Second: Write queries that directly answer predetermined questions from a business stakeholder

As discussed in the previous section, the pandas dataframes are loaded into my MYSQL environment after creating the necessary Database and Tables. **Users, Brands, Receipts, Rewards_Receipts** tables are created in the FETCH_DB Database. The loading of these tables to MYSQL Database happens at the end of 'DateQualityIssues_Loading_Data.ipynb' notebook.

Now the SQL queries are written in query_x_y.ipynb notebooks to answer the following Business questions.

Question 1: What are the top 5 brands by receipts scanned for most recent month?

Answer:

Brands with the highest number of occurrences on all receipts: This calculates the total number of times items from a brand are present across all scanned receipts, counting every instance where an item from the brand is included.

As mentioned in the Data Quality Section regarding brand and brandCode, the best possible way to associate receipts with brands is through brandCode, which has a good percentage of missing values. As discussed before, the first word from the description is extracted to impute the missing brandCode. Additionally, brandCode remains the primary identifier for naming purposes.

The most recent month, **2021-03-01**, is dynamically determined. The query ensures that this is appropriately handled in the analysis. It can be observed that only two brands are present in the data for the most recent month. To effectively answer the question of the top 5 brands by receipts scanned, it would be beneficial to extend the query to include data from a few previous months as well, allowing for a more comprehensive analysis. The queries are written for the most recent month, **2021-02-01** and **2021-01-01.**

With **CTE COUNT_BRAND_RECEIPTS**, the tables **Rewards_Receipts** and **Receipts** are joined. The scanned date and time is obtained from **the Receipts** table and used for querying the most recent month's data. After grouping by *brandCode* information from **Rewards_Receipts** table, ranks are assigned based on the **COUNT(Rewards_Receipts.receipt_id).** DENSE_RANK () is used to assign ranks.

```sql
-- Please comment and uncomment the recent_month variable accordingly

SET @MOST_RECENT_MONTH = (SELECT DATE_FORMAT(MAX(scanned_date_time), '%Y-%m-01') FROM receipts);

-- Since there are only 2 Distinct brands for the most recent month,
-- it is worthwhile to check out previous months as well

-- SET @MOST_RECENT_MONTH = '2021-02-01';

-- SET @MOST_RECENT_MONTH = '2021-01-01';

WITH COUNT_BRAND_RECEIPTS AS(
    SELECT RR.brandCode AS Brand,
    COUNT(RR.receipt_id) AS ReceiptCount,
    DENSE_RANK() OVER(ORDER BY COUNT(RR.receipt_id) DESC) AS BrandRank
    FROM rewards_receipts RR
    INNER JOIN receipts R
    ON RR.receipt_id = R.receipt_id
    WHERE DATE_FORMAT(R.scanned_date_time,'%Y%m') = DATE_FORMAT(@MOST_RECENT_MONTH, '%Y%m')
      AND RR.brandCode IS NOT NULL
      AND RR.brandCode != 'ITEM'
    GROUP BY Brand
)

SELECT Brand, ReceiptCount, BrandRank
FROM COUNT_BRAND_RECEIPTS
WHERE BrandRank <= 5
ORDER BY BrandRank ASC;
```

Here are the results:

For the Most Recent Month:

brandCode	Receipts Scanned	Brand_Rank
MUELLER	11	1
THINDUST	11	1

Brands Mueller and Thindust share the first rank in terms of the count of Receipts Scanned.

For the Month: '2021-02-01'

brandCode	Receipts Scanned	Brand_Rank
MUELLER	29	1
THINDUST	29	1
FLIPBELT	28	2
HEINZ	10	3
BRAND	3	4
DORITOS	3	4
DELETED	2	5
JIF	2	5
SUAVE	2	5
MISSION	2	5
CAPRI	2	5
SPIGEN	2	5

From the above list, brands like 'DELETED','BRAND' do not make sense. This is due to the limitation of the imputation method discussed earlier. In some cases, the brand codes may not be accurate when extracted from the description.

For the Month: '2021-01-01'

brandCode	Receipts Scanned	Brand_Rank
HY-VEE	296	1
BEN AND JERRYS	180	2
PC	138	3
KLARBRUNN	133	4
PEPSI	103	5

The brand 'PC' might not be a reasonable result, which highlights the limitation of the imputation method discussed earlier. In certain cases, extracting brand codes from the description may not yield accurate results. However, brands like ‘KLARBRUNN’ appearing in the query results demonstrate the effectiveness of the imputation technique. Without this method, brands like KLARBRUNN would likely not have surfaced in the analysis.

