MSIS 618 – Database Management Fall 2023

Project: Blue Cross Insurance Database management and its implementation

Abhishek. Anant. Cholkar

MSBA

ID:02073868

Table of contents

Executive Summary

This project's goal was Creating a robust database system to support Blue Cross Insurance's key business processes was the project's main objective. Blue Cross offers a range of insurance products to clients in the area, including business, home, auto, life, and health insurance.

A normalized structure is used in the design to ensure data integrity and minimize redundancy. Capturing client 360-degree data, policy and claims information, enabling precise commission monitoring, and optimizing provider partnership administration are the main goals of the new database system. The solution, which is organized into 13 main tables, provides a centralized source of truth that enables. Through main key and foreign key associations, these entities are linked together in order to capture the connections between entities. Because of the database's effective data storage and retrieval capabilities, Blue Cross Company is able to manage customer information, track transactions, examine policies, and securely maintain information.

Agile methodology is used in the implementation process, which includes requirements gathering, database/app development, user testing, and migration/integration. Before the system is used by the entire organization, a gradual rollout begins with the claims workflow for validation.

Once implemented, the database will serve as the analytical foundation, facilitating cross-functional insight through business intelligence (BI) for improved operational decision making, risk assessments, fraud detection, and strategic planning. It will also support everyday transactions.

Project Description

Identification of the company:

For this project, I have selected Blue Cross Insurance Company as a business. I selected this particular organization as I have done one project in Change Management which relates to patient's emergency data. Blue Cross Insurance Company, a leading health insurance provider, plays a crucial role in managing a vast amount of sensitive health and customer data. The company deals with a diverse set of stakeholders, including policyholders, healthcare providers, and regulatory bodies.

Important Business activities using Database:

By creating this database, the company can maintain records of their customers and their details to identify what kind of model they want, their transaction details, and other details which is beneficial for them in such a way that they can just run the query and get details for whatever they want. One of the critical business activities at Blue Cross is the processing and management of insurance claims. This involves handling large volumes of data, including personal client information, policy details, healthcare services availed, billing, and payments. This database helps ask critical business questions leading to efficiencies in claim processing, customer service, and compliance reporting.

Business Requirements:

The proposed database project aims to streamline the claims management process. It will offer a unified platform for storing and accessing all relevant data, ensuring data integrity and security. The database system is designed to support: Targeted Audience, i.e. Claims processors, customer service representatives, data analysts, compliance officers, and management staff.

It will reduce processing time for claims, improve accuracy in data handling, and enhance customer service experiences. It will also facilitate easier compliance reporting and data analysis for strategic decision-making. The system aims to automate many manual processes, reducing the scope for human error and increasing the overall operational efficiency.

Milestone of the project:

I divided this project into 1 week, the first milestone was first day when I decided on the company, The second milestone was identifying the entities and attributes and relationships while the third was creating the tables and fourth was running SQL queries, One of the challenges I faced was creating attributes which are absolutely necessary for this schema for which I had to do some research on the website and refer some tutorials.

Database Design: ER Diagram

Explanation of the SQL code and how it relates to an ER diagram: (source code is in Appendix)

Customer: This table stores the basic information of customers. Each row in this tables represents a unique customer. The CustomerID is the primary keys that uniquely identify each record. Each customer may have multiple policies or have filed various claims over time.

Policy: This table stores the basic information of a Policy. Each row in this tables represents a unique Policy. The PolicyID is the primary keys that uniquely identify each record. This central policy table contains all policy agreements between the insurance provider and its customers.

Claims: The claim table tracks and manages individual claims submitted against active insurance policies. Key details include info on the related policy, the filing date, current status, and claimed dollar amounts for the incident being covered.

PolicyPlan: This describes the different insurance plans and product bundles that customers can be subscribed to via a policy agreement. The PlanID is the primary keys that uniquely identify each record.

ClaimService: Provides granular information on various services delivered related to an insurance claim, tying to a provider's system. Links to claim and provider databases to associate medical, automotive or property services to the originating claim. The ServiceID is the primary keys that uniquely identify each record.

Provider: Stores information on service providers in the insurance network, which supply services paid for under member claims. Can include attributes like specialty, contact info and contracted reimbursement rates. The ProviderID is the primary keys that uniquely identify each record.

Payment: To track settlement of claims, a payment table logs disbursements related to an approved insurance claim. Would include links to originating claim details, payment date and amount, and method of disbursement. The PaymentID is the primary keys that uniquely identify each record.

