Retail Industry Customer Behavior Database Management System

Author: Yu Qiu

Northeastern University

Date: 12/12/2021

Introduction

My project is to help retail companies manage their customers' information of their profile, purchase history, and campaign activities on different channels for customer analysis.

- **Better Understand Customers**
- **Generating Sales Reports**
- **Measure the Performance of Campaigns**

User Profile

Product Team

Product team could edit and retrieve any data to analyze user behaviors and optimize the product and services according to the analysis; tag users for future campaigns and provide references for the sales team;

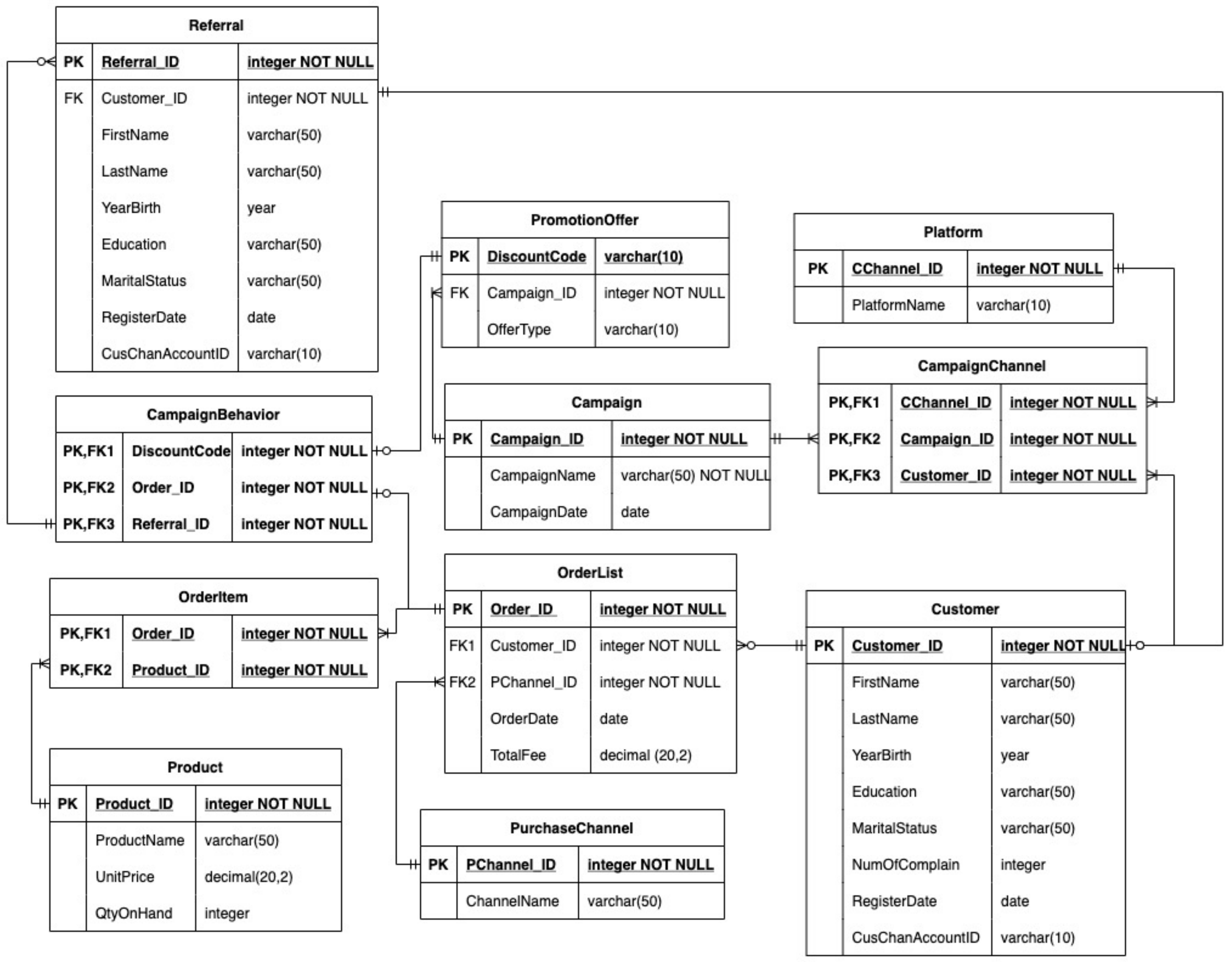
Marketing Team

Marketing team could retrieve part of the data to measure campaigns' performances and report to leadership team, also tailor-made new campaigns;

Sales and Leadership Team

Sales team and leadership team could send requests to product or marketing team for any specific needs and view the data retrieved.

