CVS Retail Store

MGT 4058-A TEAM 5:
Mateen Rasoli, Chuandong Liu, Max Oglesby, Danny Schall, Anish Yennam

INTRODUCTION:

Our project focused on CVS and more specifically their retail store. CVS, also known as “Consumer Value Stores,” was first founded in 1963 in Massachusetts where they primarily sold health and beauty products to consumers. The company was founded by brothers Stanley and Sidney Goldstein along side their partner Ralph Hoagland.

Our group has designed a database intended for the CVS company to manage their online and in-person sales. The database created includes several items such as information relating to orders that customers have placed either online or in-person, customer information, retail store information, stock/storage information, etc. Our goal for this project is to allow the CVS company to manage their products and orders more efficiently while also satisfying customer requirements based on the sales of their products. It is evident that CVS stores may have products out of stock so we will keep track of their storage for their retail stores to ensure that there is enough stock for each product. This system in place will aid CVS to better manage their product stock and ultimately reduce the likelihood of goods being out of stock to increase customer purchases. In addition to this, we plan on keeping track of customer information for both online and in-person customers using the CVS card. Furthermore, we can also keep track of any damages, theft, lost goods, and any other outlying circumstances regarding products sold at CVS. We will also create a table of CVS’s special offers and discounts to customers.

Overall, our common goal with creating this database is to assist the CVS company and upper-level management to better manage their products, customers, and to make decisions related to these two while satisfying customer needs and desires.

DESIGNING OUR E-R DIAGRAM:

For the design of the E-R diagram, we included members, which are our customers as an entity to keep track of the customer information. Members have simple attributes, such as member_id, which is the primary key for member, customer name, their address, email, phone number, and their date of birth. Then, we also included another important element, which is orders. Orders have three attributes, order id, which is the primary key, order date, and order type. Since we are CVS retail store, discount code is another important aspect given that a CVS membership gets a lot of discount codes after check out. For the discount in CVS store, we keep track of its discount code, rate of the discount, and the category of the product it works on. Members are the people who have orders in CVS, so their is a one-to-many relationship between the members and orders. Discount code will be given to customers, so every customer can have more than one discount code, but each discount code can only be held by one customer. Discount can be used in the orders. One discount code can not apply to orders or apply to exactly one order. One order can use zero or one discount code. We also include a supertype and subtype relationship in it. The orders that we have are separate into online and in-person orders. Online orders are stored in the warehouse and the warehouse will send the item out. The in person order is mainly focused on in-store spending. We mainly focused on retail in our database. For the in person orders, it is a subtype of orders, with disjoint and total participation since one order cannot be in person order and online order at the same time. Additionally, all the orders must be either an online order or an in-person order. Since we have in person orders, we create a store entity to indicate the location where the purchase happened. The store entity have store id, as a primary key, the address, which is a composite key. Also, we include the most important entity of CVS, our product. For each product, we have the product name, product id, their price, and their category. In-person orders can contain one or many product. One product can be in zero or many in-person orders. For this contain relationship, we also have a entity called quantity to keep track of the quantity of each product in that order. One in person order can be processed in one and only one store, one store can process one or many in person orders. Furthermore, we also keep track of the remaining stock of the stores, by adding a hold relationship. This is a relationship between store and product. One store can hold zero or many products, and one product can be held by zero or many stores. We also have a simple attribute, remaining stock, for this many to many relationship to keep track of the remaining quantity of each product the store holds.

CREATING TABLES:

Once we created and modified our E-R Diagram, we then began the process of creating our tables. We started off by creating the tables for each of the entities displayed in our E-R Diagram and adding columns with their respective attributes. When we got to the Orders entity and its Supertype/Subtype relationship, we made sure to distinctly identify the Online Order table and In-Person table, and had their respective primary keys - O_Order_ID and P_Order_ID - be the first column, followed by their remaining attributes. Since there were two M:N relationships, we had to create two additional tables, representing the Store to Product relationship and the Product to In-Person Order relationship. We made sure the primary keys of the relationships were the primary keys in the respective M:N tables, with an additional column added for the unique attribute for each relationship. Once we did all of this, we then went back and added the necessary foreign keys to the tables. Since we had only three 1:M relationships, we only needed to add three foreign keys to our data. We took the primary key of the one side and made it a foreign key to the many side. We did have one 1:1 relationship, so we took the primary key of the mandatory table, which was Discount, and made it a foreign key to the optional table, which was Orders.

NORMALIZING OUR TABLES:

Having created our tables, we then proceeded to normalize them. We first started by seeing if our tables satisfied the 1NF constraints. We found that there were no repeating attributes between any of our relations and each table had a defined primary key that uniquely identified each row in each relation. Once we established 1NF, we then proceeded to satisfy the 2NF requirements by making sure there were no partial dependencies in our tables. We only had two tables that had a composite primary key, and the following attributes were determinant on both the primary keys for the respective tables. Store_ID and Product_ID together could identify the Remaining_Stock, and P_Order_ID and Product_ID could identify the Quantity of the purchase. We then established our tables as being 2NF and proceeded to satisfy the 3NF requirements. We looked through each table and each attribute to make sure there were no transitive dependencies, and found that no nonkey attribute is functionally dependant on another. Having gone through all these steps and confirming our tables to be in 3NF, we successfully normalized all of our tables.

