In the last decade the use of Electronic Medical Records (EMR) by American hospitals and physicians has grown dramatically. Interest in utilizing EMR data for clinical-outcomes research has kept pace with this growth. The objective of this project was two-fold: First, the team analyzed Electronic Medical Records data to measure the effectiveness of various anti-hypertensive medications in a complex and uncontrolled “real-world” environment. Second, we created a tool to measure the severity of discontinued prescriptions and compared this tool against data from the University of Utah’s “Drug Informer” database to compare findings from disparate data sets.
Equation Consulting – Background

Equation Consulting is a 45 person consulting firm based in Salt Lake City, UT with a core focus on data-driven solutions to improve physician economics within hospital, private, or academic settings. Our core service is helping hospitals and physicians understand and improve physician economics. Due to the Meaningful Use Initiative piece of the 2009 HITECH Act, there is an increased interest in incorporating EMR data into our solutions. Accordingly, Equation is seeking to increase their exposure to and understanding of clinical EMR data.

Project Description

We will analyze the outcomes of different anti-hypertensive medications using EMR data from Equation’s Client Data Repository. Analysis will introduce several statistical tools not currently used at Equation and also compare results against data from the University of Utah’s recently created DrugInformer.com website. The project will also include privacy protection education and analysis to help Equation determine how to simultaneously preserve data integrity and patient privacy.

Project Goals

- Understand what information is available and discover possible applications in future products and projects that could be offered to Equation’s clients.
- “Measuring effectiveness of anti-hypertension medication” chosen as first-look analysis
- Protect privacy by determining how to create meaningful datasets for research that meet HIPAA requirements for “de-identified” data sets

New Skills and Learning

- Improve understanding of statistical techniques in data analysis including: t-test, regression, and data-mining methods
- Gain understanding of clinical data contained within EMR data.
- Learn privacy protection laws and understand “Safe Harbor” regulations for determining to what extent data must be limited to be safely exempt from HIPPA regulation.

Introduction to Key Concepts

- EMR
- Patient Privacy
- Hypertension
**Introduction to EMR**

Electronic Medical Records (EMR), also referred to as Electronic Health Records (EHR), are computerized medical record in a hospital or physician’s office that take the place of the traditional paper “chart”. EMR systems typically contain patient demographics and vitals, medical history, lab results, and sometimes integrate X-ray and other images as well. Major benefits over paper charts include improving efficiency and accuracy of data sharing, decreased cost of storing cumbersome paper files, and opportunities to prompt physicians with automated warnings prior to performing orders.

Long before Obama-care, Congress signed the Health Information Technology for Economic and Clinical Health Act of 2009, also known as the HITECH Act. A key piece of this legislation was the “Meaningful Use Initiative”, which created $34 billion in financial incentives for hospitals and physicians who can demonstrate meaningful use of certified electronic health records (EHRs). This law also includes substantial payment reductions if they are not meaningful users of health IT after 2015.

As a result of the Meaningful Use Initiative, EMR adoption rates have increased dramatically and as of late 2011 adoption rates range from 39% for solo practitioners to 77% for large multi-specialty practices. Another important piece of the HITECH act was a universal minimum standard required of all EMR systems in order to qualify for the Meaningful Use incentive. Not only has the adoption of EMR data increased, but the richness of this data has also substantially improved.

**Introduction to Patient Privacy**

With the increased use of EMR, there is greater need to be vigilant in protecting patient privacy as highly sensitive patient information being captured and stored electronically is increasing. Several historical high-profile blunders in failed anonymization have resulted in justifiably strict rules to certify data as “de-identified”. The “Safe Harbor” outlined by HIPPA for creating a “de-identified” dataset that by definition has no Personal Health Information (PHI) requires the removal of all of the following 18 elements:

1. Names.
2. All geographic subdivisions smaller than a state, including street address, city, county, precinct, ZIP Code, and their equivalent geographical codes, except for the initial three digits of a ZIP Code
3. All elements of dates (except year) for dates directly related to an individual, including birth date, admission date, discharge date, date of death and all ages over 89 except that such ages are aggregated into a single category of age 90 or older.
4. Telephone numbers.
5. Facsimile numbers.
6. Electronic mail addresses.
7. Social security numbers.
8. Medical record numbers.
9. Health plan beneficiary numbers.
10. Account numbers.
12. Vehicle identifiers and serial numbers, including license plate numbers.
15. Internet protocol (IP) address numbers.
16. Biometric identifiers, including fingerprints and voiceprints.
17. Full-face photographic images and any comparable images.
18. Any other unique identifying number, characteristic, or code, unless otherwise permitted by the Privacy Rule for re-identification.

Only if a dataset is classified as “de-identified” is it free from the requirements of HIPPA regulations designed to protect such data. Understandably, a result of the de-identification process is that there is substantial loss of data richness and integrity for research purposes. It is the goal of Equation to determine how to generate meaningful data extracts that maintain this data integrity, but while still following the Safe-Harbor requirement for de-identification.

Introduction to Hypertension

Hypertension, specifically the effectiveness of anti-hypertensive medication, was chosen as a first-look analysis for EMR data. From a data perspective hypertension was a good choice because:

- Blood pressure is a very basic and almost universal stored in EMR data
- Hypertension is defined by a specific set of diagnosis codes that can be used to easily identify the “universe” of hypertensive patients
- Data collection by hospitals and physician practices doesn’t require additional work or a behavior change by physicians, but instead requires only on medical staff involvement.

Clinical support for choosing hypertension as a research metric abound. For example, the Department of Health & Human Services’ Agency for Healthcare Research and Quality reported that: “...Even small improvements in blood pressure control can have major public health impact. A 1990 systematic review of 14 randomized treatment trials for hypertensive patients showed that lowering diastolic blood pressure (DBP) by 5 to 6 points reduced stroke rates by 42%. Another recent study showed that lowering DBP by only 2 points could result in a 6% reduction in the risk of coronary heart disease, along with a 15% reduction in the risk of stroke and one type of heart attack...”[4]

For reference, Table 1 displays the current clinical categorization of hypertension by blood pressure measurement. Most analysis performed focused only on patients who fit into the categories of Hypertension Stages 1 & 2.

Table 1

<table>
<thead>
<tr>
<th>CATEGORY</th>
<th>Systolic</th>
<th>Diastolic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hypotension</td>
<td>&lt;90</td>
<td>&lt;60</td>
</tr>
<tr>
<td>Normal</td>
<td>90-119</td>
<td>60-79</td>
</tr>
<tr>
<td>Prehypertension</td>
<td>120-139</td>
<td>80-89</td>
</tr>
<tr>
<td>High Blood Pressure (Hypertension) Stage 1</td>
<td>140-159</td>
<td>90-99</td>
</tr>
<tr>
<td>High Blood Pressure (Hypertension) Stage 2</td>
<td>160-180</td>
<td>100-110</td>
</tr>
<tr>
<td>Hypertensive Crisis (Emergency care needed)</td>
<td>&gt;180</td>
<td>&gt;110</td>
</tr>
</tbody>
</table>
The Data

All data used in our analysis comes from Equation’s Client Data Repository, and was only accessible to Nathan Manwaring as a full-time employee of Equation Consulting. Patient information (de-identified or not) was housed exclusively on Equation’s servers and was not shared with any third party, including the University of Utah, its students, or any employees.

The EMR data used in this analysis is from a popular EMR data warehouse with data spread over 149 unique tables and more than 14,000 columns. Items included in the analysis include: Patient Demographics, Medications, Diagnosis Codes, Procedure Codes, Provider Information, Location and Department information, Dates of Service, Medication Orders, etc.

Data Processing

Data was housed in a SQL Server environment that was also the primary method of data aggregation, scrubbing, and analysis. Before the data could be analyzed we had to pull the appropriate metrics and measures from the EMR data repository and format it in a way that was conducive to the analysis that we wanted to perform on the data.

The first goal was to generate a clean list of patients that would be the starting universe for potential analysis. Criteria for meeting this requirement is an encounter with a primary diagnosis of hypertension.

The second goal was to generate a dataset that would tie together various medications, encounters, BP readings, and physician encounters in an ordered way that allowed for comparing consecutive items on a single line.

The following steps were taken:

SQL Data Manipulation

1. Pull All Orders for Patients with an encounter having PrimaryDX of Hypertension (grain = prescription ID) (table = Order Summary)
   a. Exclude Test Patients
   b. Exclude Cancelled Orders
   c. Exclude Historical Orders
2. Rank all Patient Encounters chronologically by service date for each patient
   a. Exclude Encounters where Blood Pressure is not recorded
3. Populate Order Summary with the encounter ID for the encounter closest to the medication date, and the 4 previous and following encounters with BP measured
4. Populate Order Summary with provider information and clinical metrics from each of the encounters in step 3
5. Measure Change in Diastolic BP following medication orders
6. Summarize multiple orders given on the same date into a single line-item (grain = patient+order date), include all medications in pipe delimited medication field (table = Patient Summary)
7. Populate Patient Summary with information on each of the orders
8. Determine primary authorizing provider for each Order Date
9. Use dynamic SQL to create binary columns for top 50 medications (for this population) and populate those columns to indicate a medication is part of a particular order date.
10. Populate Patient Summary with current medication list information from encounter table.
11. Populate Patient Summary with information on medication orders that could potentially interfere with impact from a particular order, exclude orders without a clean before/after reading.
12. Set Parameters for acceptable timeline between BP readings and prescription writing. Second Reading between 10 and 60 days, first reading within 10 days previous to the prescription.
13. Create BP categorization using industry standards for BP measurement.
14. Clean BP measurements that we incorrect numbers (BP readings clearly too high or too small), create identifier for outliers.
15. Populate Patient Summary with BP category information.
16. Final Research Population: Exclude all BP outliers, and only include patient orders that have no orders in-between previous and next BP measurement.
17. Populate Patient Summary with Primary Diagnosis code from Encounter table, both for current and next encounter following medication order.
18. Create limited Patient Summary table with only information on 4 core drugs: Lisinopril, Hydrochlorothiazide, Lis-Hctz (combo), and Furosemide.