ERD Design



Referral

PK	Referral_ID	integer NOT NULL
FK	Customer_ID	integer NOT NULL
	FirstName	varchar(50)
	LastName	varchar(50)
	YearBirth	year
	Education	varchar(50)
	MaritalStatus	varchar(50)
	RegisterDate	date
	CusChanAccountID	varchar(10)

PromotionOffer

PK	DiscountCode	varchar(10)
FK	Campaign_ID	integer NOT NULL
	OfferType	varchar(10)

Platform

PK	CChannel_ID	integer NOT NULL
	PlatformName	varchar(10)

CampaignBehavior

PK,FK1	DiscountCode	integer NOT NULL
PK,FK2	Order_ID	integer NOT NULL
PK,FK3	Referral_ID	integer NOT NULL

Campaign

PK	Campaign_ID	integer NOT NULL
	CampaignName	varchar(50) NOT NULL
	CampaignDate	date

CampaignChannel

PK,FK1	CChannel_ID	integer NOT NULL
PK,FK2	Campaign_ID	integer NOT NULL
PK,FK3	Customer_ID	integer NOT NULL

OrderItem

PK,FK1	Order_ID	integer NOT NULL
PK,FK2	Product_ID	integer NOT NULL

OrderList

PK	Order_ID	integer NOT NULL
FK1	Customer_ID	integer NOT NULL
FK2	PChannel_ID	integer NOT NULL
	OrderDate	date
	TotalFee	decimal (20,2)

Customer

PK	Customer_ID	integer NOT NULL
	FirstName	varchar(50)
	LastName	varchar(50)
	YearBirth	year
	Education	varchar(50)
	MaritalStatus	varchar(50)
	NumOfComplain	integer
	RegisterDate	date
	CusChanAccountID	varchar(10)

Product

PK	Product_ID	integer NOT NULL
	ProductName	varchar(50)
	UnitPrice	decimal(20,2)
	QtyOnHand	integer

PurchaseChannel

PK	PChannel_ID	integer NOT NULL
	ChannelName	varchar(50)

Business Statements & Queries

Business Query Statement 1:

The **product team** wants to know the satisfactory rate of the married customers who registered between 2012-06-01 and 2013-06-01.
Generate a list of customers registered during the period with the marital status married. Display the result in the order of the birth year. The columns should include Customer ID, Full Name, Year Birth, Marital Status, Number of Complain, and Register Date.

```sql
SELECT Customer_ID AS CustomerID, LastName || ", "|| FirstName AS FullName,
YearBirth, MaritalStatus, NumOfComplain AS NumberOfComplain, RegisterDate
FROM Customer
WHERE RegisterDate BETWEEN "2012-06-01" AND "2013-06-01"
AND MaritalStatus = "Married"
ORDER BY NumOfComplain DESC, RegisterDate
```

19 records

CustomerID	FullName	YearBirth	MaritalStatus	NumberOfComplain	RegisterDate
1050	Gordon, Red	1952	Married	2	2013-04-30
1993	Dubols, Marc	1949	Married	1	2012-12-03
9360	Silk, Martha	1982	Married	0	2012-08-08
1402	Peacock, Mike	1954	Married	0	2012-09-11
10629	Liu, David	1973	Married	0	2012-09-14
2569	Smith, Jack	1987	Married	0	2012-10-12
6177	Caliahan, Laura	1985	Married	0	2012-11-12

Business Query Statement 2:

The **leadership team** wants to know which type of promotional offer is the most favored one.
List the frequency of the offer types that have been used by customers in the descending order of frequency.

```sql
SELECT OfferType, COUNT(PromotionOffer.DiscountCode) AS
FrequencyOfOfferType
FROM PromotionOffer JOIN CampaignBehavior
WHERE PromotionOffer.DiscountCode = CampaignBehavior.DiscountCode GROUP
BY OfferType ORDER BY FrequencyOfOfferType DESC
```

3 records

OfferType	FrequencyOfOfferType
Buy One Get One Free	17
50% Discount	9
20% Discount	3

Business Query Statement 3:

The **sales team** wants to see the sales performance of all the clock products. List all the information on the clock products like Product Name, Unit Price, QtySold.

```sql
SELECT ProductName, "$" || UnitPrice AS UnitPrice,
COUNT(Product.Product_ID)AS QtySold FROM Product JOIN OrderItem
WHERE Product.Product_ID = OrderItem.Product_ID AND
ProductName LIKE "%clock%" GROUP BY ProductName
```

3 records

ProductName	UnitPrice	QtySold
ALARM CLOCK BAKELIKE GREEN	$3.75	1
ALARM CLOCK BAKELIKE PINK	$3.75	1
ALARM CLOCK BAKELIKE RED	$3.75	1

Business Statements & Queries

Business Query Statement 4:

Sales team wants to know the number of products that have been sold out, including those which has no order history.

```
-- Step 1: Generate a list of products which has order history with their quantity sold.
SELECT DISTINCT OrderItem.Product_ID, ProductName,
COUNT(OrderItem.Product_ID) AS QuantitySold, QtyOnHand AS QuantityOnHand
FROM OrderItem JOIN Product WHERE OrderItem.Product_ID = Product.Product_ID
GROUP BY OrderItem.Product_ID
```

```
-- Step 2: Generate a list of products which have no order history by using EXCEPT
operator.
SELECT Product_ID, ProductName, 0 AS QuantitySold, QtyOnHand AS QuantityOnHand
FROM Product GROUP BY Product_ID EXCEPT
SELECT DISTINCT OrderItem.Product_ID, ProductName,
COUNT(OrderItem.Product_ID) AS QuantitySold, QtyOnHand AS QuantityOnHand FROM
OrderItem JOIN Product WHERE OrderItem.Product_ID = Product.Product_ID GROUP
BY OrderItem.Product_ID
```

```
-- Step 3: Combine the above 2 lists by using UNION Operator.
SELECT Product_ID, ProductName, 0 AS QuantitySold, QtyOnHand AS QuantityOnHand
FROM Product GROUP BY Product_ID EXCEPT
SELECT DISTINCT OrderItem.Product_ID, ProductName, COUNT(OrderItem.Product_ID)
AS QuantitySold, QtyOnHand AS QuantityOnHand FROM OrderItem JOIN Product
WHERE OrderItem.Product_ID = Product.Product_ID GROUP BY OrderItem.Product_ID
UNION
SELECT DISTINCT OrderItem.Product_ID, ProductName, COUNT(OrderItem.Product_ID)
AS QuantitySold, QtyOnHand AS QuantityOnHand FROM OrderItem JOIN Product
WHERE OrderItem.Product_ID = Product.Product_ID GROUP BY OrderItem.Product_ID
ORDER BY QuantitySold DESC
```

90 records

Product_ID	ProductName	QuantitySold	QuantityOnHand
121	POPPY'S PLAYHOUSE KITCHEN	10	1300000
172	RED WOOLLY HOTTIE WHITE HEART	8	1100000
110	WHITE HANGING HEART T-LIGHT HOLDER	6	200000
123	IVORY KNITTED MUG COSY	4	1500000
131	JAM MAKING SET WITH JARS	3	2300000
134	BLUE COAT RACK PARIS FASHION	3	2600000

Business Query Statement 5:

Sales team and leadership team want to see the quarterly sales performances.

```
SELECT strftime("%Y", OrderDate) AS Year, CASE
WHEN strftime("%m", OrderDate) IN ("01", "02", "03") THEN "Q1"
WHEN strftime("%m", OrderDate) IN ("04", "05", "06") THEN "Q2"
WHEN strftime("%m", OrderDate) IN ("07", "08", "09") THEN "Q3"
ELSE "Q4" END AS Quarter, SUM(TotalFee) AS QuarterlySales FROM OrderList
GROUP BY Year, Quarter ORDER BY Quarter, Year
```

12 records

Year	Quarter	QuarterlySales
2012	Q1	1446.00
2013	Q1	17764.61
2014	Q1	7654.37
2012	Q2	1666.10
2013	Q2	10818.53
2014	Q2	762.73
2012	Q3	4393.20
2013	Q3	5003.50
2014	Q3	1394.00
2012	Q4	10680.13
2013	Q4	4501.66
2014	Q4	213.00

Business Statements & Queries

Business Query Statement 6:

Create a **view** of labeled customer by the risk level which could be used to further check their behaviors in sales and campaigns.

```
CREATE VIEW IF NOT EXISTS CustGroup AS
SELECT Customer_ID, LastName || ", "|| FirstName AS FullName, YearBirth, Education,
MaritalStatus, RegisterDate, NumOfComplain, CusChanAccountID,
CASE WHEN NumOfComplain >1 THEN " Risky" WHEN NumOfComplain =1 THEN
"Attention" ELSE "Normal" END AS Tag FROM Customer GROUP BY Customer_ID
ORDER BY NumOfComplain DESC
```

