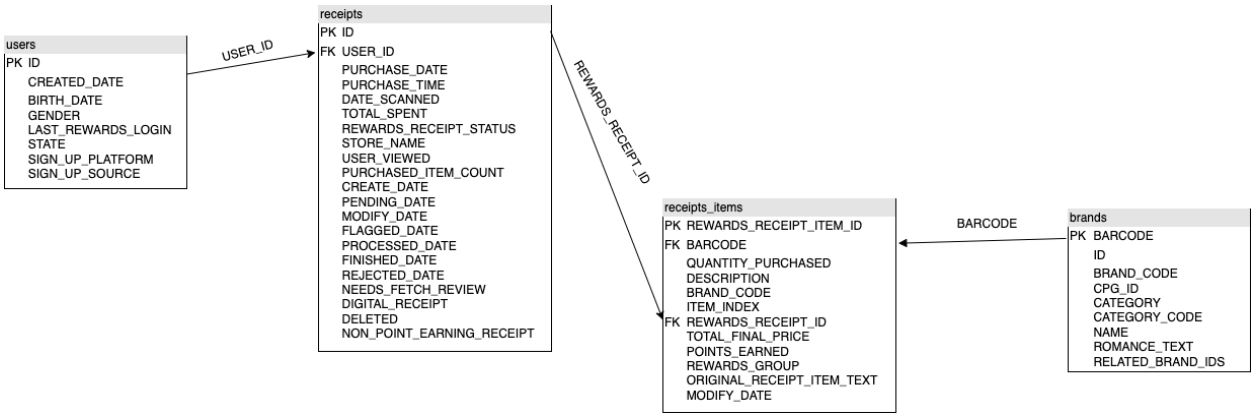
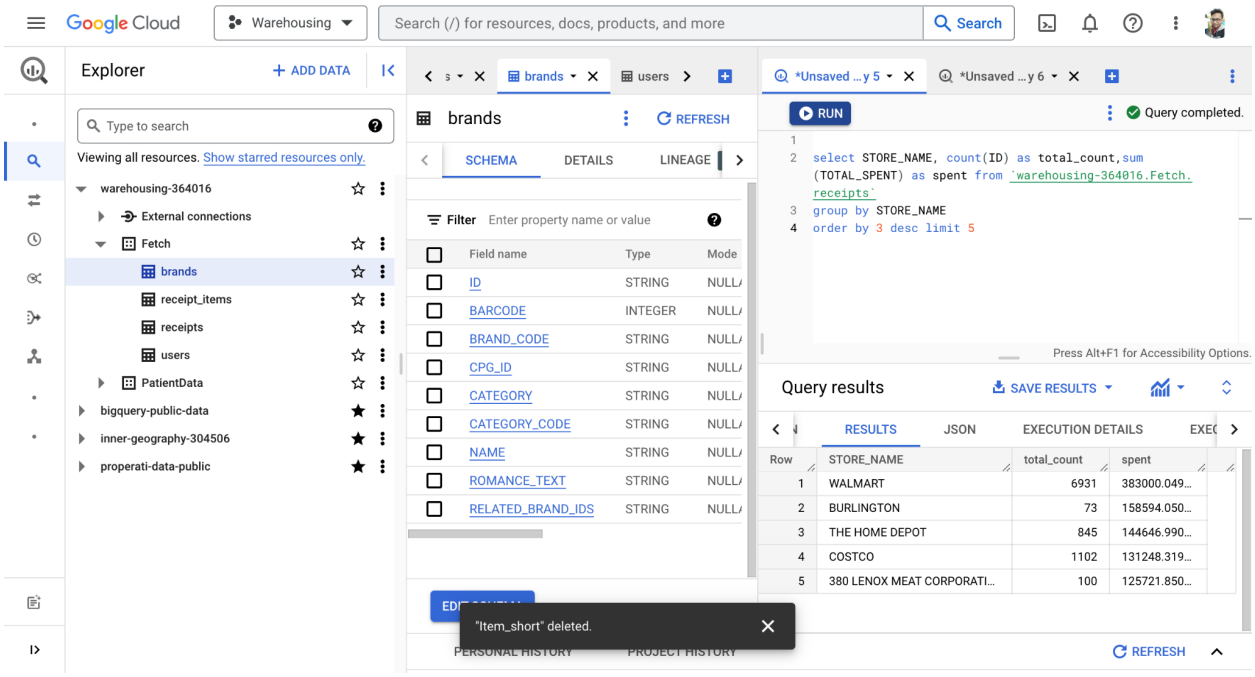
First: Review Existing Data and Diagram a New Structured Relational Data Model

I have created a UML to create a relational model for the given data



Design Considerations:

- I have linked receipts table with users table using USER_ID as foreign key.
- I have linked receipt_items table to receipt table using REWARD_RECEIPT_ID as foreign key.
- For brands table, I have selected BARCODE as primary key because its unique and not null.
- I have linked receipt_items table to brands table using BARCODE as a foreign key
- I have designed a data warehouse in Google Bigquery to test queries and optimize query performance.
- I have set table expiration to 10 days to avoid any additional Bigquery billing charges



Snapshot of Bigquery console hosting the data warehouse

Second: Write a query that directly answers question(s) from a business stakeholder

Which brand saw the most dollars spent in the month of June?

