Project Part 2 Report

You Wang, yw6127@nyu.edu, Luoyao Chen, lc4866@nyu.edu

1. **Data lake**

   1.1 Data Collection and data lake design

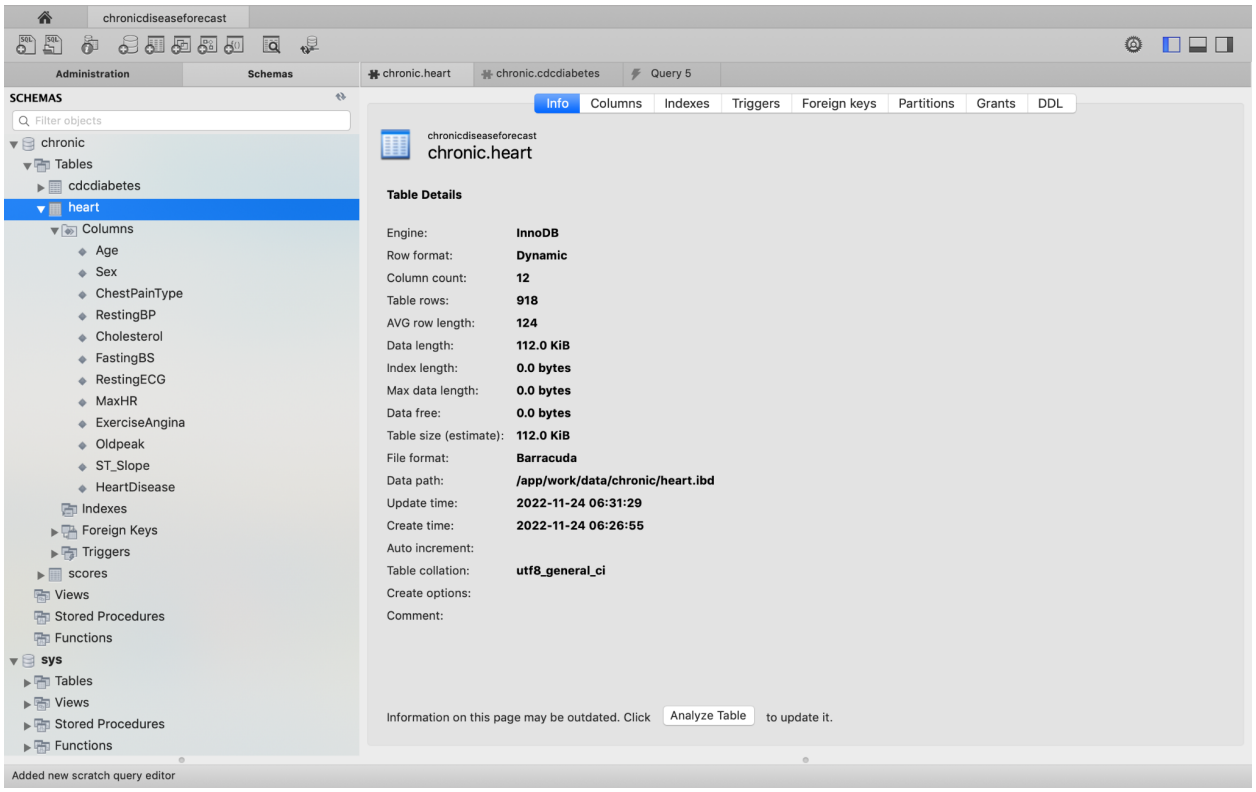
   We have collected obesity, diabetes and cardiovascular patient data from CDC and Kaggle. We have downloaded the datasets and put together a small data lake (in the screen shot) on Microsoft Azure.

   Data lake design:

   Create a MS Azure, MySQL data server, and use MySQL workbench to connect to the server, imported data tables as indicated in the screenshot.