See Appendix A for the complete set of SQL code used to generate the table used in final analysis.

High Level Review of Data

The initial pull of patients was limited to those with a documented primary diagnosis of hypertension. Total hypertensive patients totaled 28,178. We reviewed 29 demographic data points to understand the population and the robustness of the data: median patient age: 60, age range: 40-87, gender as 57% female, median weight: 190lbs, average blood pressure measurements per patient: 7.

The initial review of medication revealed 6,818 unique drugs prescribed within the data set. Of the 1.9 million prescriptions recorded in the medication data set, there were 30,590 orders that were populated with a “Discontinued Reason” code that we considered potential adverse events.

For the primary research population of hypertensive patients, the top 50 drugs were prescribed a total of 19,151 times over the course of 5 years. Of these drugs, the 4 most common drugs prescribed with antihypertensive properties were chosen as the focus of our analysis. They are:

1. Lisinopril
2. Hydrochlorothiazide (HCTZ)
3. Lisinopril-Hydrochlorothiazide combination drug
4. Furosemide

Note: ultimately drugs 3 & 4 were excluded from analysis as further review determined that the intended patient demographic was sufficiently unique that it wasn’t meaningful to compare their impact, however Lisinopril and HCTZ appeared to have identical demographic distribution and were thus chosen as the key comparison drugs.
Analysis: Multiple Linear Regression

The first method used to analyze the effectiveness of antihypertensive medication was Multiple Linear Regression. Linear regression is a modeling approach that determines the closest linear relationship between one or more independent variables and a single dependent variable.\(^5\) In linear regression we want the coefficients to show variation between the different independent variables. We also want to see a very low P-value (.05) for most or all of the variables analyzed. The P-value represents the % chance that the impact of the variable described by the coefficient is due simply to chance.

Regression modeling is facilitated through binary columns that indicate the presence of a particular independent variable (see Table 2 below)

Table 2

<table>
<thead>
<tr>
<th>Patient ID</th>
<th>Medication ID</th>
<th>Levothyroxine</th>
<th>Atenolol</th>
<th>Metoprolol</th>
<th>Glibenclamide</th>
<th>Furosemide</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

We initially attempted to model the relationship between many (14) medications and their impact on blood pressure. While this method did successfully identify several drugs known to increase/decrease BP (see Table 3), it also determined that the effect of the antihypertensive drug “Furosemide” was actually increasing diastolic BP. Additional data scrubbing would show that Furosemide is not associated with increased BP.

Table 3

<table>
<thead>
<tr>
<th>Drug</th>
<th>Impact on Diastolic BP</th>
<th>Coefficients</th>
<th>Standard Error</th>
<th>t Stat</th>
<th>P Value</th>
<th>Lower 95%</th>
<th>Upper 95%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Furosemide</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Levothyroxine</td>
<td></td>
<td>0.47</td>
<td>0.12</td>
<td>3.92</td>
<td>0.0001</td>
<td>0.23</td>
<td>0.84</td>
</tr>
<tr>
<td>Metoprolol</td>
<td></td>
<td>0.34</td>
<td>0.11</td>
<td>3.10</td>
<td>0.0023</td>
<td>0.19</td>
<td>0.64</td>
</tr>
<tr>
<td>Atenolol</td>
<td></td>
<td>0.36</td>
<td>0.12</td>
<td>3.01</td>
<td>0.0044</td>
<td>0.17</td>
<td>0.66</td>
</tr>
<tr>
<td>Glibenclamide</td>
<td></td>
<td>0.39</td>
<td>0.11</td>
<td>3.62</td>
<td>0.0003</td>
<td>0.26</td>
<td>0.65</td>
</tr>
<tr>
<td>Levothyroxine</td>
<td></td>
<td>0.47</td>
<td>0.12</td>
<td>3.92</td>
<td>0.0001</td>
<td>0.23</td>
<td>0.84</td>
</tr>
<tr>
<td>Metoprolol</td>
<td></td>
<td>0.34</td>
<td>0.11</td>
<td>3.10</td>
<td>0.0023</td>
<td>0.19</td>
<td>0.64</td>
</tr>
<tr>
<td>Atenolol</td>
<td></td>
<td>0.36</td>
<td>0.12</td>
<td>3.01</td>
<td>0.0044</td>
<td>0.17</td>
<td>0.66</td>
</tr>
<tr>
<td>Glibenclamide</td>
<td></td>
<td>0.39</td>
<td>0.11</td>
<td>3.62</td>
<td>0.0003</td>
<td>0.26</td>
<td>0.65</td>
</tr>
</tbody>
</table>

\(\text{Furosemide}\)
Conclusion

We knew that the linear regression model was likely subjected to too many variables to build a strong model, however we did find that in general, linear regression was able to identify those drugs which had a positive, or negative, effect on hypertension. Specifically, the regression model easily found that Lisinopril and Hydrochlorothiazide or the combination of the two were the most effective drugs at lowering blood pressure. In all 3 cases the P-value for these drugs was very low (<.01). However, the model also identified Furosemide as a drug with a negative impact on high blood pressure. This was error due to the inclusion of very many variables as the negative relationship disappeared once we separated the consideration of Furosemide alone and when used in tandem with other medication. While we gain some useful information from linear regression modeling, it does not appear to be the most appropriate modeling method for this dataset.

Analysis: Data Mining - Classification

The second method used to analyze the data was data mining through classification. For this analysis our software of choice was Weka 3, an open source data mining tool maintained by the University of Waikato. Classification uses data mining techniques: Decision Tree & Neural Networks, and attempts to classify an object into sets of pre-defined object classes for example:

- Based on specific demographics can we classify patients as sick or not
- Based on customer attributes classify customers as will or won’t purchase

Two of the main goals of classification are to a) discover relationships between independent and dependent variables not previously known and b) better understand and quantify already known relationships between variables.

We tried Data Mining modeling – J48 decision tree and NaïveBays to try to determine which patients would experience a favorable drop in their Category of Hypertension (e.g. Type 2 -> Type 1, or Pre-hypertensive -> Normal). The potential independent variables given were: Gender, Age, HypertensionDX_mrkr, DiagnosisGroup, DaysFromOrder, Weight, BMI, Temp, Pulse, Diastolic Category, various medication markers. The J48 decision tree was the most accurate with 65% of patients correctly predicted as having or not having a favorable drop in hypertensive BP category. Precision (% of patients predicted to drop that actually did) and Recall (% of patients that actually dropped that were predicted to do so by the decision tree) were both at %66 and %65 respectively. (see Table 4)
Conclusion

Our initial model was able to accurately predict 65% of all patients as either having or not having a beneficial drop in DBP category. However, since roughly half of all patients experienced this drop in BP category, an accuracy of 65% is only 15% better than if someone were to assign each patient into a category through random selection. When performing classification analysis we seek to maximize the accuracy of the model, but we also need to minimize decisions along the decision tree in order to preserve sample size for each “decision”. This is clearly an area to continue exploring in the future, however the initial result shows too many decision points with too little data to be useful (see Table 5). Needs additional review.

Analysis: Data Mining – Classification Try #2

For our second try modeling with the J48 decision tree, we modified the minimum objects per leaf to 10 patients, and changed to 66% training set rather than 10-fold. The same variables were given, and the same data was used. The J48 decision tree was still correct 65% of the time, but this time with only 5 leaves (decision points) compared to the 48+ that were used in the first model. (see Table 6 and Table 7)
An interesting, and unexplained, phenomenon in this data set was that the J48 decision tree identified starting BP and age as the only necessary decision points. We originally would have guessed that weight, BMI, or other vital metrics would play a more obvious role in BP outcomes. We note that the accuracy of the decision tree is still only 65%, and that further investigation is necessary to better understand the initial findings of the classification analysis.