Query:

```
select BRAND_CODE,SUM(TOTAL_FINAL_PRICE) AS TOTAL_SPENT from
`warehousing-364016.Fetch.receipt_items`
where extract(year from MODIFY_DATE) = 2022 and extract(month from MODIFY_DATE) = 6
GROUP BY 1 ORDER BY 2 DESC
LIMIT 5
```

Results:

Row	BRAND_CODE	TOTAL_SPENT
1	*null*	112145.740...
2	KIRKLAND SIGNATURE	1822.17000...
3	GREAT VALUE	1185.48999...
4	ANDERSEN	706.0
5	CARDELL	556.98

Since brandcode is null for first observation, we can consider KIRKLAND SIGNATURE as the brand that saw most dollar spent in June 2022 for this assignment

Comments:

- In this query, I found the brand that saw the most dollar spent in June 2022.
- I have used MODIFY_DATE in receipt_items instead of PURCHASE_DATE in receipts in order to avoid joins and increase query performance. As per my analysis, these dates are the same in both tables.
- BRANDCODE is NULL for brand with most spend

Which user spent the most money in the month of August?

Query:

```
SELECT USER_ID,sum(TOTAL_SPENT) as total FROM `warehousing-364016.Fetch.receipts`
where extract(year from PURCHASE_DATE) = 2022 and extract(month from PURCHASE_DATE) =
8
group by 1 order by 2 desc
limit 5
```

Results:

Row	USER_ID	total
1	5ffb49a847903912705e9a64	12742.6899...
2	61757c3da9619d4881912d84	7427.65999...
3	6134c726bb1615636197f943	5135.44999...
4	6115880fa009af1799ef9104	4807.57999...
5	60047e8a2d7db612a69d2e18	4392.51999...

User, 5ffb49a847903912705e9a64 spent most in August 2022.

Comments:

- In this query, I found the users with the most dollar spent in August 2022.
- We were not provided with user names in this database

What user bought the most expensive item?

Query:

```
WITH cte AS
(select REWARDS_RECEIPT_ITEM_ID,REWARDS_RECEIPT_ID, DESCRIPTION,
(TOTAL_FINAL_PRICE/QUANTITY_PURCHASED) AS ITEM_PRICE
from `warehousing-364016.Fetch.receipt_items`
WHERE QUANTITY_PURCHASED IS NOT NULL AND QUANTITY_PURCHASED NOT IN (-1,0))

select USER_ID from `warehousing-364016.Fetch.receipts` where ID = (select
REWARDS_RECEIPT_ID from cte order by ITEM_PRICE desc limit 1)
```

Results:

Row	USER_ID
1	617376b8a9619d488190e0b6

User, 617376b8a9619d488190e0b6 purchased the most expensive item.

Comments:

- We were not provided with user names in this database thus only the USER_ID.
- I didn't join the receipts and receipts_item table to increase query performance. CTE resulted in better execution time in Bigquery.
- There were some errors in the values of QUANTITY_PURCHASED column they have values like NULL, -1,

and 0.
- To find the price of individual items, I divided the TOTAL_FINAL_PRICE by QUANTITY_PURCHASED and ordered it descendingly to find the item with the highest price.

What user bought the most expensive item?

Query:

```
WITH cte AS
(select REWARDS_RECEIPT_ITEM_ID,REWARDS_RECEIPT_ID, DESCRIPTION,
(TOTAL_FINAL_PRICE/QUANTITY_PURCHASED) AS ITEM_PRICE
from `warehousing-364016.Fetch.receipt_items`
WHERE QUANTITY_PURCHASED IS NOT NULL AND QUANTITY_PURCHASED NOT IN (-1,0))

select USER_ID from `warehousing-364016.Fetch.receipts` where ID = (select
REWARDS_RECEIPT_ID from cte order by ITEM_PRICE desc limit 1)
```

Results:

Row	USER_ID
1	617376b8a9619d488190e0b6

608749aac63a95130a45fbf4

User, 617376b8a9619d488190e0b6 purchased the most expensive item.

Comments:

- We were not provided with user names in this database thus only the USER_ID.
- I didn't join the receipts and receipts_item table to increase query performance. CTE resulted in better execution time in Bigquery.
- There were some errors in the values of QUANTITY_PURCHASED column they have values like NULL, -1, and 0.
- To find the price of individual items, I divided the TOTAL_FINAL_PRICE by QUANTITY_PURCHASED and ordered it descendingly to find the item with the highest price.