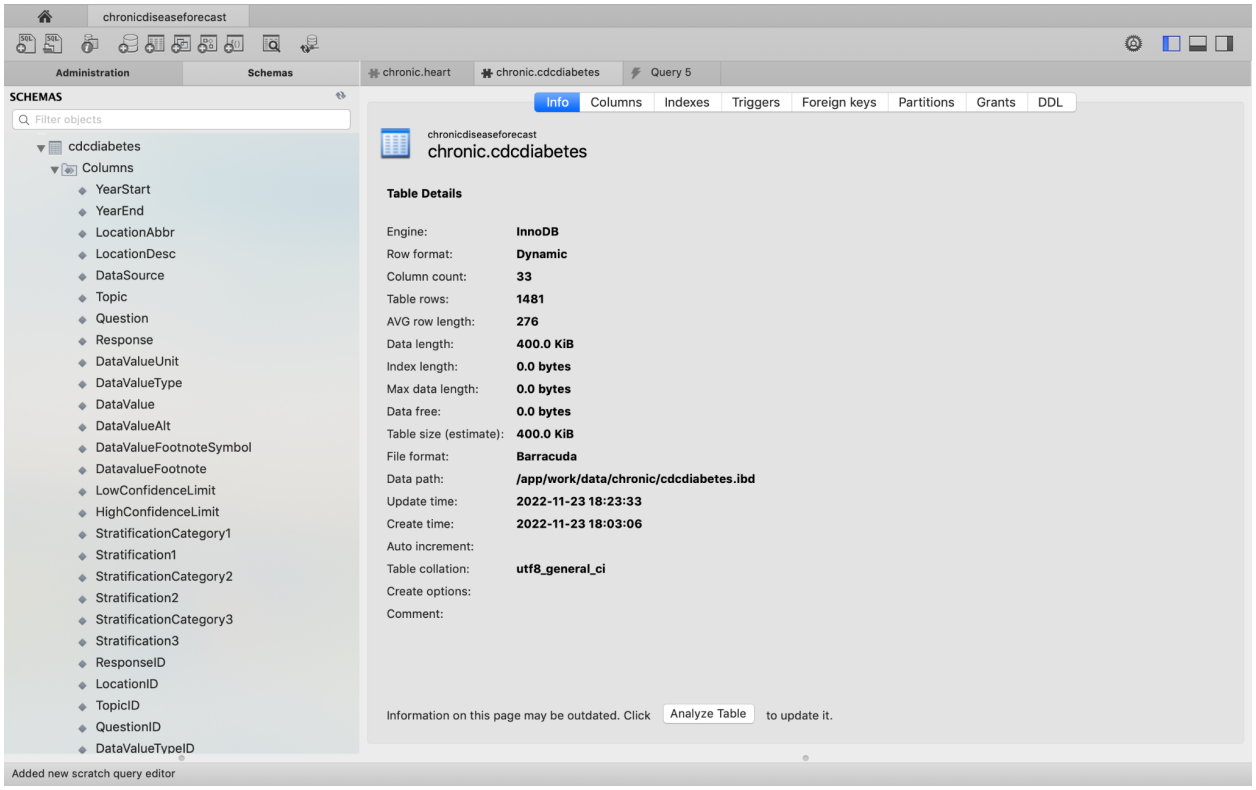
   Heart, from Kaggle:

   https://www.kaggle.com/datasets/fedesoriano/heart-failure-prediction



   CDC Diabetes:

   https://chronicdata.cdc.gov/Chronic-Disease-Indicators/U-S-Chronic-Disease-Indicators-Diabetes/f8ti-h92k



