Write queries that directly answer predetermined questions from a business stakeholder

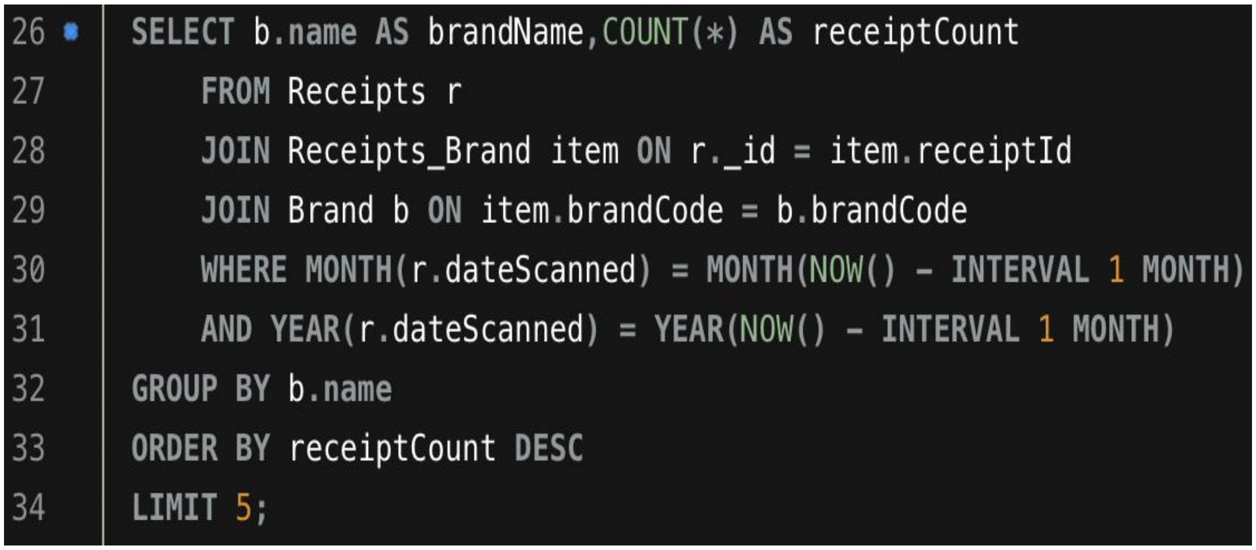
Note

- Created a new table Receipts_Brand as mentioned in ER diagram to make a relation between Receipts and Brand
- All codes were written in MySQL Workbench

Q1. What are the top 5 brands by receipts scanned for most recent month?

Code Explanation

- Inner joining between Receipts and Brand
- Filtering for the previous most recent month since there will be data for the whole month. (For example, since today is June 18th, I am not considering June as most recent month since it is not an entire month hence May will be considered)
- Counting the number of receipts for each brand and displaying top 5 brands