What is the name of the most expensive item purchased?

Query:

```
select REWARDS_RECEIPT_ITEM_ID,BRAND_CODE,BARCODE, DESCRIPTION,
(TOTAL_FINAL_PRICE/QUANTITY_PURCHASED) AS ITEM_PRICE
from `warehousing-364016.Fetch.receipt_items`
WHERE QUANTITY_PURCHASED IS NOT NULL AND QUANTITY_PURCHASED NOT IN (-1,0)
order by 5 desc limit 5
```

Results:

Row	REWARDS_RECEIPT_ITEM_ID	BRAND_CODE	BARCODE	DESCRIPTION	ITEM_PRICE
1	1efd6d7c75ecbae32214acb6cda41d12	*null*	*null*	RLGULAR SALE	31005.99
2	79482a8fa3bd0eef3d626f1c862042e8	*null*	240292012	82 GOURMET HOUSEW	31005.99
3	b26669cf4ce90cc9d7d3b0ab588cb04b	*null*	*null*	GOLDILOCKS NOPIA R BLAGK	31003.84
4	b4fafd04d8274a1e95b97155edaade2f	*null*	*null*	KURI-IRI DORAYAKI CAKE	31003.0
5	39694b0880b511e8a12bfb76cf2c20f3	*null*	*null*	YIZMANG FISH BALL	31001.5

For the most expensive item on the list, BRANDCODE and BARCODE is null. Thus obtaining the brandname from brands table is not possible in this case. Thus, I used description to identify the item.
Most expensive item: 82 GOURMET HOUSEW, since the description of first item is inconclusive
Comments:

- For second item in the list, no similar barcode or brandcode exists in brands table.
- Length of barcode is inconsistent.
- Data needs to be cleaned before analysis which is outside the scope of this assignment.

How many users scanned in each month?

Query:

```
select extract(year from date_scanned) as year, extract(month from date_scanned) as
month, count(USER_ID) as user_count from `warehousing-364016.Fetch.receipts`
group by 1,2
order by 1 desc,2 desc
```

Results:

Row	year	month	user_count
1	2023	1	535
2	2022	12	4552
3	2022	11	4013
4	2022	10	3855
5	2022	9	3673
6	2022	8	3755
7	2022	7	3913
8	2022	6	3581
9	2022	5	3789
10	2022	4	3654
11	2022	3	3819
12	2022	2	3305
13	2022	1	3385
14	2021	12	3895

Comments:

- For second item in the list, no similar barcode or brandcode exists in brands table.
- Length of barcode is inconsistent.
- Data needs to be cleaned before analysis which is outside the scope of this assignment.

Third: Choose something noteworthy about the data and share with a non-technical stakeholder

Data Governance:

- Data governance should be applied strictly. Data is highly inconsitent
- Formatting of date in PURCHASE_TIME should be done correctly to support querying operations
- There are a lot of null values that makes data analysis difficult.
- Erros in accounting found. For example, in some cases the TOTAL_FINAL_PRICE of individual items is higher than the TOTAL_SPENT in receipt
- In the receipts_items table, QUANTITY_PURCHASED is negative in some observations
- BARCODE formatting is inconsistent in the rewards_receipt table

Data Analysis Inference:

- The number of users scan each month peaks around November and December during the holiday season. Stores can maintain extra inventory and host sales for maximum profit during the holiday season.
- The highest total spend is from users that signed up via Facebook. Thus, brands can increase their marketing activities in Facebook to draw more user spend via the platform.

Row	SIGN_UP_SOURCE	spent
1	Facebook	1131176.47…
2	Email	889313.669…
3	Apple	668355.679…
4	Google	633474.449…

- Majority of sales come from non-point earning receipts. Thus, brands need to rework oin their reward strategies to lure customers to opt for rewards.

Row	NON_P	total_count	spent
1	true	22490	914584.949…
2	*null*	8986	350210.519…
3	fal…	39125	2057524.80…

- Walmart generated the highest sales.

Row	STORE_NAME	total_count	spent
1	WALMART	6931	383000.049…
2	BURLINGTON	73	158594.050…
3	THE HOME DEPOT	845	144646.990…
4	COSTCO	1102	131248.319…
5	380 LENOX MEAT CORPORATI…	100	125721.850…

Warehouse Design and Optimizations:

- Tables should be denormalized to avoid joins and improve query performance for analytical purpose
- Since, the data size is large, tables receipt and receipt_item can be partitioned by date to improve query performance and reduce cost of running queries.
- Before loading data to warehouse like Bigquery, data should be cleaned to improve performance and business intelligence. We need to enforce strict data governance to avoid data inconsistencies