CRITIQUE:

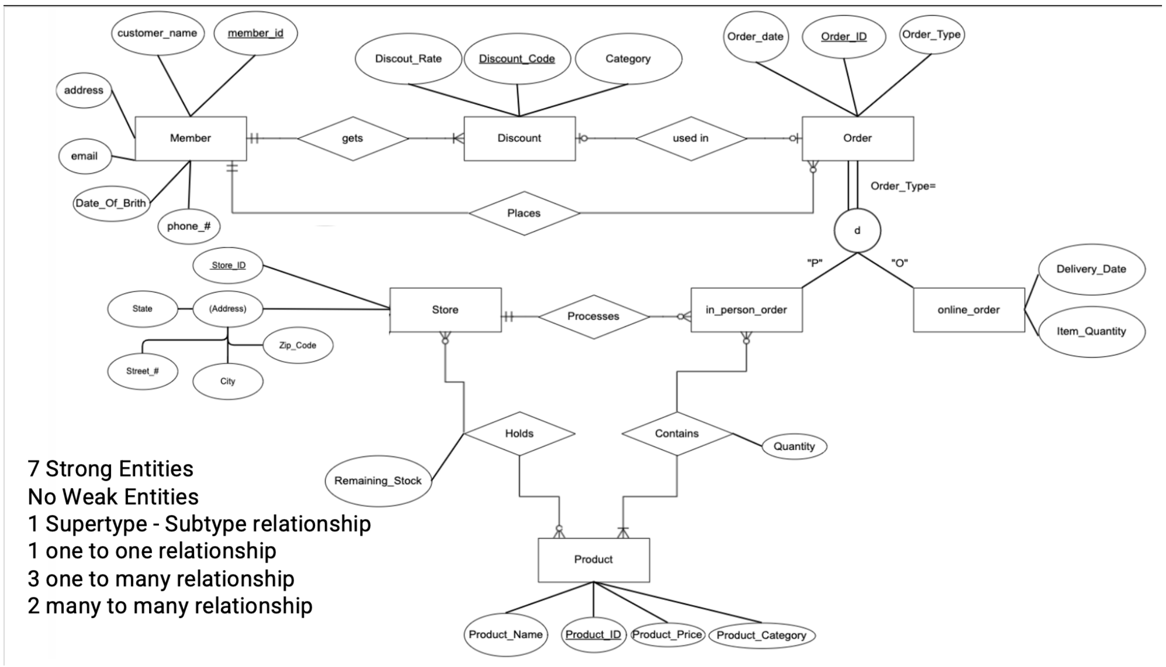
Even though our E-R Diagram turned out really good and served its purpose, there were a few issues we experienced when utilizing SQL. The first issue we had was that our Order table from our original E-R Diagram was getting confused for the ORDER keyword when we attempted to do some SQL queries. To fix this, we had to change Order to Orders, so the SQL could better differentiate between the two.

Our E-R Diagram also had some weaknesses that we had to overcome. One weakness was that our tables were not the friendliest to use when using SQL. A lot of our tables are long and take up many characters, which makes the SQL codes longer and harder to read than they should be. If we had more time, we would go back and shorten the length of the table names and some of the attributes, so that writing and reading SQL code would become easier. Another weakness with our E-R Diagram is that we did not really focus deeply on some of the other functions that CVS offers. CVS also has pharmacies and MinuteClinics, which we could have included into our Database had we had more time and better knowledge of what kinds of information CVS would stores in those tables. Adding these other functions would make it more useful for CVS, as all of their key components would be stored within a central database and easily able to answer even more business problems.

If we had more time implement more tables and entities to our database, we would have added a Warehouse entity. We would have used this entity to highlight where the products from the Online Orders table comes from, as they do not originate from particular stores. We could have also used the Warehouse table to highlight which store used which warehouse when filling up on stock. It would have been useful when calculating which warehouses are the most used and which warehouse CVS should consider shutting down due to low usage.

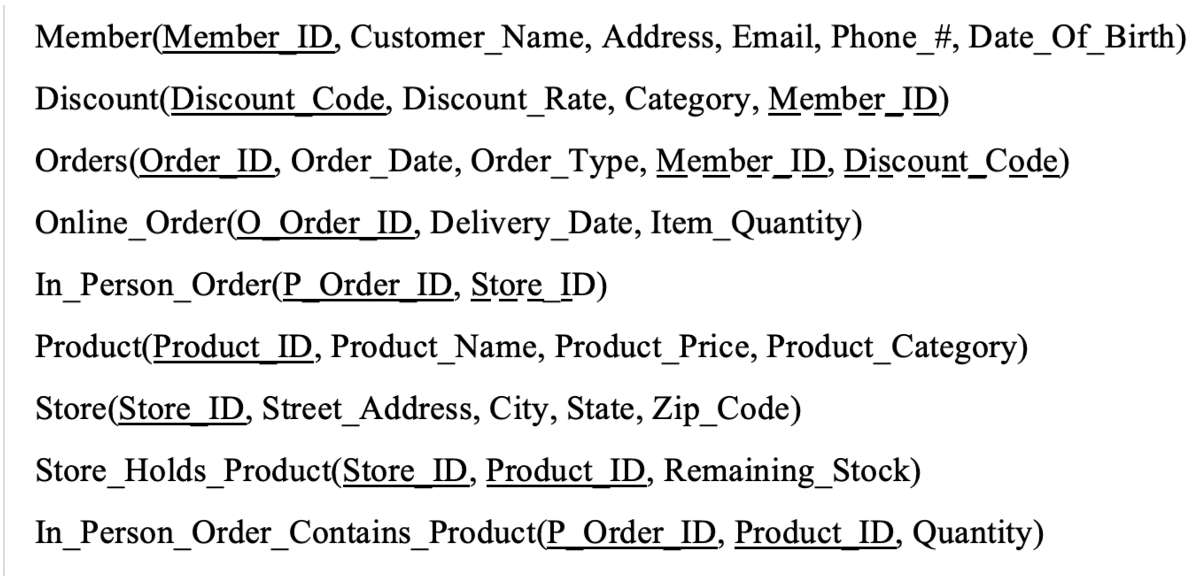
APPENDIX:

E-R Diagram:



7 Strong Entities
No Weak Entities
1 Supertype - Subtype relationship
1 one to one relationship
3 one to many relationship
2 many to many relationship

Normalized Table:



Member(Member_ID, Customer_Name, Address, Email, Phone_#, Date_Of_Birth)

Discount(Discount_Code, Discount_Rate, Category, Member_ID)

Orders(Order_ID, Order_Date, Order_Type, Member_ID, Discount_Code)

Online_Order(O_Order_ID, Delivery_Date, Item_Quantity)

In_Person_Order(P_Order_ID, Store_ID)

Product(Product_ID, Product_Name, Product_Price, Product_Category)

Store(Store_ID, Street_Address, City, State, Zip_Code)

Store_Holds_Product(Store_ID, Product_ID, Remaining_Stock)

In_Person_Order_Contains_Product(P_Order_ID, Product_ID, Quantity)

SQL QUERIES:

```
#Query 1
SELECT Product.Product_Name, SUM(In_Person_Order_Contains_Product.Quantity) AS
'Total Orders'
FROM Product, In_Person_Order_Contains_Product
WHERE Product.Product_ID = In_Person_Order_Contains_Product.Product_ID
GROUP BY Product.Product_Name
ORDER BY SUM(In_Person_Order_Contains_Product.Quantity) DESC
LIMIT 1
```

```
#Query 2
SELECT Store_Holds_Product.Store_ID ,Store_Holds_Product.Product_ID,
Product.Product_Name, Store_Holds_Product.Remaining_Stock FROM
Store_Holds_Product, Product
WHERE Remaining_Stock = 0
AND Product.Product_ID = Store_Holds_Product.Product_ID
GROUP BY Store_ID, Product_ID

#Query 3
SELECT Store_ID, SUM(In_Person_Order_Contains_Product.Quantity *
Product.Product_Price) AS 'Sales'
FROM In_Person_Order LEFT JOIN In_Person_Order_Contains_Product
on In_Person_Order.P_Order_ID = In_Person_Order_Contains_Product.P_Order_ID
LEFT JOIN Product on In_Person_Order_Contains_Product.Product_ID =
Product.Product_ID
GROUP BY Store_ID
ORDER BY Sales DESC

#Query 4
SELECT Store_ID, Product_Category, SUM(In_Person_Order_Contains_Product.Quantity)
AS Total_Quantity
FROM In_Person_Order LEFT JOIN In_Person_Order_Contains_Product
on In_Person_Order.P_Order_ID = In_Person_Order_Contains_Product.P_Order_ID
LEFT JOIN Product on In_Person_Order_Contains_Product.Product_ID =
Product.Product_ID
GROUP BY Store_ID, Product_Category
HAVING Total_Quantity > 10

#Query 5
SELECT Member.Customer_Name, Online_Order.Item_Quantity FROM Online_Order LEFT
JOIN Orders ON Online_Order.O_Order_ID = Orders.Order_ID LEFT JOIN Member ON
Member.Member_ID =Orders.Member_ID ORDER BY (item_quantity * 1) DESC LIMIT 1
#Query 6
SELECT Member.Customer_Name, COUNT(Member.Customer_Name) AS "Number of
Orders" FROM In_Person_Order LEFT JOIN Orders ON In_Person_Order.P_Order_ID =
Orders.Order_ID LEFT JOIN Member ON Member.Member_ID = Orders.Member_ID
GROUP BY Member.Customer_Name ORDER BY COUNT(Member.Customer_Name)
DESC

#Query 7
SELECT In_Person_Order.Store_ID, (SUM(In_Person_Order_Contains_Product.Quantity)/
COUNT(In_Person_Order_Contains_Product.P_Order_ID)) AS 'Avg_Quantity_Sold'
FROM In_Person_Order_Contains_Product, In_Person_Order
WHERE In_Person_Order.P_Order_ID=In_Person_Order_Contains_Product.P_Order_ID
GROUP BY Store_ID
```

```
#Query 8
SELECT In_Person_Order_Contains_Product.P_Order_ID,
SUM(In_Person_Order_Contains_Product.Quantity) AS 'Items_in_Order'
FROM In_Person_Order_Contains_Product, In_Person_Order
WHERE In_Person_Order.P_Order_ID=In_Person_Order_Contains_Product.P_Order_ID
GROUP BY P_Order_ID
ORDER BY Items_in_Order DESC
LIMIT 10

#Query 9
SELECT Product.Product_Category, SUM(Store_Holds_Product.Remaining_Stock) AS
'Total_Remaining_Stock'
FROM Product, Store_Holds_Product
WHERE Store_Holds_Product.Product_ID=Product.Product_ID
GROUP BY Product_Category
ORDER BY Total_Remaining_Stock DESC

#Query 10
SELECT Product.Product_Name, SUM(In_Person_Order_Contains_Product.Quantity) AS
'Products_Sold'
FROM In_Person_Order_Contains_Product, Product
WHERE In_Person_Order_Contains_Product.Product_ID = Product.Product_ID
GROUP BY Product_Name
ORDER BY Products_Sold
LIMIT 10
```