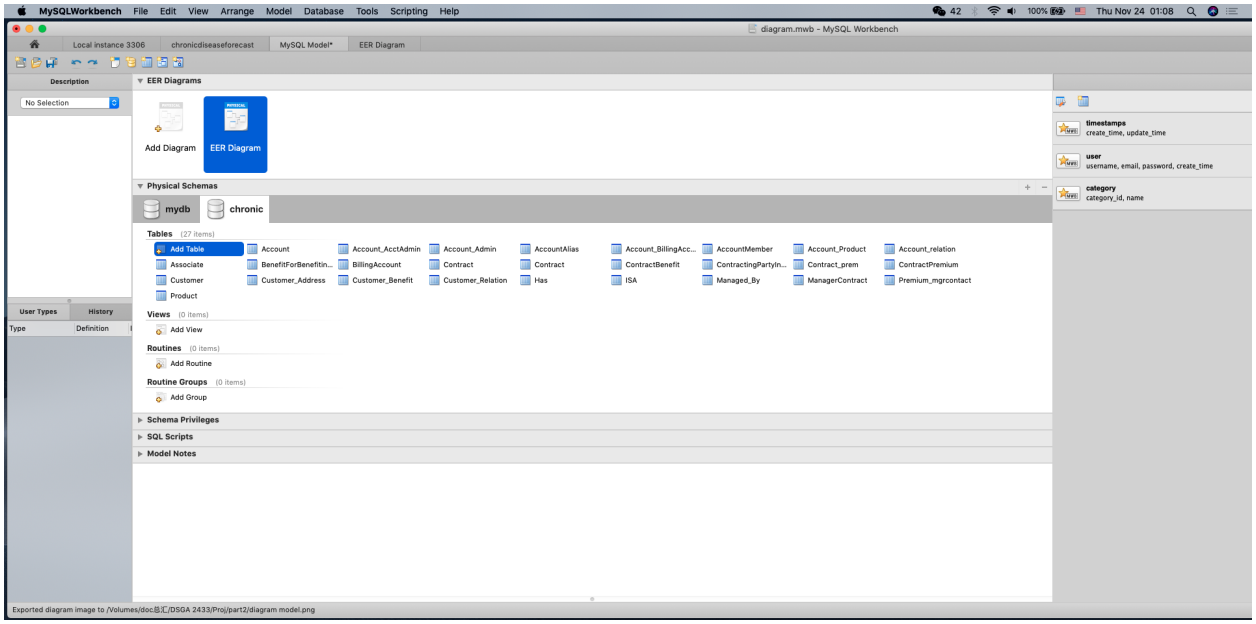
1.2 identify how insights can be extracted from the dataset we have collected and how to feed into the EDA