Relationship between them:

Customer and Policy: One Customer can have multiple Policies (1: M relationship).

Policy and PolicyPlan: One PolicyPlan can be associated with multiple Policies (1: N relationship).

Policy and Claim: One Policy can cover multiple Claims (1: N relationship).

Claim and ClaimService: One Claim can have multiple ClaimServices (1: N relationship).

ClaimService and Provider: One Provider can provide multiple ClaimServices (1: N relationship).

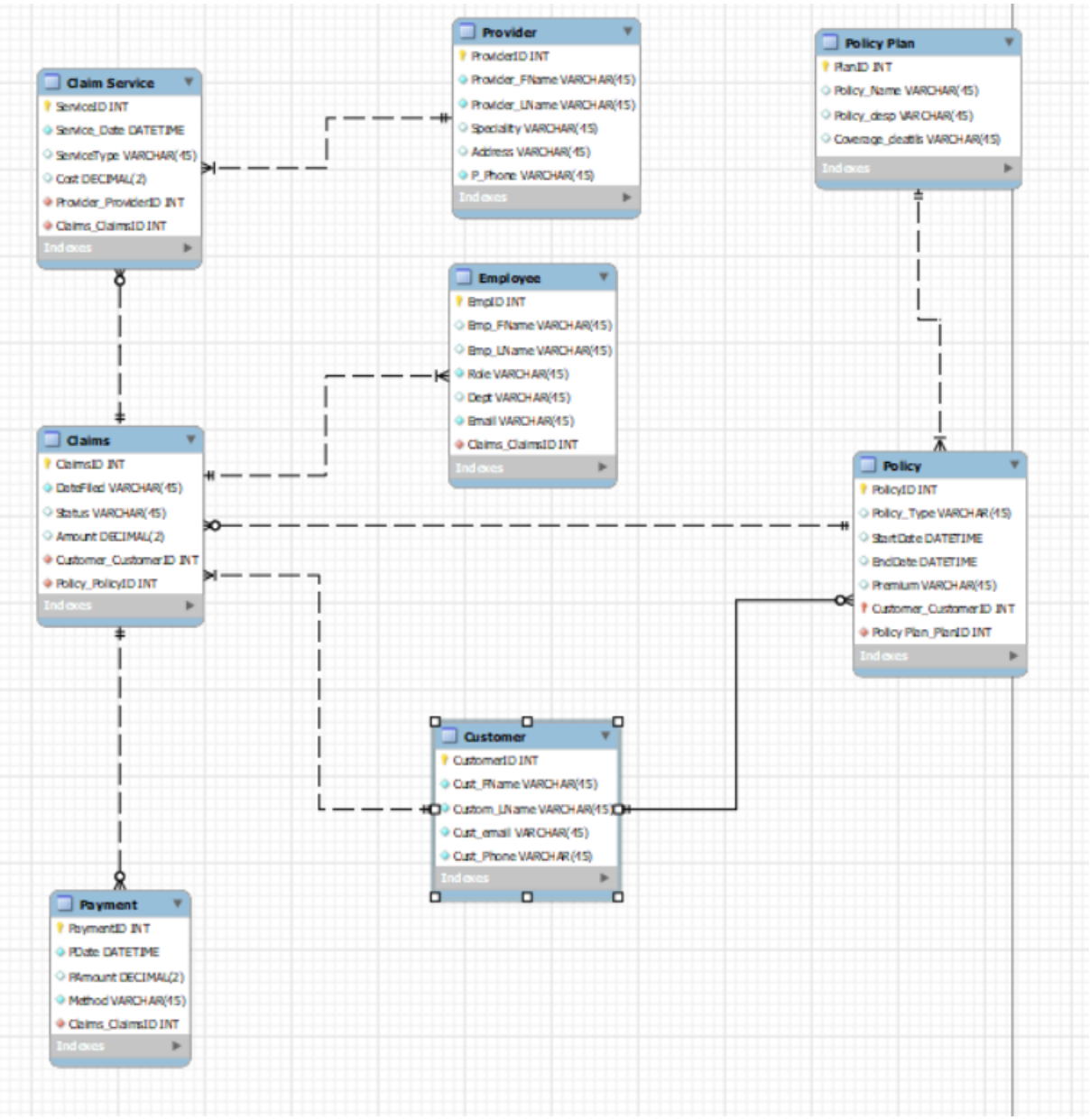
Claim and Payment: One Claim can be associated with one Payment (1:1 relationship).

