Group 7

Novartis Healthcare Data Management System

Krish Engineer
Monica Martinez
Pratik Gawli
Sai Bhargav Tetali
Vivian Wang

Table of Contents

Data Models

Companies in the healthcare industry hold an immense amount of data. However, due to the nature of the industry, these data have to be kept private and de-identified. For our project we chose to work with Novartis Healthcare, a Swiss-American multinational pharmaceutical corporation based in Basel, Switzerland and Cambridge, Massachusetts, United States. Novartis Healthcare has a data management system storing tons of data corresponding to various transactional operations ranging from drug sales to drug promotional campaigns. Even though the patient information is kept private, we can still perform some analysis to get insights about the sales figures in different regions, their inventory system, and the effectiveness of drug promotional campaigns. To address these questions, we generated synthetic data on drug sales, inventory, and drug promotional campaigns since healthcare data is not publicly available.

Overview of Dataset

- Patient Data: Patient ID, Patient Age
- Drug List Data: Drug ID, Drug Name, Drug Price, Disease Area
- Pharmacy List Data: Pharmacy ID, Pharmacy Name, Pharmacy Address
- Physician List Data: Physician ID, Physician Type, Physician First Name, Physician Last Name

Our three main transactional datasets contains information listed above and are composed of:

- **Transaction List Data**: Transaction ID, Patient ID, Physician ID, Pharmacy ID, Drug ID, Transaction Date, Quantity
- **Inventory Out Data**: Drug ID, Delivery Date, Units, Pharmacy ID
- **Inventory In Data**: Drug ID, Received Date, Units
- **Promotion List Data:** Call ID, Drug ID, Physician ID, Date of Call

The transaction list data records all transactions at Novartis. Each transaction has a transaction ID. The patient ID refers to which patient this transaction is for. The drug ID tells us which drug was prescribed. The pharmacy ID shows which pharmacy the drug was shipped from with what quantity. The physician ID maps to our database of all the physicians recommending Novartis drugs. Finally, a transaction date is associated with each transaction in our record.

The inventory data keeps track of the inflow and outflow of each drug. We divided our inventory table into two parts: InventoryIn and InventoryOut. The InventoryIn table records only shipments from our factory and contains only positive numbers for the quantities we received. The InventoryOut table records our shipments to customers as a positive number, and drug returns from pharmacies as a negative number. The pharmacy ID keeps track of which pharmacy initiated the return. We also recorded a date for each of these transactions.

The promotion list data records information for every promotional call. For each call, we have a call ID, the date of call, the drug ID for the drug that was promoted, and the physician ID for the physician that made the call.

After constructing the data warehouse which is to be in a specific format to be used for analytical purposes, we want to answer the following questions:

- For different zip codes, which drug had the highest sales in a given time period; which regions have the highest sales in general?
- What is the effectiveness of promotional calls on our sales?
- Are there certain pharmacies that are returning our drugs more frequently?

DDL Code

Below are the DDL code commands that were used for constructing the Transactional and core tables:

```
create table Physicians(
    PhysicianID number(6,0) NOT NULL PRIMARY KEY,
    PhysicianType varchar2(255),
    PhysicianFirstName varchar2(255),
    PhysicianLastName varchar2(255)
)
```

```
create table Patients(
    PatientID number(6,0) NOT NULL PRIMARY KEY,
    PatientAge varchar2(255)
)
```

```
create table Drugs(
    DrugID number(6,0) NOT NULL PRIMARY KEY,
    DrugName varchar2(255),
    DrugPrice number(5,0),
    DiseaseArea varchar2(255)
)
```

```
create table Pharmacy_list(
    PharmacyId number(6,0) not null Primary Key,
    PharmacyName Varchar2(255),
    PharmacyAddress Varchar2(255)
)
```

