dim_item

Granularity:

Each row represents a unique **item_uuid** with attributes such as **item_name**, **item_code**, **category**, **brand**, and **pricing**. There are **~5,000 unique items** in total.

Data Integrity Issues:

- **itemCode:** Several rows contain **NA** or invalid values, impacting item identification and analysis.
- **Missing Pricing:** A significant number of items lack **pricing data**, which affects revenue and promotional analysis.
- **Category/Brand Mismatches:** Some items are incorrectly associated with **category** or **brand**, potentially causing incorrect reporting in marketing or inventory analysis.
- **Description Errors:** Inconsistent or incomplete **item descriptions**, which can impact product searchability and data consistency.

• **Opportunities:**

- **Item Performance Analysis:** Investigate top-selling and underperforming items by **category** and **brand**. Insights can inform inventory management and sales strategies.
- **Pricing Optimization:** Analyze pricing trends over time to identify potential areas for pricing adjustments, offering discounts, or optimizing pricing models.
- **Inventory Management:** Leverage item data to monitor **stock levels** and identify slow-moving or high-demand items, facilitating better supply chain and procurement decisions.
- **Product Segmentation:** Segment items based on performance or customer preference to improve targeted marketing efforts and optimize promotions for specific groups.

• **Improvement Suggestions:**

- **Standardize itemCode:** Cleanse data to remove **NA** values and ensure each **item_code** is valid and consistent across the dataset.
- **Implement Price Validation:** Ensure **pricing data** is captured for all items, setting rules to flag missing or incorrect pricing.
- **Fix Category/Brand Mapping:** Correct **category/brand mismatches** by linking items to the right categories and brands using a reference table.
- **Clean Item Descriptions:** Review and standardize **item descriptions**, ensuring they are complete and accurate, and removing any invalid characters.

---

dim_brand

Granularity:

Captures brand-level details, where each row represents a unique **brand_uuid**. It includes the brand name, brand code, category, category code, a flag indicating top brand status, and the associated **brandCPGId**. There are **1167 unique brands** in the dataset.

Data Integrity Issues:

- **brandCode and brandCategoryCode:** Many rows have "NA" values or duplicate values from brandCategory or brandName. These need to be standardized with heavy pre-processing.
- **Test Brands:** Several rows contain test brand names (e.g., "@xxx" format) that need to be removed. The exact count is not specified.
- **Ref Value Errors:** Instances where CPGs are mistakenly entered as COGs. The exact count is not specified.

Opportunities:

- **Market Segmentation:** The dataset can support broad analysis opportunities to identify different types of brands and categories, segmenting them by their unique characteristics, product offerings, and CPG groupings. This can help optimize marketing efforts, product placement, and target different demographics effectively.
- **Brand Performance Analysis:** Analysis of top-performing brands, examining metrics like market share, growth trends, and customer loyalty across different categories. By focusing on the correlation between brand codes, top brand status, and product offerings, companies can identify opportunities to push successful products or brands to the forefront of their strategy.
- **CPG Grouping Trends:** Analyzing how brands fall under similar CPGs can help understand category-wide performance trends and create opportunities for product bundling, targeted campaigns, and future partnerships across brands within the same CPG group.

Data Improvement Recommendations:

- **SCD Type 1 for topBrandFlag:** Consider using **SCD Type 1** for the **topBrandFlag**. Currently, it is stored as **SCD Type 0**, meaning it is immutable. If the top brand status evolves over time, **SCD Type 1** would allow overwriting of the previous value. If frequent changes are required, **SCD Type 2** might be a better fit to track changes historically.
- **Vertical Scaling with Brand Offerings: 11 brands** offer two product offerings, while around **1,145 brands** provide only one. There is a business opportunity to scale vertically by increasing partnerships with existing brands to expand their product offerings.
- **Brand Code and Category Code Standardization:** Ensure that all **brandCode** and **brandCategoryCode** values are mapped to easily recognizable, standardized codes. This can be achieved through a reference table linking the textual values of **brandName** and **brandCategory** to their corresponding codes.
- **Validation Checks:** Implement validation checks to ensure that every record has a valid, non-null **brandCode** and **brandCategoryCode**. Automate this process to ensure data consistency.
- **Test Data Removal:** Set up a cleaning process to identify and remove test brand entries (e.g., those with "@xxx" format) from the dataset

---

fact_receipt_items_bridge

Granularity:

This entity links the receipt data with the item and brand details, connecting barcodes and brand codes to items.