Business Query Statement 7:

Create a **view** of customer consumption activity which contains the total order amount in $, including those who have not consumed yet.

```
CREATE VIEW IF NOT EXISTS CustActivity AS
SELECT Customer_ID, FullName, RegisterDate, Tag, 0 AS TotalOrderAmount_$ FROM
CustGroup WHERE Customer_ID NOT IN
 (SELECT Customer_ID FROM (SELECT b.Customer_ID, FullName, RegisterDate, Tag,
SUM(b.TotalFee) AS TotalOrderAmount_$ FROM CustGroup a JOIN OrderList b ON
a.Customer_ID = b.Customer_ID
GROUP BY b.Customer_ID))
UNION
SELECT b.Customer_ID, FullName, RegisterDate, Tag, SUM(b.TotalFee) AS
TotalOrderAmount_$ FROM CustGroup a JOIN OrderList b ON a.Customer_ID =
b.Customer_ID GROUP BY b.Customer_ID
```

```
SELECT *FROM CustActivity
```

75 records

Customer_ID	FullName	RegisterDate	Tag	TotalOrderAmount_$
387	Rocha, Alexande	2012-11-13	Normal	758.00
503	Jones, Monica	2013-08-14	Normal	12.00
965	King, Robert	2012-11-12	Attention	1446.00
1012	Smith, Lucas	2013-02-18	Normal	0.00

Business Query Statement 8:

Marketing Team wants to rank the campaign channel according to the total number of referrals happened through them.

```
SELECT a.CChannel_ID, PlatformName, COUNT(f.Referral_ID) AS NumOfReferral,
RANK () OVER (ORDER BY COUNT(f.Referral_ID) DESC) AS ReferralRank
FROM Platform a JOIN CampaignChannel b JOIN Campaign c JOIN PromotionOffer d
JOIN CampaignBehavior e JOIN Referral f WHERE a.CChannel_ID = b.CChannel_ID AND
b.Campaign_ID = c.Campaign_ID AND c.Campaign_ID = d.Campaign_ID AND
d.DiscountCode = e.DiscountCode AND e.Referral_ID = f.Referral_ID GROUP BY
a.CChannel_ID
```

4 records

CChannel_ID	PlatformName	NumOfReferral	ReferralRank
2	facebook	25	1
3	instagram	22	2
1	email	8	3
5	message	2	4

Business Query Statement 9:

Rank the customers according to their total order amount for each type of the risk tag.

```
SELECT FullName, RegisterDate, Tag, TotalOrderAmount_$,
RANK () OVER(PARTITION BY Tag ORDER BY TotalOrderAmount_$ DESC)
ActiveRankByTag FROM CustActivity GROUP BY Customer_ID
```

75 records

FullName	RegisterDate	Tag	TotalOrderAmount_$	ActiveRankByTag
Chen, Hanna	2014-03-03	Risky	250.77	1
Gordon, Red	2013-04-30	Risky	0.00	2
Dubols, Marc	2012-12-03	Attention	2666.00	1
King, Robert	2012-11-12	Attention	1446.00	2
Park, Margaret	2014-02-01	Attention	785.70	3
Smith, William	2013-06-17	Attention	236.46	4
Smith, Jack	2014-04-28	Attention	222.15	5
Murray, John	2013-08-14	Attention	213.00	6

Conclusion

Challenges Faced

- To create an ideal ERD is not easy, it needs to well know the customers' behavior, the analysis needs of targeted users and different campaign types.
- Populating database is also a challenge, since I need to combine data from different data sets and use fictitious data, then to make everything logical in different tables.
- This design still has some limitations due to the limit on the number of tables. Future design could consider to connect the database with social media platforms to collect more data on customers for further analysis.

Lesson Learned

- The business rules could help to define the relationships of each entities and set the limitations on how to store data.
- The database design should be based on the needs of the targeted users, it is essential to clarify the users before designing it.
- Third Normal Form is an important design objective that could avoid most of the bad relational design problems;
- If there is a Many2Many relationship, we need to split it to make it third normal form;
- Creating business statements is very helpful to perfecting the ERD, so keep improving the ERD design during the process of running the report.

Thank you!