Claim and Customer: One Customer can be associated with multiple claims (1: N relationship).

Rationale:

In order to minimize redundancy and enhance data integrity, the design attempts to normalize the data. Blue Cross's business logic is reflected in the relationships between each separate entity. We make sure the system can readily handle new Policies, Plans, and Services without requiring changes to the table structure by segregating separate entities such as Policy, Policy Plan, and Claims and Claims Service. Additionally, the design makes sure that all data is appropriately represented in relationships between the data and that all information required for the Blue Cross to operate is saved.

In terms of an ER diagram Customer, Policy, Policy Plan, Claims, Claim Service, Provider, Payment and Employee would be entities. The relationships between entities are represented by the foreign keys. For example, a Customer is associated with a Policy and Claims, forming a relationship between these three entities. Similarly, a Claim Service is associated with a Provider and Claims forming relationships between these entities.



This is the basic layout of the ER diagram. I have shown relationships highlighted earlier.

Based on the tables, the relationships could be represented using Crow's Foot Notation.

Data types and other columns- and table-level constraints

Customer:

CustomerID: INT PRIMARY KEY
FName: VARCHAR (45) NOT NULL
LName: VARCHAR (45) NOT NULL
Address: VARCHAR (45)
Email: VARCHAR (45) UNIQUE
Phone: VARCHAR (15).

Policy:

PolicyID: INT PRIMARY KEY
CustomerID: INT FOREIGN KEY REFERENCES Customer (CustomerID)
Type: VARCHAR (45) NOT NULL
StartDate: DATE NOT NULL
EndDate: DATE NOT NULL
Premium: DECIMAL NOT NULL

PolicyPlan:

PlanID: INT PRIMARY KEY
Name: VARCHAR (45) NOT NULL
Description: LONGTEXT
CoverageDetails: TEXT.

Claim:

ClaimID: INT PRIMARY KEY

PolicyID: INT FOREIGN KEY REFERENCES Policy (PolicyID)

DateFiled: DATE NOT NULL

Status: VARCHAR (45) NOT NULL

Amount: DECIMAL

ClaimService:

ServiceID: INT PRIMARY KEY

ClaimID: INT FOREIGN KEY REFERENCES Claim (ClaimID)

ServiceDate: DATE NOT NULL

ProviderID: INT FOREIGN KEY REFERENCES Provider (ProviderID)

ServiceType: VARCHAR (45) NOT NULL

Cost: DECIMAL

Provider:

ProviderID: INT PRIMARY KEY

Name: VARCHAR (255) NOT NULL

Specialty: VARCHAR (255)

Address: VARCHAR (255)

Phone: VARCHAR (15)

Employee

EmployeeID: INT PRIMARY KEY

FName: VARCHAR (45) NOT NULL

LName: VARCHAR (45) NOT NULL

Role: VARCHAR (45) NOT NULL

Department: VARCHAR (45)

Email: VARCHAR (45) UNIQUE

Payment:

PaymentID: INT PRIMARY KEY

ClaimID: INT FOREIGN KEY REFERENCES Claim (ClaimID)

Date: DATE NOT NULL

Amount: DECIMAL NOT NULL

Method: VARCHAR (45)

Normalization:

A standard for relational database schema design, Boyce-Codd Normal Form (BCNF) seeks to minimize redundancy in relational databases. A more powerful variation of the third normal form is called BCNF (3NF). If and only if X is a super key for any non-trivial functional dependency X -> Y, then a relation is in BCNF. In other words, a candidate key is the determinant of every functional dependency.

Taking a look at the given schema,

We can see that every table has a primary key, which guarantees that each table's rows can be uniquely identified by that key.

Each table's foreign key fields, which establish the connections between them, are independent of any candidate key and have no influence over the values of any non-key fields within the table in which they are found.

Dependencies between tables are minimal because non-key fields—those not included in primary key or foreign key relationships—do not affect the values of any other fields within the same table.

The abstract form of all the relations based on the provided schema is as follows and

The functional dependencies for each relation:

1.Customer (CustomerID, FName, LName, Address, Email, Phone)

CustomerID → FName, LName, Address, Email, Phone.

2. Policy (PolicyID, CustomerID, Type, StartDate, EndDate, Premium)

PolicyID → CustomerID, Type, StartDate, EndDate, Premium

3. PolicyPlan (PlanID, Name, Description, CoverageDetails)

PlanID → Name, Description, CoverageDetails.

