Fetch Rewards Analytical Engineer Coding Exercise

Summary:

There are 3 JSON compressed files in this challenge, which need to store in a data warehouse. Some of the JSON files are nested and need to flatten to extract data and load it into the data model. The data model of the Datawarehouse needs to be structured, relational, and simple. The data warehouse needs to be queried for the analytical questions asked as per the challenge. Data Quality issues should also be pointed and addressed.

Understanding of the use case:

Fetch reward is an app that rewards its users when they upload the receipt of purchases made in different stores. Fetch reward uses the OCR application to convert the image into text and output the text into a JSON object from an application API containing various information on the receipt. Also, the users uploading the receipt information and the participating brands' information are provided.

Given JSON

User

Users	Description
_id	user Id
state	state abbreviation
createdDate	when the user created their account
lastLogin	last time the user was recorded logging in to the app
role	constant value set to 'CONSUMER'
active	indicates if the user is active; only Fetch will de-activate an account with this flag

Brands

Brands	Description
_id	brand uuid
barcode	the barcode on the item
brandCode	String that corresponds with the brand column in a partner product file
category	The category name for which the brand sells products in
categoryCode	The category code that references a BrandCategory
cpg	reference to CPG collection
topBrand	Boolean indicator for whether the brand should be featured as a 'top brand'
Name	Brand name

Receipts

Receipts	Description
_id	uuid for this receipt
bonusPointsEarned	Number of bonus points that were awarded upon receipt completion
bonusPointsEarnedReason	event that triggered bonus points
createDate	The date that the event was created
dateScanned	Date that the user scanned their receipt
finishedDate	Date that the receipt finished processing
modifyDate	The date the event was modified
pointsAwardedDate	The date we awarded points for the transaction
pointsEarned	The number of points earned for the receipt
purchaseDate	the date of the purchase
purchasedItemCount	Count of number of items on the receipt
rewardsReceiptItemList	The items that were purchased on the receipt
rewardsReceiptStatus	status of the receipt through receipt validation and processing
totalSpent	The total amount on the receipt
userId	string id back to the User collection for the user who scanned the receipt

rewardsReceiptItemList

Another JSON expanded. new fields, no description

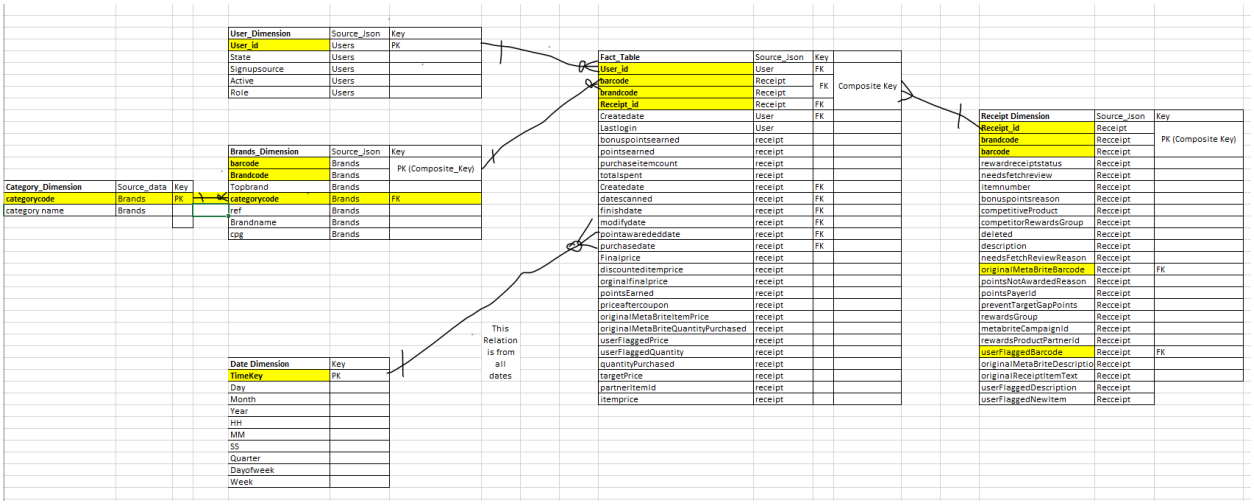
rewardsReceiptItemList	Description
barcode	
brandCode	
competitiveProduct	
competitorRewardsGroup	
deleted	
description	
discountedItemPrice	
finalPrice	
itemNumber	
itemPrice	
metabriteCampaignId	
needsFetchReview	
needsFetchReviewReason	
originalFinalPrice	
originalMetaBriteBarcode	
originalMetaBriteDescription	
originalMetaBriteItemPrice	
originalMetaBriteQuantityPurchased	
originalReceiptItemText	

partnerItemId	
pointsEarned	
pointsNotAwardedReason	
pointsPayerId	
preventTargetGapPoints	
priceAfterCoupon	
quantityPurchased	
rewardsGroup	
rewardsProductPartnerId	
targetPrice	
userFlaggedBarcode	
userFlaggedDescription	
userFlaggedNewItem	
userFlaggedPrice	
userFlaggedQuantity	

Solution:

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

Data Model ROLAP



The above is the dimensional modeling that I developed that can store the data in Datawarehouse. The Fact table contains different numeric fields, dates involved and id's that can be connected to dimensions.

