Hello, below I have copied and rewritten queries, analysis, or writing otherwise needed to complete all 5 steps of this exercise. I have not used GitHub before, so hopefully uploading two separate PDFs is sufficient for this exercise. If anything is needed in a different format or requires more detail, I am happy to redo any steps in this exercise. An assumption for the date was made, it is being calculated relative to the last date in the files rather than the real world date.

First step: visualization
All 3 drawn in lucid on a single canvas and saved as pdf, uploaded as a separate document.

Second step: answering 2 of the questions (5 and 6 chosen)
SQL query written in DuckDB

```
WITH receipts AS (
  SELECT
    *,
    (timestamp 'epoch' + dateScanned."$date" / 1000 * interval '1 second') AS scanned_ts,
    (timestamp 'epoch' + createDate."$date" / 1000 * interval '1 second') AS user_created_ts
  FROM receipts_data
  WHERE rewardsReceiptStatus = 'FINISHED'
),

max_user_created AS (
  SELECT MAX(user_created_ts) AS max_created_ts FROM receipts
),

recent_users_receipts AS (
  SELECT r.*
  FROM receipts r
  CROSS JOIN max_user_created m
  WHERE r.user_created_ts >= m.max_created_ts - INTERVAL '6 months'
),

items AS (
  SELECT
    r._id."$oid" AS receipt_id,
    i.brandCode,
    CAST(r.totalSpent AS DOUBLE) AS total_spent,
    r.userId
  FROM recent_users_receipts r
  CROSS JOIN UNNEST(r."rewardsReceiptItemList") AS t(i)
  WHERE i.brandCode IS NOT NULL
```

```
),

brand_spend AS (
  SELECT
    brandCode,
    SUM(total_spent) AS total_spend,
    COUNT(DISTINCT receipt_id) AS transaction_count
  FROM items
  GROUP BY brandCode
),

max_spend AS (
  SELECT brandCode, total_spend
  FROM brand_spend
  ORDER BY total_spend DESC
  LIMIT 1
),

max_transactions AS (
  SELECT brandCode, transaction_count
  FROM brand_spend
  ORDER BY transaction_count DESC
  LIMIT 1
)

SELECT
  'Most Spend' AS metric,
  brandCode,
  total_spend AS value
FROM max_spend

UNION ALL

SELECT
  'Most Transactions' AS metric,
  brandCode,
  transaction_count AS value
FROM max_transactions;
```

Output:

metric	brandCode	value
varchar	varchar	double

```
│ Most Spend        │ KROGER         │ 290213.43999999994 │
│ Most Transactions │ BEN AND JERRYS │               30.0 │
└───────────────────┴────────────────┴────────────────────┘
```