4. Claim (ClaimID, PolicyID, DateFiled, Status, Amount)

ClaimID → PolicyID, DateFiled, Status, Amount.

5. ClaimService (ServiceID, ClaimID, ServiceDate, ProviderID, ServiceType, Cost)

ServiceID → ClaimID, ServiceDate, ProviderID, ServiceType, Cost.

6. Provider (ProviderID, Name, Specialty, Address, Phone)

ProviderID → Name, Specialty, Address, Phone.

7. Employee (EmployeeID, Name, Role, Department, Email)

EmployeeID → Name, Role, Department, Email.

8. Payment (PaymentID, ClaimID, Date, Amount, Method)

PaymentID → ClaimID, Date, Amount, Method.

Database Implementation:

The database implementation is done by creating a database for each table that includes 6 rows: (see Appendix)

Below is the Query which can help the company to get the details: (for source code check appendix)

1. List of Active Policies.

Cust_FName	Custom_LName	PolicyID	Policy_Type	StartDate	EndDate
John	Doe	101	Basic	2023-01-01 00:00:00	2024-01-01 00:00:00
Jane	Smith	102	Premium	2023-02-01 00:00:00	2024-02-01 00:00:00
George	Michel	103	Family	2023-03-01 00:00:00	2024-03-01 00:00:00
Michelle	Davis	104	Dental	2023-04-01 00:00:00	2024-04-01 00:00:00
Dave	Olsen	105	Vision	2023-05-01 00:00:00	2024-05-01 00:00:00
Amy	Kim	106	Senior	2023-06-01 00:00:00	2024-06-01 00:00:00
Carlos	Hernandez	107	Student	2023-07-01 00:00:00	2024-07-01 00:00:00
Karen	Smith	108	Travel	2023-08-01 00:00:00	2024-08-01 00:00:00
Linda	Park	109	Catastropic	2023-09-01 00:00:00	2024-09-01 00:00:00
Chris	Lee	110	Employee Welness	2023-10-01 00:00:00	2024-10-01 00:00:00

2. Claims Summary by Customer.

Cust_FName	Custom_LName	Number_Of_Claims	TotalClaimedAmount
John	Doe	1	500.00
Jane	Smith	1	1000.00
George	Michel	1	750.00
Michelle	Davis	1	0.00
Dave	Olsen	1	600.00
Amy	Kim	1	1100.00
Carlos	Kim dez	1	850.00
Karen	Smith	1	700.00
Linda	Park	1	950.00
Chris	Lee	1	1200.00

3. Provider Services and Costs

John	Smith	Consultation	100.000000
Sarah	Lee	X-Ray	200.000000
Mark	Jones	Surgery	300.000000
Jessica	Davis	Skin Test	50.000000
Mike	Brown	Heart Checkup	150.000000
Karen	Miller	Prenatal Care	250.000000
James	Moore	Endoscopy	350.000000
Patricia	Taylor	Bra Endoscopy	400.000000
David	Thomas	General Surgery	450.000000
Nancy	Young	Psychiatric Evaluation	500.000000

4. Policy Renewal Upcoming

Cust_FName	Custom_LName	PolicyID	EndDate
John	Doe	101	2024-01-01 00:00:00

5. Claims Processing Status

ClaimsID	Cust_FName	Custom_LName	DateFiled	Status
202	Jane	Smith	2023-04-10	Processing
206	Amy	Kim	2023-08-15	Processing
209	Linda	Park	2023-11-01	Processing

Stored Procedures and triggers.

Calculating Policy Premium

```
DELIMITER //
CREATE PROCEDURE CalculatePolicyPremium(IN policy_id INT, IN customer_age INT, IN coverage_amount DECIMAL)
BEGIN
    DECLARE premium_rate DECIMAL;
    -- Example calculation logic (simplified)
    IF customer_age < 30 THEN
        SET premium_rate = 0.02;
    ELSE
        SET premium_rate = 0.03;
    END IF;
    UPDATE Policy SET Premium = coverage_amount * premium_rate WHERE PolicyID = policy_id;
END //
DELIMITER ;
```

2. Audit Claim Updates

```
DELIMITER //
CREATE TRIGGER AuditClaimUpdates
BEFORE UPDATE ON Claims
FOR EACH ROW
BEGIN
    IF OLD.Status <> NEW.Status THEN
        INSERT INTO AuditLog (ClaimsID, OldStatus, NewStatus, ChangeDate)
        VALUES (NEW.ClaimsID, OLD.Status, NEW.Status, NOW());
    END IF;
END //
DELIMITER ;
```

This was the end of queries.