**Analysis: Deep-Dive Comparison of Lisinopril and Hydrochlorothiazide**

Preliminary analysis shows comparable results from LIS and HCTZ, especially among Category 2-3 combined populations. LIS has slightly better performance overall. (See Table 8)
The T-test is commonly used to calculate the significance of observed differences between the means of two samples. T-test result is the percent probability of the null hypothesis, that there is no significant difference between the means of the two populations. Using the T-test on two identical categories of patients who were prescribed one of the two drugs we were able to determine that there was a 48% probability that there was no difference between the outcomes achieved by the two drugs for that population. (see Table 9)

### Table 9

<table>
<thead>
<tr>
<th>Lisinopril Dataset</th>
<th>Hydrochlorothiazide Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean (10.9)</td>
<td>Mean (9.9)</td>
</tr>
<tr>
<td>Median (10.3)</td>
<td>Median (10.3)</td>
</tr>
<tr>
<td>Std Dev 9.4</td>
<td>Std Dev 7.9</td>
</tr>
</tbody>
</table>

**T-Test** 48%

### Comparing EMR Analysis Findings to Industry Research

Two studies were found comparing effectiveness of LIS vs HCTZ at lowering blood pressure[1,2]. Both studies found that LIS was 6-7 points more effective than HCTZ. However, our EMR Analysis shows only
2 points of difference between the impacts of the two drugs (see Table 10). We determined that we would attempt to identify the potential causes of difference between EMR findings and research.

**Table 10**

<table>
<thead>
<tr>
<th></th>
<th>HCTZ Dias Drop</th>
<th>LIS Dias Drop</th>
<th>LIS vs HCTZ</th>
<th>Total Patients</th>
</tr>
</thead>
<tbody>
<tr>
<td>Study1</td>
<td>(7.0)</td>
<td>(14.0)</td>
<td>(7.0)</td>
<td>16</td>
</tr>
<tr>
<td>Study2</td>
<td>(6.8)</td>
<td>(12.5)</td>
<td>(5.7)</td>
<td>394</td>
</tr>
<tr>
<td>Study Average</td>
<td>(6.8)</td>
<td>(12.6)</td>
<td>(5.6)</td>
<td>410</td>
</tr>
</tbody>
</table>

**Differences in Demographic Distributions**

Comparison of the patient population to the study population revealed that the study’s age distribution was significantly under representing the “>64” category compared to the actual patient population. The patient population gender was also significantly different from the study gender distribution. The study was 75% male whereas the actual patient population was more than 65% female. Both of these differences are especially meaningful if it can be shown that the effectiveness of the medications is influenced by either of these metrics: Age or Gender.

**Conclusion: The Impact of Different Gender Distributions is Important**

Splitting the drug comparison by gender reveals that Males have less favorable outcomes from HCTZ than females. T-test results suggest a high degree of similarity for females using the medications, but not for Males. If EMR data had similar demographics to the Study, the results of the comparison would be more comparable. (see Table 11)
Conclusion: Real-World vs. Research

EMR Research is not to replace studies, nor is it competing with them. EMR tells the physician a different story. Since clinical trials use highly controlled environments, they aren’t necessarily “in touch” with reality in the conclusions that they deliver.

EMR analysis, on the other hand, answers for a particular provider: “What actually happens to my patient’s BP when I prescribe this medication?” While there could be a number of factors influencing patient BP outcomes, the actual impact of the medication, measured by change in BP, allows the physician to know if the drug appears to work for his or her patients as it is currently being prescribed.

Future Research: Clinical Questions

In addition to blood pressure, clinical areas that should be studied include the more critical and more popular research items:

1. What are predictors of Heart Attack, Stroke, Diabetes, etc.?
2. What medications seem to have the best track record for reducing Heart Attack Risk?
3. What physicians seem to be the most efficient providers? (best outcomes relative to patient cost)

Future Research: Social Media and Adverse Events Integration

One possible factor influencing the comparison of Lis and HCTZ in large populations of patients is the frequency of side-effects influencing whether patients are unable or unwilling to finish the prescription due to side effects. Below we see Losartan listed with a high rate of patients reporting “coughing” as a negative side-effect of the drug. [3]
EMR data captures information when a medication is discontinued due to dose adjustment, patient preference, allergic reactions, etc. By analyzing this data, we can start to compare the picture to findings from the DrugInformer.com database. The user of the tool simply sets the weighting criteria and then the excel sheet will display the “top ranked” drugs by severity based on the weighting chosen by the user.

### Protecting Privacy and Data Integrity

After the initial SQL data scrubbing, we determined that none of the 18 “Safe Harbor” fields were required to complete the analysis, and that little value is lost by removing dates. Furthermore, it is...
possible to construct a data extraction script that creates the Summary Analysis tables, but retained order of and time in between encounters and measurements without including any actual dates. Hospitals or physician practices would control the execution of the extract and would then certify that the end-result was free form PHI before releasing to researchers.

By constructing special data extraction tools for clients, it’s possible to both guarantee patient privacy and achieve rich data sets with high research value. Conversely, unless a specific query is created for a particular data set, it is unlikely that a hospital would be able to provide PHI-free data that retained this information.


APPENDIX A – SQL Code Developed
---01 Create MedicationOrder Table

IF OBJECT_ID('MedicationOrder') <> 0 DROP TABLE MedicationOrder
GO
CREATE TABLE MedicationOrder(
    Order_ID VARCHAR(255) PRIMARY KEY,
    PatientID VARCHAR(255) NOT NULL,
    Order_Date DATE,
    Order_DateUpdate DATE,
    Order_DateStart DATE,
    Order_DateEnd DATE,
    Order_PrescriptionProvider VARCHAR(255),
    Order_AuthorizingProvider VARCHAR(255),
    Order_MedicationID VARCHAR(255),
    Order_MedicationDescription VARCHAR(255),
    Order_MedicationSimpleGenericCode VARCHAR(255),
    Order_MedicationSimpleGenericName VARCHAR(255),
    Order_MedicationForm VARCHAR(255),
    Order_MedicationRoute VARCHAR(255),
    Order_MedicationLastDose VARCHAR(255),
    Order_MedicationPharmClass VARCHAR(255),
    Order_MedicationTheraclass VARCHAR(255),
    Order_BP_EncounterCurr00_Rank INT,
    Order_BP_EncounterPrev01_CSN_ID VARCHAR(255),
    Order_BP_EncounterPrev01_DepartmentID VARCHAR(255),
    Order_BP_EncounterPrev01_PCPProvID VARCHAR(255),
    Order_BP_EncounterPrev01_Pulse MONEY,
    Order_BP_EncounterPrev01_Temp MONEY,
    Order_BP_EncounterPrev01_Weight MONEY,
    Order_BP_EncounterPrev01_Systolic MONEY,
    Order_BP_EncounterPrev01_DaysFromOrder MONEY,
    Order_BP_EncounterPrev01_DateService DATE,
    Order_BP_EncounterPrev02_CSN_ID VARCHAR(255),
    Order_BP_EncounterPrev02_DepartmentID VARCHAR(255),
    Order_BP_EncounterPrev02_PCPProvID VARCHAR(255),
    Order_BP_EncounterPrev02_VisitProvID VARCHAR(255),
    Order_BP_EncounterPrev02_Pulse MONEY,
    Order_BP_EncounterPrev02_Temp MONEY,
    Order_BP_EncounterPrev02_BMI MONEY,
    Order_BP_EncounterPrev02_Weight MONEY,
    Order_BP_EncounterPrev02_Systolic MONEY,
    Order_BP_EncounterPrev02_Diastolic MONEY,
    Order_BP_EncounterPrev02_Diastolic MONEY,
    Order_BP_EncounterPrev03_CSN_ID VARCHAR(255),
    Order_BP_EncounterPrev03_DepartmentID VARCHAR(255),
    Order_BP_EncounterPrev03_PCPProvID VARCHAR(255),
    Order_BP_EncounterPrev03_Pulse MONEY,
    Order_BP_EncounterPrev03_Temp MONEY,
    Order_BP_EncounterPrev03_BMI MONEY,
    Order_BP_EncounterPrev03_Weight MONEY,
    Order_BP_EncounterPrev03_Systolic MONEY,
    Order_BP_EncounterPrev03_DaysFromOrder MONEY,
    Order_BP_EncounterPrev03_DateService DATE,
    Order_BP_EncounterPrev04_CSN_ID VARCHAR(255),
    Order_BP_EncounterPrev04_DepartmentID VARCHAR(255),
    Order_BP_EncounterPrev04_PCPProvID VARCHAR(255),
    Order_BP_EncounterPrev04_VisitProvID VARCHAR(255),
    Order_BP_EncounterPrev04_Pulse MONEY,
    Order_BP_EncounterPrev04_Temp MONEY,
    Order_BP_EncounterPrev04_BMI MONEY,
    Order_BP_EncounterPrev04_Weight MONEY,
    Order_BP_EncounterPrev04_Systolic MONEY,
    Order_BP_EncounterPrev04_Diastolic MONEY,
    Order_BP_EncounterPrev04_DaysFromOrder MONEY,
    Order_BP_EncounterPrev04_DateService DATE,
    Order_BP_EncounterPrev04_VisitProvID VARCHAR(255),
    Order_BP_EncounterPrev04_DepartmentID VARCHAR(255),
    Order_BP_EncounterPrev04_PCPProvID VARCHAR(255),
    Order_BP_EncounterPrev04_Pulse MONEY,
    Order_BP_EncounterPrev04_Temp MONEY,
    Order_BP_EncounterPrev04_BMI MONEY,
    Order_BP_EncounterPrev04_Weight MONEY,
INSERT INTO MedicationOrder(
  Order_ID,
  PatientID,
  Order_Date,
  Order_DateUpdate,
  Order_DateStart,
  Order_DateEnd,
  Order_PrescriptionProvider,
  Order_AuthorizingProvider,
  Order_MedicationID,
  Order_MedicationDescription,
  Order_MedicationSimpleGenericCode,
  Order_MedicationSimpleGenericName,
  Order_MedicationForm,
  Order_MedicationRoute,
  Order_MedicationLastDose
)
GO
CREATE
GO
ORDER
GROUP
FROM
SELECT
IF
-------------------
----------------------------------------------------------------------------------------------------------------
--
----------------------------------------------------------------------------------------------------------------
GO
PAT_NAME
WHERE
FROM
)
END
ORDER
PARTITION
PAT_ENC
PAT_ENC
BP_DIASTOLIC
WHERE
db_0175_08_ODS
INNER
db_0
(db_0175_08_ODS)
LEFT
db_0175_08_ODS
LEFT
db_0175_08_ODS
LEFT
db_0175_08_ODS
WHERE
a.PAT_ID IN(SELECT DISTINCT PAT_ID FROM db_0175_08_ODS..Hypertension_Patient_List)
AND
a.PAT_ID NOT IN(SELECT DISTINCT PAT_ID FROM db_0175_08_ODS..PATIENT WHERE PAT_NAME LIKE '%ZZ%' AND PAT_NAME LIKE '%TEST%')
AND
a.ORDER_STATUS_C NOT IN('1','4','12')
AND
a.ORDER_CLASS_C NOT IN('3','4')
GO
--------------------------------------------------------------------------------------------------------------------------
--03 Create TEMP_BP_Encounter_Rank to Rank all BP Encounters for Each Patient
--------------------------------------------------------------------------------------------------------------------------
IF OBJECT_ID('TEMP_BP_Encounter_Rank') <> 0 DROP TABLE TEMP_BP_Encounter_Rank
SELECT
PAT_ENC.PAT_ID
,CONTACT_DATE
,MAX(PAT_ENC_CSN_ID) AS PAT_ENC_CSN_ID
,row_number () OVER(
PARTITION BY
PAT_ENC.PAT_ID
ORDER BY
CONTACT_DATE
) AS BP_Visit_Rank
INTO TEMP_BP_Encounter_Rank
FROM
db_0175_08_ODS..PAT_ENC
INNER JOIN
(db_0175_08_ODS..Hypertension_Patient_List)
ON PAT_ENC.PAT_ID = Hypertension_Patient_List.PAT_ID
WHERE
BP_DIASTOLIC > 0
GROUP BY
PAT_ENC.PAT_ID
,CONTACT_DATE
ORDER BY
PAT_ENC.PAT_ID, CONTACT_DATE
GO
CREATE UNIQUE INDEX ix_PAT_ID_BP_Visit_Rank ON TEMP_BP_Encounter_Rank (PAT_ID, BP_VISIT_RANK)
GO
--------------------------------------------------------------------------------------------------------------------------
UPDATE MedicationOrder
SET Order_BP_EncounterCurr00_Rank = c.MinRank
FROM MedicationOrder
INNER JOIN
  ( SELECT a.Order_ID, MIN(b.BP_Visit_Rank) AS MinRank
    FROM MedicationOrder a
    INNER JOIN TEMP_BP_Encounter_Rank b
    ON a.PatientID = b.PAT_ID AND a.Order_Date BETWEEN DATEADD(day,-3,b.CONTACT_DATE) AND DATEADD(day,3,b.CONTACT_DATE)
    GROUP BY a.Order_ID ) c
ON MedicationOrder.Order_ID = c.Order_ID
GO

