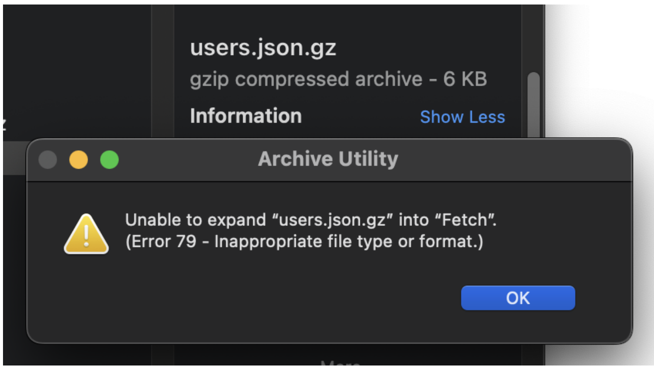
02. Write a query that directly answers a predetermined question from a business stakeholder

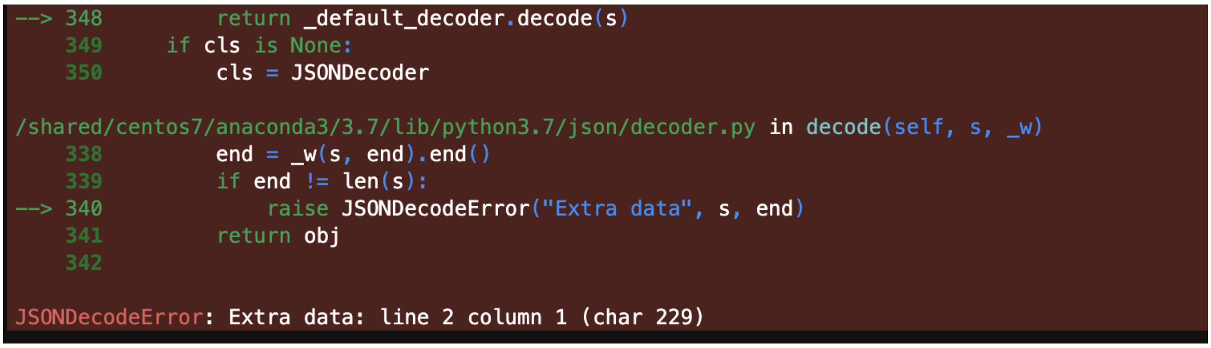
I tried to work on PostgreSQL by inserting the json files, and by converting it to csv, but encountered the following JSON and file errors, the screenshot of which I've also uploaded to the repository as "Error Screenshot 01" & "Error Screenshot 02".

users.json



Unable to expand “users.json.gz” into “Fetch”.
(Error 79 - Inappropriate file type or format.)

brands.json

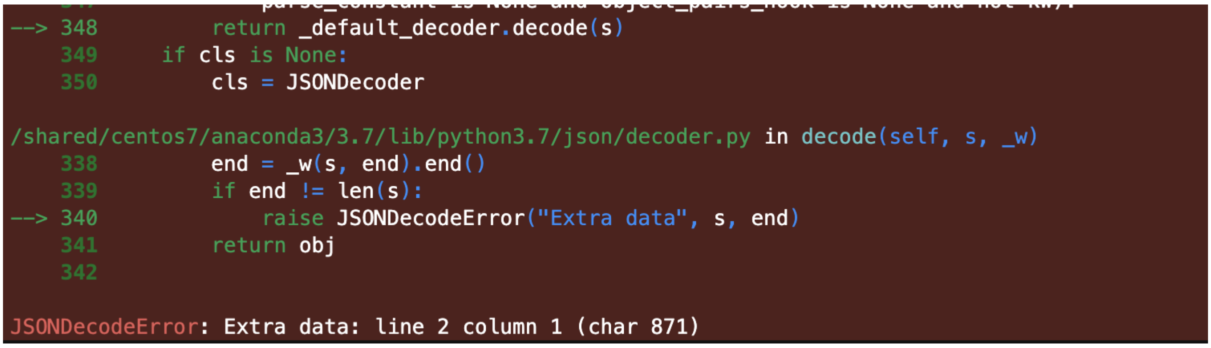


```
--> 348         return _default_decoder.decode(s)
    349     if cls is None:
    350         cls = JSONDecoder

/shared/centos7/anaconda3/3.7/lib/python3.7/json/decoder.py in decode(self, s, _w)
    338         end = _w(s, end).end()
    339         if end != len(s):
--> 340             raise JSONDecodeError("Extra data", s, end)
    341         return obj
    342 

JSONDecodeError: Extra data: line 2 column 1 (char 229)
```