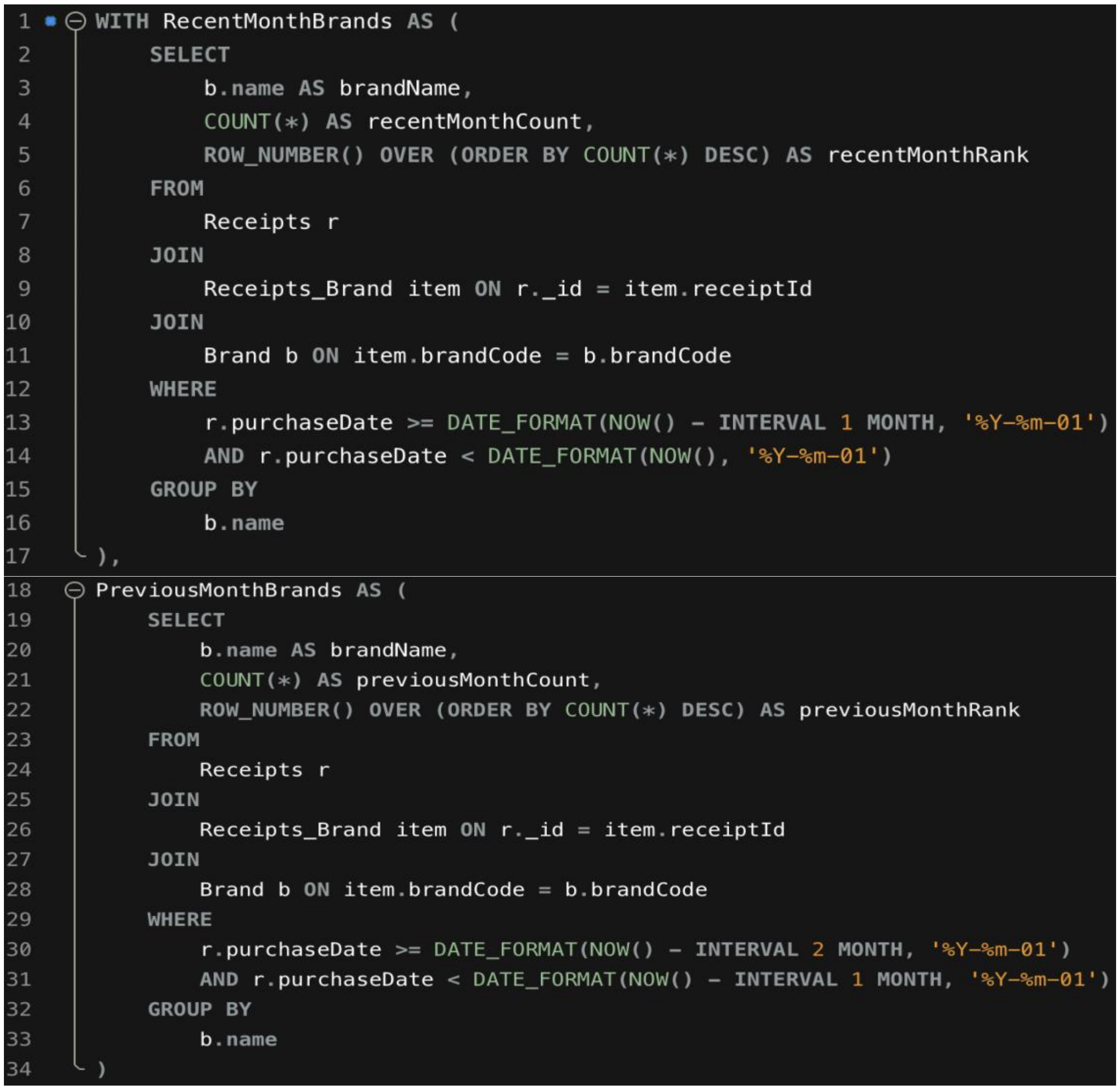


```sql
SELECT b.name AS brandName,COUNT(*) AS receiptCount
    FROM Receipts r
    JOIN Receipts_Brand item ON r._id = item.receiptId
    JOIN Brand b ON item.brandCode = b.brandCode
    WHERE MONTH(r.dateScanned) = MONTH(NOW() - INTERVAL 1 MONTH)
    AND YEAR(r.dateScanned) = YEAR(NOW() - INTERVAL 1 MONTH)
GROUP BY b.name
ORDER BY receiptCount DESC
LIMIT 5;
```