Third step: identifying errors
Query also written in DuckDB, a few comments added to show what I looked for more clearly

```
WITH
-- 1. Items in receipts with and without brandCode
items_brandCode AS (
  SELECT
    COUNT(*) AS total_items,
    COUNT(CASE WHEN item.brandCode IS NOT NULL THEN 1 END) AS
items_with_brandCode,
    COUNT(CASE WHEN item.brandCode IS NULL THEN 1 END) AS
items_missing_brandCode
  FROM receipts_data r
  CROSS JOIN UNNEST(r.rewardsReceiptItemList) AS t(item)
),

-- 2. Receipts with missing user
receipts_missing_user AS (
  SELECT COUNT(*) AS count_missing_user
  FROM receipts_data r
  LEFT JOIN users_data u ON r.userId = u._id."$oid"
  WHERE u._id."$oid" IS NULL
),

-- 3. Users with missing or invalid createdDate
users_createdDate AS (
  SELECT
    COUNT(*) AS total_users,
    COUNT(CASE WHEN createdDate."$date" IS NOT NULL THEN 1 END) AS
users_with_createdDate,
    COUNT(CASE WHEN createdDate."$date" IS NULL THEN 1 END) AS
users_missing_createdDate,
    COUNT(CASE WHEN createdDate."$date" < 0 THEN 1 END) AS
users_with_invalid_createdDate
  FROM users_data
),

-- 4. Brands missing brandCode and duplicates
brands_brandCode AS (
```

```
  SELECT
    COUNT(*) AS total_brands,
    COUNT(CASE WHEN brandCode IS NOT NULL THEN 1 END) AS brands_with_brandCode,
    COUNT(CASE WHEN brandCode IS NULL THEN 1 END) AS brands_missing_brandCode,
    COUNT(DISTINCT brandCode) AS distinct_brandCodes,
    COUNT(*) - COUNT(DISTINCT brandCode) AS duplicate_brandCodes
  FROM brands_data
),

-- 5. Receipt brandCodes unmatched in brands_data
unmatched_brands AS (
  SELECT
    COUNT(DISTINCT receipt_brands.brandCode) AS receipt_brandCodes,
    COUNT(DISTINCT b.brandCode) AS matching_brandCodes,
    COUNT(DISTINCT receipt_brands.brandCode) - COUNT(DISTINCT b.brandCode) AS
unmatched_brandCodes
  FROM (
    SELECT DISTINCT item.brandCode
    FROM receipts_data r
    CROSS JOIN UNNEST(r.rewardsReceiptItemList) AS t(item)
    WHERE item.brandCode IS NOT NULL
  ) receipt_brands
  LEFT JOIN brands_data b ON receipt_brands.brandCode = b.brandCode
),

-- 6a. Duplicate receipt IDs
dup_receipts AS (
  SELECT COUNT(*) AS dup_receipt_ids
  FROM (
    SELECT _id."$oid"
    FROM receipts_data
    GROUP BY _id."$oid"
    HAVING COUNT(*) > 1
  )
),

-- 6b. Duplicate user IDs
dup_users AS (
  SELECT COUNT(*) AS dup_user_ids
  FROM (
    SELECT _id."$oid"
    FROM users_data
    GROUP BY _id."$oid"
    HAVING COUNT(*) > 1
```

```
  )
),

-- 6c. Duplicate brand IDs
dup_brands AS (
  SELECT COUNT(*) AS dup_brand_ids
  FROM (
    SELECT _id."$oid"
    FROM brands_data
    GROUP BY _id."$oid"
    HAVING COUNT(*) > 1
  )
)

SELECT 'Total items in receipts' AS check_description, CAST(total_items AS VARCHAR) AS
value FROM items_brandCode
UNION ALL
SELECT 'Items with brandCode', CAST(items_with_brandCode AS VARCHAR) FROM
items_brandCode
UNION ALL
SELECT 'Items missing brandCode', CAST(items_missing_brandCode AS VARCHAR) FROM
items_brandCode
UNION ALL
SELECT 'Receipts with missing user', CAST(count_missing_user AS VARCHAR) FROM
receipts_missing_user
UNION ALL
SELECT 'Total users', CAST(total_users AS VARCHAR) FROM users_createdDate
UNION ALL
SELECT 'Users with createdDate', CAST(users_with_createdDate AS VARCHAR) FROM
users_createdDate
UNION ALL
SELECT 'Users missing createdDate', CAST(users_missing_createdDate AS VARCHAR)
FROM users_createdDate
UNION ALL
SELECT 'Users with invalid createdDate', CAST(users_with_invalid_createdDate AS
VARCHAR) FROM users_createdDate
UNION ALL
SELECT 'Total brands', CAST(total_brands AS VARCHAR) FROM brands_brandCode
UNION ALL
SELECT 'Brands with brandCode', CAST(brands_with_brandCode AS VARCHAR) FROM
brands_brandCode
UNION ALL
SELECT 'Brands missing brandCode', CAST(brands_missing_brandCode AS VARCHAR)
FROM brands_brandCode
```

```
UNION ALL
SELECT 'Distinct brandCodes', CAST(distinct_brandCodes AS VARCHAR) FROM
brands_brandCode
UNION ALL
SELECT 'Duplicate brandCodes', CAST(duplicate_brandCodes AS VARCHAR) FROM
brands_brandCode
UNION ALL
SELECT 'Receipt brandCodes total', CAST(receipt_brandCodes AS VARCHAR) FROM
unmatched_brands
UNION ALL
SELECT 'Receipt brandCodes matched in brands', CAST(matching_brandCodes AS
VARCHAR) FROM unmatched_brands
UNION ALL
SELECT 'Receipt brandCodes unmatched', CAST(unmatched_brandCodes AS VARCHAR)
FROM unmatched_brands
UNION ALL
SELECT 'Duplicate receipt IDs', CAST(dup_receipt_ids AS VARCHAR) FROM dup_receipts
UNION ALL
SELECT 'Duplicate user IDs', CAST(dup_user_ids AS VARCHAR) FROM dup_users
UNION ALL
SELECT 'Duplicate brand IDs', CAST(dup_brand_ids AS VARCHAR) FROM dup_brands;
```