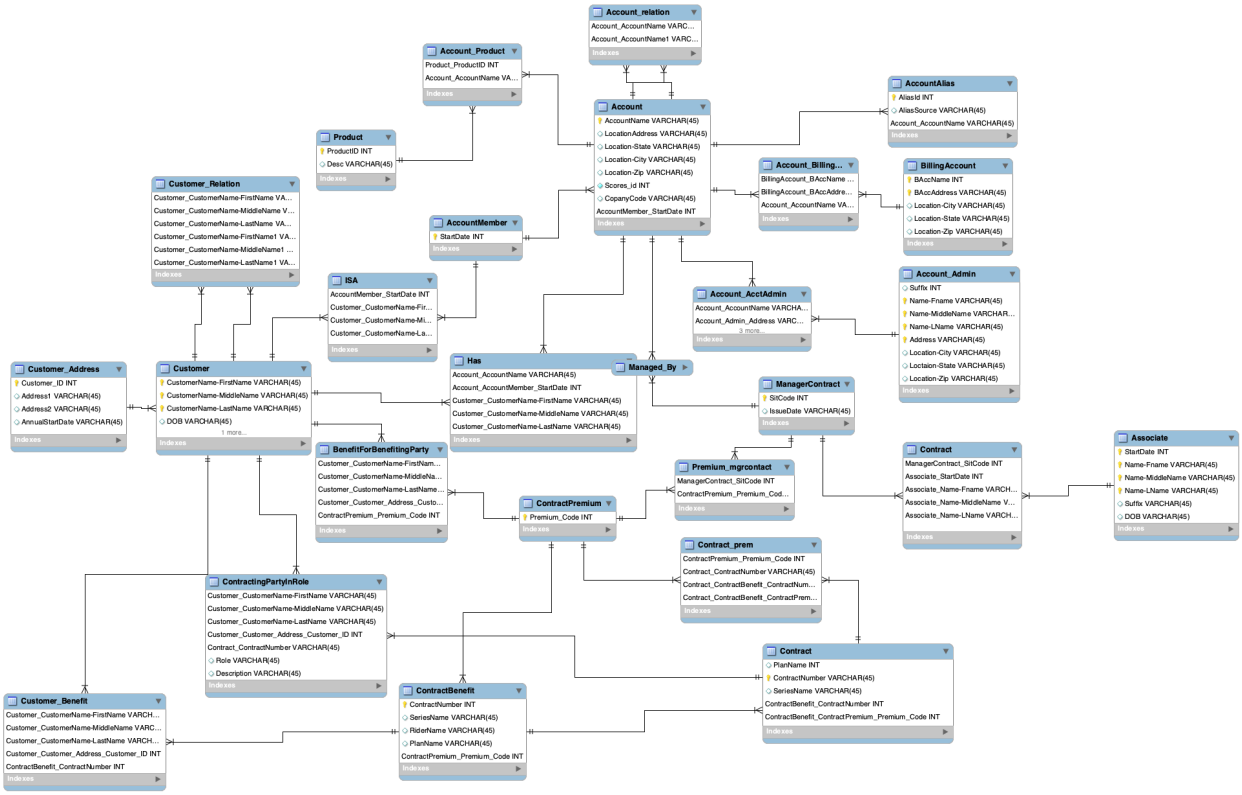
Take this dataset we collected from kaggle as an example:https://www.kaggle.com/datasets/fedesoriano/heart-failure-prediction We can store this dataset on Microsoft Azure Cloud, then we can use the data to develop a machine learning model to predict whether a person is likely to have diabetes. We can use this model to predict the probabilities of customers in the insurance company database of having diabetes. For customers who are likely to have diabetes, we can label them as a high risk group in the insurance company database. From the model, we can also find out what features are more important in predicting diabetes. If we find out there are important diabetes indicators that are not included in the EDA, we can further optimize our design and create corresponding features.

2. Logical schema and optimization

2.1 Create logical schema for previous conceptual model

Create a logical schema corresponding to the ER conceptual diagram in Part1, ref https://hevodata.com/learn/data-modeling-in-azure/#41





2.2 Optimization of the logical schema

When creating the logical schema, we found some redundancy in our original conceptual design. Take the ‘Contract’ entity and the ‘ContractBenefit’ part as an example. In our original design, the ContractBenfit table contained features like ‘PlanName’ from ‘Contract’ and ‘Name’ from customer. To normalize the design, we decided that we don’t

need these features in ‘ContractBenefit’ because these features are redundant in ‘ContractBenefit’ information as they already exist in tables like ‘Contract’ and ‘Customer_Benefit’. In general, we try to make sure each table only contains the necessary information and features and reduce the redundancy.

In order to optimize our model, we have also decomposed some of the tables. We created tables for every many-to-many relationships so that no records are duplicated in each table. For example, at first we plan to put a foreign key ‘Companycode’ from ‘Acount’ to the ‘Account_Admin’ table and describe the relationship between ‘Account’ and ‘Account_Admin’ in only one table. This design choice would result in duplicate records. Therefore, we decided to create another table ‘Acct_AcctAdmin’ to store the relationship between records in these two tables.

3. **Most suitable Reference architecture**

We have stored the collected datasets on MS Azure MySQL data server. And we have created the insurance database tables on MySQL workbench. MS Azure can be connected to MySQL workbench, and csv data can be imported into MySQL workbench. Through this architecture, the insurance company data can interact with the collected data successfully.