UPDATE MedicationOrder
SET Order_BP_EncounterCurr00_Rank = c.MinRank-.5
FROM MedicationOrder
INNER JOIN
  ( SELECT a.Order_ID, MIN(b.BP_Visit_Rank) AS MinRank
    FROM MedicationOrder a
    INNER JOIN TEMP_BP_Encounter_Rank b
    ON a.PatientID = b.PAT_ID AND a.Order_Date < DATEADD(day,-3,b.CONTACT_DATE)
    GROUP BY a.Order_ID ) c
ON MedicationOrder.Order_ID = c.Order_ID
WHERE Order_BP_EncounterCurr00_Rank IS NULL
GO

UPDATE MedicationOrder
SET Order_BP_EncounterCurr00_Rank = c.MaxRank + 1
FROM MedicationOrder
INNER JOIN
  ( SELECT a.Order_ID, MAX(b.BP_Visit_Rank) AS MaxRank
    FROM MedicationOrder a
    INNER JOIN TEMP_BP_Encounter_Rank b
    ON a.PatientID = b.PAT_ID
    GROUP BY a.Order_ID ) c
ON MedicationOrder.Order_ID = c.Order_ID
WHERE Order_BP_EncounterCurr00_Rank IS NULL
GO

--04 Update all Previous and Next CSN_IDs Where Possible
-----------------------------------------------
UPDATE MedicationOrder
SET
  Order_BP_EncounterPrev04_DateService = b.CONTACT_DATE,
  Order_BP_EncounterPrev04_DaysFromOrder = DATEDIFF(day, a.Order_Date, b.CONTACT_DATE),
  Order_BP_EncounterPrev04_Diastolic = b.BP_DIASTOLIC,
  Order_BP_EncounterPrev04_Systolic = b.BP_SYSTOLIC,
  Order_BP_EncounterPrev04_Weight = CASE WHEN CAST(b.WEIGHT AS MONEY) > 0 THEN CAST(b.WEIGHT AS MONEY)/16 ELSE NULL END,
  Order_BP_EncounterPrev04_BMI = CASE WHEN CAST(b.BMI AS MONEY) > 0 THEN CAST(b.BMI AS MONEY) ELSE NULL END,
FROM MedicationOrder a
INNER JOIN db_0175_08_ODS..PAT_ENC b ON a.Order_BP_EncounterPrev04_CSN_ID = b.PAT_ENC_CSN_ID
GO
UPDATE MedicationOrder
SET
  Order_BP_EncounterPrev03_DateService = b.CONTACT_DATE,
  Order_BP_EncounterPrev03_DaysFromOrder = DATEDIFF(day, a.Order_Date, b.CONTACT_DATE),
  Order_BP_EncounterPrev03_Diastolic = b.BP_DIASTOLIC,
  Order_BP_EncounterPrev03_Systolic = b.BP_SYSTOLIC,
  Order_BP_EncounterPrev03_Weight = CASE WHEN CAST(b.WEIGHT AS MONEY) > 0 THEN CAST(b.WEIGHT AS MONEY)/16 ELSE NULL END,
  Order_BP_EncounterPrev03_BMI = CASE WHEN CAST(b.BMI AS MONEY) > 0 THEN CAST(b.BMI AS MONEY) ELSE NULL END,
FROM MedicationOrder a
INNER JOIN db_0175_08_ODS..PAT_ENC b ON a.Order_BP_EncounterPrev03_CSN_ID = b.PAT_ENC_CSN_ID
GO
UPDATE MedicationOrder
SET Order_BP_EncounterPrev02_DateService = b.CONTACT_DATE,
    Order_BP_EncounterPrev02_DaysFromOrder = DATEDIFF(day, a.Order_Date, b.CONTACT_DATE),
    Order_BP_EncounterPrev02_Diastolic = b.BP_DIASTOLIC,
    Order_BP_EncounterPrev02_Systolic = b.BP_SYSSTOLIC,
    Order_BP_EncounterPrev02_Weight = CASE WHEN CAST(b.WEIGHT AS MONEY) > 0 THEN CAST(b.WEIGHT AS MONEY)/16 ELSE NULL END,
    Order_BP_EncounterPrev02_BMI = CASE WHEN CAST(b.BMI AS MONEY) > 0 THEN CAST(b.BMI AS MONEY) ELSE NULL END,
FROM MedicationOrder a
INNER JOIN db_0175_08_ODS..PAT_ENC b ON a.Order_BP_EncounterPrev02_CSN_ID = b.PAT_ENC_CSN_ID
GO
UPDATE MedicationOrder
SET Order_BP_EncounterPrev01_DateService = b.CONTACT_DATE,
    Order_BP_EncounterPrev01_DaysFromOrder = DATEDIFF(day, a.Order_Date, b.CONTACT_DATE),
    Order_BP_EncounterPrev01_Diastolic = b.BP_DIASTOLIC,
    Order_BP_EncounterPrev01_Systolic = b.BP_SYSSTOLIC,
    Order_BP_EncounterPrev01_Weight = CASE WHEN CAST(b.WEIGHT AS MONEY) > 0 THEN CAST(b.WEIGHT AS MONEY)/16 ELSE NULL END,
    Order_BP_EncounterPrev01_BMI = CASE WHEN CAST(b.BMI AS MONEY) > 0 THEN CAST(b.BMI AS MONEY) ELSE NULL END,
FROM MedicationOrder a
INNER JOIN db_0175_08_ODS..PAT_ENC b ON a.Order_BP_EncounterPrev01_CSN_ID = b.PAT_ENC_CSN_ID
GO
UPDATE MedicationOrder
SET Order_BP_EncounterCurr00_DateService = b.CONTACT_DATE,
    Order_BP_EncounterCurr00_DaysFromOrder = DATEDIFF(day, a.Order_Date, b.CONTACT_DATE),
    Order_BP_EncounterCurr00_Diastolic = b.BP_DIASTOLIC,
    Order_BP_EncounterCurr00_Systolic = b.BP_SYSSTOLIC,
    Order_BP_EncounterCurr00_Weight = CASE WHEN CAST(b.WEIGHT AS MONEY) > 0 THEN CAST(b.WEIGHT AS MONEY)/16 ELSE NULL END,
    Order_BP_EncounterCurr00_BMI = CASE WHEN CAST(b.BMI AS MONEY) > 0 THEN CAST(b.BMI AS MONEY) ELSE NULL END,
FROM MedicationOrder a
INNER JOIN db_0175_08_ODS..PAT_ENC b ON a.Order_BP_EncounterCurr00_CSN_ID = b.PAT_ENC_CSN_ID
GO
UPDATE MedicationOrder
SET Order_BP_EncounterNext01_DateService = b.CONTACT_DATE,
    Order_BP_EncounterNext01_DaysFromOrder = DATEDIFF(day, a.Order_Date, b.CONTACT_DATE),
    Order_BP_EncounterNext01_Diastolic = b.BP_DIASTOLIC,
    Order_BP_EncounterNext01_Systolic = b.BP_SYSSTOLIC,
    Order_BP_EncounterNext01_Weight = CASE WHEN CAST(b.WEIGHT AS MONEY) > 0 THEN CAST(b.WEIGHT AS MONEY)/16 ELSE NULL END,
    Order_BP_EncounterNext01_BMI = CASE WHEN CAST(b.BMI AS MONEY) > 0 THEN CAST(b.BMI AS MONEY) ELSE NULL END,
FROM MedicationOrder a
INNER JOIN db_0175_08_ODS..PAT_ENC b ON a.Order_BP_EncounterNext01_CSN_ID = b.PAT_ENC_CSN_ID
GO
-- 05 Populate Several Diastolic Comparison Metrics