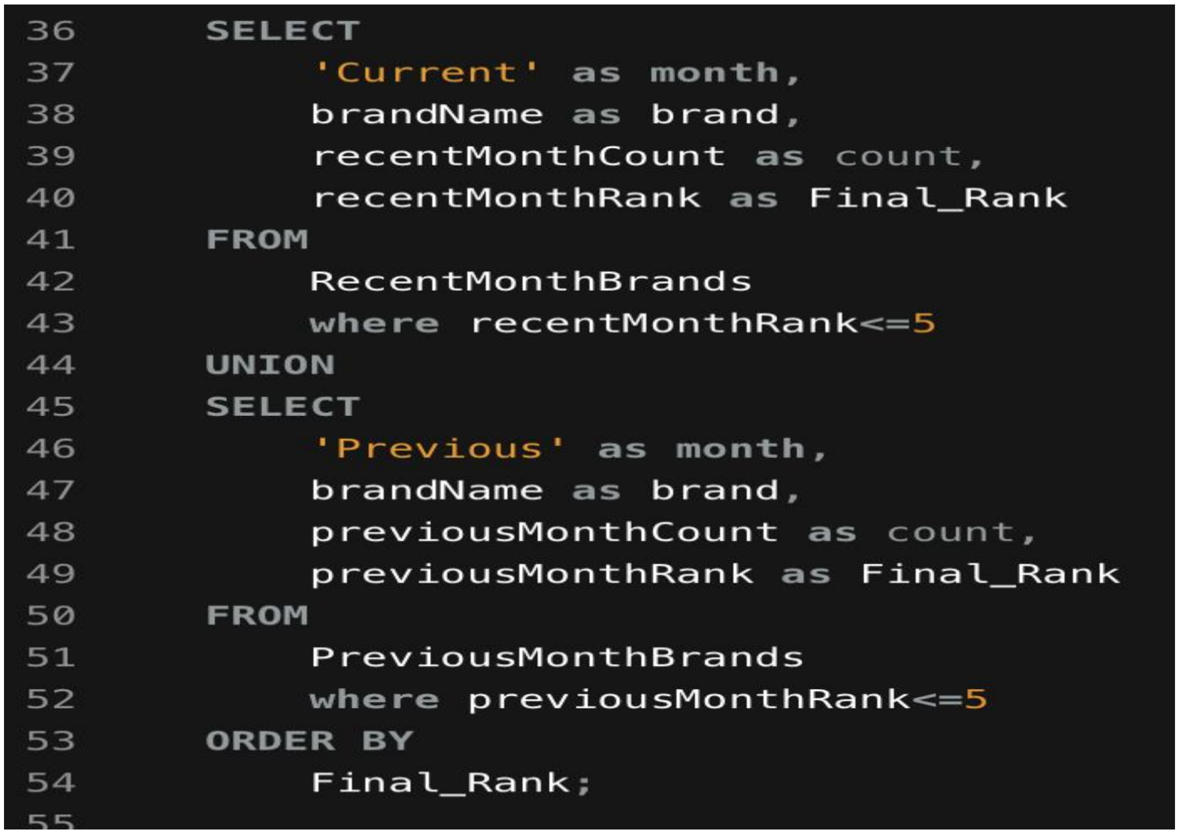
Q2. How does the ranking of the top 5 brands by receipts scanned for the recent month compare to the ranking for the previous month?

Code Explanation – Same logic as Q1

- Ranking based on the count of scanned receipts
- Filtering for the previous most recent month since there will be data for the whole month and the month before that in 2 separate CTEs
- Union the count and brand from both CTEs to display the top 5 brands for the most recent and previous month
- Final Output will have 10 rows, 5 each for recent month and previous month along with top 5 brands respectively



```sql
WITH RecentMonthBrands AS (
    SELECT
        b.name AS brandName,
        COUNT(*) AS recentMonthCount,
        ROW_NUMBER() OVER (ORDER BY COUNT(*) DESC) AS recentMonthRank
    FROM
        Receipts r
    JOIN
        Receipts_Brand item ON r._id = item.receiptId
    JOIN
        Brand b ON item.brandCode = b.brandCode
    WHERE
        r.purchaseDate >= DATE_FORMAT(NOW() - INTERVAL 1 MONTH, '%Y-%m-01')
        AND r.purchaseDate < DATE_FORMAT(NOW(), '%Y-%m-01')
    GROUP BY
        b.name
),
PreviousMonthBrands AS (
    SELECT
        b.name AS brandName,
        COUNT(*) AS previousMonthCount,
        ROW_NUMBER() OVER (ORDER BY COUNT(*) DESC) AS previousMonthRank
    FROM
        Receipts r
    JOIN
        Receipts_Brand item ON r._id = item.receiptId
    JOIN
        Brand b ON item.brandCode = b.brandCode
    WHERE
        r.purchaseDate >= DATE_FORMAT(NOW() - INTERVAL 2 MONTH, '%Y-%m-01')
        AND r.purchaseDate < DATE_FORMAT(NOW() - INTERVAL 1 MONTH, '%Y-%m-01')
    GROUP BY
        b.name
)
```