Question 2: What are the top 5 brands by receipts scanned for most recent month?

Answer:

This question extends the previous analysis. By calculating the ranks for the current month (already computed in the previous step) and comparing them to the ranks from the previous month, we can measure whether there has been any improvement or decline in the ranks of the brands over time.

The CTE to calculate ranks:

```
WITH CURRENT_RANK_TABLE AS (
    SELECT RR.brandCode AS Brand,
    SUM(IF(
    DATE_FORMAT(R.scanned_date_time,'%Y%m') = DATE_FORMAT(@MOST_RECENT_MONTH,'%Y%m'),1,0
    )) AS Current_Month_ReceiptsCount,
    SUM(IF(
    DATE_FORMAT(R.scanned_date_time, '%Y%m') = DATE_FORMAT(DATE_SUB(@MOST_RECENT_MONTH, INTERVAL 1 MONTH), '%Y%m'), 1, 0
    )) AS Previous_Month_ReceiptsCount,

    -- Apply DENSE_RANK() for Current Month Count
    DENSE_RANK() OVER
    (ORDER BY SUM(IF(DATE_FORMAT(R.scanned_date_time,'%Y%m') = DATE_FORMAT(@MOST_RECENT_MONTH,'%Y%m'), 1, 0)) DESC)
    AS Current_Brand_Rank,

    -- Apply DENSE_RANK() for Previous Month Count
    DENSE_RANK() OVER
    (ORDER BY SUM(IF(DATE_FORMAT(R.scanned_date_time, '%Y%m') = DATE_FORMAT(DATE_SUB(@MOST_RECENT_MONTH, INTERVAL 1 MONTH), '%Y%m'), 1, 0))
    AS Previous_Brand_Rank

    FROM rewards_receipts RR
    INNER JOIN receipts R
    ON RR.receipt_id = R.receipt_id
    WHERE RR.brandCode IS NOT NULL AND RR.brandCode != 'ITEM'
    GROUP BY Brand
)
```

Followed by the select statements:

```
SELECT Brand,
Current_Month_ReceiptsCount,
Previous_Month_ReceiptsCount,
Current_Brand_Rank,
Previous_Brand_Rank,
CAST(Previous_Brand_Rank AS SIGNED) - CAST(Current_Brand_Rank AS SIGNED) Rank_Difference
FROM CURRENT_RANK_TABLE
WHERE Current_Brand_Rank <= 5 AND Current_Month_ReceiptsCount!= 0
ORDER BY Current_Brand_Rank ASC;
```

Rewards_Receipts and Receipts tables are joined. SUM IF statements are used to calculate the receipt counts for the current month and previous month. They are also used to calculate the respective ranks.

Ranks are cast as SIGNED so that if there are negative ranks, there will not be any BIGINT out of range errors.

Here are the results:

For the most recent Month:

Brand	Current_Month_ReceiptsCount	Previous_Month_ReceiptsCount	Current_Month_Rank	Previous_Month_Rank	rank_Difference
MUELLER	11	29	1	1	0
THINDUST	11	29	1	1	0

There has been no rank difference for the brands Mueller and Thindust. However, analyzing the rank differences for previous months to obtain the top 5 brands information.

For the most recent Month as '2021-02-01':

Brand	Current_Month_ReceiptsCount	Previous_Month_ReceiptsCount	Current_Month_Rank	Previous_Month_Rank	rank_Difference
MUELLER	29	4	1	51	50
THINDUST	29	4	1	51	50
FLIPBELT	28	22	2	33	31
HEINZ	10	22	3	33	30
DORITOS	3	92	4	10	6
BRAND	3	19	4	36	32
MISSION	2	17	5	38	33
CAPRI	2	12	5	43	38
SUAVE	2	9	5	46	41
DELETED	2	7	5	48	43
JIF	2	7	5	48	43
SPIGEN	2	0	5	55	50

For the most recent Month as '2021-01-01':

Brand	Current_Month_ReceiptsCount	Previous_Month_ReceiptsCount	Current_Month_Rank	Previous_Month_Rank	rank_Difference
HY-VEE	296	0	1	1	0
BEN AND JERRYS	180	0	2	1	-1
PC	138	0	3	1	-2
KLARBRUNN	133	0	4	1	-3
PEPSI	103	0	5	1	-4

Since there were no records of any receipt scan in the month of Dec-2020; the Previous month receipt counts are ‘0’ and the query assumes rank 1 for it. Due to these corner cases, it is always important to analyze the quantity (Receipt count here) along with the ranks or rank differences.

