IE76700 PROJECT REPORT

Direct to Consumer E-Commerce Database

Rahul Dixit

Abhishek Taware

dixit.ra@northeastern.edu

taware.ab@northeastern.edu

Percentage of Effort Contributed by Student 1: 50%

Percentage of Effort Contributed by Student 2: 50%

Signature of Student 1: *Rahul Dixit*

Signature of Student 2: *Abhishek Taware*

Submission Date: 05/01/22

Business Definition

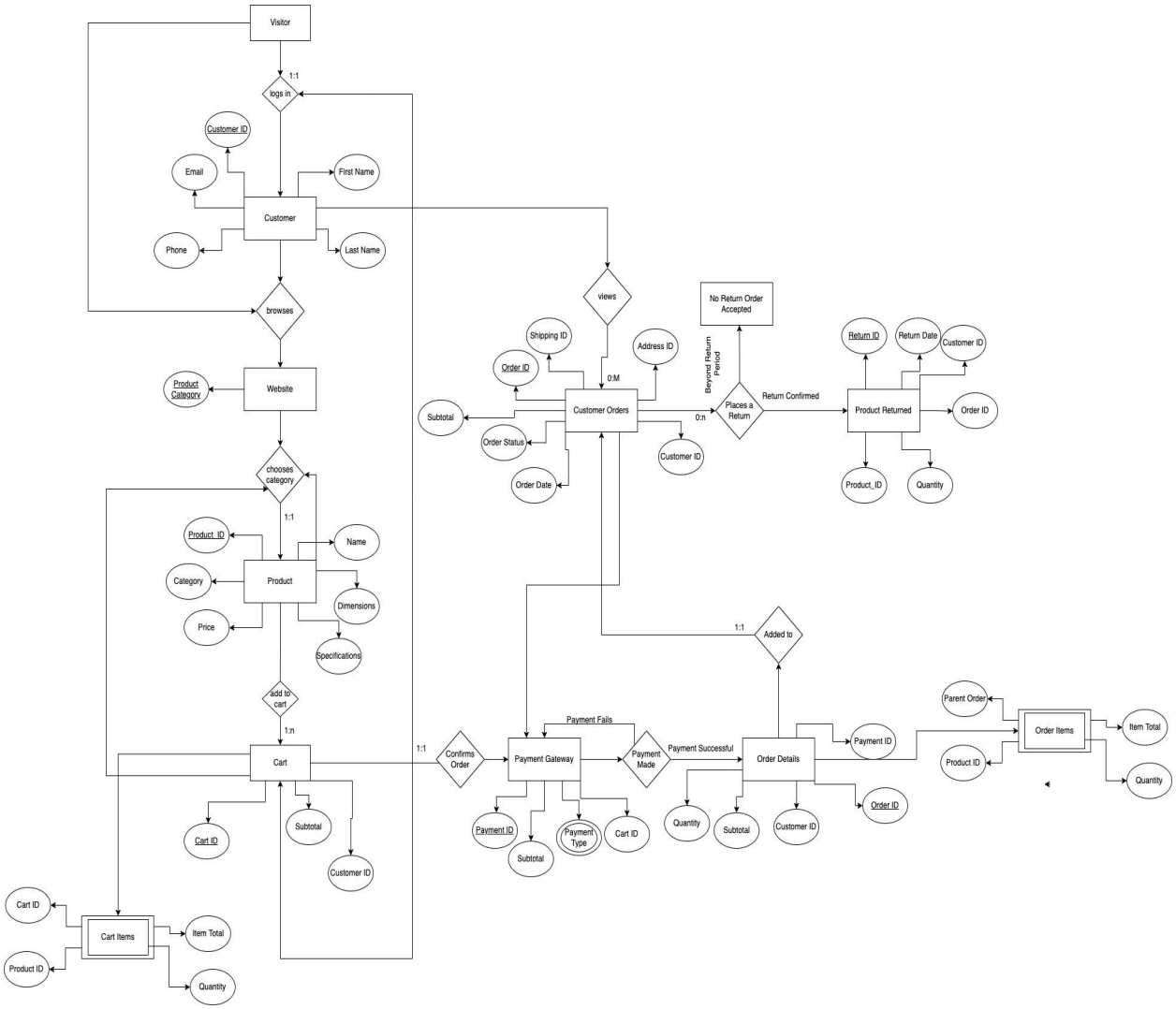
According to eMarketer's forecast, US direct-to-consumer (D2C) ecommerce sales will reach $151.20 billion in 2022, an increase of 16.9% compared to this year.The purpose of the database is to maintain the data used to run an D2C brand which includes datasets such as their Product Details , buyer details, shipment details, and payment information etc. We have included a full fledged online shopping model for a brand which sells products of its own direct to consumer.