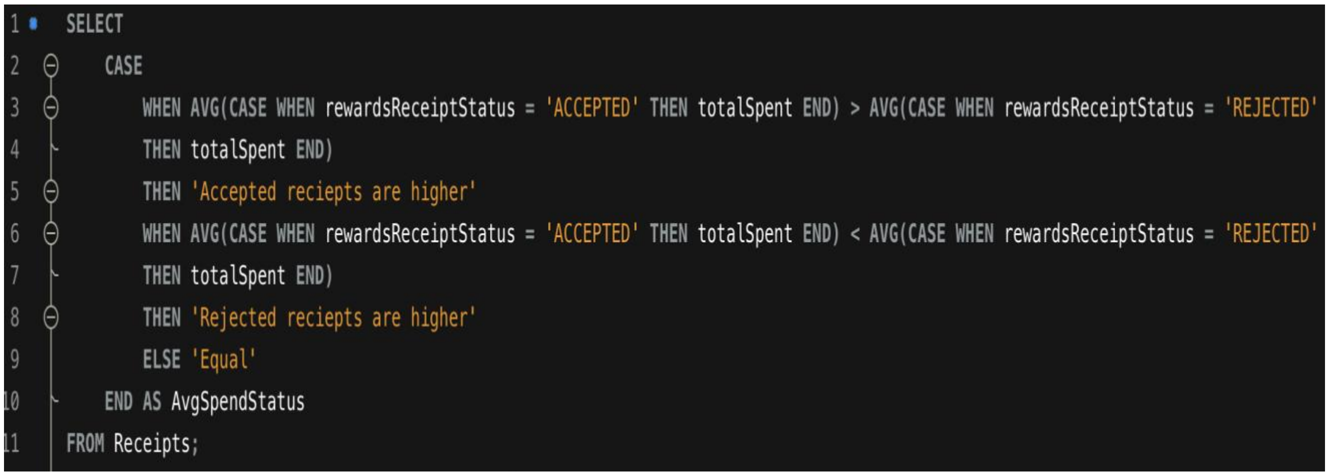


```
    SELECT
        'Current' as month,
        brandName as brand,
        recentMonthCount as count,
        recentMonthRank as Final_Rank
    FROM
        RecentMonthBrands
        where recentMonthRank<=5
    UNION
    SELECT
        'Previous' as month,
        brandName as brand,
        previousMonthCount as count,
        previousMonthRank as Final_Rank
    FROM
        PreviousMonthBrands
        where previousMonthRank<=5
    ORDER BY
        Final_Rank;
```

Q3. When considering average spend from receipts with 'rewardsReceiptStatus' of 'Accepted' or 'Rejected', which is greater?

<u>Code Explanation </u>–

- Using case when to check if status is accepted or rejected and then directly comparing the average of totalspend to check which is greater

