# Database Systems Project Part II

Name: Chenyi Lyu NID: N15220231

Name: Zoe Tan NID: N15948322

EDA Logical Schema Creation

To handle unstructured and structured dataset, we configured two servers and several databases on Azure. This allows us to migrate the data from the local MySQL server and fulfill the requirement of the data lake.

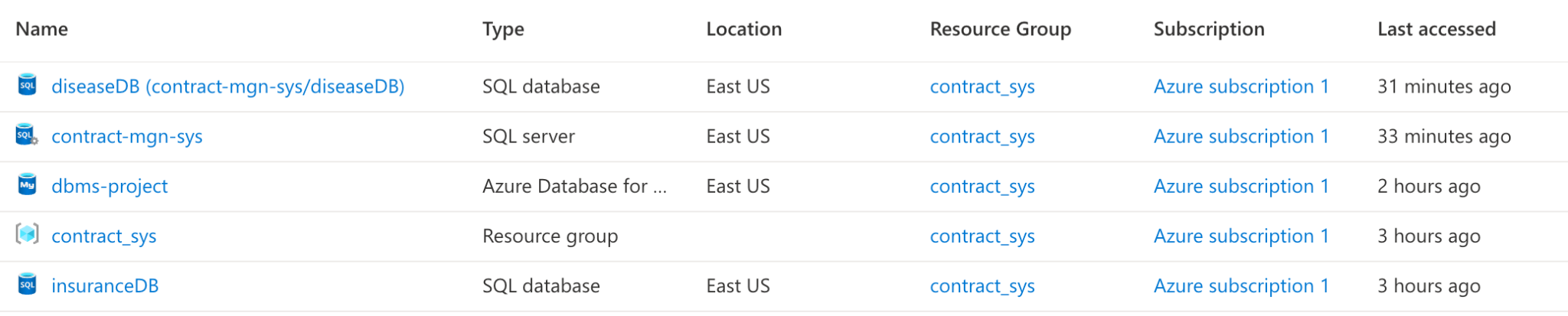


Figure 1. Azure configuration

The disease factors dataset was adopted from [online resources](https://www.kaggle.com/datasets/kamilpytlak/personal-key-indicators-of-heart-disease). We used Azure SQL database to store the .csv file (Figure 1). The dataset contains 18 variables as shown in Figure 2.