ALTER TABLE MedicationOrder
ADD
DiastolicDrop_00_01 INT,
DiastolicDrop_01_02 INT,
DiastolicDrop_00_02 INT
GO
UPDATE MedicationOrder
SET
DiastolicDrop_00_01 = Order_BP_EncounterNext01_Diastolic - Order_BP_EncounterCurr00_Diastolic,
DiastolicDrop_01_02 = Order_BP_EncounterNext02_Diastolic - Order_BP_EncounterNext01_Diastolic,
DiastolicDrop_00_02 = Order_BP_EncounterNext02_Diastolic - Order_BP_EncounterCurr00_Diastolic
GO

--- 06 Create PatientOrder table that groups all medications in a given order into a single item

IF OBJECT_ID('TEMP_Order_MedRank') <> 0 DROP TABLE TEMP_Order_MedRank
GO
SELECT DISTINCT
PatientID, Order_Date, Order_MedicationSimpleGenericCode, row_number() OVER(PARTITION BY PatientID, Order_Date ORDER BY Order_MedicationSimpleGenericCode) AS MedicationOrder_Rank
INTO TEMP_Order_MedRank
FROM MedicationOrder

IF OBJECT_ID('PatientOrder') <> 0 DROP TABLE PatientOrder
SELECT DISTINCT
PatientID, Order_Date, CAST(NULL AS VARCHAR(500)) AS MedicationList
INTO PatientOrder
FROM MedicationOrder

DECLARE @Order_Counter INT
DECLARE @Max_Order_Counter INT
SET @Order_Counter = 1
SET @Max_Order_Counter = (SELECT MAX(MedicationOrder_Rank) FROM TEMP_Order_MedRank)
WHILE @Order_Counter <= @Max_Order_Counter
BEGIN
UPDATE PatientOrder
FROM PatientOrder a
INNER JOIN TEMP_Order_MedRank b ON a.PatientID = b.PatientID AND a.Order_Date = b.Order_Date AND b.MedicationOrder_Rank = @Order_Counter
LEFT JOIN TEMP_Order_MedRank c ON a.PatientID = c.PatientID AND a.Order_Date = c.Order_Date AND c.MedicationOrder_Rank = @Order_Counter-1
SET @Order_Counter = @Order_Counter + 1
END
UPDATE PatientOrder SET MedicationList = MedicationList + '|

--- 07 Populate PatientOrder with PAT_ENC BP Detail from MedicationOrder

ALTER TABLE PatientOrder
ADD
Order_AuthorizingProvider VARCHAR(255),
Order_BP_EncounterCurr00_Rank INT,
Order_BP_EncounterPrev04_CSN_ID VARCHAR(255),
Order_BP_EncounterPrev03_CSN_ID VARCHAR(255),
Order_BP_EncounterPrev02_CSN_ID VARCHAR(255)
IF OBJECT_ID('TEMP_JOIN_MedicationOrder') <> 0 DROP TABLE TEMP_JOIN_MedicationOrder
GO
SELECT DISTINCT
  PatientID,
  Order_Date,
  Order_BP_EncounterCurr00_Rank,
  Order_BP_EncounterPrev04_CSN_ID,
  Order_BP_EncounterPrev03_CSN_ID,
  Order_BP_EncounterPrev02_CSN_ID,
  Order_BP_EncounterPrev01_CSN_ID,
  Order_BP_EncounterCurr00_CSN_ID,
  Order_BP_EncounterNext01_CSN_ID,
  Order_BP_EncounterNext02_CSN_ID,
  Order_BP_EncounterNext03_CSN_ID,
  Order_BP_EncounterNext04_CSN_ID,
  Order_BP_EncounterPrev04_DateService,
  Order_BP_EncounterPrev04_DaysFromOrder,
  Order_BP_EncounterPrev04_Diastolic,
  Order_BP_EncounterPrev04_Systolic,
  Order_BP_EncounterPrev04_Weight,
  Order_BP_EncounterPrev04_BMI,
  Order_BP_EncounterPrev04_Temp,
  Order_BP_EncounterPrev04_Pulse,
  Order_BP_EncounterPrev04_VisitProvID,
  Order_BP_EncounterPrev04_PCPProvID,
  Order_BP_EncounterPrev04_DepartmentID,
  Order_BP_EncounterPrev03_DateService,
  Order_BP_EncounterPrev03_DaysFromOrder,
  Order_BP_EncounterPrev03_Diastolic,
  Order_BP_EncounterPrev03_Systolic,
  Order_BP_EncounterPrev03_Weight,
  Order_BP_EncounterPrev03_BMI,
  Order_BP_EncounterPrev03_Temp,
  Order_BP_EncounterPrev03_Pulse,
  Order_BP_EncounterPrev03_VisitProvID,
  Order_BP_EncounterPrev03_PCPProvID,
  Order_BP_EncounterPrev03_DepartmentID,
  Order_BP_EncounterPrev02_DateService,
  Order_BP_EncounterPrev02_DaysFromOrder,
  Order_BP_EncounterPrev02_Diastolic,
  Order_BP_EncounterPrev02_Systolic,
  Order_BP_EncounterPrev02_Weight,
  Order_BP_EncounterPrev02_BMI,
  Order_BP_EncounterPrev02_Temp,
  Order_BP_EncounterPrev02_Pulse,
  Order_BP_EncounterPrev02_VisitProvID,
  Order_BP_EncounterPrev02_PCPProvID,
  Order_BP_EncounterPrev02_DepartmentID,
  Order_BP_EncounterPrev01_DateService,
  Order_BP_EncounterPrev01_DaysFromOrder,
  Order_BP_EncounterPrev01_Diastolic,
  Order_BP_EncounterPrev01_Systolic
CREATE UNIQUE INDEX ix_Pat_ID_Order_Date ON TEMP_JOIN_MedicationOrder (PatientID, Order_Date)

GO

UPDATE PatientOrder
SET Order_AuthorizingProvider = a.AuthorizingProvider
FROM PatientOrder
INNER JOIN (
    SELECT PAT_ID,
    ORDERING_DATE,
    MAX(AUTHRZING_PROV_ID) AS AuthorizingProvider
    FROM db_0175_08_ODS..ORDER_MED
) a
ON PatientOrder.PAT_ID = a.PAT_ID AND PatientOrder.ORDERING_DATE = a.ORDERING_DATE
INTO TEMP_JOIN_MedicationOrder
IF OBJECT_ID('TEMP_TOP_DRUGS') <> 0 DROP TABLE TEMP_TOP_DRUGS
GO
SELECT TOP 50
    Order_MedicationSimpleGenericCode,
    COUNT(*) AS Cnt
SELECT DISTINCT Order_BP_EncounterPrev04_CSN_ID FROM PatientOrder
UNION
SELECT DISTINCT Order_BP_EncounterPrev03_CSN_ID FROM PatientOrder
UNION
SELECT DISTINCT Order_BP_EncounterPrev02_CSN_ID FROM PatientOrder
UNION
SELECT DISTINCT Order_BP_EncounterPrev01_CSN_ID FROM PatientOrder
UNION
SELECT DISTINCT Order_BP_EncounterCurr00_CSN_ID FROM PatientOrder
UNION
SELECT DISTINCT Order_BP_EncounterNext01_CSN_ID FROM PatientOrder
UNION
SELECT DISTINCT Order_BP_EncounterNext02_CSN_ID FROM PatientOrder
UNION
SELECT DISTINCT Order_BP_EncounterNext03_CSN_ID FROM PatientOrder
UNION
SELECT DISTINCT Order_BP_EncounterNext04_CSN_ID FROM PatientOrder
)
GO