PROJECT SLIDES:

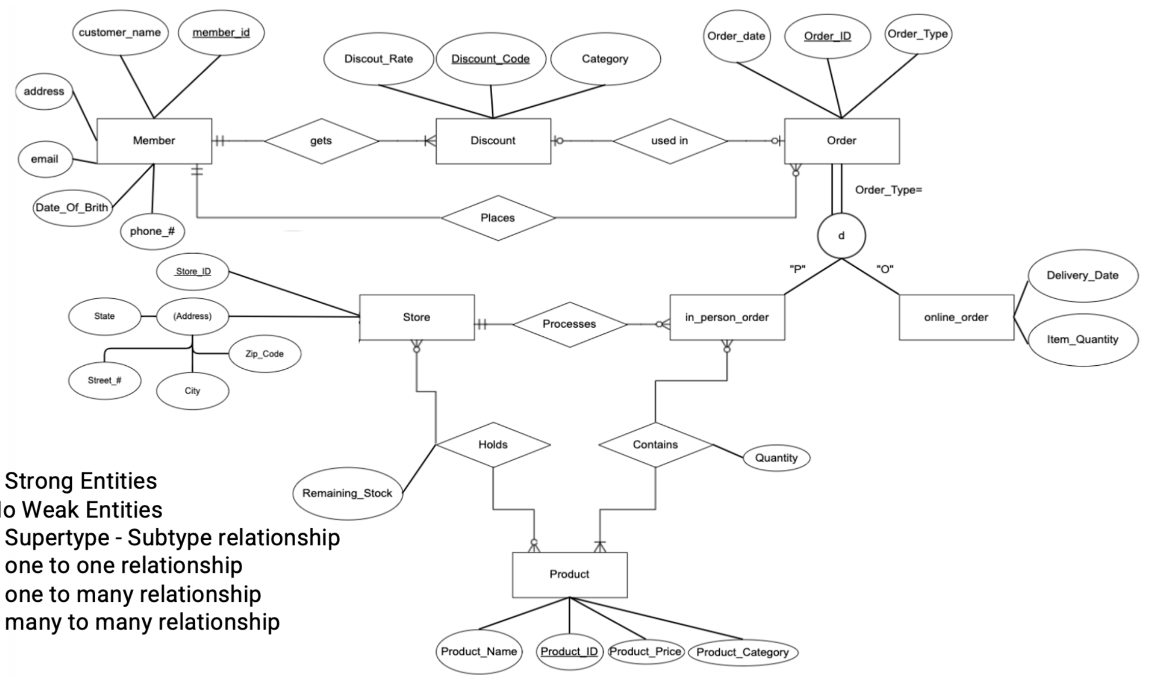
CVS Retail Store

Anish Reddy Yennam, Chuandong Liu, Daniel Gregory Schall,
Mateen Gebrael Rasoli, Maxwell Nathaniel Oglesby

Background

- CVS = “Consumer Value Stores”
- First CVS store was founded in 1963 and sold health and beauty products in Massachusetts
- Founded by brothers Stanley and Sidney Goldstein & partner Ralph Hoagland
- Designed a database to manage their online & in-person sales to satisfy customer requirements
- Keep track of storage/stock of products
- Keep track of in-person/online customers with the CVS card
- Keep track of damages, theft, etc. regarding products
- Created a table on special offers & discounts.

E-R Diagram



7 Strong Entities
No Weak Entities
1 Supertype - Subtype relationship
1 one to one relationship
3 one to many relationship
2 many to many relationship

Normalized Table

Member(Member_ID, Customer_Name, Address, Email, Phone_#, Date_Of_Birth)

Discount(Discount_Code, Discount_Rate, Category, Member_ID)

Orders(Order_ID, Order_Date, Order_Type, Member_ID, Discount_Code)

Online_Order(O_Order_ID, Delivery_Date, Item_Quantity)

In_Person_Order(P_Order_ID, Store_ID)

Product(Product_ID, Product_Name, Product_Price, Product_Category)

Store(Store_ID, Street_Address, City, State, Zip_Code)

Store_Holds_Product(Store_ID, Product_ID, Remaining_Stock)

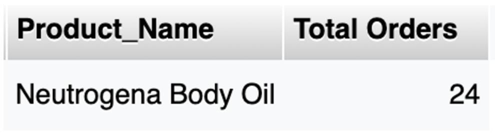
In_Person_Order_Contains_Product(P_Order_ID, Product_ID, Quantity)

Query 1

Business Problem:

Which product has the most sales?

```
SELECT Product.Product_Name,
SUM(In_Person_Order_Contains_Product.Qua
ntity) AS 'Total Orders'
FROM Product,
In_Person_Order_Contains_Product
WHERE Product.Product_ID =
In_Person_Order_Contains_Product.Product
_ID
GROUP BY Product.Product_Name
ORDER BY
SUM(In_Person_Order_Contains_Product.Qua
ntity) DESC
LIMIT 1;
```



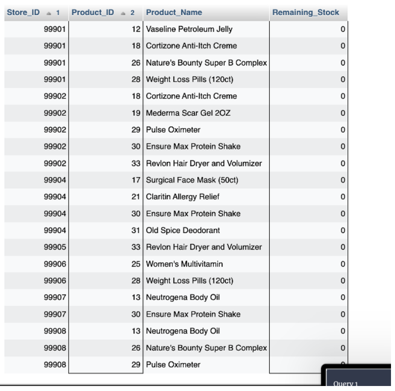
Product_Name	Total Orders
Neutrogena Body Oil	24

Query 2

Business Problem:

Which products require more inventory?

```
SELECT
Store_Holds_Product.Store_ID ,Store_Holds_
Product.Product_ID, Product.Product_Name,
Store_Holds_Product.Remaining_Stock FROM
Store_Holds_Product, Product
WHERE Remaining_Stock = 0
AND Product.Product_ID =
Store_Holds_Product.Product_ID
GROUP BY Store_ID, Product_ID;
```



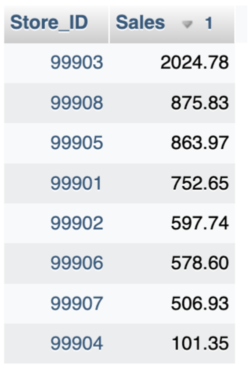
Store_ID ▲ 1	Product_ID ▲ 2	Product_Name	Remaining_Stock
99901	12	Vaseline Petroleum Jelly	0
99901	18	Cortizone Anti-Itch Creme	0
99901	26	Nature's Bounty Super B Complex	0
99901	28	Weight Loss Pills (120ct)	0
99902	18	Cortizone Anti-Itch Creme	0
99902	19	Mederma Scar Gel 2OZ	0
99902	29	Pulse Oximeter	0
99902	30	Ensure Max Protein Shake	0
99902	33	Revlon Hair Dryer and Volumizer	0
99904	17	Surgical Face Mask (50ct)	0
99904	21	Claritin Allergy Relief	0
99904	30	Ensure Max Protein Shake	0
99904	31	Old Spice Deodorant	0
99905	33	Revlon Hair Dryer and Volumizer	0
99906	25	Women's Multivitamin	0
99906	28	Weight Loss Pills (120ct)	0
99907	13	Neutrogena Body Oil	0
99907	30	Ensure Max Protein Shake	0
99908	13	Neutrogena Body Oil	0
99908	26	Nature's Bounty Super B Complex	0
99908	29	Pulse Oximeter	0

Query 3

Business Problem:

Which store has the most sales?

```
    SELECT Store_ID,
SUM(In_Person_Order_Contains_Product.Quantity *
Product.Product_Price) AS 'Sales'
    FROM In_Person_Order LEFT JOIN
In_Person_Order_Contains_Product
    on In_Person_Order.P_Order_ID =
In_Person_Order_Contains_Product.P_Order_ID
    LEFT JOIN Product on
In_Person_Order_Contains_Product.Product_ID =
Product.Product_ID
    GROUP BY Store_ID
    ORDER BY Sales DESC;
```



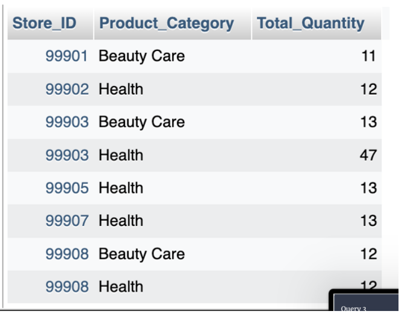
Store_ID	Sales ▼ 1
99903	2024.78
99908	875.83
99905	863.97
99901	752.65
99902	597.74
99906	578.60
99907	506.93
99904	101.35