References:

Blue cross Company / About Us : *https://www.bcbs.com/about-us*

ER Diagram (MS SQL Server) | https://www.databasestar.com/entity-relationship-diagram/

Boyce-Codd Normal Form (BCNF) of Database Normalization.
Geeksforgeeks.com. https://www.geeksforgeeks.org/boyce-codd-normal-form-bcnf/

Appendix:

Source Code for creation, insertion and quering:

- MySQL Workbench Forward Engineering

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;

SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0;

SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE
,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';


-- -----------------------------------------------------

-- Schema Bluecross_db

-- -----------------------------------------------------


-- -----------------------------------------------------

-- Schema Bluecross_db

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `Bluecross_db` DEFAULT CHARACTER SET utf8 ;

USE `Bluecross_db` ;
```

```
-- -----------------------------------------------------
-- Table `Bluecross_db`.`Customer`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Bluecross_db`.`Customer` (
  `CustomerID` INT NOT NULL,
  `Cust_FName` VARCHAR(45) NOT NULL,
  `Custom_LName` VARCHAR(45) NOT NULL,
  `Cust_email` VARCHAR(45) NOT NULL,
  `Cust_Phone` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`CustomerID`),
  UNIQUE INDEX `Cust_email_UNIQUE` (`Cust_email` ASC) VISIBLE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `Bluecross_db`.`Policy Plan`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Bluecross_db`.`Policy Plan` (
```

```
  `PlanID` INT NOT NULL,
  `Policy_Name` VARCHAR(45) NULL,
  `Policy_desp` VARCHAR(45) NULL,
  `Coverage_deatils` VARCHAR(45) NULL,
  PRIMARY KEY (`PlanID`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `Bluecross_db`.`Policy`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Bluecross_db`.`Policy` (
  `PolicyID` INT NOT NULL,
  `Policy_Type` VARCHAR(45) NULL,
  `StartDate` DATETIME NULL,
  `EndDate` DATETIME NULL,
  `Premium` DECIMAL(10,2) NULL,
  `Customer_CustomerID` INT NOT NULL,
  `Policy_Plan_PlanID` INT NOT NULL,
```

```
  PRIMARY KEY (`PolicyID`, `Customer_CustomerID`),
  INDEX `fk_Policy_Customer1_idx` (`Customer_CustomerID` ASC) VISIBLE,
  INDEX `fk_Policy_Policy Plan1_idx` (`Policy_Plan_PlanID` ASC) VISIBLE,
  CONSTRAINT `fk_Policy_Customer1`
    FOREIGN KEY (`Customer_CustomerID`)
    REFERENCES `Bluecross_db`.`Customer` (`CustomerID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Policy_Policy Plan1`
    FOREIGN KEY (`Policy_Plan_PlanID`)
    REFERENCES `Bluecross_db`.`Policy Plan` (`PlanID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;



-- -----------------------------------------------------
-- Table `Bluecross_db`.`Provider`
-- -----------------------------------------------------
```

```
CREATE TABLE IF NOT EXISTS `Bluecross_db`.`Provider` (
  `ProviderID` INT NOT NULL,
  `Provider_FName` VARCHAR(45) NOT NULL,
  `Provider_LName` VARCHAR(45) NOT NULL,
  `Speciality` VARCHAR(45) NULL,
  `Address` VARCHAR(45) NULL,
  `P_Phone` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`ProviderID`))
ENGINE = InnoDB;



-- -----------------------------------------------------
-- Table `Bluecross_db`.`Employee`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Bluecross_db`.`Employee` (
  `EmpID` INT NOT NULL,
  `Emp_FName` VARCHAR(45) NULL,
  `Emp_LName` VARCHAR(45) NULL,
  `Role` VARCHAR(45) NOT NULL,
```

```
  `Dept` VARCHAR(45) NULL,
  `Email` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`EmpID`),
  UNIQUE INDEX `Email_UNIQUE` (`Email` ASC) VISIBLE)
ENGINE = InnoDB;