Some of the business questions addressed

- Understanding Customers and their Orders having higher than their Average Order Value
- Understanding the Product/ Categories which have no takers / higher returns
- Understanding the most popular Zip Codes / States for Logistical Planning

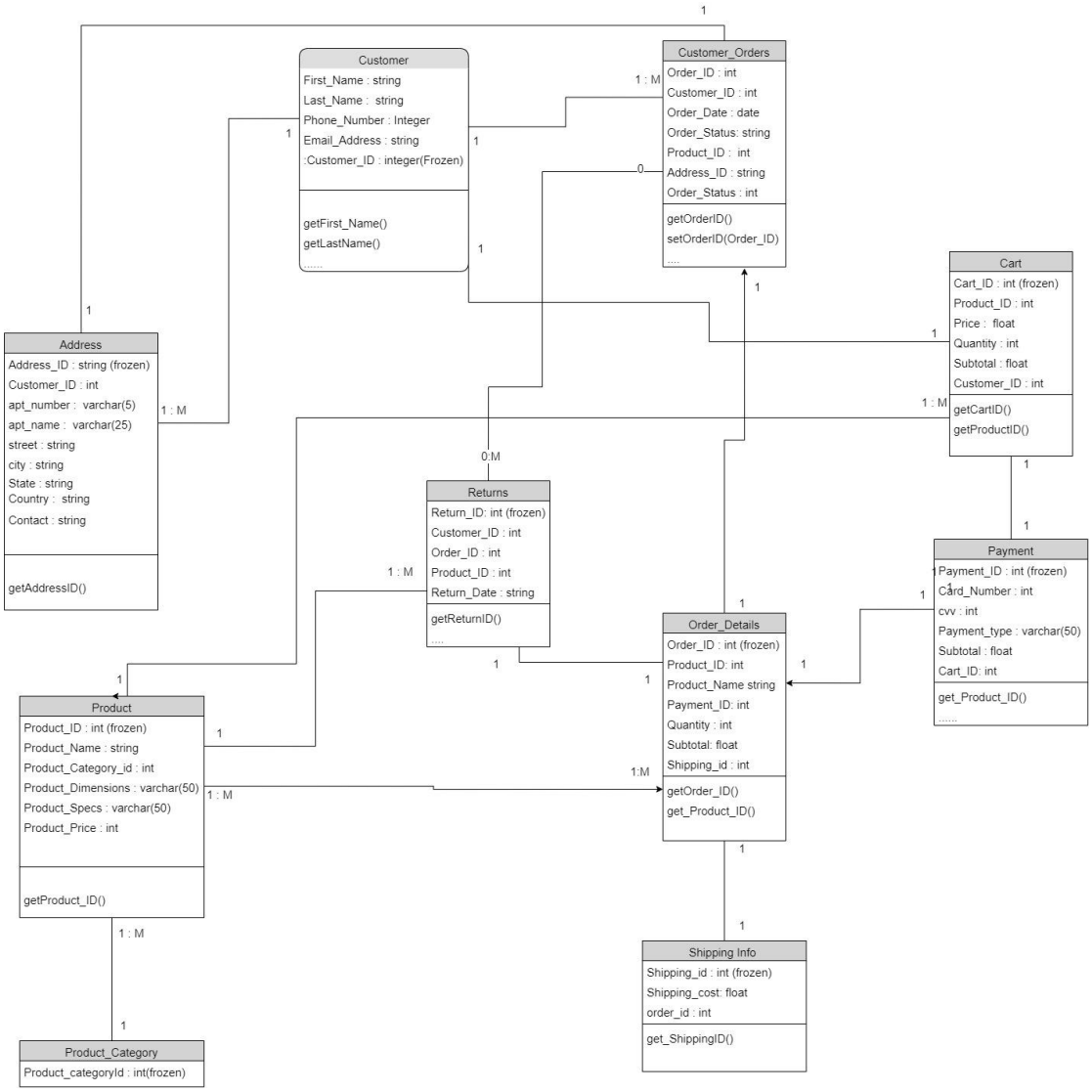
EER Diagram

To understand how we would proceed with the implementation of the Database, we worked on creating an EER Diagram. The diagram covered the entire flow of how the e-commerce site would work covering all the aspects and the entities required starting off with the Customer Entity Class.