Query 4

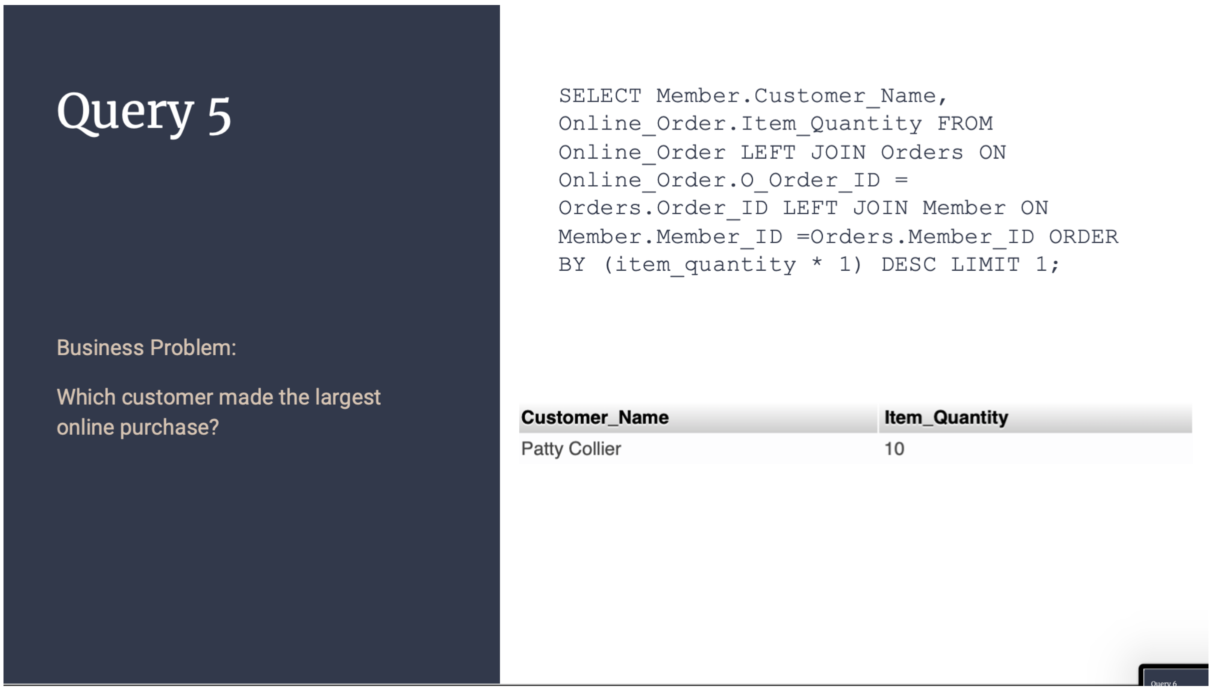
Business Problem:

Which Product Categories have more than 10 items sold for all of the stores?

```
    SELECT Store_ID, Product_Category,
SUM(In_Person_Order_Contains_Product.Quantity) AS
Total_Quantity
    FROM In_Person_Order LEFT JOIN
In_Person_Order_Contains_Product
    on In_Person_Order.P_Order_ID =
In_Person_Order_Contains_Product.P_Order_ID
    LEFT JOIN Product on
In_Person_Order_Contains_Product.Product_ID =
Product.Product_ID
    GROUP BY Store_ID, Product_Category
    HAVING Total_Quantity > 10;
```



Store_ID	Product_Category	Total_Quantity
99901	Beauty Care	11
99902	Health	12
99903	Beauty Care	13
99903	Health	47
99905	Health	13
99907	Health	13
99908	Beauty Care	12
99908	Health	12



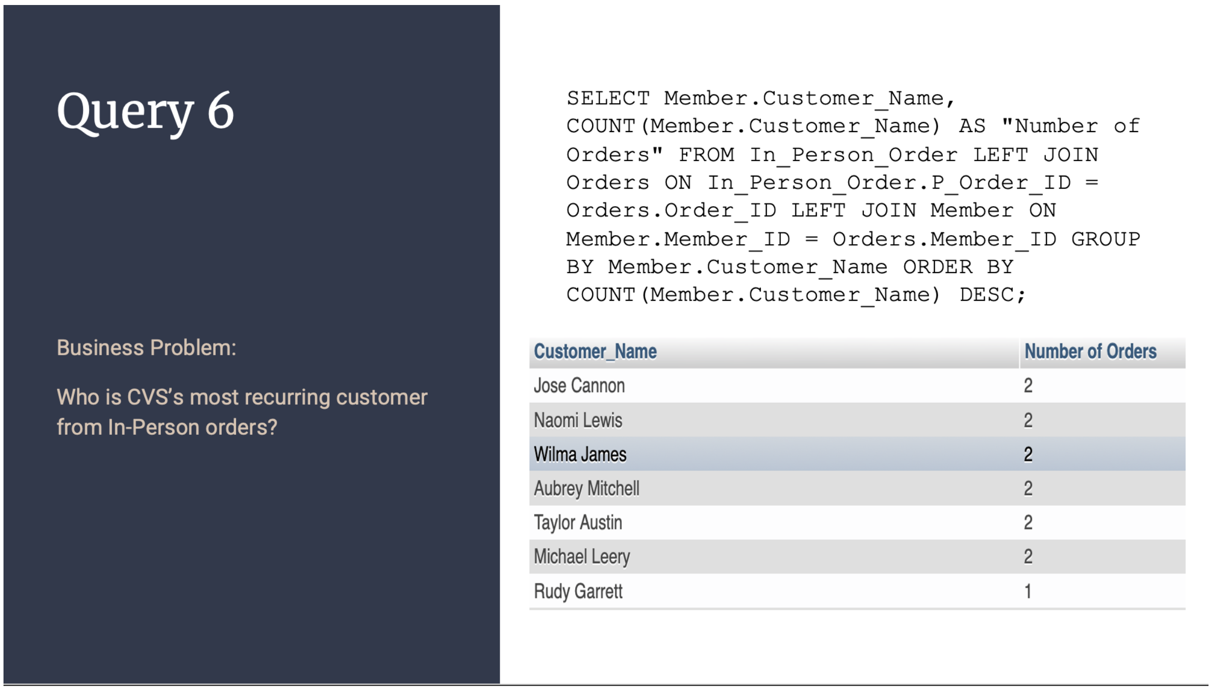
Query 5

Business Problem:

Which customer made the largest online purchase?

```
SELECT Member.Customer_Name,
Online_Order.Item_Quantity FROM
Online_Order LEFT JOIN Orders ON
Online_Order.O_Order_ID =
Orders.Order_ID LEFT JOIN Member ON
Member.Member_ID =Orders.Member_ID ORDER
BY (item_quantity * 1) DESC LIMIT 1;
```

Customer_Name	Item_Quantity
Patty Collier	10



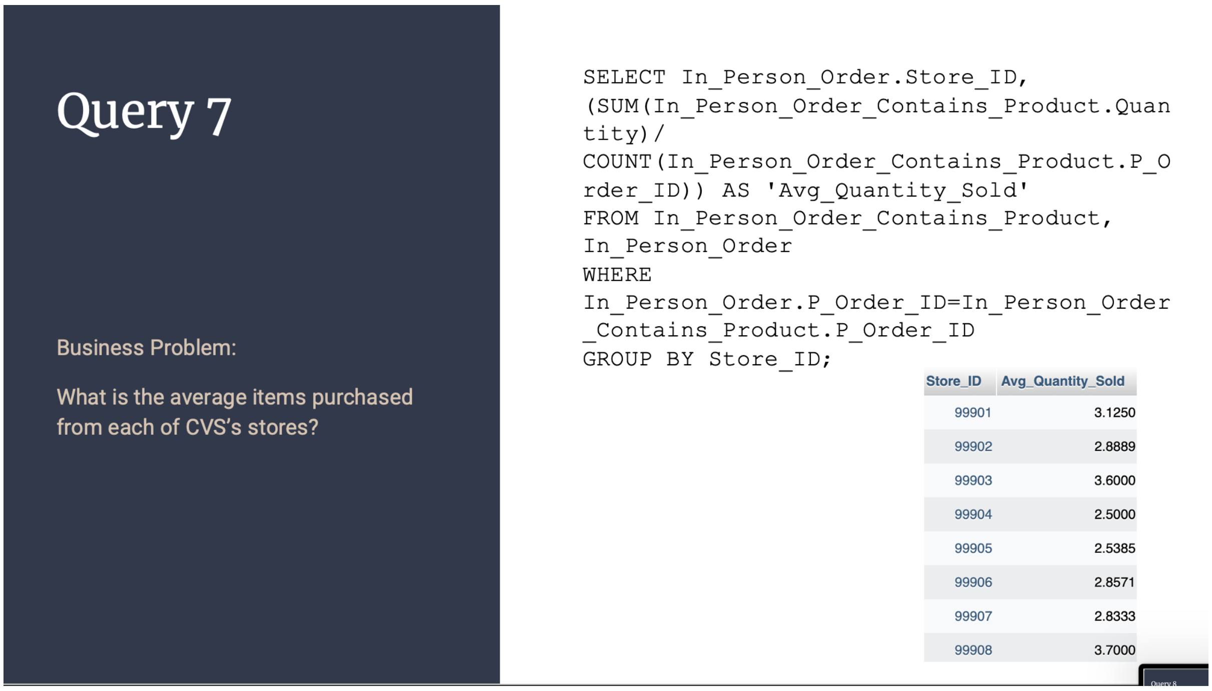
Query 6

Business Problem:

Who is CVS's most recurring customer from In-Person orders?

```
SELECT Member.Customer_Name,
COUNT(Member.Customer_Name) AS "Number of
Orders" FROM In_Person_Order LEFT JOIN
Orders ON In_Person_Order.P_Order_ID =
Orders.Order_ID LEFT JOIN Member ON
Member.Member_ID = Orders.Member_ID GROUP
BY Member.Customer_Name ORDER BY
COUNT(Member.Customer_Name) DESC;
```

Customer_Name	Number of Orders
Jose Cannon	2
Naomi Lewis	2
Wilma James	2
Aubrey Mitchell	2
Taylor Austin	2
Michael Leery	2
Rudy Garrett	1