-- -----------------------------------------------------
-- Table `Bluecross_db`.`Claims`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Bluecross_db`.`Claims` (
  `ClaimsID` INT NOT NULL,
  `DateFiled` VARCHAR(45) NOT NULL,
  `Status` VARCHAR(45) NULL,
  `Amount` DECIMAL(10,2) NULL,
  `Customer_CustomerID` INT NOT NULL,
  `Policy_PolicyID` INT NOT NULL,
  `Employee_EmpID` INT NOT NULL,
  PRIMARY KEY (`ClaimsID`),
```

```
  INDEX `fk_Claims_Customer1_idx` (`Customer_CustomerID` ASC) VISIBLE,
  INDEX `fk_Claims_Policy1_idx` (`Policy_PolicyID` ASC) VISIBLE,
  INDEX `fk_Claims_Employee1_idx` (`Employee_EmpID` ASC) VISIBLE,
  CONSTRAINT `fk_Claims_Customer1`
    FOREIGN KEY (`Customer_CustomerID`)
    REFERENCES `Bluecross_db`.`Customer` (`CustomerID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Claims_Policy1`
    FOREIGN KEY (`Policy_PolicyID`)
    REFERENCES `Bluecross_db`.`Policy` (`PolicyID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Claims_Employee1`
    FOREIGN KEY (`Employee_EmpID`)
    REFERENCES `Bluecross_db`.`Employee` (`EmpID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `Bluecross_db`.`Payment`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Bluecross_db`.`Payment` (
  `PaymentID` INT NOT NULL,
  `PDate` DATETIME NOT NULL,
  `PAmount` DECIMAL(10,2) NULL,
  `Method` VARCHAR(45) NOT NULL,
  `Claims_ClaimsID` INT NOT NULL,
  PRIMARY KEY (`PaymentID`),
  INDEX `fk_Payment_Claims1_idx` (`Claims_ClaimsID` ASC) VISIBLE,
  CONSTRAINT `fk_Payment_Claims1`
    FOREIGN KEY (`Claims_ClaimsID`)
    REFERENCES `Bluecross_db`.`Claims` (`ClaimsID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `Bluecross_db`.`Claim Service`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Bluecross_db`.`Claim Service` (
  `ServiceID` INT NOT NULL,
  `Service_Date` DATETIME NOT NULL,
  `ServiceType` VARCHAR(45) NULL,
  `Cost` DECIMAL(10,2) NULL,
  `Provider_ProviderID` INT NOT NULL,
  `Claims_ClaimsID` INT NOT NULL,
  PRIMARY KEY (`ServiceID`),
  INDEX `fk_Claim Service_Provider_idx` (`Provider_ProviderID` ASC) VISIBLE,
  INDEX `fk_Claim Service_Claims1_idx` (`Claims_ClaimsID` ASC) VISIBLE,
  CONSTRAINT `fk_Claim Service_Provider`
    FOREIGN KEY (`Provider_ProviderID`)
    REFERENCES `Bluecross_db`.`Provider` (`ProviderID`)
    ON DELETE NO ACTION
```

```
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Claim Service_Claims1`
    FOREIGN KEY (`Claims_ClaimsID`)
    REFERENCES `Bluecross_db`.`Claims` (`ClaimsID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;



SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

Insertion:

```
INSERT INTO Customer (CustomerID,Cust_FName , Custom_LName, Cust_email,Cust_Phone
)
VALUES
  (1, 'John', 'Doe', 'johndoe@email.com', '555-1234'),
  (2, 'Jane', 'Smith', 'janesmith@email.com', '555-5678'),
  (5, 'George', 'Michel', 'gm@email.com', '555-6398'),
```

(6, 'Michelle ', 'Davis', 'mdavis@email.com', '555-5176'),

(7, 'Dave', 'Olsen', 'dolsen@email.net', '555-9265'),

(8, 'Amy', 'Kim', 'amykim@email.com', '555-1059'),

(9, 'Carlos', 'Hernandez', 'carloshdz@email.com', '555-8254'),

(10, 'Karen', 'Smith', 'ksmith23@email.com', '555-1663'),

(12, 'Linda', 'Park', 'lpark81@email.net', '555-6412'),

(13, 'Chris', 'Lee', 'chrislee@email.com', '555-2288');

INSERT INTO `bluecross_db`.`Policy Plan`(PlanID, Policy_Name, Policy_desp, Coverage_deatils)

VALUES

(21, 'Basic Plan', 'Standard coverage', 'Medical, Dental, Vision'),

(22, 'Premium Plan', 'Comprehensive coverage', 'Medical, Dental, Vision, Prescription Drugs'),

