**SQL queries**

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

SELECT p.topBrand, COUNT(\*) AS receiptsScanned

FROM Products p

JOIN RewardsReceiptItem as rri ON p.barcode = rri.barcode

JOIN RewardsReceipt as rr ON rri.rewardsReceiptId = rr.\_id

WHERE EXTRACT(YEAR\_MONTH FROM rr.dateScanned) = (

SELECT MAX(EXTRACT(YEAR\_MONTH FROM dateScanned))

FROM RewardsReceipt

)

GROUP BY p.topBrand

ORDER BY receiptsScanned DESC

LIMIT 5;

This query joins the **Products**, **RewardsReceiptItem**, and **RewardsReceipt** tables based on the corresponding relationships. It filters the data to include only the receipts scanned in the most recent month by comparing the year and month extracted from the **dateScanned** column. The results are then grouped by the top brand and sorted in descending order of the number of receipts scanned. Finally, the **LIMIT 5** clause limits the output to the top 5 brands.

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

WITH recent\_month AS (

SELECT p.topBrand, COUNT(\*) AS receiptsScanned

FROM Products p

JOIN RewardsReceiptItem as rri ON p.barcode = rri.barcode

JOIN RewardsReceipt as rr ON rri.rewardsReceiptId = rr.\_id

WHERE EXTRACT(YEAR\_MONTH FROM rr.dateScanned) = (

SELECT MAX(EXTRACT(YEAR\_MONTH FROM dateScanned))

FROM RewardsReceipt

)

GROUP BY p.topBrand

ORDER BY receiptsScanned DESC

LIMIT 5

), previous\_month AS (

SELECT p.topBrand, COUNT(\*) AS receiptsScanned

FROM Products p

JOIN RewardsReceiptItem as rri ON p.barcode = rri.barcode

JOIN RewardsReceipt as rr ON rri.rewardsReceiptId = rr.\_id

WHERE EXTRACT(YEAR\_MONTH FROM rr.dateScanned) = (

SELECT MAX(EXTRACT(YEAR\_MONTH FROM dateScanned)) - INTERVAL 1 MONTH

FROM RewardsReceipt

)

GROUP BY p.topBrand

ORDER BY receiptsScanned DESC

LIMIT 5

)

SELECT rm.topBrand AS recent\_month\_brand, rm.receiptsScanned AS recent\_month\_scanned,

pm.topBrand AS previous\_month\_brand, pm.receiptsScanned AS previous\_month\_scanned

FROM recent\_month as rm

LEFT JOIN previous\_month as pm ON rm.topBrand = pm.topBrand;

This query uses common table expressions (CTEs) to retrieve the top 5 brands and the corresponding number of receipts scanned for both the recent month and the previous month. The **recent\_month** CTE calculates the rankings for the most recent month, while the **previous\_month** CTE calculates the rankings for the month prior to the most recent month.

The final select statement joins the two CTEs on the **topBrand** column and retrieves the brand, the number of receipts scanned for the recent month, and the number of receipts scanned for the previous month. It uses a left join to include all the top brands from the recent month, even if they are not present in the previous month's rankings.

The result will provide a comparison of the top 5 brands by receipts scanned for the recent month and the previous month, showing the brand name, the number of receipts scanned for each month, and the corresponding rankings.

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

SELECT rewardsReceiptStatus, AVG(totalSpent) AS averageSpend

FROM RewardsReceipt

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

GROUP BY rewardsReceiptStatus;

This query retrieves the 'rewardsReceiptStatus' and calculates the average spend ('totalSpent') for receipts with the 'rewardsReceiptStatus' values of 'Accepted' or 'Rejected'. The **AVG()** function is used to calculate the average.

The result will provide two rows, one for 'Accepted' and one for 'Rejected', along with their corresponding average spend values. You can compare the average spend values to determine which one is greater.

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

SELECT rewardsReceiptStatus, SUM(purchasedItemCount) AS totalItemsPurchased

FROM RewardsReceipt

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

GROUP BY rewardsReceiptStatus;

This query retrieves the 'rewardsReceiptStatus' and calculates the sum of purchased item counts ('purchasedItemCount') for receipts with the 'rewardsReceiptStatus' values of 'Accepted' or 'Rejected'. The **SUM()** function is used to calculate the total.

The result will provide two rows, one for 'Accepted' and one for 'Rejected', along with their corresponding total item counts. You can compare the total item counts to determine which one is greater.

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

SELECT p.brandCode, SUM(rr.totalSpent) AS totalSpend

FROM Products p

JOIN RewardsReceiptItem rri ON p.barcode = rri.barcode

JOIN RewardsReceipt rr ON rri.rewardsReceiptId = rr.\_id

JOIN User u ON rr.userId = u.\_id

WHERE u.createdDate >= DATE\_SUB(CURDATE(), INTERVAL 6 MONTH)

GROUP BY p.brandCode

ORDER BY totalSpend DESC

LIMIT 1;

This query joins the 'Products', 'RewardsReceiptItem', 'RewardsReceipt', and 'User' tables. It calculates the total spend for each brand among the users who were created within the past 6 months. The **SUM()** function is used to calculate the total spend, and the results are grouped by the 'brandCode'. The **ORDER BY** clause sorts the results in descending order of total spend, and the **LIMIT 1** ensures that only the brand with the highest total spend is returned.

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

SELECT p.brandCode, COUNT(\*) AS transactionCount

FROM Products as p

JOIN RewardsReceiptItem as rri ON p.barcode = rri.barcode

JOIN RewardsReceipt as rr ON rri.rewardsReceiptId = rr.\_id

JOIN User u ON rr.userId = u.\_id

WHERE u.createdDate >= DATE\_SUB(CURDATE(), INTERVAL 6 MONTH)

GROUP BY p.brandCode

ORDER BY transactionCount DESC

LIMIT 1;

This query joins the 'Products', 'RewardsReceiptItem', 'RewardsReceipt', and 'User' tables. It counts the number of transactions for each brand among the users who were created within the past 6 months. The **COUNT(\*)** function is used to calculate the transaction count, and the results are grouped by the 'brandCode'. The **ORDER BY** clause sorts the results in descending order of transaction count, and the **LIMIT 1** ensures that only the brand with the highest transaction count is returned.