receipts.json



```
    347             parse_constant is None and object_pairs_hook is None and not kw):
--> 348         return _default_decoder.decode(s)
    349     if cls is None:
    350         cls = JSONDecoder

/shared/centos7/anaconda3/3.7/lib/python3.7/json/decoder.py in decode(self, s, _w)
    338         end = _w(s, end).end()
    339         if end != len(s):
--> 340             raise JSONDecodeError("Extra data", s, end)
    341         return obj
    342 

JSONDecodeError: Extra data: line 2 column 1 (char 871)
```

I’ve thus answered all the mentioned queries theoretically based on the previously created normalized structured starflake relational data model in question 01.

I’ve kept the queries simple and concise, and made it as performant as possible based on the existing requirements which I had, with basic assumptions.All the used variables & entities are consistent with the normalized structured starflake relational data model created.

.

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

```
SELECT b.name AS brand_name, COUNT(*) AS scanned_count
FROM Brand b
JOIN Receipt r ON b.brand_id = r.brand_id
WHERE DATE_TRUNC('month', r.createDate) = DATE_TRUNC('month',
CURRENT_DATE)
GROUP BY b.name
ORDER BY scanned_count DESC
LIMIT 5;
```

2. 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 b.name AS brand_name, COUNT(*) AS scanned_count
   FROM Brand b
   JOIN Receipt r ON b.brand_id = r.brand_id
   WHERE DATE_TRUNC('month', r.createDate) = DATE_TRUNC('month',
CURRENT_DATE)
   GROUP BY b.name
   ORDER BY scanned_count DESC
   LIMIT 5
), previous_month AS (
   SELECT b.name AS brand_name, COUNT(*) AS scanned_count
   FROM Brand b
   JOIN Receipt r ON b.brand_id = r.brand_id
   WHERE DATE_TRUNC('month', r.createDate) = DATE_TRUNC('month',
CURRENT_DATE - INTERVAL '1 month')
   GROUP BY b.name
   ORDER BY scanned_count DESC
   LIMIT 5
)
SELECT recent.brand_name, recent.scanned_count, previous.scanned_count AS
previous_month_count
FROM recent_month AS recent
LEFT JOIN previous_month AS previous ON recent.brand_name = previous.brand_name;
```

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

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

```
FROM Receipt
WHERE rewardsReceiptStatus IN ('Accepted', 'Rejected')
GROUP BY rewardsReceiptStatus
ORDER BY average_spend DESC
LIMIT 1;
```

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

```
SELECT rewardsReceiptStatus, SUM(purchasedItemCount) AS total_items_purchased
FROM Receipt
WHERE rewardsReceiptStatus IN ('Accepted', 'Rejected')
GROUP BY rewardsReceiptStatus
ORDER BY total_items_purchased DESC
LIMIT 1;
```

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

```
SELECT b.name AS brand_name, SUM(r.totalSpent) AS total_spend
FROM Brand b
JOIN Receipt r ON b.brand_id = r.brand_id
JOIN Users u ON r.userId = u._id
WHERE u.createdDate >= CURRENT_DATE - INTERVAL '6 months'
GROUP BY b.name
ORDER BY total_spend DESC
LIMIT 1;
```

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

```
SELECT b.name AS brand_name, COUNT(*) AS transaction_count
FROM Brand b
JOIN Receipt r ON b.brand_id = r.brand_id
JOIN Users u ON r.userId = u._id
WHERE u.createdDate >= CURRENT_DATE - INTERVAL '6 months'
GROUP BY b.name
ORDER BY transaction_count DESC
LIMIT 1;
```