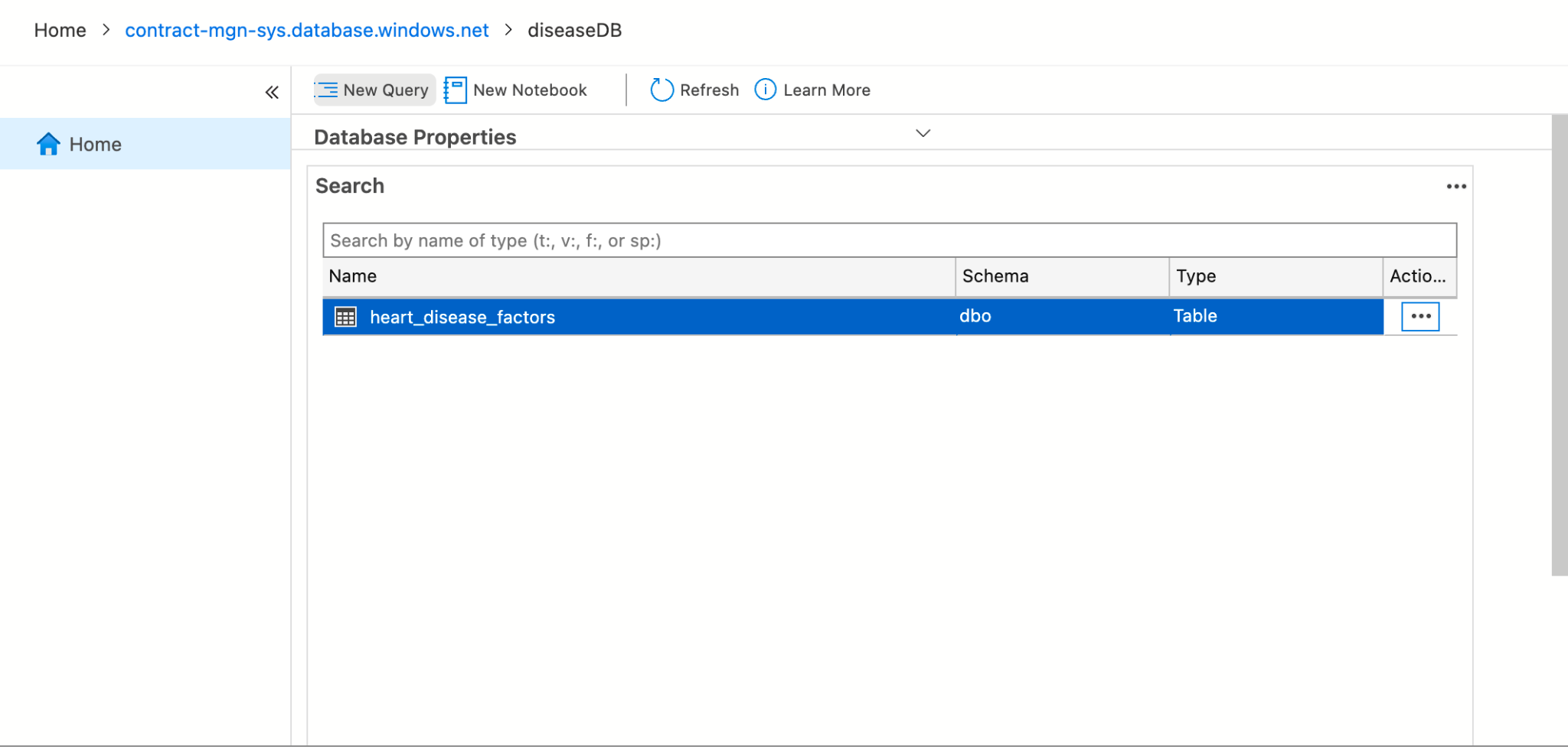


Figure 1. Screenshot of disease factor data and the storage

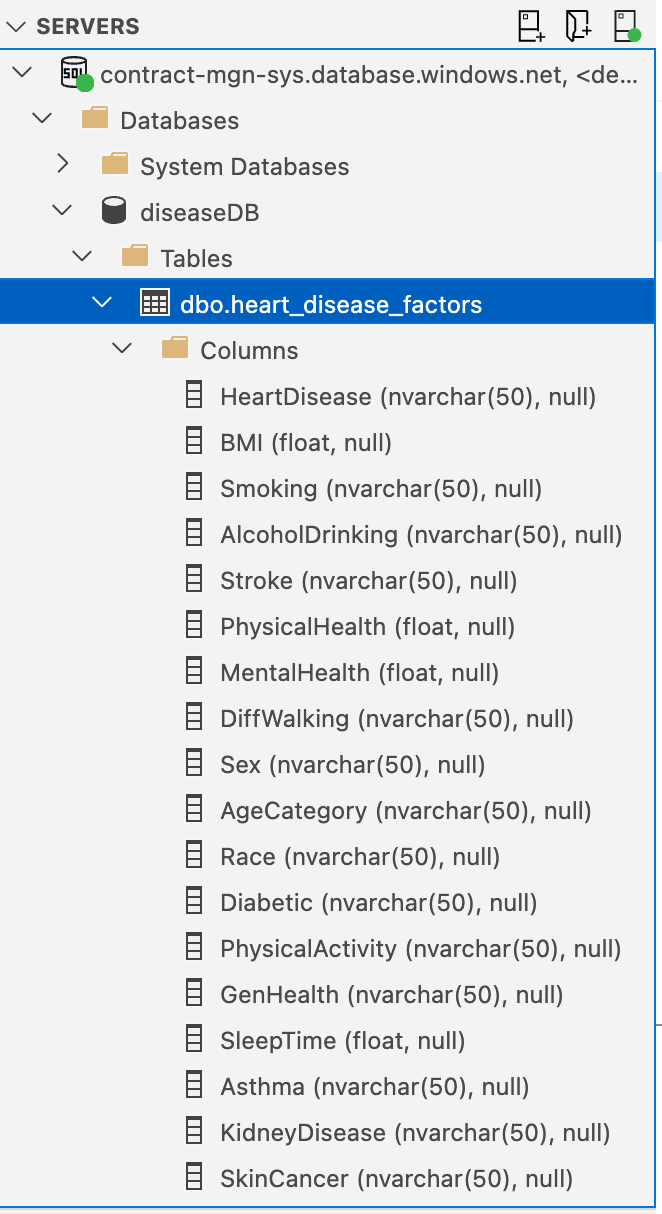


Figure 2. Screenshot of disease factor data schema

We created the Insurance logical schema in local MySQL and then migrated to Azure SQL based on the ER diagram generated in project part I (Figure 3). The insuranceDB contains 27 tables and the relations are as shown in Figure 4.