Fact table components –

The data in this table are at receipts item-level granularity.

Id's –

User_id -> is connected to user dimension.

Brandcode + Barcode -> is connected to brands dimension.

Brandcode + Barcode + receipt_id a composite can be connected to receipt dimension.

Receipt_Id not required but may solve some business queries based on receipt id

Dates – Various dates related to receipt and user are connected to date dimension. This can quickly look up the epoch date to a Georgian calendar date that can solve some business questions.

Numeric values – These are many other numerical values in the fact table relating to the receipt and item level numerical values like prices and quantity.

Dimension Tables –

User dimension – unique Users, is identified by user_id a unique key

Receipt dimension – individual records in this table are identified by receipt_id, barcode and brandcode.

Brands dimension – individual records can be identified by barcode and brandcode.

Date dimension – individual dates can be identified by date key

Second: Write a query that directly answers a predetermined question from a business stakeholder

Written in MSSQL

- **What are the top 5 brands by receipts scanned for most recent month?**

  #assumption – top brands are top used brands in receipt itemlist

  ```
  With cte as(

  Select Fact_table.barcode, Fact_table.brandcode

   from Fact_table

  where Fact_table.datescanned in (select Date_dimension .Timekey

   From Date_dimension where Date_dimension.month = month(getdate())-1 and Date_dimension .year=year(getdate())  )
  ```

)

Select brands_dimension.brandname, count(cte.barcode)

From cte join brands_dimension

on brands_dimension.barcode= cte. barcode, brands_dimension.brandcode= cte. brandcode groupby cte.brandname

order by count(cte. barcode) desc limit 5

- **How does the ranking of the top 5 brands by receipts scanned for the recent month compare to the ranking for the previous month?**

to store recent month items sale

With rec_cte as(

Select Fact_table.barcode, Fact_table.brandcode

from Fact_table

where Fact_table.datescanned in (select Date_dimension .Timekey

From Date_dimension where Date_dimension.month = month(getdate())-1 and Date_dimension .year=year(getdate()))

)

to store previous month items sale

With prev_cte as(

Select Fact_table.barcode, Fact_table.brandcode

from Fact_table

where Fact_table.datescanned in (select Date_dimension .Timekey

From Date_dimension where Date_dimension.month = month(getdate())-2 and Date_dimension .year=year(getdate()))

```
)

# joining brands and count of their brand name for both previous and recent months

With rec_cte_2 as(

Select rec_cte.brandname, count(rec_cte.barcode) as brand_count

From rec_cte join brands_dimension

on brands_dimension .barcode= rec_cte. Barcode and

brands_dimension .brandcode= rec_cte.brandcode

groupby rec_cte.brandname order by count(rec_cte. barcode) desc limit 5)

With prev_cte_2 as(

Select prev_cte.brandname, count(prev_cte. barcode) as brand_count

From prev_cte join brands_dimension

on brands_dimension .barcode= prev_cte. barcode,
brands_dimension.brandcode= prev_cte.brandcode

 groupby prev_cte.brandname order by count(prev_cte. barcode) desc limit 5)

# ranking the result based on count of usage

With rec_cte_3 as(

Select *, rank() over(partition by rec_cte_2.brandname order by
rec_cte_2.brand_count desc) as brandrank from rec_cte_2 )

With prev_cte_3 as(

Select *, rank() over(partition by prev_cte_2.brandname order by
prev_cte_2.brand_count desc) as brandrank from prev_cte_2)

# joining 2 cte (results based on count) on rank to show them side by side
```

Select rec_cte_3.brandrank, rec_cte_3. brandname as recent_brand, prev_cte_3.brand.brandname as previous_brand

from rec_cte_3 join prev_cte_3

on rec_cte_3.Brandrank= prev_cte_3.brandrank

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

  # storing cte with the barcode and brandcode based on status accepted/rejected

  With cte as(

  Select receipt_dimension.brandcode, receipt_dimension .barcode, receipt_dimension.receipt_id, receipt_dimension.rewardreceiptstatus

  from receipt_dimension

  where receipt_dimension.rewardreceiptstatus=’Accepted' or receipt_dimension .rewardreceiptstatus=’Rejected'

)

  #joining fact table and cte results

  Select cte.Rewardreceiptstatus, avg(fact_table.totalspent) as average_spend

  From fact_table join cte

  on fact_table.barcode=cte.barcode and fact_table.brandcode=cte.brandcode and fact_table. receipt_id = cte. receipt_id order by avg(fact_table.totalspent) desc limit 1