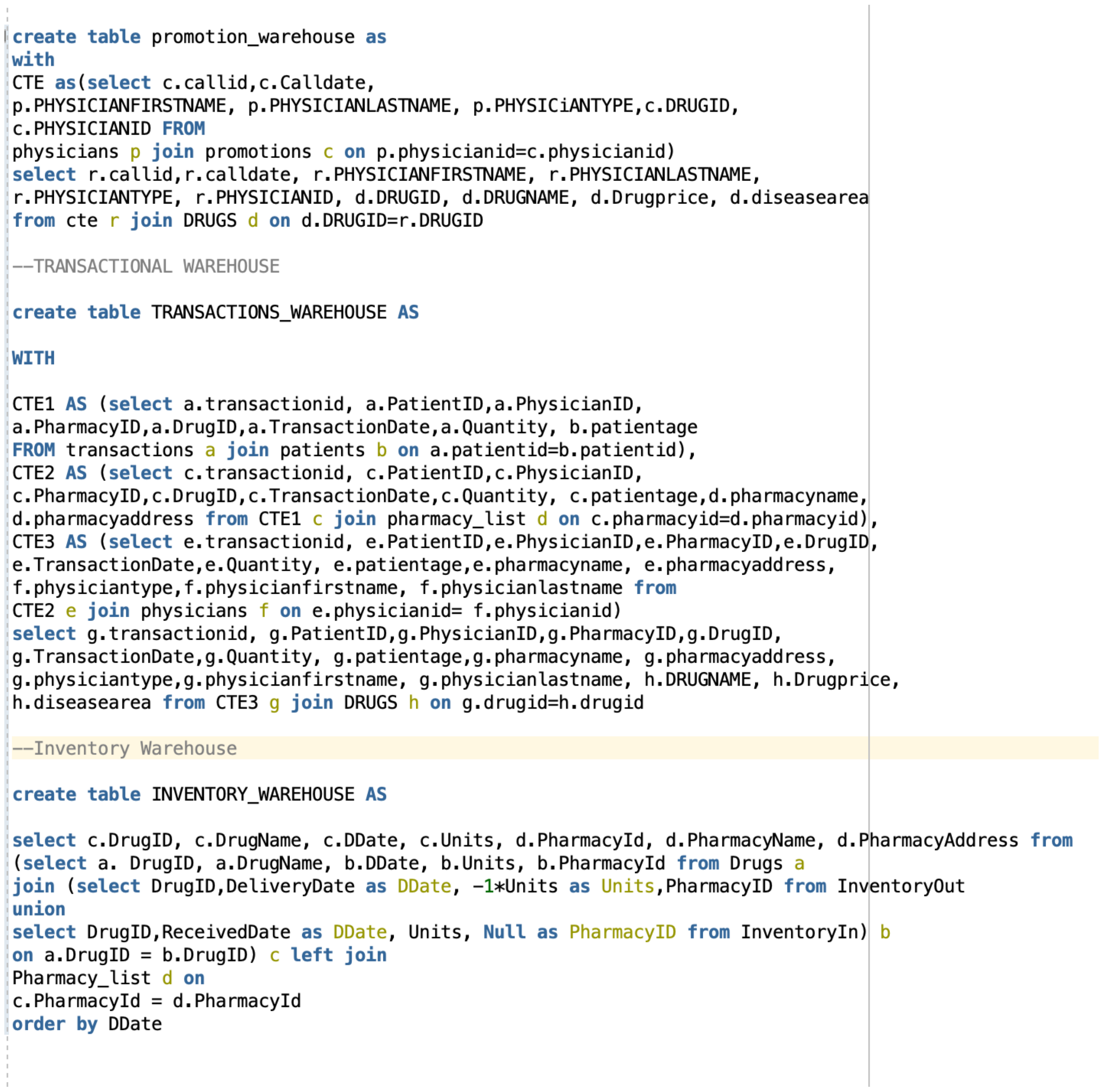
```
create table Transactions(
    TransactionID number(6,0) NOT NULL PRIMARY KEY,
    PatientID number(6,0) NOT NULL,
    PhysicianID number(6,0),
    PharmacyId number(6,0) NOT NULL,
    DrugID number(6,0) NOT NULL,
    TransactionDate date NOT NULL,
    Quantity number(4,0) NOT NULL,
    FOREIGN KEY (PatientID) REFERENCES Patients(PatientID),
    FOREIGN KEY (PhysicianID) REFERENCES Physicians(PhysicianID),
    FOREIGN KEY (PharmacyId) REFERENCES Pharmacy_list(PharmacyId),
    FOREIGN KEY (DrugID) REFERENCES Drugs(DrugID)
)
```

```
create table Promotions(
    CallID number(6,0) NOT NULL PRIMARY KEY,
    PhysicianID number(6,0),
    DrugID number(6,0) NOT NULL,
    CallDate date NOT NULL,
    FOREIGN KEY (PhysicianID) REFERENCES Physicians(PhysicianID),
    FOREIGN KEY (DrugID) REFERENCES Drugs(DrugID)
)
```

```
create table InventoryOut(
    DrugID number(6,0) NOT NULL,
    DeliveryDate date NOT NULL,
    Units number (6,0) NOT NULL,
    PharmacyId number(6,0),
    FOREIGN KEY (PharmacyId) REFERENCES Pharmacy_list(PharmacyId),
    FOREIGN KEY (DrugID) REFERENCES Drugs(DrugID)
)
```

```
create table InventoryIn(
    DrugID number(6,0) NOT NULL,
    ReceivedDate date NOT NULL,
    Units number (6,0) NOT NULL,
    FOREIGN KEY (DrugID) REFERENCES Drugs(DrugID)
)
```

