SELECT
  c.name,
  sum(s.total_price) AS total_amount
FROM customers c
JOIN sales s ON c.customer_id = s.customer_id
GROUP BY c.name
ORDER BY total_amount DESC
LIMIT 1;


--Find customers who made purchases in every quarter of 2025.
WITH customer_quarter AS (
  SELECT
  c.customer_id,
    c.name,
    CASE
      WHEN strftime('%m', s.sale_date) BETWEEN '01' AND '03' THEN 'Q1'
      WHEN strftime('%m', s.sale_date) BETWEEN '04' AND '06' THEN 'Q2'
      WHEN strftime('%m', s.sale_date) BETWEEN '07' AND '09' THEN 'Q3'
      WHEN strftime('%m', s.sale_date) BETWEEN '10' AND '12' THEN 'Q4'
    END AS quarter
  FROM customers c
  JOIN sales s ON s.customer_id = c.customer_id
  WHERE strftime('%Y', s.sale_date ) = '2025'
)
SELECT
  name,
  count(distinct quarter) AS quarter_covered
FROM customer_quarter
```

```
GROUP BY name
HAVING quarter_covered = 4;

-- 🏪 Store Performance

--Which store had the most diverse product sales (sold the most different products)?
SELECT
  st.store_name,
  count(DISTINCT p.product_id) AS product_variety,
  count(s.sale_id) AS times_sales
FROM sales s
JOIN stores st ON s.store_id = st.store_id
JOIN products p ON s.product_id = p.product_id
GROUP BY st.store_name
ORDER BY product_variety DESC
LIMIT 1;

--Find the peak sales day for each store.
with peak_sale_day AS (
  SELECT
    st.store_name,
    date(s.sale_date) as sale_day,
    sum(s.total_price) as total_sales
  FROM sales s
  JOIN stores st on s.store_id = st.store_id
  group by st.store_name, sale_day
),
ranked_sale AS (
  SELECT *,
        rank() over (PARTITION by store_name ORDER BY total_sales DESC) AS rank
FROM peak_sale_day
```

```
)
SELECT
  store_name,
  sale_day,
  total_sales
FROM ranked_sale
WHERE rank = 1

--Which store had the highest average order value?
SELECT
 st.store_name,
 count(s.sale_id) AS total_sales,
 round(avg(s.total_price), 2) AS avg_order_value
FROM stores st
JOIN sales s ON s.store_id = st.store_id
GROUP BY st.store_name
ORDER BY avg_order_value DESC
LIMIT 1;

-- 📈 Trends & Comparisons

--Show monthly revenue trends over 2024–2025.
SELECT
 strftime('%Y', sale_date) AS year,
 strftime('%m', sale_date) AS month,
 sum(total_price) AS monthly_revenue
FROM sales
WHERE strftime('%Y', sale_date) IN ('2024', '2025')
GROUP BY year, month
ORDER by year, month
```

```
--Compare average revenue per store in 2024 vs 2025.
SELECT
  st.store_name,
  strftime('%Y', s.sale_date) AS year,
  round(avg(s.total_price), 2) AS avg_revenue
FROM sales s
JOIN stores st on s.store_id = st.store_id
WHERE strftime('%Y', s.sale_date) IN ('2024','2025')
GROUP BY st.store_name
ORDER BY year;

--What is the trend of total units sold per month?
SELECT
  strftime('%Y - %m', sale_date) AS year_month,
  sum(quantity_sold) AS total_quantity,
  sum(total_price) AS total_revenue
FROM sales s
GROUP BY year_month
ORDER BY year_month;

-- 🔎 Advanced Challenge

--Which products were consistently top-selling (in top 3) every month?
WITH monthly_product_sales AS (
  SELECT
    p.product_id,
    p.product_name,
    strftime('%Y-%m', s.sale_date) AS year_month,
    SUM(s.quantity_sold) AS total_sold
  FROM sales s
  JOIN products p ON s.product_id = p.product_id
```

```
  GROUP BY p.product_id, year_month
),
ranked_sales AS (
  SELECT *,
    DENSE_RANK() OVER (
      PARTITION BY year_month
      ORDER BY total_sold DESC
    ) AS rank
  FROM monthly_product_sales
)
SELECT product_name
FROM ranked_sales
WHERE rank <= 3
GROUP BY product_id
HAVING COUNT(DISTINCT year_month) = (SELECT COUNT(DISTINCT strftime('%Y-%m', sale_date))
FROM sales);


--Find products whose sales dropped 3 months in a row.
WITH monthly_sales AS (
  SELECT
    p.product_id,
    p.product_name,
    strftime('%Y-%m', s.sale_date) AS year_month,
    SUM(s.total_price) AS monthly_sales
  FROM sales s
  JOIN products p ON p.product_id = s.product_id
  GROUP BY p.product_id, year_month
),
sales_with_lags AS (
  SELECT
    *,
```

```
    LAG(monthly_sales, 1) OVER (PARTITION BY product_id ORDER BY year_month) AS prev_1,
    LAG(monthly_sales, 2) OVER (PARTITION BY product_id ORDER BY year_month) AS prev_2
  FROM monthly_sales
),
dropped_3_months AS (
  SELECT
    product_name,
    year_month,
    monthly_sales,
    prev_1,
    prev_2
  FROM sales_with_lags
  WHERE monthly_sales < prev_1
    AND prev_1 < prev_2
)
SELECT DISTINCT product_name
FROM dropped_3_months;

--          Find customers who only purchased from one store

SELECT
  c.customer_id,
  c.name,
  COUNT(DISTINCT s.store_id) AS store_count
FROM customers c
JOIN sales s ON c.customer_id = s.customer_id
GROUP BY c.customer_id, c.name
HAVING COUNT(DISTINCT s.store_id) = 1;
```