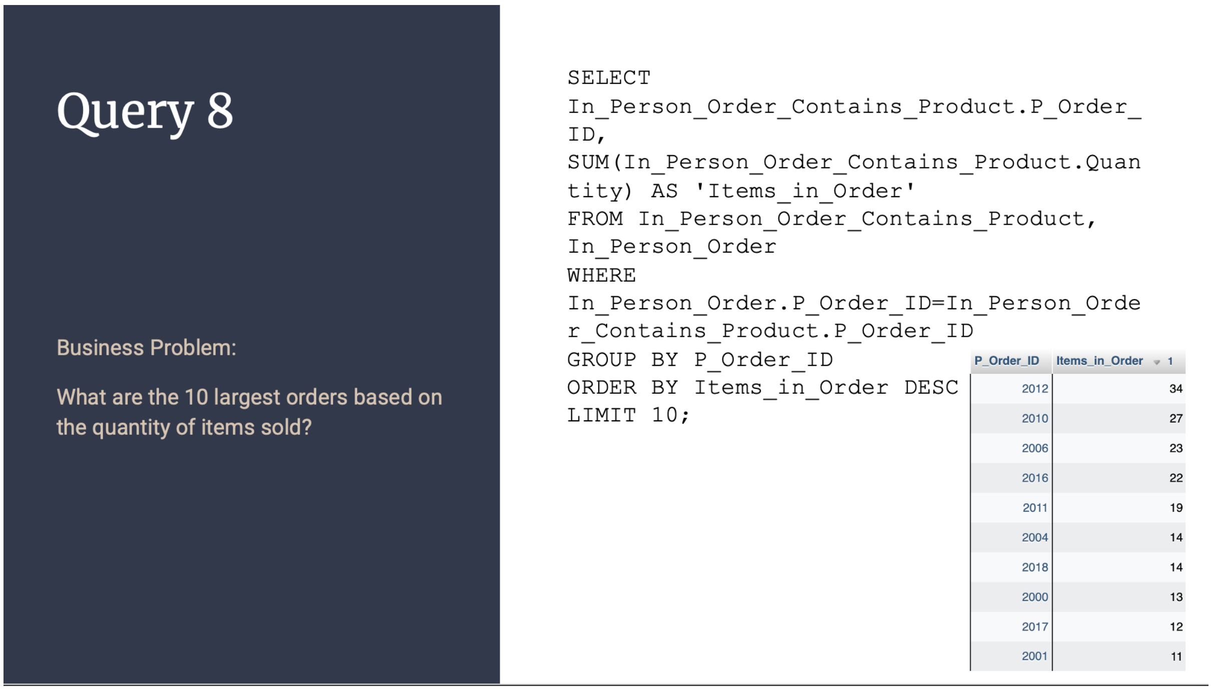
Query 7

Business Problem:

What is the average items purchased from each of CVS's stores?

```
SELECT In_Person_Order.Store_ID,
(SUM(In_Person_Order_Contains_Product.Quan
tity)/
COUNT(In_Person_Order_Contains_Product.P_O
rder_ID)) AS 'Avg_Quantity_Sold'
FROM In_Person_Order_Contains_Product,
In_Person_Order
WHERE
In_Person_Order.P_Order_ID=In_Person_Order
_Contains_Product.P_Order_ID
GROUP BY Store_ID;
```

Store_ID	Avg_Quantity_Sold
99901	3.1250
99902	2.8889
99903	3.6000
99904	2.5000
99905	2.5385
99906	2.8571
99907	2.8333
99908	3.7000



Query 8

Business Problem:

What are the 10 largest orders based on the quantity of items sold?

```
SELECT
In_Person_Order_Contains_Product.P_Order_
ID,
SUM(In_Person_Order_Contains_Product.Quan
tity) AS 'Items_in_Order'
FROM In_Person_Order_Contains_Product,
In_Person_Order
WHERE
In_Person_Order.P_Order_ID=In_Person_Orde
r_Contains_Product.P_Order_ID
GROUP BY P_Order_ID
ORDER BY Items_in_Order DESC
LIMIT 10;
```

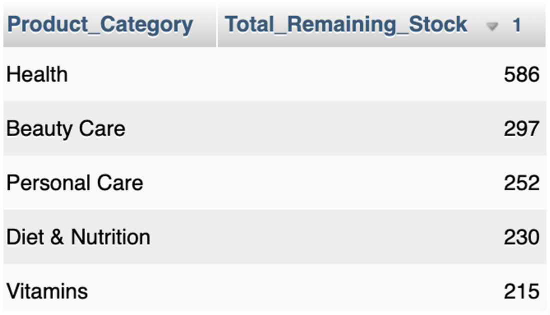
P_Order_ID	Items_in_Order ▼ 1
2012	34
2010	27
2006	23
2016	22
2011	19
2004	14
2018	14
2000	13
2017	12
2001	11

Query 9

Business Problem:

How much inventory does each Product Category have left in stock?

```
SELECT Product.Product_Category,
SUM(Store_Holds_Product.Remaining_Stock)
AS 'Total_Remaining_Stock'
FROM Product, Store_Holds_Product
WHERE
Store_Holds_Product.Product_ID=Product.P
roduct_ID
GROUP BY Product_Category
ORDER BY Total_Remaining_Stock DESC;
```



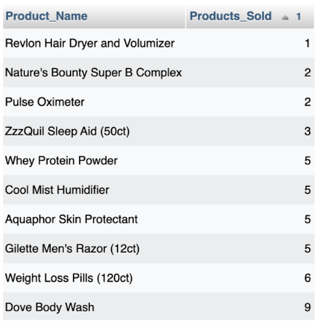
Product_Category	Total_Remaining_Stock 1
Health	586
Beauty Care	297
Personal Care	252
Diet & Nutrition	230
Vitamins	215

Query 10

Business Problem:

What are the Top 10 lowest selling products based on recent In-Person sales?

```
SELECT Product.Product_Name,
SUM(In_Person_Order_Contains_Product.Quan
tity) AS 'Products_Sold'
FROM In_Person_Order_Contains_Product,
Product
WHERE
In_Person_Order_Contains_Product.Product_
ID = Product.Product_ID
GROUP BY Product_Name
ORDER BY Products_Sold
LIMIT 10;
```



Product_Name	Products_Sold 1
Revlon Hair Dryer and Volumizer	1
Nature's Bounty Super B Complex	2
Pulse Oximeter	2
ZzzQuil Sleep Aid (50ct)	3
Whey Protein Powder	5
Cool Mist Humidifier	5
Aquaphor Skin Protectant	5
Gilette Men's Razor (12ct)	5
Weight Loss Pills (120ct)	6
Dove Body Wash	9

Thank You

Questions ?