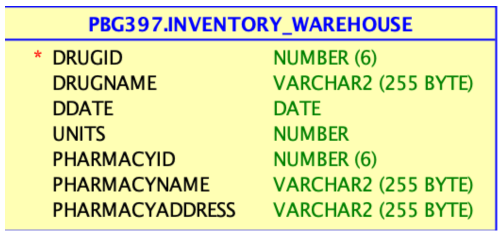
Below is the code used to build the data warehouse:



```
create table promotion_warehouse as
with
CTE as(select c.callid,c.Calldate,
p.PHYSICIANFIRSTNAME, p.PHYSICIANLASTNAME, p.PHYSICiANTYPE,c.DRUGID,
c.PHYSICIANID FROM
physicians p join promotions c on p.physicianid=c.physicianid)
select r.callid,r.calldate, r.PHYSICIANFIRSTNAME, r.PHYSICIANLASTNAME,
r.PHYSICIANTYPE, r.PHYSICIANID, d.DRUGID, d.DRUGNAME, d.Drugprice, d.diseasearea
from cte r join DRUGS d on d.DRUGID=r.DRUGID

--TRANSACTIONAL WAREHOUSE

create table TRANSACTIONS_WAREHOUSE AS

WITH

CTE1 AS (select a.transactionid, a.PatientID,a.PhysicianID,
a.PharmacyID,a.DrugID,a.TransactionDate,a.Quantity, b.patientage
FROM transactions a join patients b on a.patientid=b.patientid),
CTE2 AS (select c.transactionid, c.PatientID,c.PhysicianID,
c.PharmacyID,c.DrugID,c.TransactionDate,c.Quantity, c.patientage,d.pharmacyname,
d.pharmacyaddress from CTE1 c join pharmacy_list d on c.pharmacyid=d.pharmacyid),
CTE3 AS (select e.transactionid, e.PatientID,e.PhysicianID,e.PharmacyID,e.DrugID,
e.TransactionDate,e.Quantity, e.patientage,e.pharmacyname, e.pharmacyaddress,
f.physiciantype,f.physicianfirstname, f.physicianlastname from
CTE2 e join physicians f on e.physicianid= f.physicianid)
select g.transactionid, g.PatientID,g.PhysicianID,g.PharmacyID,g.DrugID,
g.TransactionDate,g.Quantity, g.patientage,g.pharmacyname, g.pharmacyaddress,
g.physiciantype,g.physicianfirstname, g.physicianlastname, h.DRUGNAME, h.Drugprice,
h.diseasearea from CTE3 g join DRUGS h on g.drugid=h.drugid

--Inventory Warehouse

create table INVENTORY_WAREHOUSE AS

select c.DrugID, c.DrugName, c.DDate, c.Units, d.PharmacyId, d.PharmacyName, d.PharmacyAddress from
(select a. DrugID, a.DrugName, b.DDate, b.Units, b.PharmacyId from Drugs a
join (select DrugID,DeliveryDate as DDate, -1*Units as Units,PharmacyID from InventoryOut
union
select DrugID,ReceivedDate as DDate, Units, Null as PharmacyID from InventoryIn) b
on a.DrugID = b.DrugID) c left join
Pharmacy_list d on
c.PharmacyId = d.PharmacyId
order by DDate
```

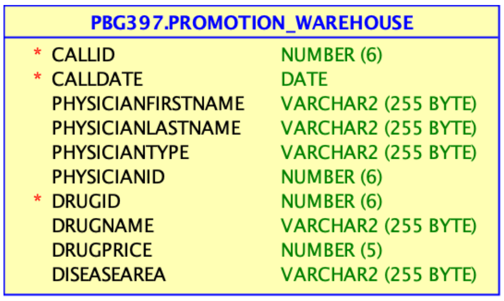
Data Warehouse - SSOT

Our data is grouped into three main data warehouses. To construct a SSOT, we created an inventory warehouse, a promotion warehouse, and a transactions warehouse.