CREATE TABLE TEMP_All_CSN_ID

SELECT Order_BP_EncounterPrev04_CSN_ID AS CSN_ID
INTO TEMP_All_CSN_ID
FROM
SELECT DISTINCT Order_BP_EncounterPrev04_CSN_ID FROM PatientOrder
UNION
SELECT DISTINCT Order_BP_EncounterPrev03_CSN_ID FROM PatientOrder
UNION
SELECT DISTINCT Order_BP_EncounterPrev02_CSN_ID FROM PatientOrder
UNION
SELECT DISTINCT Order_BP_EncounterPrev01_CSN_ID FROM PatientOrder
UNION
SELECT DISTINCT Order_BP_EncounterCurr00_CSN_ID FROM PatientOrder
UNION
SELECT DISTINCT Order_BP_EncounterNext01_CSN_ID FROM PatientOrder
UNION
SELECT DISTINCT Order_BP_EncounterNext02_CSN_ID FROM PatientOrder
UNION
SELECT DISTINCT Order_BP_EncounterNext03_CSN_ID FROM PatientOrder
UNION
SELECT DISTINCT Order_BP_EncounterNext04_CSN_ID FROM PatientOrder

GO

ALTER TABLE TEMP_All_CSN_ID
ADD MedList VARCHAR(500)
GO

DECLARE @Line INT
DECLARE @Max_Line INT

SET @Line = 1
SET @Max_Line = (SELECT MAX(Line) FROM db_0175_08_0S..PAT_ENC_CURR_MEDS INNER JOIN TEMP_All_CSN_ID b ON a.PAT_ENC_CSN_ID = b.CSN_ID AND a.IS_ACTIVE_YN = 'Y')

WHILE @Line <= @Max_Line
BEGIN
UPDATE TEMP_All_CSN_ID
SET MedList = ISNULL(a.MedList, '') + '|' + d.SIMPLE GENERIC_C
FROM TEMP_All_CSN_ID a
    INNER JOIN db_0175_08_ODS..PAT ENC_CURR_MEDS b ON a.CSN_ID = b.PAT ENC_CSNS_ID AND b.LINE = @Line AND b.IS ACTIVE_YN = 'Y'
    INNER JOIN db_0175_08_ODS..CLARITY_MEDICATION d ON b.MEDICATION_ID = d.MEDICATION_ID
WHERE CHARINDEX(d.SIMPLE GENERIC_C, ISNULL(a.MedList, ''), 1) = 0
SET @Line = @Line + 1
END

UPDATE TEMP_All_CSN_ID SET MedList = MedList + '''
GO

ALTER TABLE PatientOrder
GO
UPDATE PatientOrder
FROM PatientOrder a
    INNER JOIN TEMP_All_CSN_ID b ON a.Order_BP_EncounterPrev04_CSN_ID = b.CSN_ID
GO
UPDATE PatientOrder
SET Order_BP_EncounterPrev03_Med_List = b.MedList
FROM PatientOrder a
    INNER JOIN TEMP_All_CSN_ID b ON a.Order_BP_EncounterPrev03_CSN_ID = b.CSN_ID
GO
UPDATE PatientOrder
SET Order_BP_EncounterPrev02_Med_List = b.MedList
FROM PatientOrder a
    INNER JOIN TEMP_All_CSN_ID b ON a.Order_BP_EncounterPrev02_CSN_ID = b.CSN_ID
GO
UPDATE PatientOrder
SET Order_BP_EncounterPrev01_Med_List = b.MedList
FROM PatientOrder a
    INNER JOIN TEMP_All_CSN_ID b ON a.Order_BP_EncounterPrev01_CSN_ID = b.CSN_ID
GO
UPDATE PatientOrder
SET Order_BP_EncounterCurr00_Med_List = b.MedList
FROM PatientOrder a
    INNER JOIN TEMP_All_CSN_ID b ON a.Order_BP_EncounterCurr00_CSN_ID = b.CSN_ID
GO
UPDATE PatientOrder
SET Order_BP_EncounterNext04_Med_List = b.MedList
FROM PatientOrder a
    INNER JOIN TEMP_All_CSN_ID b ON a.Order_BP_EncounterNext04_CSN_ID = b.CSN_ID
GO
UPDATE PatientOrder
SET Order_BP_EncounterNext03_Med_List = b.MedList
FROM PatientOrder a
INNER JOIN TEMP_All_CSN_ID b ON a.Order_BP_EncounterNext03_CSN_ID = b.CSN_ID
GO
UPDATE PatientOrder
SET Order_BP_EncounterNext02_Med_List = b.MedList
FROM PatientOrder a
INNER JOIN TEMP_All_CSN_ID b ON a.Order_BP_EncounterNext02_CSN_ID = b.CSN_ID
GO
UPDATE PatientOrder
SET Order_BP_EncounterNext01_Med_List = b.MedList
FROM PatientOrder a
INNER JOIN TEMP_All_CSN_ID b ON a.Order_BP_EncounterNext01_CSN_ID = b.CSN_ID
GO
------------------------------------------------------------------------------
----------------------------------
------------------
-- 09 Create Columns to Determine Acceptable Drug Order to Monitor BP Changes
------------------------------------------------------------------------------
----------------------------------
------------------
ALTER TABLE PatientOrder
ADD
  NextMedicationOrderDate DATE,
  PreviousMedicationOrderDate DATE,
  Order_BP_EncounterNext01_OK TINYINT,
  Order_BP_EncounterNext02_OK TINYINT,
  Order_BP_EncounterNext03_OK TINYINT,
  Order_BP_EncounterNext04_OK TINYINT
GO
IF OBJECT_ID('TEMP_NextOrderDate') <> 0 DROP TABLE TEMP_NextOrderDate
SELECT PatientID,
  Order_Date,
  row_number () OVER(
    PARTITION BY PatientID
    ORDER BY Order_Date
  ) AS PatientEncounterRanks
INTO TEMP_NextOrderDate
FROM PatientOrder
GO
CREATE INDEX ix_Patient_OrderDate ON TEMP_NextOrderDate (PatientID,Order_Date)
CREATE INDEX ix_Patient_PatientEncounterRanks ON TEMP_NextOrderDate (PatientID,PatientEncounterRanks)
GO
UPDATE PatientOrder
SET
  NextMedicationOrderDate = c.Order_Date,
  PreviousMedicationOrderDate = d.Order_Date
FROM PatientOrder a
INNER JOIN TEMP_NextOrderDate b ON a.PatientID = b.PatientID AND a.Order_Date = b.Order_Date
LEFT JOIN TEMP_NextOrderDate c ON b.PatientID = c.PatientID AND b.PatientEncounterRanks = c.PatientEncounterRanks-1
LEFT JOIN TEMP_NextOrderDate d ON b.PatientID = d.PatientID AND b.PatientEncounterRanks = d.PatientEncounterRanks+1
GO
DECLARE @MaxDaysAfterOrder INT
DECLARE @MaxDaysBeforeOrder INT
DECLARE @MinDaysAfterOrder INT
SET @MinDaysAfterOrder = 10
SET @MaxDaysAfterOrder = 60
SET @MaxDaysBeforeOrder = -30

UPDATE PatientOrder
SET
    Order_BP_EncounterNext01_OK =
        CASE
            WHEN NextMedicationOrderDate >= Order_BP_EncounterNext01_DateService
                AND PreviousMedicationOrderDate < Order_BP_EncounterCurr00_DateService
                AND Order_BP_EncounterNext01_DaysFromOrder BETWEEN @MinDaysAfterOrder AND @MaxDaysAfterOrder
                AND Order_BP_EncounterCurr00_DaysFromOrder > @MaxDaysBeforeOrder
            THEN 1
            ELSE 0
        END,
    Order_BP_EncounterNext02_OK =
        CASE
            WHEN NextMedicationOrderDate >= Order_BP_EncounterNext02_DateService
                AND PreviousMedicationOrderDate < Order_BP_EncounterCurr00_DateService
                AND Order_BP_EncounterNext02_DaysFromOrder BETWEEN @MinDaysAfterOrder AND @MaxDaysAfterOrder
                AND Order_BP_EncounterCurr00_DaysFromOrder > @MaxDaysBeforeOrder
            THEN 1
            ELSE 0
        END,
    Order_BP_EncounterNext03_OK =
        CASE
            WHEN NextMedicationOrderDate >= Order_BP_EncounterNext03_DateService
                AND PreviousMedicationOrderDate < Order_BP_EncounterCurr00_DateService
                AND Order_BP_EncounterNext03_DaysFromOrder BETWEEN @MinDaysAfterOrder AND @MaxDaysAfterOrder
                AND Order_BP_EncounterCurr00_DaysFromOrder > @MaxDaysBeforeOrder
            THEN 1
            ELSE 0
        END,
    Order_BP_EncounterNext04_OK =
        CASE
            WHEN NextMedicationOrderDate >= Order_BP_EncounterNext04_DateService
                AND PreviousMedicationOrderDate < Order_BP_EncounterCurr00_DateService
                AND Order_BP_EncounterNext04_DaysFromOrder BETWEEN @MinDaysAfterOrder AND @MaxDaysAfterOrder
                AND Order_BP_EncounterCurr00_DaysFromOrder > @MaxDaysBeforeOrder
            THEN 1
            ELSE 0
        END
GO