UML Diagram

Based on our ER Diagram, we modeled a UML before implementing the relational model in the database. The UML Model gave us an idea of what classes we need to create and how the cardinalities would relate between different classes.



Tables

Below, we have created some records for reference which showcase the schema of our database along with some records of over 10 tables.

Customers

PK: Customer_ID

First_Name	Last_Name	Email_Address	Customer_ID	Phone_Number
Raj	Subramanian	raj.sub@gmail.com	101	8572832214
Rupith	Baburaj	rupith.baburaj@gmail.com	102	8572832212
Shwetha	Subramanian	shwetha.sub@gmail.com	103	8572832211

Products

PK: Product_ID, FK: product_Category_id

Product_ID	Product_Name	Product_Category_Id	Product_Dimensions	Product_Specs	Product_Price
301	Apple iPhone 13 Pro	203	7 inch x 3.5 inch	A15 Bionic Chip, 4GB RAM, Triple 13MP Camera	999
302	Apple iPhone 12 Pro	203	7 inch x 3.5 inch	A15 Bionic Chip, 4GB RAM, Triple 13MP Camera	979
303	Apple iPhone 11 Pro	203	7 inch x 3.5 inch	A15 Bionic Chip, 4GB RAM, Triple 13MP Camera	929
304	Ipad Air	204	10 inch x 7.5 inch	A15 Bionic Chip, 4GB RAM, Triple 13MP Camera	699
305	Ipad Pro	204	11 inch x 8.5 inch	A15 Bionic Chip, 4GB RAM, Triple 15MP Camera	799

Order Details

Order_ID	Payment_ID	Subtotal	Shipping_ID	Customer_ID
601	501	1599	701	101

Customer_Orders

Order_ID	Customer_ID	Order_Date	Address_ID	Order_Statu
601	101	2022-04-20 00:00:00	A1	Shipped
602	102	2022-04-20 00:00:00	A2	Shipped

Product_Category

product_category_id	category_name
201	Laptop
202	Camera
203	Phones

Returns

Return_ID	Customer_ID	Order_ID	Product_ID	Quantity	Return_Date
801	104	605	303	1	2022-04-30
802	103	604	301	1	2022-04-28

PK : Return_ID, FK: Order_ID, Customer_ID, Product_ID

Payment

Payment_ID	card_number	cvv	Payment_type	Subtotal	Cart_ID
501	1111222244445555	123	Credit Card	1599	401
502	2783278327683687	393	Debit Card	4998	402

PK : Payment_ID, FK: Cart_ID

Order_Items

Parent_Order	product_id	quantity	product_subtotal
601	312	1	1599

FK: Parent_Order

Cart_Items

Parent_Cart	product_id	quantity	product_subtotal
401	312	1	1599

FK : Parent_Cart, Product_ID

Addresses

Address_ID	Customer_ID	apt_number	street_address	city	State	zip
A1	101	2	20 S Huntington Avenue	Boston	Massachusetts	12132

PK: Address_ID, FK: Customer_ID

Cart

Cart_ID	Subtotal	Customer_ID
401	1599	101

PK: Cart_ID

SQL QUERIES

1. **Customers and number of orders per customer**

```
SELECT first_name, c1.customer_id, count(c2.order_id) as cnt

FROM customer c1 join customer_orders c2

ON c1.customer_id = c2.customer_id

GROUP BY c1.customer_id

ORDER BY cnt;
```

first_name	customer_id	cnt
Rupith	102	1
Shwetha	103	1
Aniruddh	104	1
Raj	101	4

2. **Customers with the most number of orders**

```
select T.customer_id,

    T.First_Name,

    T.TOTAL_ORDERS

FROM

(

SELECT Row_number() over(order by COUNT(Customer_Orders.Order_ID)
DESC) As RN,

     customer.customer_id,
```

```
    customer.first_name,
    COUNT(customer_orders.order_id) AS "TOTAL_ORDERS"
FROM Customer
INNER JOIN customer_orders
ON customer.customer_id = customer_orders.customer_id
GROUP BY customer.customer_id, customer.first_name
) AS T
Where RN=1;
```