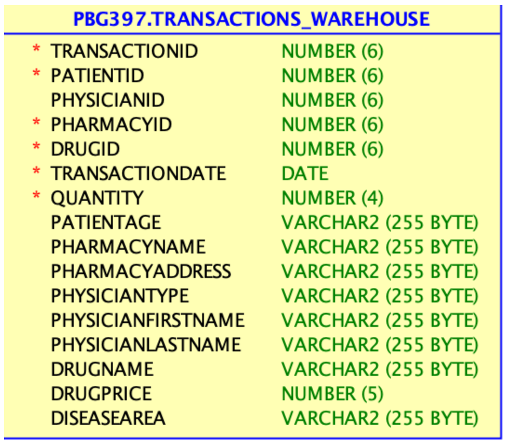
PBG397.INVENTORY_WAREHOUSE		
*	DRUGID	NUMBER (6)
	DRUGNAME	VARCHAR2 (255 BYTE)
	DDATE	DATE
	UNITS	NUMBER
	PHARMACYID	NUMBER (6)
	PHARMACYNAME	VARCHAR2 (255 BYTE)
	PHARMACYADDRESS	VARCHAR2 (255 BYTE)

This is a table of all inventory data. It includes DrugID, DrugName, Date, Units, PharmacyID, PharmacyName, and PharmacyAddress.



PBG397.PROMOTION_WAREHOUSE	
* CALLID	NUMBER (6)
* CALLDATE	DATE
PHYSICIANFIRSTNAME	VARCHAR2 (255 BYTE)
PHYSICIANLASTNAME	VARCHAR2 (255 BYTE)
PHYSICIANTYPE	VARCHAR2 (255 BYTE)
PHYSICIANID	NUMBER (6)
* DRUGID	NUMBER (6)
DRUGNAME	VARCHAR2 (255 BYTE)
DRUGPRICE	NUMBER (5)
DISEASEAREA	VARCHAR2 (255 BYTE)

The table contains promotion data. It includes the CallID, CallDate, PhysicianFirstName, PhysicianLastName, PhysicianType, PhysicianID, DrugID, DrugName, DrugPrice, and DiseaseArea.

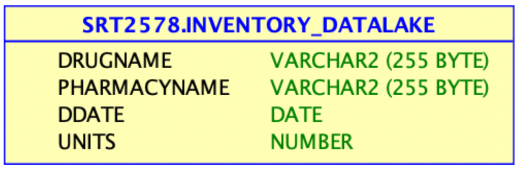


PBG397.TRANSACTIONS_WAREHOUSE	
* TRANSACTIONID	NUMBER (6)
* PATIENTID	NUMBER (6)
PHYSICIANID	NUMBER (6)
* PHARMACYID	NUMBER (6)
* DRUGID	NUMBER (6)
* TRANSACTIONDATE	DATE
* QUANTITY	NUMBER (4)
PATIENTAGE	VARCHAR2 (255 BYTE)
PHARMACYNAME	VARCHAR2 (255 BYTE)
PHARMACYADDRESS	VARCHAR2 (255 BYTE)
PHYSICIANTYPE	VARCHAR2 (255 BYTE)
PHYSICIANFIRSTNAME	VARCHAR2 (255 BYTE)
PHYSICIANLASTNAME	VARCHAR2 (255 BYTE)
DRUGNAME	VARCHAR2 (255 BYTE)
DRUGPRICE	NUMBER (5)
DISEASEAREA	VARCHAR2 (255 BYTE)

Lastly, we have our transactions_warehouse table.
We created a SSOT for Novartis Healthcare. We constructed these tables using ETL operations on the transactional tables. The data required for any analytical operations can be extracted using these tables.

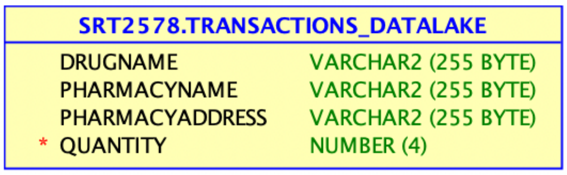
Data Marts - MVOT

We formed our data marts using parts of the information from our data warehouses. The goal is to use these data marts to answer the questions we've posted above. These serve as MVOT of our data as they are derived from the SSOT and included additional variables we created for analytical operations.



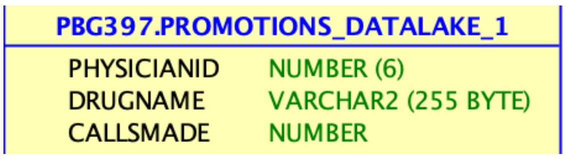
SRT2578.INVENTORY_DATALAKE	
DRUGNAME	VARCHAR2 (255 BYTE)
PHARMACYNAME	VARCHAR2 (255 BYTE)
DDATE	DATE
UNITS	NUMBER

Our inventory data mart contains information from the inventory warehouse. We will perform some analysis to find out how our inventory system is looking. The analysis will display the current stocks of each of our drugs.