----------------------------------------------------------------------------------------------------------------
-- 10 Create BP Category Table
----------------------------------------------------------------------------------------------------------------

CREATE TABLE lkp_BloodPressureCategory(
    BP_Code INT PRIMARY KEY,
    BP_Category VARCHAR(255),
    SystolicRange VARCHAR(255),
    DiastolicRange VARCHAR(255))

/*
CATEGORY  Systolic  Diastolic
0  Hypotension  <90  <60
1  Normal  90-119  60-79
2  Prehypertension  120-139  80-89
3  High Blood Pressure (Hypertension) Stage 1  140-159  90-99
4  High Blood Pressure (Hypertension) Stage 2  160-180  100-110
5  Hypertensive Crisis (Emergency care needed)  >180  >110
*/
INSERT INTO lkp_BloodPressureCategory VALUES('0', 'Hypotension', '<90', '<60')
INSERT INTO lkp_BloodPressureCategory VALUES('1', 'Normal', '90-119', '60-79')
INSERT INTO lkp_BloodPressureCategory VALUES('2', 'Prehypertension', '120-139', '80-89')
INSERT INTO lkp_BloodPressureCategory VALUES('3', 'High Blood Pressure (Hypertension) Stage 1', '140-159', '90-99')
INSERT INTO lkp_BloodPressureCategory VALUES('4', 'High Blood Pressure (Hypertension) Stage 2', '160-180', '100-110')
INSERT INTO lkp_BloodPressureCategory VALUES('5', 'Hypertensive Crisis (Emergency care needed)', '>', '180', '>', '110')

SELECT * FROM lkp_BloodPressureCategory

/*
---------------------------------------------------------------
------11a BP Clean-up
---------------------------------------------------------------
UPDATE PatientOrder
SET Order_BP_EncounterCurr00_Diastolic = ROUND(Order_BP_EncounterCurr00_Diastolic/10, 0)
WHERE Order_BP_EncounterCurr00_Diastolic >= 500

UPDATE PatientOrder
SET Order_BP_EncounterCurr00_Systolic = ROUND(Order_BP_EncounterCurr00_Systolic/10, 0)
WHERE Order_BP_EncounterCurr00_Systolic >= 800

UPDATE PatientOrder
SET Order_BP_EncounterCurr00_Systolic = ROUND(Order_BP_EncounterCurr00_Systolic*10, 0)
WHERE Order_BP_EncounterCurr00_Systolic BETWEEN 10 AND 15

ALTER TABLE PatientOrder
ADD OutlierDiastolic_mrkr TINYINT,
OutlierSystolic_mrkr TINYINT

GO
UPDATE PatientOrder
SET OutlierDiastolic_mrkr =
CASE WHEN Order_BP_EncounterCurr00_Diastolic < 40 OR Order_BP_EncounterCurr00_Diastolic > 130 THEN 1
END,
OutlierSystolic_mrkr =
CASE WHEN Order_BP_EncounterCurr00_Systolic < 60 OR Order_BP_EncounterCurr00_Systolic > 250 THEN 1
END