(23, 'Family Plan', 'Coverage for entire family', 'Medical, Dental, Vision'),

(24, 'Dental Plan', 'Focused on dental care', 'Dental services and procedures'),

(25, 'Vision Plan', 'Focused on vision care', 'Vision tests, eyeglasses, contact lenses'),

(26, 'Senior Plan', 'Tailored for senior citizens', 'Medical, Prescription Drugs, Elderly Care'),

(27, 'Student Plan', 'Coverage for students', 'Medical, Mental Health, Preventive Care'),

(28, 'Travel Plan', 'Coverage during travel', 'Emergency Medical, Trip Cancellation'),

(29, 'Catastrophic Plan', 'High-deductible coverage', 'Major Medical Expenses'),

(30, 'Employee Wellness Plan', 'Corporate wellness coverage', 'Medical, Fitness Programs');

INSERT INTO Provider (ProviderID, Provider_FName, Provider_LName, Speciality, Address, P_Phone)

VALUES

(1, 'John', 'Smith', 'Family Medicine', '123 Main St, City, State', '555-0111'),

(2, 'Sarah', 'Lee', 'Pediatrics', '456 Park Ln, City, State', '555-0222'),

(3, 'Mark', 'Jones', 'Orthopedics', '789 Church St, City, State', '555-0333'),

(4, 'Jessica', 'Davis', 'Dermatology', '135 Oak Rd, City, State', '555-0444'),

(5, 'Mike', 'Brown', 'Cardiology', '246 Pine St, City, State', '555-0555'),

(6, 'Karen', 'Miller', 'Obstetrics', '357 Valley Rd, City, State', '555-0666'),

(7, 'James', 'Moore', 'Gastroenterology', '159 Spring Ln, City, State', '555-0777'),

(8, 'Patricia', 'Taylor', 'Neurology', '753 Hills Dr, City, State', '555-0888'),

(9, 'David', 'Thomas', 'General Surgery', '456 Glen Creek Wy, City, State', '555-0999'),

(10, 'Nancy', 'Young', 'Psychiatry', '632 Ridge Dr, City, State', '555-1122');

INSERT INTO Policy (PolicyID,Customer_CustomerID , Policy_Plan_PlanID,Policy_Type, StartDate, EndDate, Premium)

VALUES

(101, 1, 21,'Basic', '2023-01-01', '2024-01-01', 100.00),

(102, 2, 22,'Premium','2023-02-01', '2024-02-01', 200.00),

(103, 5, 23,'Family','2023-03-01', '2024-03-01', 150.00),

(104, 6, 24,'Dental', '2023-04-01', '2024-04-01', 80.00),

(105, 7, 25,'Vision', '2023-05-01', '2024-05-01', 90.00),

(106, 8, 26,'Senior', '2023-06-01', '2024-06-01', 120.00),

(107, 9, 27,'Student', '2023-07-01', '2024-07-01', 110.00),

(108, 10, 28,'Travel', '2023-08-01', '2024-08-01', 130.00),

(109, 12, 29,'Catastropic', '2023-09-01', '2024-09-01', 140.00),

(110, 13, 30,'Employee Welness', '2023-10-01', '2024-10-01', 160.00);

INSERT INTO Employee (EmpID,Emp_FName ,Emp_LName , Role, Dept, Email)

VALUES

(1, 'Emma', 'Thomas', 'Claims Adjuster', 'Claims', 'ethomas@email.com'),

(2, 'Noah', 'Johnson', 'Underwriter', 'Underwriting', 'njohnson@email.com'),

(3, 'Olivia', 'Brown', 'Customer Service Rep', 'Customer Service', 'obrown@email.com'),

(4, 'Liam', 'Davis', 'Actuary', 'Actuarial', 'ldavis@email.com'),

(5, 'Sophia', 'Miller', 'Agent', 'Sales', 'smiller@email.com'),

(6, 'Mason', 'Wilson', 'Risk Analyst', 'Risk Management', 'mwilson@email.com'),

(7, 'Isabella', 'Martinez', 'HR Manager', 'Human Resources', 'imartinez@email.com'),

(8, 'Jacob', 'Anderson', 'IT Specialist', 'Information Technology', 'janderson@email.com'),

(9, 'Mia', 'Hernandez', 'Marketing Director', 'Marketing', 'mhernandez@email.com'),

(10, 'William', 'Jones', 'CEO', 'Executive', 'wjones@email.com');