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

  ## storing cte with the barcode and brandcode based on status accepted/rejected

With cte as(
Select receipt_dimension.brandcode, receipt_dimension.barcode,
receipt_dimension. receipt_id, receipt_dimension.rewardreceiptstatus
from receipt_dimension where
receipt_dimension.rewardreceiptstatus=’Accepted' or receipt_dimension
.rewardreceiptstatus=’Rejected'
)
##joining fact table and cte results
Select cte.rewardreceiptstatus,count(fact_table.barcode) as
total_number_of_items_purchased
From fact_table
join cte on fact_table.barcode = cte.barcode and fact_table.brandcode =
cte.brandcode and fact_table. receipt_id = cte. receipt_id order by
count(fact_table.barcode) desc limit 1

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

With cte as(

Select Fact_table.barcode, Fact_table.brandcode, sum(Fact_table.finalprice) as brandspend as from Fact_table where datescanned in (select Timekey from Date_dimension where month in (month(getdate())-1, month(getdate())-2,month(getdate())-3,month(getdate())-4,month(getdate())-5,month(getdate())-6) and year=year(getdate()))

group by Fact_table.barcode, Fact_table.brandcode order by sum(Fact_table .finalprice) desc limit 1)

Select brands_dimension.name From cte join brands_dimension on cte.barcode= brands_dimension.barcode and cte.brandcode= brands_dimension.barcode

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

With cte as(

Select Fact_table.barcode, Fact_table.brandcode, count(Fact_table.barcode) as brandspend as from Fact_table where datescanned in (select Timekey from Date_dimension where month in (month(getdate())-1, month(getdate())-2, month(getdate())-3, month(getdate())-4, month(getdate())-5, month(getdate())-6) and year=year(getdate())) group by Fact_table.barcode, Fact_table.brandcode order by Fact_table.count(Fact_table.barcode) desc limit 1)

Select brands_dimension.name From cte join brands_dimension on cte.barcode= brands_dimension.barcode and cte.brandcode= brands_dimension.barcode

These queries can be better written and evaluated if we load data into a real time OLAP database and execute with data.

Third: Evaluate Data Quality Issues in the Data Provided

Using the programming language of your choice (SQL, Python, R, Bash, etc...) identify at least one data quality issue. We are not expecting a full blown review of all the data provided, but instead want to know how you explore and evaluate data of questionable provenance.

There are two major data quality issues, I found and needs to mentioned.

1) **Duplicate records in User accounts**.
   There are many duplicate user records in users table. These needs to be removed before loading into data warehouse based on the recent record. The duplication may be due to multiple submissions to create an account through the application.

2) **Inconsistent Barcode and Brandcode values**.
   The main joining link between receipt and brands are barcode and brandcodes. In receipts these can be derived from expanding Receipt item list which has list of items purchased in a receipt. In general, an item will have one and unique barcode and are produced by a brand. But there are instances where the same barcode has different brandcodes and this will lead to inconsistency in join. However, during preprocessing these has been captured by try exept block clause in python.
   (Check Jupyter notebook in the Github Link for preprocessing)

Fourth: Communicate with Stakeholders

Construct an email or slack message that is understandable to a product or business leader who isn't familiar with your day to day work. This part of the exercise should show off how you communicate and reason about data with others. Commit your answers to the git repository along with the rest of your exercise.

- **What questions do you have about the data?**
- **How did you discover the data quality issues?**
- **What do you need to know to resolve the data quality issues?**
- **What other information would you need to help you optimize the data assets you're trying to create?**
- **What performance and scaling concerns do you anticipate in production and how do you plan to address them?**

Hello Dave,

I hope you're doing well.!!

I recently went through your requirement documents to create a data warehouse model that would best answer the analytical question related to business decision-making. I enjoyed looking through the problem; it was a complex yet exciting problem.

I understood that three files are to be loaded into the warehouse, and these three files would be interrelated. I have a couple of queries and suggestions regarding the problem. I found duplicate records (~280) in the user file, indicating there might be some problem with the application, which prompted users to submit multiple responses during account creation. Also, the users have both consumers and fetch staff records, contradicting the data warehouse design to limit information only to consumers; this might led to discarding some valuable data.

Also, while joining receipts (containing items with barcode and brand codes) and brands with the same fields, I found that the brand code has many null entries and duplicate entries. These would lead to inaccurate joins with multiple records. However, this has been avoided for now, with first joining non-duplicated barcodes and using brandcodes

as a next level of joining with duplicated barcodes. But, any null value for the barcode column could lead to inconsistency and potential loss. Hence application needs to be revised to capture these.

I have not found any description regarding the fields in the reciptitemlist, which hindered progress in further optimizing the tables. This would also help me further optimize the data warehouse prototype model provided by normalization (reducing redundant data).

When it comes to deploying the data model into a database, two particular databases come into my mind, i.e., Teradata and Redshift. Both of them are highly scalable, relational and are better suited for Analytical processing databases. Further, I suggest Redshift because of its peculiar columnar storage mechanism, which increases the processing speeds and available on AWS cloud - reducing efforts to procure hardware when scaled.

I hope I explained to you in concise significant challenges and finding. I would like to discuss my findings over a call further. Kindly let me know when you are free over the week.

Thanks & Regards

Sai Suraj Argula