Question 3 and 4:

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

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

There is no information about the reward being ‘accepted’ in the table. Here is the distribution of the Rewards_Receipt_status:

FINISHED	518
SUBMITTED	434
REJECTED	71
PENDING	50
FLAGGED	46

Based on the above information, 'Accepted' status could be understood as 'Finished' status.

The information of *Total Money spent* on receipts and the *reward status* is present in **Receipts** table

```
SELECT R.rewardsReceiptStatus As RewardStatus,
ROUND(AVG(COALESCE(R.totalSpent,0)),4) AS Avg_total_spent
FROM receipts R
GROUP BY R.rewardsReceiptStatus
HAVING R.rewardsReceiptStatus = 'FINISHED' or R.rewardsReceiptStatus = 'REJECTED'
ORDER BY Avg_total_spent DESC;
```

Here is the output:

```
+---------------+-----------------+
| RewardsStatus | Avg_total_spent |
+---------------+-----------------+
|    FINISHED   |     80.8543     |
|    REJECTED   |     23.3261     |
+---------------+-----------------+
```

Finished or Accepted Status has the higher average of the total money spent.

The information of *Total number of items purchased* from receipts and the *reward status* is present in **Receipts** table.

Query:

```
SELECT R.rewardsReceiptStatus As RewardStatus,
ROUND(SUM(COALESCE(R.purchasedItemCount,0)),4) AS Total_number_of_items_purchased
FROM receipts R
GROUP BY R.rewardsReceiptStatus
HAVING R.rewardsReceiptStatus = 'FINISHED' or R.rewardsReceiptStatus = 'REJECTED'
ORDER BY Total_number_of_items_purchased DESC;
```

Here is the output:

```
+---------------+---------------------------------+
| RewardsStatus | Total_number_of_items_purchased |
+---------------+---------------------------------+
|    FINISHED   |              8184.0             |
|    REJECTED   |              173.0              |
+---------------+---------------------------------+
```

Finished or Accepted Status has the higher total sum of the number of purchased items.

Question 5 and 6:

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

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

The tables **Rewards_Receipts**, **Receipts**, and **Users** are joined to filter the information for users who were created within the past 6 months, allowing us to extract the total money spent on each brand. The money spent is derived from the **Rewards_Receipts**.*finalPrice*, as this column represents the actual price users paid for purchasing products from each brand.

Query:

```
SELECT RR.brandCode AS Brand, SUM(RR.finalPrice) AS total_money_spent
FROM rewards_receipts RR
INNER JOIN receipts R
ON RR.receipt_id = R.receipt_id
INNER JOIN users U
ON R.userId = U.user_id
WHERE DATE_FORMAT(U.created_date_time,'%Y%m') <= (SELECT DATE_FORMAT(MAX(created_date_time),'%Y%m') FROM users)
AND DATE_FORMAT(U.created_date_time,'%Y%m') > (SELECT DATE_FORMAT(DATE_SUB(MAX(created_date_time), INTERVAL 6 MONTH), '%Y%m') FROM users)

GROUP BY RR.brandCode
ORDER BY total_money_spent DESC
LIMIT 1;
```

Here are the results:

brand	total_money_spent
HUGGIES	1931.92

Huggies brand has the highest spend (with 1931.92 dollars) among the users that were created within the past 6 months (from the latest month)

To query the most transactions for a brand that was bought by users that were created within the last 6 months (from the latest month), it is similar to the previous uestion only that instead of Final Price, we query based on the count (To obtain the number of transactions)

Query:

```
SELECT RR.brandCode AS Brand, COUNT(*) AS total_transactions
FROM rewards_receipts RR
INNER JOIN receipts R
ON RR.receipt_id = R.receipt_id
INNER JOIN users U
ON R.userId = U.user_id
WHERE DATE_FORMAT(U.created_date_time,'%Y%m') <= (SELECT DATE_FORMAT(MAX(created_date_time),'%Y%m') FROM users)
AND DATE_FORMAT(U.created_date_time,'%Y%m') > (SELECT DATE_FORMAT(DATE_SUB(MAX(created_date_time), INTERVAL 6 MONTH), '%Y%m') FROM users)
GROUP BY RR.brandCode
ORDER BY total_transactions DESC
LIMIT 1;
```

Result:

brand	total_transactions
HY-VEE	296

HY-VEE brand has the most number of transactions (with 296 transactions) amongst the brands that were bought by the users who were created within the last 6 months.