customer_id	First_Name	TOTAL_ORDERS
101	Raj	4

3. **Zip Codes / States with the highest number of sales**

```
select a.zip, a.state, sum(subtotal) as total_order_value, a.customer_id
from customer_orders c join orderdetails o
on c.customer_id = o.customer_id
join address a on o.customer_id = a.customer_id
group by 2
order by 3 desc
limit 1;
```

zip	state	total_order_value	customer_id
12140	Massachusetts	14232	101

4. **Select the highest selling products for each product category**

```
with cte1 as(

SELECT product_Category_Id, p.product_id, MAX(o.product_subtotal/o.quantity)
as max_price

FROM products p

JOIN order_items o

ON p.product_id = o.product_id

GROUP BY product_Category_id

)

SELECT cte1.product_id, p.product_name, cte1.Product_Category_Id

FROM cte1

JOIN products p

ON cte1.product_id = p.product_id;
```

product_id	product_name	Product_Category_Id
312	Alienware m15 r3	206
305	Ipad Pro	204
301	Apple iPhone 13 Pro	203
315	Nikon D3300	202

5. **Average spend per category for products that have been shipped by product_Category_ID for orders which have been complete**

```
WITH cte2 AS(

SELECT AVG(product_subtotal) AS avg_price, product_category_id,
o.product_id, parent_order

FROM products p LEFT JOIN order_items o
```

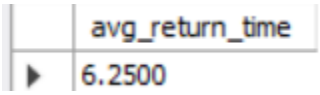
```
ON p.product_id = o.product_id
GROUP BY product_category_id
)
SELECT cte2.avg_price, cte2.product_Category_id, order_status
FROM cte2 JOIN customer_orders co
ON cte2.parent_order = co.order_id
WHERE order_status ='SHIPPED';
```

avg_price	product_Category_id	order_status
1085.6666666666667	206	Shipped
1148.5	204	Shipped
1944.6666666666667	203	Shipped
399	202	Shipped

6. **Average Time Duration within which a Customer places a return order**

```
WITH cte3
AS(
SELECT DATEDIFF(Return_Date, Order_Date) AS difference,
r.Order_ID,
product_id
FROM customer_orders c
RIGHT JOIN returns r
ON c.order_id = r.order_id
```

```
group by r.product_id)
SELECT AVG(difference) AS avg_return_time
FROM cte3;
```



avg_return_time
6.2500

7. **Customers and the Orders that they have placed which have costed more than their average total.**

```
SELECT * from OrderDetails O
WHERE
Subtotal >
(SELECT AVG(Subtotal) FROM OrderDetails OD
WHERE OD.Customer_ID = O.Customer_ID);
```

8. **Stored Procedure to change the Zip Code of a Customer's Address**

```
DELIMITER //
CREATE PROCEDURE Update_Zip (IN ZIP1 INT, IN addr varchar(50))
UPDATE Address
SET zip = ZIP1 WHERE street_address = addr;
//
CALL Update_Zip(12140, '20 S Huntington Avenue');
DELIMITER;
```

No- SQL Implementation in MongoDB

Tables created using shell:

Addresses

Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
20.48 kB	10	221.00 B	1	36.86 kB

CUSTOMER

Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
20.48 kB	10	155.00 B	1	36.86 kB

Order_Details

Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
20.48 kB	5	93.00 B	1	36.86 kB

Payment

Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
20.48 kB	5	149.00 B	1	36.86 kB

Products

Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
20.48 kB	14	221.00 B	1	36.86 kB

Queries and Results:

Query1 : Getting the subtotal of the order grouped by order_id

Query2 : Getting the Subtotal for each payment type used

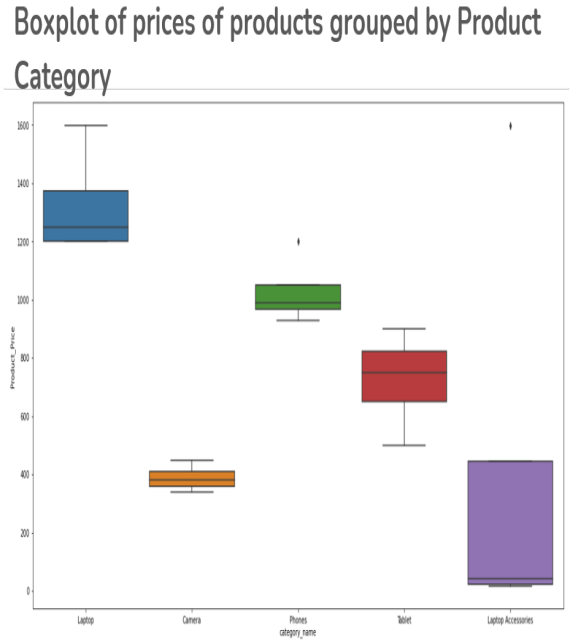
Query3 : Query1 using Map Reduce

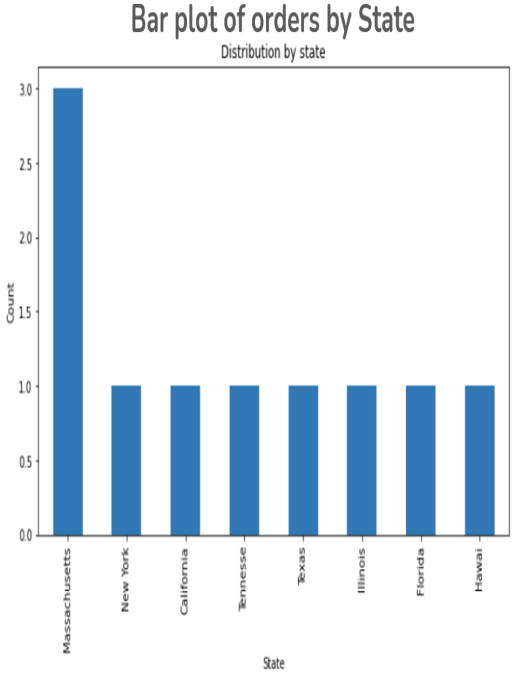
```
> show collections
Addresses
CUSTOMER
Order_Details
Payment
Products
> db.Order.aggregate([{$group:{_id:'$Order_ID', total:{$sum:"$Subtotal"}}}]);
> db.Order_Details.aggregate([{$group:{_id:'$Order_ID', total:{$sum:"$Subtotal"}}}]);
{ "_id" : 705, "total" : 3499 }
{ "_id" : 704, "total" : 2997 }
{ "_id" : 702, "total" : 4998 }
{ "_id" : 703, "total" : 1599 }
{ "_id" : 701, "total" : 1599 }
> db.Payment.aggregate([{$group:{_id:'$Payment_type', total:{$sum:"$Subtotal"}}}]);
{ "_id" : "Gift Card", "total" : 3499 }
{ "_id" : "Debit Card", "total" : 4998 }
{ "_id" : "Credit Card", "total" : 4596 }
{ "_id" : "Prepaid Card", "total" : 1599 }
>
> var map = function() { emit(this.Order_ID, this.Subtotal)};
> var reduce = function(Order_ID,Subtotal){return Array.sum(Subtotal);};
> db.Order_Details.mapReduce(map,reduce, {out:'Result'});
{ "result" : "Result", "ok" : 1 }
> db.Result.find()
{ "_id" : 704, "value" : 2997 }
{ "_id" : 705, "value" : 3499 }
{ "_id" : 703, "value" : 1599 }
{ "_id" : 702, "value" : 4998 }
{ "_id" : 701, "value" : 1599 }
```

Implementation of Python Script

The first visualization i.e the Box Plot gives us an idea about the price ranges of different categories. On the Other hand, the second Plot i.e the Bar Chart gives us an idea of which States have seen most of the Order being placed wherein Massachusetts leads the tally.

Python Script - EDA





Summary

Since the D2C Space is seeing such traction, we thought about modeling the database used by any D2C Site to understand how the flow of data happens on such websites. Using these tables and records, we can also visually answer some of the business questions that these companies might have which can improve their efficiency, margins and help them target the right customer base and alter their strategies accordingly.