Data Integrity Issues:

- **Barcode Issues:** 3851 instances of NA values for barcode in the rewardsReceiptItemList. This implies that several receipts don't have or improperly scan barcodes.
- **Barcode Matching:**
  Matches: 82
  Mismatches: 6859
  Total barcodes in receipts data: 568
  Total barcodes in brands data in database: 1160
  This is a high rate of barcodes mismatch - if barcodes don't match for valid items, spenders may not receive rewards for purchases which can lead to customer dissatsisfaction
- **BrandCode Matching:**
  Matches: 629
  Mismatches: 6312
  Total brandCodes in receipts data: 227
  Total brandCodes in brands data in database: 897
  This is a high rate of barcodes mismatch - if brand Codes don't match for valid cases, Brands may not get credit for purchases, affecting negotiations and partnership opportunities

Opportunities:

- **Product Performance and Sales Insights:** Improved barcode and brand code matching will enable better analysis of top-selling products, seasonal trends, and sales performance across different categories.
- **Vendor and Brand Management:** Accurate brand code matching can help track brand performance more effectively, providing valuable insights for brand negotiations and strategic vendor partnerships.
- **Customer Purchase Behavior:** With accurate item data, we can analyze customer purchasing habits, identify popular items, and tailor personalized promotions or loyalty programs to increase retention and engagement.
- **Operational Efficiency:** Addressing data gaps will help improve the accuracy of inventory tracking and reporting, enabling better stock management and minimizing the risk of stockouts or overstocking

Data Improvement Recommendations:

- **Pre-validation Before Ingestion:** Implement checks during data ingestion to ensure barcode and brand code exist in the master database.
- **Standardization & Cleaning:** Trim spaces and ensure barcodes and brand codes are consistently formatted (e.g., uppercase). For missing barcodes, use alternative identifiers like partnerItemId or itemNumber for matching.
- **Improve OCR Accuracy:** If barcodes are extracted from receipts using OCR, improve OCR accuracy and implement confidence thresholds for accepting barcode data.

---

fact_receipts

Granularity:

This entity captures the receipt data, including information such as bonusPointsEarnedReason, pointsAwardedDate, and the status of the receipt (rewardsReceiptStatus).

Data Integrity Issues:

- **Missing bonusPointsEarnedReason:** 575 out of 1179 records have NA for bonusPointsEarnedReason, which impacts the understanding of why certain points were earned. Fortunately, bonusPointsEarned are 0 for these cases.
- **Incorrect pointsAwardedDate:** Points should not have an awarded date if no points are earned, instances where reverse is true
- **Overwritten rewardsReceiptStatus:** It appears that the receipt status is overwritten (SCD Type 1), losing the history of status transitions, which hinders tracking receipt progress and trend analysis.
- **rewardsReceiptStatus Rejected status:** instances when there are status is rejected but pointsEarned is non-zero
- **Date Mismatch:** One instance where createDate differs from dateScanned (potential data entry issue, (oid => 6000c74b0a7214ad4c000060)
- **Missing rewardsReceiptItemList:** 440 receipts out of 1119 have "NaN" for rewardsReceiptItemList. There are 2 rows among these that do not have rewardsReceiptItemList and yet have non-zero vaues in bonusPointsEarned and pointsEarned
- **Barcode and Brand Code Issues:** Missing or incorrect values in barcode (NA for 3851), brandCode (4341 missing), and itemNumber (6788 NA) for scanned items.
- **Review Flag:** needsFetchReview is True for 813 receipts, indicating further data issues.

Opportunities:

- **Receipt Lifecycle Analysis:** By implementing SCD Type 2 for rewardsReceiptStatus, we can track receipt status transitions over time. This will help us analyze the full receipt lifecycle, identify bottlenecks, and improve operational efficiency by understanding where delays or issues occur in the process.
- **Customer Reward Trends:** Analyzing bonusPointsEarned and pointsAwardedDate can provide valuable insights into reward program effectiveness and customer engagement. We can identify peak redemption periods and trends to better tailor promotions and rewards strategies.
- **Fraud Detection & Prevention:** Analyzing receipts with missing or incorrect data (e.g., missing item numbers or brand codes) can help identify potential fraud cases or system exploitation, leading to the development of better fraud detection mechanisms.

