1) Monthly revenue and orders

```
-- 1. Monthly revenue and number of transactions

SELECT
  DATE_FORMAT(Time_of_Purchase, '%Y-%m') AS year_month,
  COUNT(*) AS total_transactions,
  SUM(Purchase_Amount) AS total_revenue,
  AVG(Purchase_Amount) AS avg_order_value
FROM transactions_cleaned
GROUP BY DATE_FORMAT(Time_of_Purchase, '%Y-%m')
ORDER BY year_month;
```

2) Revenue by year, month, weekday (thoda richer)

```
-- 2. Revenue by year, month and day of week

SELECT
  YEAR(Time_of_Purchase) AS year,
  MONTH(Time_of_Purchase) AS month,
  DATE_FORMAT(Time_of_Purchase, '%W') AS day_of_week,
  SUM(Purchase_Amount) AS total_revenue,
  COUNT(*) AS transactions
FROM transactions_cleaned
GROUP BY
  YEAR(Time_of_Purchase),
  MONTH(Time_of_Purchase),
  DATE_FORMAT(Time_of_Purchase, '%W')
ORDER BY year, month;
```

3) Top 10 customers by revenue

```
-- 3. Top 10 customers by total revenue

SELECT
  Customer_ID,
  COUNT(*) AS transactions,
  SUM(Purchase_Amount) AS total_revenue,
  AVG(Purchase_Amount) AS avg_order_value
FROM transactions_cleaned
GROUP BY Customer_ID
ORDER BY total_revenue DESC
LIMIT 10;
```

4) Revenue by purchase category

```
-- 4. Revenue and average order value by purchase category

SELECT
  Purchase_Category,
  COUNT(*) AS transactions,
  SUM(Purchase_Amount) AS total_revenue,
  AVG(Purchase_Amount) AS avg_order_value
FROM transactions_cleaned
GROUP BY Purchase_Category
ORDER BY total_revenue DESC;
```

5) Revenue by purchase channel

```
-- 5. Revenue by purchase channel (e.g., Online, Mobile App, In-store)

SELECT
  Purchase_Channel,
  COUNT(*) AS transactions,
  SUM(Purchase_Amount) AS total_revenue,
  AVG(Purchase_Amount) AS avg_order_value
FROM transactions_cleaned
GROUP BY Purchase_Channel
ORDER BY total_revenue DESC;
```

6) Loyalty member vs non-member revenue summary

Assume `Customer_Loyalty_Program_Member` values `'Yes'`/`'No'`:

```
-- 7. Loyalty program impact on revenue

SELECT
  Customer_Loyalty_Program_Member AS loyalty_member,
  COUNT(*) AS transactions,
  COUNT(DISTINCT Customer_ID) AS unique_customers,
  SUM(Purchase_Amount) AS total_revenue,
  AVG(Purchase_Amount) AS avg_order_value
FROM transactions_cleaned
GROUP BY Customer_Loyalty_Program_Member
ORDER BY total_revenue DESC;
```

This query shows whether loyalty members spend more and how many of our customers are enrolled.

7) Discount bucket performance (optional but strong)

Agar DB me `Discount_Used` numeric hai, tum SQL me bucket bana sakte ho:

```
-- 8. Performance by discount bucket

SELECT
  CASE
    WHEN Discount_Used < 0.10 THEN '0-10%'
    WHEN Discount_Used < 0.20 THEN '10-20%'
    WHEN Discount_Used < 0.30 THEN '20-30%'
    WHEN Discount_Used < 0.50 THEN '30-50%'
    ELSE '50%+'
  END AS discount_bucket,
  COUNT(*) AS transactions,
  SUM(Purchase_Amount) AS total_revenue,
  AVG(Purchase_Amount) AS avg_order_value
FROM transactions_cleaned
GROUP BY
  CASE
    WHEN Discount_Used < 0.10 THEN '0-10%'
    WHEN Discount_Used < 0.20 THEN '10-20%'
    WHEN Discount_Used < 0.30 THEN '20-30%'
    WHEN Discount_Used < 0.50 THEN '30-50%'
    ELSE '50%+'
    END
ORDER BY discount_bucket;
```