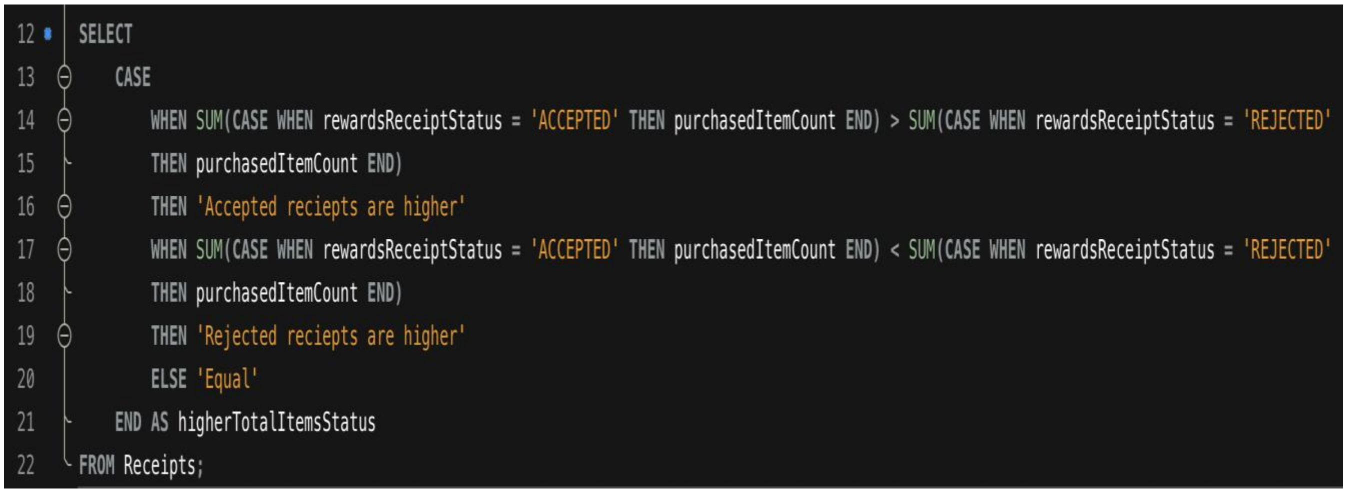


```
SELECT
    CASE
        WHEN AVG(CASE WHEN rewardsReceiptStatus = 'ACCEPTED' THEN totalSpent END) > AVG(CASE WHEN rewardsReceiptStatus = 'REJECTED'
        THEN totalSpent END)
        THEN 'Accepted reciepts are higher'
        WHEN AVG(CASE WHEN rewardsReceiptStatus = 'ACCEPTED' THEN totalSpent END) < AVG(CASE WHEN rewardsReceiptStatus = 'REJECTED'
        THEN totalSpent END)
        THEN 'Rejected reciepts are higher'
        ELSE 'Equal'
    END AS AvgSpendStatus
FROM Receipts;
```

Q4. When considering total number of items purchased from receipts with 'rewardsReceiptStatus' of 'Accepted' or 'Rejected', which is greater?

Code Explanation –

- Using case when to check if status is accepted or rejected and then directly comparing the total number of items to check which is greater

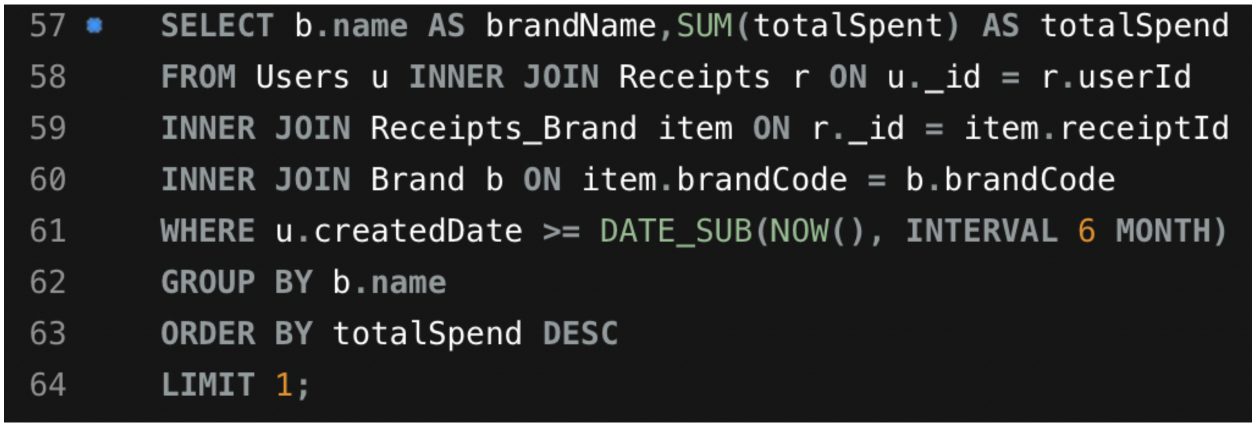


```
SELECT
    CASE
        WHEN SUM(CASE WHEN rewardsReceiptStatus = 'ACCEPTED' THEN purchasedItemCount END) > SUM(CASE WHEN rewardsReceiptStatus = 'REJECTED'
        THEN purchasedItemCount END)
        THEN 'Accepted reciepts are higher'
        WHEN SUM(CASE WHEN rewardsReceiptStatus = 'ACCEPTED' THEN purchasedItemCount END) < SUM(CASE WHEN rewardsReceiptStatus = 'REJECTED'
        THEN purchasedItemCount END)
        THEN 'Rejected reciepts are higher'
        ELSE 'Equal'
    END AS higherTotalItemsStatus
FROM Receipts;
```