Data Improvement Recommendations:

- **SCD Type 2 Implementation:** Introduce SCD Type 2 for rewardsReceiptStatus to maintain historical transitions and capture the receipt lifecycle accurately.
- **Bonus Points and Rewards Date Validation:** Ensure bonusPointsEarnedReason is populated for all records, and flag those with missing values. pointsAwardedDate should only be populated if points are awarded.
- **Barcode and Brand Code Validation:** Implement better validation at the point of data entry (e.g., receipt scanning) to ensure mandatory fields like barcodes, brand codes, and item numbers are captured correctly.
- **Data Cleansing:** Address missing rewardsReceiptItemList entries and ensure all receipt records are properly populated with item and brand data.
- **Review Flag Analysis:** Investigate why needsFetchReview is flagged for 813 records, and improve the underlying data quality process.

---

dim_user

Granularity:

Each row represents a unique user with details about their account creation, login, state, role, and status.

Data Integrity Issues:

- **Email Redaction:** Email is either redacted or not captured, which can lead to potential misuse if the same user redeems rewards multiple times.
- **Missing signUpSource:** 48 rows have "None" values in the signUpSource column, preventing accurate tracking of sign-up channels.
- Multiple entries exist with the same oid in the users, where only 212 unique oids are found among 495 entries

Opportunities:

- **Activity-based Ranking:** Users may change their activity status over time. Implementing loyalty tiers such as bronze, silver, gold, or introducing additional attributes like profileCompletion, emailVerified, or DOB could improve customer engagement.
- **Churn Detection:** Since the isActiveFlag seems tied to the last login date, users with long gaps in activity (e.g., over 6 months) could be targeted for reactivation campaigns. Only one user currently has isActiveFlag set to False, indicating outdated data in the snapshot.
- **Geographic Targeting:** 396 of 495 consumers are located in Wisconsin, suggesting an opportunity for localized campaigns. Further expansion into other states could be pursued through partnerships with new brands.

Data Improvement Recommendations:

- **Email Capture:** Ensure emails are consistently captured to prevent misuse and allow for proper cross-referencing.
- **SignUpSource Validation:** Apply domain integrity rules to the signUpSource column to only accept valid entries like "Email," "Mobile App," or "Referral."
- **Churn Detection Logic:** Implement automated processes to flag users who haven't logged in for over 6 months, and use this data for re-engagement campaigns.
- Snapshot data has 495 consumers of which 82 are classified as fetch-staff, if we are only analyzing external customers and internal **fetch-staff** data is not relevant for our analyses, it might make sense to store them separately
- Consider using SCD type 2/3 for isActiveFlag if users change status over time

---

dim_date

Granularity:

Each row represents a specific date, and additional attributes (like month, fiscal year, day of the week, and holiday flag) help categorize and organize the date.

Opportunities:

- **Additional Date Attributes:** Adding fields such as quarter, week number, and utilizing dim_time for more granular analysis would allow tracking of events such as rush-hour activity or exclusive event trends.
- **Improved Analytics:** Having more detailed time-based data could also improve insights into processing times for rewards and other event-based activities.

Data Improvement Recommendations:

- **Expand Date Fields:** Add quarter, week number, and possibly dim_time to enable more detailed tracking and analysis of time- based data, especially for events or rush periods
- Would provide insight into the turnaround or processing time of reward statuses, as most rewards are processed within a single day
- Replace dates with **datekeys** in all entity tables to improve performance

---

Couple Examples of Data Quality Issues

instances where **barcode** and **brandCode** are missing, and instead of a quantity increase, the same item appears as two to three separate line items.

```python
# Can one receipt have multiple brandCodes -- yes

# brand_count_per_oid = receipt_items_df.groupby('oid')['brandCode'].nunique().reset_index(name='brandCode_count')
# brand_count_per_oid = brand_count_per_oid.sort_values(by=['brandCode_count'], ascending=False)  # Add this line
# print(brand_count_per_oid)

# lets look at sample receipts such that oid have muliptle brandcode

oids_to_show = ['6000b2be0a7214ad4c00004d', '6008f09d0a720f05fa000128']
receipt_items_df[receipt_items_df['oid'].isin(oids_to_show)][['oid', 'dateScanned', 'brandCode', 'barcode', 'description']]
```