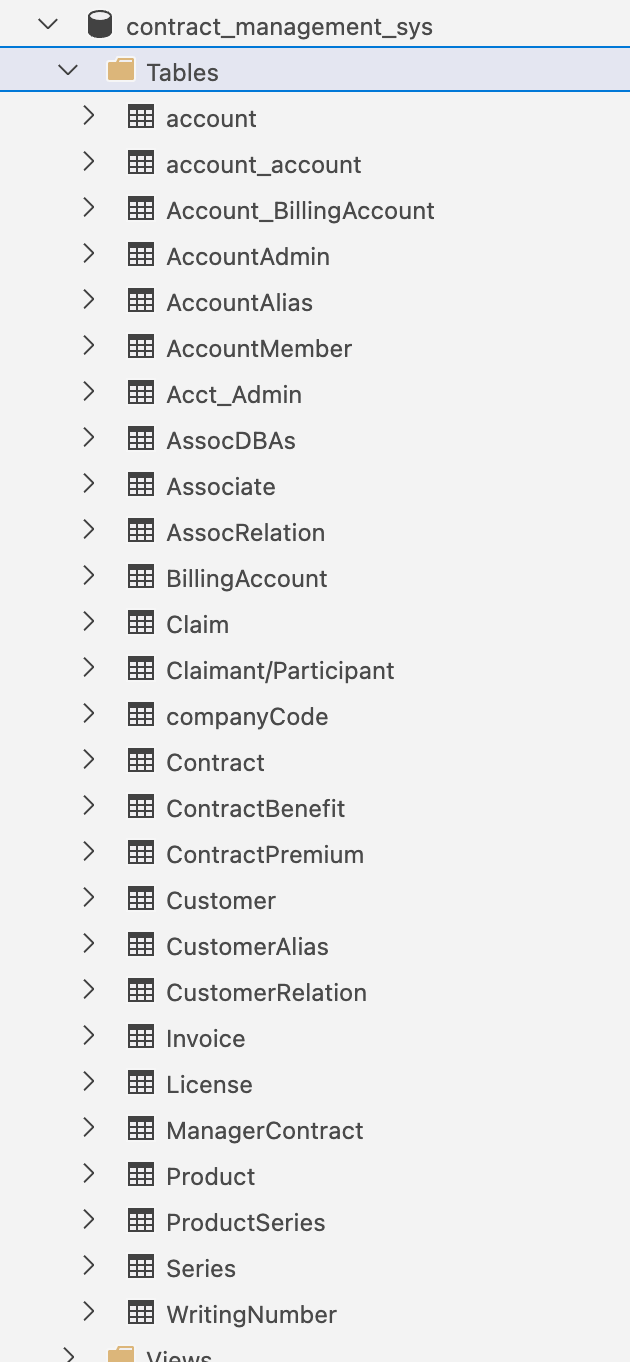


Figure 3. Screenshot of insurance data logical view

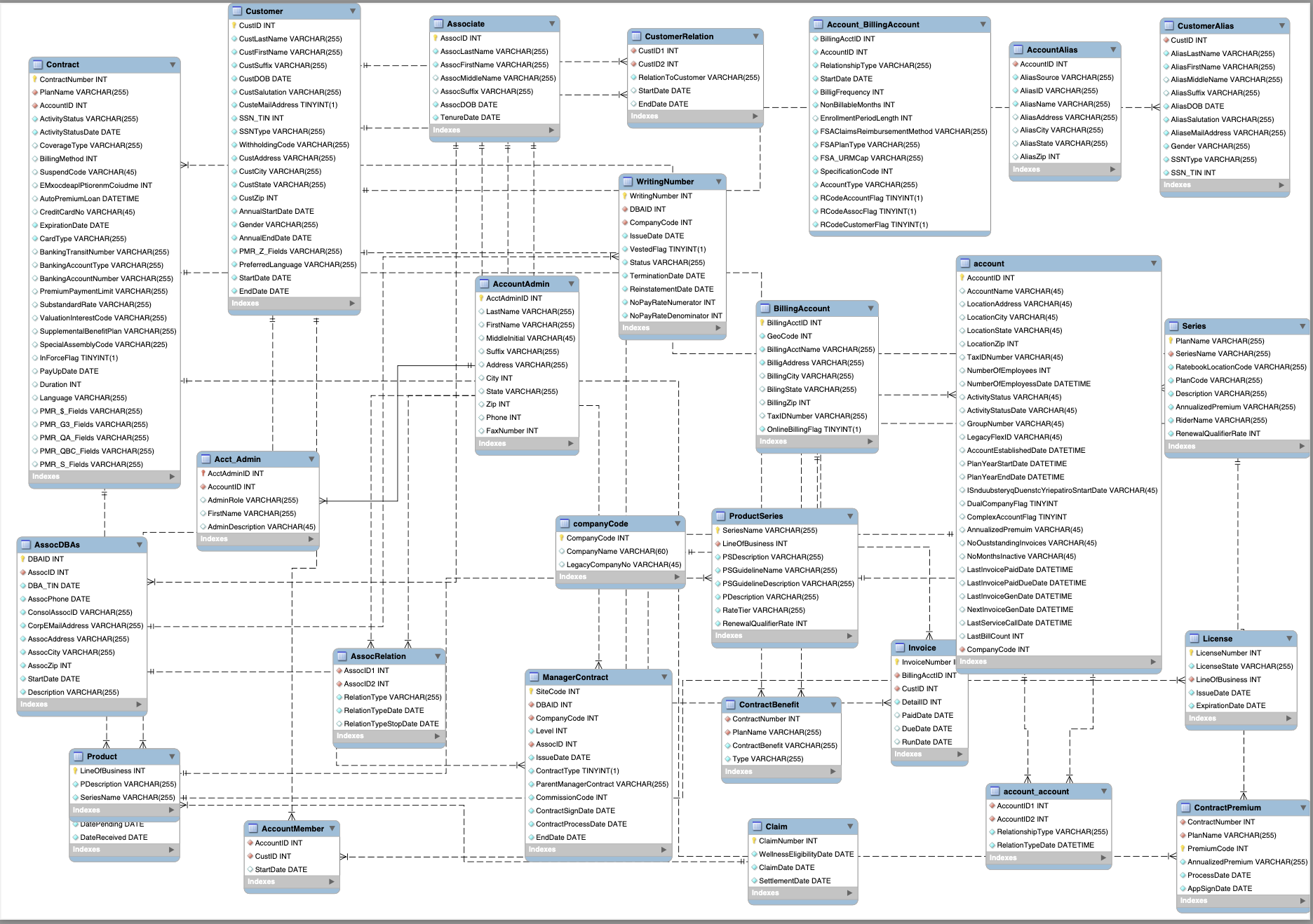


Figure 4. Insurance data logical schema

EDA Logical Schema Optimization

We conduct logical schema optimization for each relation table generated in section I above.

#### Acct\_Admin

**Optimization:** Remove FirstName in Acct\_Admin table to reduce redundant information.

**Reason:** FirstName is stored in both AccountAdmin and Acct\_Admin. We regard FirstName as an attribute of entity AccountAdmin, so it only needs to be stored in AccountAdmin relation. Besides, after removal Acct\_Admin only contains relationship attributes AcctAdmin, AccountID, AdminRole, and AdminDescription, which make the relation Acct\_Admin more explainable.

#### account and companyCode

**Optimization:** Move the attributes (LocationAddress, LocationCity, LocationState, LocationZip, NumberOfEmployees, NumberOfEmployeesDate) from account relation to companyCode relation. So the companyCode table becomes companyCode(CompanyCode, CompanyName, LocationAddress, LocationCity, LocationState, LocationZip, NumberOfEmployees, NumberOfEmployeesDate, LegacyCompanyNo).

**Reason:** By definition attributes (LocationAddress, LocationCity, LocationState, LocationZip, NumberOfEmployees, NumberOfEmployeesDate) describe the company entity, so it is more reasonable to move it to the companyCode relation.