-------------------------------
------11b Add BP Categories
-------------------------------
ALTER TABLE PatientOrder
ADD
    Orig_BP_Systolic_Cat VARCHAR(255),
    Orig_BP_Diastolic_Cat VARCHAR(255),
    New_BP_Systolic_Cat VARCHAR(255),
    New_BP_Diastolic_Cat VARCHAR(255),
    BP_Systolic_Change INT,
    BP_Diastolic_Change INT
UPDATE PatientOrder
SET
    Orig_BP_Systolic_Cat =
        CASE
            WHEN Order_BP_EncounterCurr00_Systolic < 90 THEN 0
            WHEN Order_BP_EncounterCurr00_Systolic BETWEEN 90 AND 119 THEN 1
            WHEN Order_BP_EncounterCurr00_Systolic BETWEEN 120 AND 139 THEN 2
            WHEN Order_BP_EncounterCurr00_Systolic BETWEEN 140 AND 159 THEN 3
            WHEN Order_BP_EncounterCurr00_Systolic BETWEEN 160 AND 180 THEN 4
            WHEN Order_BP_EncounterCurr00_Systolic > 180 THEN 5
        END,
    Orig_BP_Diastolic_Cat =
        CASE
            WHEN Order_BP_EncounterCurr00_Diastolic < 60 THEN 0
            WHEN Order_BP_EncounterCurr00_Diastolic BETWEEN 60 AND 79 THEN 1
            WHEN Order_BP_EncounterCurr00_Diastolic BETWEEN 80 AND 89 THEN 2
            WHEN Order_BP_EncounterCurr00_Diastolic BETWEEN 90 AND 99 THEN 3
            WHEN Order_BP_EncounterCurr00_Diastolic BETWEEN 100 AND 110 THEN 4
            WHEN Order_BP_EncounterCurr00_Diastolic > 110 THEN 5
        END,
    New_BP_Systolic_Cat =
        CASE
            WHEN Order_BP_EncounterNext04_Systolic < 90 THEN 0
            WHEN Order_BP_EncounterNext04_Systolic BETWEEN 90 AND 119 THEN 1
            WHEN Order_BP_EncounterNext04_Systolic BETWEEN 120 AND 139 THEN 2
            WHEN Order_BP_EncounterNext04_Systolic BETWEEN 140 AND 159 THEN 3
            WHEN Order_BP_EncounterNext04_Systolic BETWEEN 160 AND 180 THEN 4
            WHEN Order_BP_EncounterNext04_Systolic > 180 THEN 5
        END,
    New_BP_Diastolic_Cat =
        CASE
            WHEN Order_BP_EncounterNext03_Systolic < 90 THEN 0
            WHEN Order_BP_EncounterNext03_Systolic BETWEEN 90 AND 119 THEN 1
            WHEN Order_BP_EncounterNext03_Systolic BETWEEN 120 AND 139 THEN 2
            WHEN Order_BP_EncounterNext03_Systolic BETWEEN 140 AND 159 THEN 3
            WHEN Order_BP_EncounterNext03_Systolic BETWEEN 160 AND 180 THEN 4
            WHEN Order_BP_EncounterNext03_Systolic > 180 THEN 5
        END,
    BP_Systolic_Change =
        CASE
            WHEN Order_BP_EncounterNext02_Systolic < 90 THEN 0
            WHEN Order_BP_EncounterNext02_Systolic BETWEEN 90 AND 119 THEN 1
            WHEN Order_BP_EncounterNext02_Systolic BETWEEN 120 AND 139 THEN 2
            WHEN Order_BP_EncounterNext02_Systolic BETWEEN 140 AND 159 THEN 3
            WHEN Order_BP_EncounterNext02_Systolic BETWEEN 160 AND 180 THEN 4
            WHEN Order_BP_EncounterNext02_Systolic > 180 THEN 5
        END,
    BP_Diastolic_Change =
        CASE
            WHEN Order_BP_EncounterNext01_Systolic < 90 THEN 0
            WHEN Order_BP_EncounterNext01_Systolic BETWEEN 90 AND 119 THEN 1
            WHEN Order_BP_EncounterNext01_Systolic BETWEEN 120 AND 139 THEN 2
            WHEN Order_BP_EncounterNext01_Systolic BETWEEN 140 AND 159 THEN 3
            WHEN Order_BP_EncounterNext01_Systolic BETWEEN 160 AND 180 THEN 4
            WHEN Order_BP_EncounterNext01_Systolic > 180 THEN 5
        END
END
END
END
END
END
END
END
END
END


```sql
-- Select only relevant cases

TRUNCATE TABLE PatientOrderClean
GO

INSERT INTO [db_0175_08_Hypertension].[dbo].[PatientOrderClean]
  ([PatientID], [Order Date], [MedicationList], [OrderAuthorizingProvider], [OrderBP.EncounterPrev04.CSN_ID],
   [OrderBP.EncounterPrev03.CSN_ID])
SELECT *
FROM (SELECT CAST(NULL AS TINYINT) AS [BP_Diastolic_Change],
  CASE WHEN Order_BP_EncounterNext04.Diastolic < 60 THEN 0
     WHEN Order_BP_EncounterNext04.Diastolic BETWEEN 60 AND 79 THEN 1
     WHEN Order_BP_EncounterNext04.Diastolic BETWEEN 80 AND 89 THEN 2
     WHEN Order_BP_EncounterNext04.Diastolic BETWEEN 90 AND 99 THEN 3
     WHEN Order_BP_EncounterNext04.Diastolic BETWEEN 100 AND 110 THEN 4
     WHEN Order_BP_EncounterNext04.Diastolic > 110 THEN 5
  END AS [BP_Systolic_Change],
  CASE WHEN Order_BP_EncounterNext03.Diastolic < 60 THEN 0
     WHEN Order_BP_EncounterNext03.Diastolic BETWEEN 60 AND 79 THEN 1
     WHEN Order_BP_EncounterNext03.Diastolic BETWEEN 80 AND 89 THEN 2
     WHEN Order_BP_EncounterNext03.Diastolic BETWEEN 90 AND 99 THEN 3
     WHEN Order_BP_EncounterNext03.Diastolic BETWEEN 100 AND 110 THEN 4
     WHEN Order_BP_EncounterNext03.Diastolic > 110 THEN 5
  END AS [Order_BP_EncounterNext02.Diastolic],
  CASE WHEN Order_BP_EncounterNext02.Diastolic < 60 THEN 0
     WHEN Order_BP_EncounterNext02.Diastolic BETWEEN 60 AND 79 THEN 1
     WHEN Order_BP_EncounterNext02.Diastolic BETWEEN 80 AND 89 THEN 2
     WHEN Order_BP_EncounterNext02.Diastolic BETWEEN 90 AND 99 THEN 3
     WHEN Order_BP_EncounterNext02.Diastolic BETWEEN 100 AND 110 THEN 4
     WHEN Order_BP_EncounterNext02.Diastolic > 110 THEN 5
  END AS [Order_BP_EncounterNext01.Diastolic],
  CASE WHEN Order_BP_EncounterNext01.Diastolic < 60 THEN 0
     WHEN Order_BP_EncounterNext01.Diastolic BETWEEN 60 AND 79 THEN 1
     WHEN Order_BP_EncounterNext01.Diastolic BETWEEN 80 AND 89 THEN 2
     WHEN Order_BP_EncounterNext01.Diastolic BETWEEN 90 AND 99 THEN 3
     WHEN Order_BP_EncounterNext01.Diastolic BETWEEN 100 AND 110 THEN 4
     WHEN Order_BP_EncounterNext01.Diastolic > 110 THEN 5
  END AS [Order_AuthorizingProvider],
  CASE WHEN Order_BP_EncounterNext01.Diastolic < 60 THEN 0
     WHEN Order_BP_EncounterNext01.Diastolic BETWEEN 60 AND 79 THEN 1
     WHEN Order_BP_EncounterNext01.Diastolic BETWEEN 80 AND 89 THEN 2
     WHEN Order_BP_EncounterNext01.Diastolic BETWEEN 90 AND 99 THEN 3
     WHEN Order_BP_EncounterNext01.Diastolic BETWEEN 100 AND 110 THEN 4
     WHEN Order_BP_EncounterNext01.Diastolic > 110 THEN 5
  END AS [Order_Date],
  CASE WHEN Order_BP_EncounterNext01.Diastolic < 60 THEN 0
     WHEN Order_BP_EncounterNext01.Diastolic BETWEEN 60 AND 79 THEN 1
     WHEN Order_BP_EncounterNext01.Diastolic BETWEEN 80 AND 89 THEN 2
     WHEN Order_BP_EncounterNext01.Diastolic BETWEEN 90 AND 99 THEN 3
     WHEN Order_BP_EncounterNext01.Diastolic BETWEEN 100 AND 110 THEN 4
     WHEN Order_BP_EncounterNext01.Diastolic > 110 THEN 5
  END AS [MedicationList],
  CASE WHEN Order_BP_EncounterNext01.Diastolic < 60 THEN 0
     WHEN Order_BP_EncounterNext01.Diastolic BETWEEN 60 AND 79 THEN 1
     WHEN Order_BP_EncounterNext01.Diastolic BETWEEN 80 AND 89 THEN 2
     WHEN Order_BP_EncounterNext01.Diastolic BETWEEN 90 AND 99 THEN 3
     WHEN Order_BP_EncounterNext01.Diastolic BETWEEN 100 AND 110 THEN 4
     WHEN Order_BP_EncounterNext01.Diastolic > 110 THEN 5
  END AS [PatientID])
  ORDER BY [Order_Date]
```
SELECT *
FROM PatientOrder
WHERE
(Order_BP_EncounterNext01_OK = 1 OR Order_BP_EncounterNext02_OK = 1 OR Order_BP_EncounterNext03_OK = 1 OR Order_BP_EncounterNext04_OK = 1)
AND OutlierDiastolic_mrkr = 0
AND OutlierSystolic_mrkr = 0
GO
--After choosing 3 popular drugs, create dataset for T-testing to determine similarity between populations
--create index ix_pat_enc_csn_id on db_0175_08_ODS.dbo.ARPB_TRANSACTIONS (PAT_ENC_CSN_ID) include (Primary_dx_ID, TX_ID)
if OBJECT_ID(‘temp_orig_Primary_DX’) <> 0 drop table temp_orig_Primary_DX
select
apt.PAT_ENC_CSN_ID,
min(apt.TX_ID) as TX_ID
into temp_orig_Primary_DX
from PatientOrderClean a
inner join db_0175_08_ODS.dbo.ARPB_TRANSACTIONS apt on a.Order_BP_EncounterCurr00_CSN_ID = apt.PAT_ENC_CSN_ID
where nullif(apt.PRIMARY_DX_ID, '') is not null
group by
apt.PAT_ENC_CSN_ID
GO
create unique index ix_PAT_ENC_CSN_ID on temp_orig_Primary_DX (PAT_ENC_CSN_ID)
GO
if OBJECT_ID(‘DetailHypertensionComparison’) <> 0 drop table DetailHypertensionComparison
SELECT PatientID,
p.SEX_C,
p.BIRTH_DATE,
Order_Date,
MedicationList,
OrderAuthorizingProvider,
Order_BP_EncounterCurr00_CSN_ID AS Orig_CSN_ID,
.dx_ID as orig_DiagnosisID,
.dx_NAME as orig_DiagnosisName,
.dx_GROUP as orig_DiagnosisGroup,
case when d.dx_ID is not null then 1 else 0 end as orig_HypertensionDX_mrkr,
e.ICD9_CODE AS orig_ICD9_Code,
Order_BP_EncounterCurr00_DaysFromOrder AS Orig_DaysFromOrder,
Order_BP_EncounterCurr00_Diastolic AS Orig_Diastolic
CASE
  WHEN Order_BP_EncounterNext04_OK = 1 THEN Order_BP_EncounterNext04_CSN_ID
  WHEN Order_BP_EncounterNext03_OK = 1 THEN Order_BP_EncounterNext03_CSN_ID
  WHEN Order_BP_EncounterNext02_OK = 1 THEN Order_BP_EncounterNext02_CSN_ID
  WHEN Order_BP_EncounterNext01_OK = 1 THEN Order_BP_EncounterNext01_CSN_ID
END AS New_CSN_ID,

CASE when d.DX_ID is not null then 1 else 0 end as new_HypertensionDX_mrkr
from DetailHypertensionComparison a
left join db_0175_08_ODS.dbo.PATIENT p on a.patientid = p.PAT_ID
left join temp_orig_Primary_DX c on a.Order_BP_EncounterCurr00_CSN_ID = c.PAT_ENC_CSN_ID
left join db_0175_08_ODS.dbo.ARPB_TRANSACTIONS cc on c.TX_ID = cc.TX_ID
left join db_0175_08_ODS.dbo.Hypertension_DX_ID d on cc.PRIMARY_DX_ID = cast(d.DX_ID as varchar(255))
left join db_0175_08_ODS.dbo.CLARITY_EDG e on cc.PRIMARY_DX_ID = e.DX_ID

GO
if OBJECT_ID('temp_new_Primary_DX') <> 0 drop table temp_new_Primary_DX

GO
create unique index ix_PAT_ENC_CSN_ID on temp_new_Primary_DX (PAT_ENC_CSN_ID)
GO
update DetailHypertensionComparison set
  new_DiagnosisID = e.DX_ID
  ,new_DiagnosisName = e.DX_NAME
  ,new_DiagnosisGroup = e.DX_GROUP
  ,new_HypertensionDX_mrkr = case when d.DX_ID is not null then 1 else 0 end
  ,new_ICD9_Code = e.ICD9_CODE
from DetailHypertensionComparison a
left join temp_new_Primary_DX c on a.New_CSN_ID = c.PAT_ENC_CSN_ID
left join db_0175_08_ODS.dbo.PATIENT p on a.patientid = p.PAT_ID
left join db_0175_08_ODS.dbo.ARPB_TRANSACTIONS apt on a.New_CSN_ID = apt.PAT_ENC_CSN_ID

GO
db_0175_08_ODS.dbo.ARPB_TRANSACTIONS cc on c.TX_ID = cc.TX_ID
left join
db_0175_08_ODS.dbo.Hypertension_DX_ID d on cc.PRIMARY_DX_ID = cast(d.DX_ID as varchar(255))
left join
db_0175_08_ODS.dbo.CLARITY_EDG e on cc.PRIMARY_DX_ID = e.DX_ID

--select * from DetailHypertensionComparison