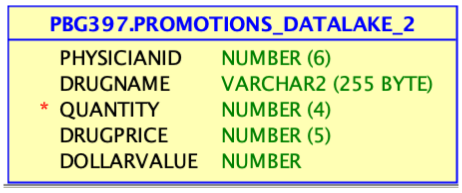
SRT2578.TRANSACTIONS_DATALAKE	
DRUGNAME	VARCHAR2 (255 BYTE)
PHARMACYNAME	VARCHAR2 (255 BYTE)
PHARMACYADDRESS	VARCHAR2 (255 BYTE)
* QUANTITY	NUMBER (4)

Our transactions data mart contains information from the transactions warehouse. It records the drug name, the pharmacy name, the pharmacy address, and the drug quantity for each transaction. We want to use this subset of information to create a summary of our sales in different regions.



PBG397.PROMOTIONS_DATALAKE_1	
PHYSICIANID	NUMBER (6)
DRUGNAME	VARCHAR2 (255 BYTE)
CALLSMADE	NUMBER

Our first promotions data mart includes information on the physician ID and drug name from the promotion warehouse. We created an additional variable for the total number of calls made to each physician for each drug. We want to use this data mart to keep track of the drug promotional calls made to different physicians and for different drugs.



PBG397.PROMOTIONS_DATALAKE_2	
PHYSICIANID	NUMBER (6)
DRUGNAME	VARCHAR2 (255 BYTE)
* QUANTITY	NUMBER (4)
DRUGPRICE	NUMBER (5)
DOLLARVALUE	NUMBER

In the second promotions data mart, we have information on physician ID, drug name, drug quantity and drug price from the promotion warehouse. We calculated the dollar value for each promotional sale and stored that as the dollar value variable in the data mart. We want to evaluate the success of our promotional campaign using this data mart.
All the above tables created are suffixed with ‘Datalake’ as they form the csv’s which are stored in Novartis Healthcare Datalake on AWS along with unstructured data of reviews for each transactions which can be used for various analytical purposes by various analytics teams with proper permissions to access data.

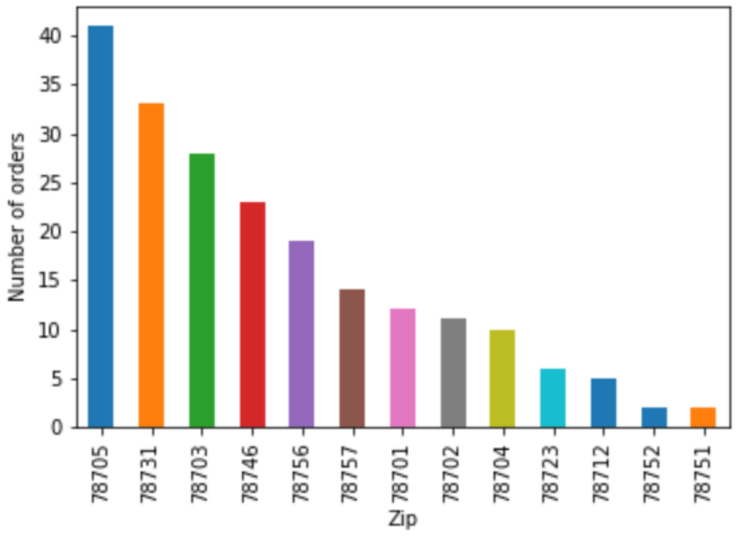
Analytics Code

Zip Code Analysis

```
pd.pivot_table(df, index = ["Zipcode"], values = ["Quantity","Amount"], aggfunc = np.sum).sort_values(by = 'Amount', as
```

