Final Project Guide: Healthcare Analytics Scenario

Chapter 6 Video 2: Demographics Scenario

We're taking a look at the distribution of patients in our healthcare system based on their age groups and genders.

1. First, we're selecting the gender of each patient from our patients table.
2. Then, we're using a CASE statement to categorize each patient into one of three age groups: Pediatric, Adult, or Senior.
3. We are determining the age group by calculating the difference in years between the patient's date of birth and the current date using the DATEDIFF function.
4. On the DATEDIFF function, first we want to select the format that we want our outcome to be. In this case, we select YEAR. Then we pass the parameters for calculating the outcome. We use date_of_birth as our initial parameter, and GETDATE() to retrieve our current date.
    - If the age falls between 0 and 17 years, classify as Pediatric.
    - If the age falls between 18 and 64 years, classify as Adult.
    - If the age falls above 64 years, classify as Senior.
5. Next, we're counting the number of patients in each gender-age group combination using the COUNT(*) function.
6. Finally, we're grouping the results by gender and age group using the GROUP BY clause to get a summary of patient counts for each combination.

Chapter 6 Video 3: Demographics and Diagnosis Scenario

Which diagnoses are most prevalent among patients, and how do they vary across the different demographic groups, including gender and age? And remember our tip: we can slightly modify the previous query to bring in the diagnosis.

1. Start with the previous query.
2. This is very similar to the first question, however, the diagnosis is in another table. To retrieve the diagnosis, we have to join the "Outpatient Visits" table to the "Patients" table.
3. Bring in the diagnosis field.
4. Don't forget to add a diagnosis to GROUP BY.

Adding the diagnosis to the query adds more insights into this population's needs. This information can aid in capacity planning, staffing decisions, and resource allocation to ensure that the healthcare facility can adequately serve its patient population.

Chapter 6 Video 4: Appointments Scenario

What are the most common appointment times throughout the day, and how does the distribution of appointment times vary across different hours?

1. Use SELECT combined with DATEPART.
2. Use aggregate function COUNT.
3. Specify the table.
4. Use GROUP BY and optional ORDER BY in descending order.

Chapter 6 Video 5: Laboratory Scenario

What are the most commonly ordered lab tests?

1. To find the most commonly ordered lab tests, we need the test name and the count of how many of those tests were ordered.
2. Use SELECT to select the columns.
3. Specify the table the data is being pulled from.
4. Use GROUP BY to group the data by the test name, and ORDER BY to order the results by the number of tests, in this case, the alias "test_count" in descending order.

Asking about the most commonly ordered lab tests provides insights in healthcare administration and patient care. It aids in resource allocation, quality improvement, cost management, and clinical decision support.

Chapter 6 Video 6: Laboratory with Risk Scenario

We're investigating the lab dataset to examine patients' blood sugar levels. Typically, fasting blood sugar levels fall between 70–100 mg/dL. Our goal is to identify patients whose lab results are outside this normal range to implement early interventions. How would you write a query to identify those patients?

1. First, we inspect the tables to select the columns we want to use. In this case, we need the patient_id, patient_name and result_value.
2. result_value is in lab_tests, and patient_id is in the patients table. The problem is that those tables don't have a common key. In this case, we can use the table 'Outpatient Visits', which contains both fields, patient_id and visit_id to work as a link between those tables.

3. We start by joining the patients table doing an INNER JOIN on the ‘Outpatient Visits’ table using the common key patient_id. Now we can join the lab_results table using the key visit_id.
4. Once those connections are done, we can select the columns patient_id, patient_name and result_value.
5. Now we can move to filtering this data. We want only the test_name “Fasting Blood Sugar”, and the tests that fell out of the normal range. In this case, the tests with result_value less than 70 and more than 100.

By identifying those with abnormal results, you can intervene early, potentially preventing the development of serious health conditions like diabetes or cardiovascular disease.

Chapter 6 Video 7: Risk Scenario

Cardiovascular disease is linked to smoking, among other conditions. The hospital management wants to prevent cardiovascular disease and they need to assess how many patients are considered High, Medium, and Low Risk. Here is how you need to categorize patients into high, medium, and low-risk groups:

High Risk: Patients who are smokers (‘Y’) and have been diagnosed with either hypertension or diabetes.

Medium Risk: Patients who are non-smokers (‘N’) and have been diagnosed with either hypertension or diabetes.

Low Risk: Patients who do not fall into the high or medium-risk categories. This includes patients who are non-smokers and do not have a diagnosis of hypertension or diabetes

1. Start with CASE statement.
2. Use aggregate function COUNT() to count the number of patients in each risk category.
3. Use GROUP BY to group the data by the CASE statement column.

This will give how many patients are in each category of cardiovascular disease risk.

Chapter 6 Video 8: Readmission Scenario

The hospital administration is interested in finding out information about the patients who had multiple visits within 30 days of their previous medical visit.

Write a query to identify those patients, the date of the initial visit, the reason for the initial visit, the readmission date, the reason for readmission, and the number of days between the initial visit and readmission. We only needed the table ‘Outpatient Visits’ for this challenge, which can be joined with itself to get the necessary information.

We were asked to ensure that only patients with at least one readmission within 30 days of their previous visit are included in the result set and that the readmission visit recorded happened after the initial visit.

1. Start by joining the ‘Outpatient Visits’ table with itself using aliases ‘ov_initial’ and ‘ov_readmit’.
2. Use common key “patient_id”.
3. Select columns patient ID, visit date, and reason for visit for both the initial visit and the readmission.
4. Add prefix to the columns to distinguish between the initial visit and the readmission.
5. Use the DATEDIFF functions to calculate the number of days between initial visit and readmission.
6. Use the WHERE clause to filter the results to include only instances where a patient has been readmitted within 30 days of their previous visit.
7. Use AND to ensure that the readmission date occurs after the initial visit date.