#### Contract, ContractPremium

**Optimization:** Remove attribute PlanName in Contract and ContractPremium.

**Reason:** Since the ContractBenefit relation already stores information for each plan and contract pair, there is no need to store PlanName in Contract and ContractPremium again. This optimization procedure also ensures that every non prime attribute in Contract and ContractPremium is fully functionally dependent on their respective primary keys.

#### ContractBenefit

**Optimization:** Expand ContractBenefit to two relations ContractPlan(ContractNumber, PlanName) and ContractBenefit(ContractNumber, Type, ContractBenefit)

**Reason:** Since a PlanName could have multiple ContractNumber, and a ContractNumber could have multiple benefit Type, there is a multivalued dependency in the original ContractBenefit relation. By decomposing ContractBenefit we remove a lot of redundancy.

#### ProductSeries

**Optimization**: Decompose ProductSeries into ProductSeries(SeriesName, PSDescription, Rate Tier, RenewalQualifierRate) and PSGuideline(SeriesName, PSGuidelineName, PSGuidelineDescription)

**Reason**: Since the relationship of LineOfBusiness and SeriesName is defined in Product relation, there is no need to store redundancy information LineOfBusiness in ProductSeries. A SeriesName could refer to multiple PSGuidelineName, so the original ProductSeries does not satisfy 1NF, so we decompose it.