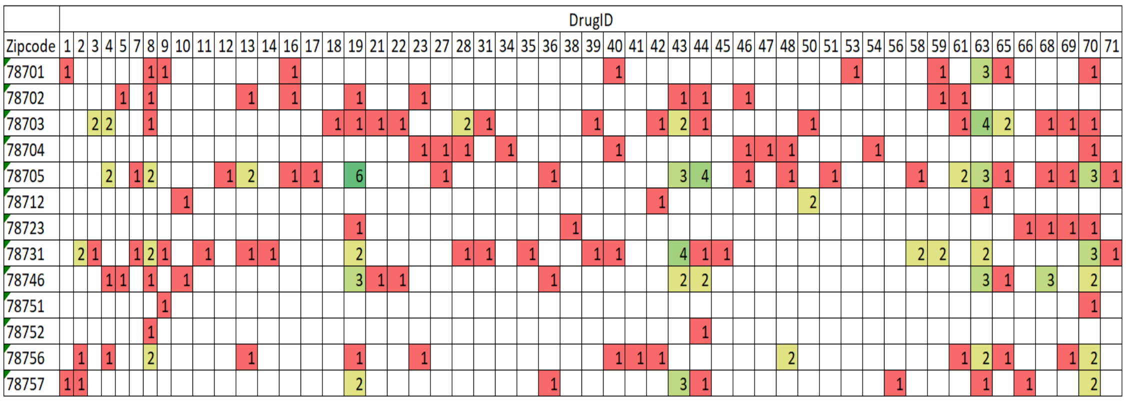
```
zip_counts = df['Zipcode'].value_counts()
zip_counts.plot(kind = 'bar')
plt.xlabel('Zip')
plt.ylabel('Number of orders')
plt.show()
```

Zipcode	Amount	Quantity
78705	130144	217
78731	109265	168
78746	72321	119
78703	63506	119
78756	50562	91
78757	40301	59
78702	38818	61
78701	36284	72
78704	22402	49
78723	15388	28
78712	14164	23
78752	10601	15
78751	7655	15



From the zip code analysis, we can see which zip codes are generating the highest sales for us. This could allow us to consider making promotions in certain areas.

Drug Sales Analysis



	DrugID																																																			
Zipcode	1	2	3	4	5	7	8	9	10	11	12	13	14	16	17	18	19	21	22	23	27	28	31	34	35	36	38	39	40	41	42	43	44	45	46	47	48	50	51	53	54	56	58	59	61	63	65	66	68	69	70	71
78701	1						1	1						1															1											1				1		3	1				1	
78702					1		1					1		1			1			1												1	1		1									1	1							
78703			2	2			1									1	1	1	1			2	1					1			1	2	1					1							1	4	2		1	1	1	
78704																				1	1	1		1					1						1	1	1				1										1	
78705				2		1	2				1	2		1	1		6				1					1						3	4		1		1		1				1		2	3	1		1	1	3	1
78712									1																						1							2								1						
78723																	1										1																					1	1	1	1	
78731		2	1			1	2	1		1		1	1				2					1	1		1			1	1			4	1	1									2	2		2					3	1
78746				1	1		1		1								3	1	1							1						2	2													3	1		3		2	
78751								1																																											1	
78752							1																										1																			
78756		1		1			2					1					1			1									1	1	1						2								1	2	1			1	2	
78757	1	1															2									1						3	1									1				1		1			2	

From the drug sales analysis, we see that overall, drug 63 has the highest sales across zip codes. This analysis will provide information for us to do a deeper dive into whether these differences in sales are significant. Was the discrepancy due to the nature of the drug (certain diseases are more rare, so certain drugs are prescribed less frequently)? If this is not the case, we can decide on whether certain drugs should be promoted more frequently to physicians. (This graph was generated using Excel).

Inventory Analysis

```
: df[['DRUGNAME','UNITS']].groupby('DRUGNAME')['UNITS'].sum().sort_values(ascending = False)
```

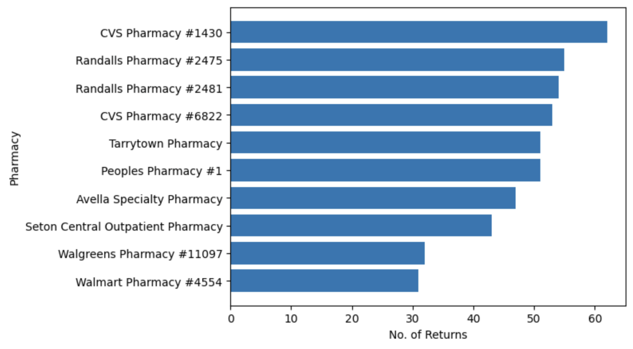
```
DRUGNAME
Simulect                    112
TOBI Podhaler                92
Trileptal                    90
Zolgensma                    88
Tegretol                     63
Travatan BAK-Free            62
Galvus                       59
Mayzent                      59
Comtan                       56
Beovu                        55
Zortress/Certican            55
Scemblix                     50
Arzerra                      46
Travatan Z                   46
Focalin XR                   45
Systane Ultra                44
Lescol                       41
Farydak                      41
Afinitor Disperz/Votubia     40
Ciprodex                     36
```