Analysis on output

check_description	value
varchar	varchar
Total items in receipts	6941
Items with brandCode	2600
Items missing brandCode	4341
Receipts with missing user	148
Total users	495
Users with createdDate	495
Users missing createdDate	0
Users with invalid createdDate	0
Total brands	1167
Brands with brandCode	933
Brands missing brandCode	234
Distinct brandCodes	897
Duplicate brandCodes	270
Receipt brandCodes total	227
Receipt brandCodes matched in brands	41
Receipt brandCodes unmatched	186
Duplicate receipt IDs	0

Duplicate user IDs	70
Duplicate brand IDs	0

19 rows | 2 columns

Items in receipts:
Total: 6,941 items
Only 2,600 have a brandCode (about 37%) - most items (4,341) are missing brand info, which may be a significant data gap.

Receipts and Users:
148 receipts have no matching user - investigate missing or orphaned receipt data.
Total users: 495, and all have a valid createdDate.
70 users have duplicate IDs - could cause analysis issues.

Brands:
1,167 total brand records, but 234 missing brandCode.
270 duplicate brandCode values - this might mean inconsistencies in brand naming or multiple records for the same brand.
227 distinct brandCode values appearing in receipts, but only 41 of those appear in the brands table - most receipt brandCodes are unmatched in your brand master list (186 unmatched). This suggests a serious mismatch problem or incomplete brand master data.

Duplicates:
No duplicate receipt or brand IDs.
Duplicate user IDs are high (70), which needs cleanup or investigation.

Step 5: sample email on questions and findings

Hi [Recipient Name],

I wanted to share some thoughts and questions I have after working with the receipt, user, and brand data.

Regarding the data itself, I'm curious about how product and brand mappings are maintained and updated. For example, I've noticed some receipt items flagged as "ITEM NOT FOUND" or with user-flagged barcodes and prices, which suggest gaps or inconsistencies in our product catalog. Additionally, brand codes are sometimes missing or inconsistent, making it challenging to accurately attribute spend and transactions to the correct brands. It would be helpful to understand what validation processes exist during ingestion and how often these mappings are reviewed or corrected.

I discovered data quality issues through direct examination of sample records where certain receipt items were marked as "ITEM NOT FOUND" or had user-flagged barcodes and prices.

Additionally, there are inconsistencies and missing values in brand codes, which make it challenging to confidently attribute spend and transactions to the correct brands. Such inconsistencies can cascade into inaccurate reporting and analytics, so understanding how exceptions are handled—whether through manual review, automated enrichment, or fallback logic—is critical to improving data fidelity.

To resolve these issues, I need to better understand the ingestion and validation workflows. Information on how exceptions are handled, what manual reviews or automated corrections exist, and how product-to-brand relationships are maintained would be invaluable. This will help us target the root causes and improve overall data reliability.

Beyond data quality, to optimize the data assets we're building, it would help to have insight into query patterns, data growth expectations, and any retention policies. Knowing typical user queries and business priorities will allow us to design more efficient data structures, such as summary tables or partitions, tailored to our business usage.

On performance and scalability, I anticipate challenges related to joining large volumes of nested receipt items with user and brand data, especially as new data accumulates. To address this, strategies like time-based partitioning, incremental aggregation, and caching will be important. Also, understanding infrastructure limits and performance SLAs will guide how aggressively we optimize. If existing monitoring tools or metrics on query performance and system health are available, access to those would help us proactively identify bottlenecks and tune accordingly.

Please let me know when we can connect to discuss these points further or if additional documentation is available.

Best regards,
Jessica Thomas