	oid	dateScanned	brandCode	barcode	description
637	6000b2be0a7214ad4c00004d	2021-01-14 21:08:14	BEN AND JERRYS	076840100354	BEN & JERRYS FROZEN CHUNKY MONKEY ICE CREAM RE...
638	6000b2be0a7214ad4c00004d	2021-01-14 21:08:14	NaN	NaN	KLARBRUNN 12PK 12 FL OZ
639	6000b2be0a7214ad4c00004d	2021-01-14 21:08:14	KLEENEX	036000391718	KLEENEX POP UP RECTANGLE BOX FACIAL TISSUE 2 P...
640	6000b2be0a7214ad4c00004d	2021-01-14 21:08:14	BORDEN	NaN	BORDEN 2% MILK, 1/2 GAL
641	6000b2be0a7214ad4c00004d	2021-01-14 21:08:14	NaN	NaN	EMIL' S SAUSAGE MUSHROOM PIZZA
642	6000b2be0a7214ad4c00004d	2021-01-14 21:08:14	KLEENEX	036000391718	KLEENEX POP UP RECTANGLE BOX FACIAL TISSUE 2 P...
643	6000b2be0a7214ad4c00004d	2021-01-14 21:08:14	BEN AND JERRYS	311111511867	BEN AND JERRYS ICE CREAM
644	6000b2be0a7214ad4c00004d	2021-01-14 21:08:14	BEN AND JERRYS	076840100354	BEN & JERRYS FROZEN CHUNKY MONKEY ICE CREAM RE...
2809	6008f09d0a720f05fa000128	2021-01-21 03:10:21	BORDEN	815473013279	Borden 2% Reduced Fat Milk
2810	6008f09d0a720f05fa000128	2021-01-21 03:10:21	BEN AND JERRYS	076840100354	BEN & JERRYS FROZEN CHUNKY MONKEY ICE CREAM RE...
2811	6008f09d0a720f05fa000128	2021-01-21 03:10:21	NaN	NaN	KLARBRUNN 12PK 12 FL OZ
2812	6008f09d0a720f05fa000128	2021-01-21 03:10:21	NaN	NaN	EMIL' S SA.U.SAGE MUSHROOM PIZZA
2813	6008f09d0a720f05fa000128	2021-01-21 03:10:21	NaN	NaN	EMIL S SAUSAGE MUSHROOM PIZZA
2814	6008f09d0a720f05fa000128	2021-01-21 03:10:21	BORDEN	815473013279	Borden 2% Reduced Fat Milk

Multiple instances of the same **oid** in the **users_df**, possibly due to different **last_login_date** entries. However, upon verification, I found that the **createDate** and **loginDate** are identical for these records with the same **oid**, indicating this is likely a data entry issue

```python
#lets look at some of the ids which are not unique
users_df[users_df.oid.isin(['5ffcb4bc04929111f6e92608', '5fff4beedf9ace121f0c17ea', '6009e60450b331
```

	active	role	signUpSource	state	oid	createdDate	lastLogin	uui
129	True	consumer	Email	WI	5ffcb4bc04929111f6e92608	2021-01-11 20:27:40.225	2021-01-11 20:27:40.265	744302ce 4117-4ea 95b3 dbe00fa1ae8
136	True	consumer	Email	WI	5ffcb4bc04929111f6e92608	2021-01-11 20:27:40.225	2021-01-11 20:27:40.265	671d88a8 d215-46b4 80f3 6a1bcc6f449
142	True	consumer	Email	WI	5fff4beedf9ace121f0c17ea	2021-01-13 19:37:18.415	2021-01-13 19:41:12.676	4c916c1e 3335-4414 a717 f6f5832f827
147	True	consumer	Email	WI	5fff4beedf9ace121f0c17ea	2021-01-13 19:37:18.415	2021-01-13 19:41:12.676	e7b1c845 f094-41b6 9c31 959beb98e46
251	True	consumer	Email	WI	6009e60450b3311194385009	2021-01-21 20:37:25.244	NaT	d7322765 6fc9-45a6 8ac7 1fa805b7a5d
252	True	consumer	Email	WI	6009e60450b3311194385009	2021-01-21 20:37:25.244	NaT	e5783d40 78bc-4acb a012 d6e447e92aa

---