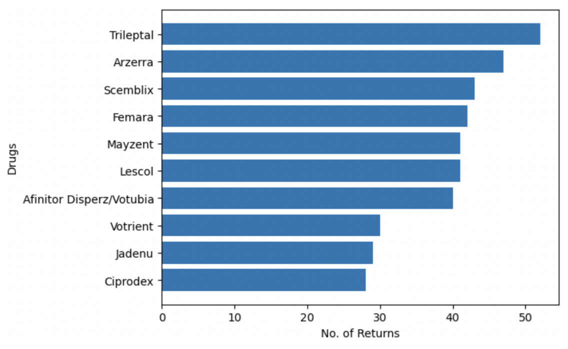
The inventory analysis tells us how much stock of a certain drug we have in hand at a given point in time. This allows us to manage our inventory effectively. If inventory for a drug is below a certain threshold, we need to work on restocking it as soon as possible. Another insight from this analysis is information on drugs that are overstocked. This is crucial because drugs are perishable products. We have to make sure that we are providing quality products to our customers. In the future, we can find ways to forecast our sales more accurately to build an efficient inventory system that meets the needs of our customers.

Drug Returns Analysis

```
: # top 10 product returns
  plt.barh(s_returns[-10:].index, s_returns[-10:])
  plt.ylabel('Drugs')
  plt.xlabel('No. of Returns')
  plt.show()
```

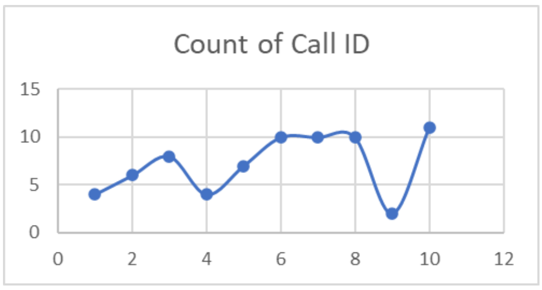
```
: # top 10 Pharmacy returns
  plt.barh(s_pharma_returns[-10:].index, s_pharma_returns[-10:])
  plt.ylabel('Pharmacy')
  plt.xlabel('No. of Returns')
  plt.show()
```

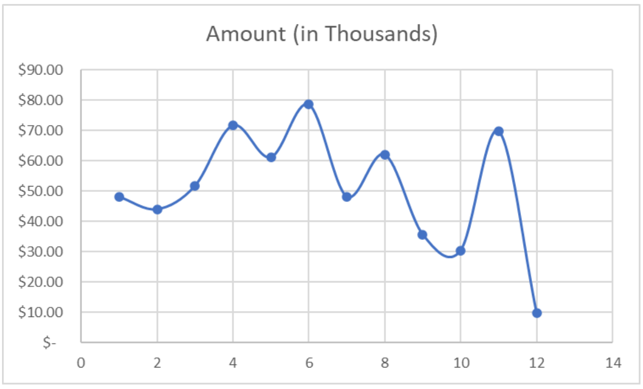




The drug returns analysis shows which pharmacy returns our products most frequently.. Using this, we can contact these pharmacies to figure out what the issues are with our drugs. Are the problems associated with drug quality, delivery speed, or customer service? We could also look at which drugs are returned most frequently and find out if there is a problem with our products. Maybe some of the drugs we shipped were expiring soon and returned to us. These insights would allow us to improve our customer relationships and hopefully boost our sales.