Q5. Which brand has the most spend among users who were created within the past 6 months?

Code Explanation –

- Inner joining all 3 tables to since we need data from each of those tables
- Filtering users who were created within the past 6 months
- Displaying the brand with highest spend

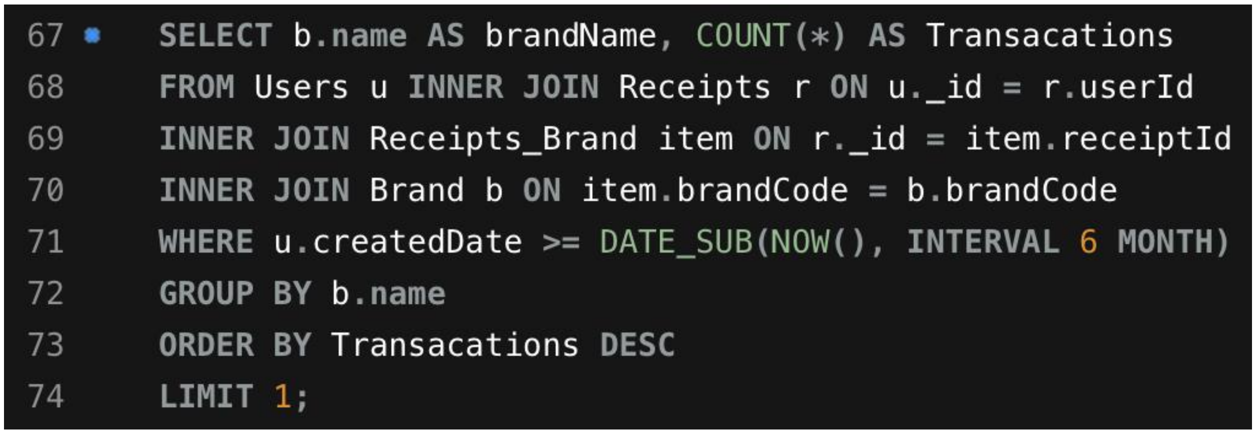


```
SELECT b.name AS brandName,SUM(totalSpent) AS totalSpend
FROM Users u INNER JOIN Receipts r ON u._id = r.userId
INNER JOIN Receipts_Brand item ON r._id = item.receiptId
INNER JOIN Brand b ON item.brandCode = b.brandCode
WHERE u.createdDate >= DATE_SUB(NOW(), INTERVAL 6 MONTH)
GROUP BY b.name
ORDER BY totalSpend DESC
LIMIT 1;
```

Q6. Which brand has the most transactions among users who were created within the past 6 months?

Code Explanation –

- Inner joining all 3 tables to since we need data from each of those tables
- Filtering users who were created within the past 6 months
- Displaying the brand with the most transactions



```sql
SELECT b.name AS brandName, COUNT(*) AS Transacations
FROM Users u INNER JOIN Receipts r ON u._id = r.userId
INNER JOIN Receipts_Brand item ON r._id = item.receiptId
INNER JOIN Brand b ON item.brandCode = b.brandCode
WHERE u.createdDate >= DATE_SUB(NOW(), INTERVAL 6 MONTH)
GROUP BY b.name
ORDER BY Transacations DESC
LIMIT 1;
```