Second: Write a query that directly answers a predetermined question from a business stakeholder

Write a SQL query against your new structured relational data model that answers one of the following bullet points below of your choosing. Commit it to the git repository along with the rest of the exercise.

Note: When creating your data model be mindful of the other requests being made by the business stakeholder. If you can capture more than one bullet point in your model while keeping it clean, efficient, and performant, that benefits you as well as your team.

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

   To determine the top 5 brands by receipts scanned for the most recent month, you can use the following SQL query:

   ```
   SELECT b.name AS brand_name, COUNT(*) AS receipts_scanned
   FROM ReceiptsData AS r
   JOIN TransactionData AS t ON r.receipt_id = t.receipt_id
   JOIN BrandData AS b ON t.brand_id = b.brand_id
   WHERE DATE_TRUNC('month', r.dateScanned) = DATE_TRUNC('month',
   CURRENT_DATE)
   GROUP BY b.name
   ORDER BY receipts_scanned DESC
   LIMIT 5;
   ```

   This query joins the ReceiptsData, TransactionData, and BrandData tables based on the corresponding keys and filters the results based on the most recent month using the DATE_TRUNC function. It then groups the results by brand name, counts the number of receipts scanned for each brand, and orders the results in descending order. Finally, it limits the output to the top 5 brands.

2. **How does the ranking of the top 5 brands by receipts scanned for the recent month compared to the ranking for the previous month?**

   To compare the ranking of the top 5 brands by receipts scanned for the recent month with the ranking for the previous month, you can use the following SQL query:

```
WITH recent_month AS (
  SELECT b.name AS brand_name, COUNT(*) AS receipts_scanned
  FROM ReceiptsData AS r
  JOIN TransactionData AS t ON r.receipt_id = t.receipt_id
  JOIN BrandData AS b ON t.brand_id = b.brand_id
  WHERE DATE_TRUNC('month', r.dateScanned) = DATE_TRUNC('month', CURRENT_DATE)
  GROUP BY b.name
  ORDER BY receipts_scanned DESC
  LIMIT 5
), previous_month AS (
  SELECT b.name AS brand_name, COUNT(*) AS receipts_scanned
  FROM ReceiptsData AS r
  JOIN TransactionData AS t ON r.receipt_id = t.receipt_id
  JOIN BrandData AS b ON t.brand_id = b.brand_id
  WHERE DATE_TRUNC('month', r.dateScanned) = DATE_TRUNC('month', CURRENT_DATE - INTERVAL '1 month')
  GROUP BY b.name
```

```
  ORDER BY receipts_scanned DESC

  LIMIT 5

)

SELECT recent_month.brand_name, recent_month.receipts_scanned,
previous_month.receipts_scanned

FROM recent_month

LEFT JOIN previous_month ON recent_month.brand_name =
previous_month.brand_name;
```

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

   To compare the average spend from receipts with a 'rewardsReceiptStatus' of 'Accepted' and 'Rejected', you can use the following SQL query:

```
SELECT rewardsReceiptStatus, AVG(totalSpent) AS average_spend

FROM ReceiptsData

WHERE rewardsReceiptStatus IN ('Accepted', 'Rejected')

GROUP BY rewardsReceiptStatus;
```

   This query retrieves the 'rewardsReceiptStatus' and calculates the average spend ('totalSpent') for receipts with a status of 'Accepted' or 'Rejected'. The results will show the average spend for each status, allowing you to determine which one is greater.

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

   To compare the total number of items purchased from receipts with a 'rewardsReceiptStatus' of 'Accepted' and 'Rejected', you can use the following SQL query:

```
SELECT rewardsReceiptStatus, SUM(purchasedItemCount) AS
total_items_purchased
FROM ReceiptsData
WHERE rewardsReceiptStatus IN ('Accepted', 'Rejected')
GROUP BY rewardsReceiptStatus;
```

   This query retrieves the 'rewardsReceiptStatus' and calculates the total number of items purchased ('purchasedItemCount') for receipts with a status of 'Accepted' or 'Rejected'. The results will show the total number of items purchased for each status, allowing you to determine which one is greater.

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

   To determine which brand has the most spend among users who were created within the past 6 months, you can use the following SQL query:

```
SELECT b.name AS brand_name, SUM(r.totalSpent) AS total_spend
FROM ReceiptsData AS r
JOIN TransactionData AS t ON r.receipt_id = t.receipt_id
JOIN BrandData AS b ON t.brand_id = b.brand_id
JOIN UsersData AS u ON r.user_id = u.user_id
WHERE u.createdDate >= CURRENT_DATE - INTERVAL '6 months'
GROUP BY b.name
ORDER BY total_spend DESC
LIMIT 1;
```

This query joins the ReceiptsData, TransactionData, BrandData, and UsersData tables based on the corresponding keys. It filters the results to include only users created within the past 6 months using the date comparison condition. Then, it groups the results by brand name and calculates the sum of total spend for each brand. The results are ordered in descending order by total spend, and only the top brand with the highest spend is returned.

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

To determine which brand has the most transactions among users who were created within the past 6 months, you can use the following SQL query:

```
SELECT b.name AS brand_name, COUNT(*) AS transaction_count
FROM TransactionData AS t
JOIN ReceiptsData AS r ON t.receipt_id = r.receipt_id
JOIN BrandData AS b ON t.brand_id = b.brand_id
JOIN UsersData AS u ON r.user_id = u.user_id
WHERE u.createdDate >= CURRENT_DATE - INTERVAL '6 months'
GROUP BY b.name
ORDER BY transaction_count DESC
LIMIT 1;
```

This query joins the TransactionData, ReceiptsData, BrandData, and UsersData tables based on the corresponding keys. It filters the results to include only users created within the past 6 months using the date comparison condition. Then, it groups the results by brand name and calculates the count of transactions for each brand. The results are ordered in descending order by transaction count, and only the brand with the highest transaction count is returned.