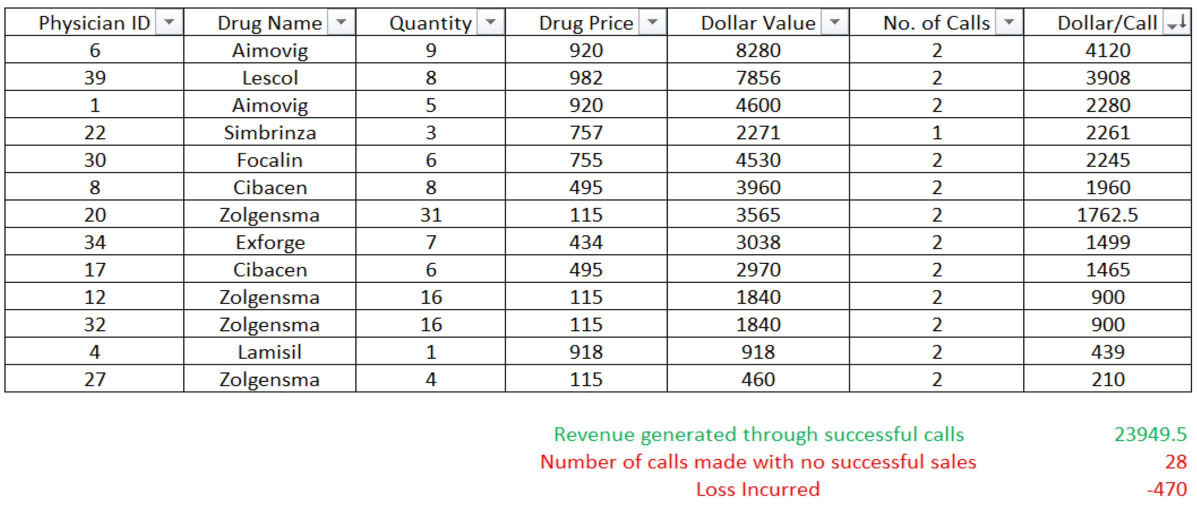
Promotion-Transaction Trends





The promotion transaction trends analysis shows the number of calls made for each month as well as the revenue for each month. The trends are behaving as we expected. If a higher number of promotional calls were made in the previous month, then we would expect to have higher sales this month. This analysis allows us to see if our promotional calls are effective. If promotional calls do not lead to higher sales, then we might have to re-evaluate our promotions. (The graphs for this analysis were generated using Excel.)

<u>Promotion Calls Profitability</u>



Physician ID	Drug Name	Quantity	Drug Price	Dollar Value	No. of Calls	Dollar/Call
6	Aimovig	9	920	8280	2	4120
39	Lescol	8	982	7856	2	3908
1	Aimovig	5	920	4600	2	2280
22	Simbrinza	3	757	2271	1	2261
30	Focalin	6	755	4530	2	2245
8	Cibacen	8	495	3960	2	1960
20	Zolgensma	31	115	3565	2	1762.5
34	Exforge	7	434	3038	2	1499
17	Cibacen	6	495	2970	2	1465
12	Zolgensma	16	115	1840	2	900
32	Zolgensma	16	115	1840	2	900
4	Lamisil	1	918	918	2	439
27	Zolgensma	4	115	460	2	210

Revenue generated through successful calls	23949.5
Number of calls made with no successful sales	28
Loss Incurred	-470

From our promotion calls profitability analysis, we can calculate the revenues and losses incurred from our promotional campaign. We see that in our dataset, we are able to generate about $23,480 of profit. However, our dataset is relatively small. When our database grows in the future, the losses incurred from unsuccessful calls/sales could be substantial if our campaign turns out to be ineffective. Closely monitoring customer response will help us develop profitable campaigns. It is also important to note that our analysis data does not include physician names for data privacy reasons. (This part of the analysis was done with Excel).

Conclusion

Pharmaceutical companies' data have many restrictions in the form of data privacy. It is important to know the intricacies behind these restrictions when building a database management system for a Pharmaceutical company. In this report, we have dealt with three out of many transaction applications a company might have. For example, in our sales transactions application we do not have Patient information like Patient name, address etc. But when a Pharma company conducts medical trials it will have Patients' information and it is essential that the companies' database management system is designed in such a way that there will be no leakage of data but the required analysis can be done on the trial data. A sophisticated data management system which supports the partly Offensive partly Defensive data management strategy of the company needs to be constructed and maintained.

Appendix

Division of Labor

Krish Engineer - Data Warehouse, Data Marts, Analytics
Monica Martinez - Analytics, Report and Presentation
Pratik Gawli - DDL code, Data Warehouse, Presentation
Sai Bhargav Tetali - Database model, DDL code, Data Marts
Vivian Wang - Analytics, Report and Presentation

Discussion of Topic Selection

Initially we had to choose between the Retail sector and Pharmaceutical sector. For the Retail sector we found a dataset on Kaggle whereas we did not find any dataset for the Pharmaceutical sector. Despite not having found any dataset we went with the Pharmaceutical sector as we thought it would be more challenging due to the mix of Offensive - Defensive strategy that we would have to think of for this sector. Also, it would make the Database modeling challenging which is the main learning objective of this course. Hence, we all unanimously decided to go with Pharmaceutical data.

Learnings:

We worked on Oracle SQL Developer for the most part of our project. We learned how to set up a database in SQL. We learned the importance of Data warehousing and making connections to the warehouse with the day to day transactions in order to keep it updated on a periodical basis. And lastly, we learned how to use the data to perform analysis while following data privacy restrictions. The most valuable lesson we learnt from this project is the importance of planning ahead in order to develop a database management system that satisfies all constraints. If not planned well in the beginning it is very difficult to go back and change the components of the system.