INSERT INTO claims (ClaimsID,Policy_PolicyID , Customer_CustomerID,Employee_EmpID, DateFiled,Status , Amount)

VALUES

(201, 101, 1, 1, '2023-03-15', 'Open', 500.00),

(202, 102, 2, 2, '2023-04-10', 'Processing', 1000.00),

(203, 103, 5, 3, '2023-05-05', 'Closed', 750.00),

(204, 104, 6, 4, '2023-06-01', 'Denied', 0.00),

(205, 105, 7, 5, '2023-07-20', 'Open', 600.00),

(206, 106, 8, 1, '2023-08-15', 'Processing', 1100.00),

(207, 107, 9, 2, '2023-09-10', 'Closed', 850.00),

(208, 108, 10, 3, '2023-10-05', 'Open', 700.00),

(209, 109, 12, 4, '2023-11-01', 'Processing', 950.00),

(210, 110, 13, 5, '2023-12-15', 'Closed', 1200.00);

```
INSERT INTO `bluecross_db`.`claim service` (ServiceID,Claims_ClaimsID ,Service_Date
,Provider_ProviderID , ServiceType, Cost)

VALUES

  (301, 201, '2023-03-16', 1, 'Consultation', 100.00),

  (302, 202, '2023-04-11', 2, 'X-Ray', 200.00),

  (303, 203, '2023-05-06', 3, 'Surgery', 300.00),

  (304, 204, '2023-06-02', 4, 'Skin Test', 50.00),

  (305, 205, '2023-07-21', 5, 'Heart Checkup', 150.00),

  (306, 206, '2023-08-16', 6, 'Prenatal Care', 250.00),

  (307, 207, '2023-09-11', 7, 'Endoscopy', 350.00),

  (308, 208, '2023-10-06', 8, 'Brain MRI', 400.00),

  (309, 209, '2023-11-02', 9, 'General Surgery', 450.00),

  (310, 210, '2023-12-16', 10, 'Psychiatric Evaluation', 500.00);



INSERT INTO Payment (PaymentID,Claims_ClaimsID ,PDate ,PAmount , Method)

VALUES

  (401, 201, '2023-03-20', 500.00, 'Check'),
```

```
(402, 202, '2023-04-15', 1000.00, 'Electronic Transfer'),

(403, 203, '2023-05-10', 750.00, 'Credit Card'),

(404, 204, '2023-06-05', 0.00, 'N/A'),

(405, 205, '2023-07-25', 600.00, 'Cash'),

(406, 206, '2023-08-20', 1100.00, 'Check'),

(407, 207, '2023-09-15', 850.00, 'Electronic Transfer'),

(408, 208, '2023-10-10', 700.00, 'Credit Card'),

(409, 209, '2023-11-05', 950.00, 'Cash'),

(410, 210, '2023-12-20', 1200.00, 'Check');



RENAME TABLE `bluecross_db`.`claim service` TO claim_service;



/*1.

List of Active Policies.

This query retrieves a list of all active policies along with customer names.

Value: Useful for customer service representatives to quickly access active policies for inquiries or follow-ups.

*/

SELECT Customer.Cust_FName,Customer.Custom_LName, Policy.PolicyID,
Policy.Policy_Type, Policy.StartDate, Policy.EndDate
```

```
FROM Customer

JOIN Policy ON Customer.CustomerID = Policy.Customer_CustomerID

WHERE Policy.EndDate >= CURDATE();



/*

2. Claims Summary by Customer.

Generates a summary of claims made by each customer, including the total number of claims and the total amount claimed.

Value: Helps in assessing the claim patterns of customers, valuable for risk assessment and customer relationship management.

*/

SELECT Customer.Cust_FName,Customer.Custom_LName, COUNT(claims.ClaimsID) AS Number_Of_Claims, SUM(claims.Amount) AS TotalClaimedAmount

FROM Customer

JOIN Policy ON Customer.CustomerID = Policy.Customer_CustomerID

JOIN Claims ON Policy.PolicyID = Claims.Policy_PolicyID

GROUP BY Customer.Cust_FName,Customer.Custom_LName;



/*

3.Provider Services and Costs
```

Provides average costs for each type of service offered by providers.

Value: Useful for the finance department in negotiating rates with providers and for cost control analysis.

*/

```
SELECT Provider.Provider_FName,Provider.Provider_LName ,claim_service.ServiceType,
AVG(claim_service.Cost) AS AverageCost

FROM Provider

JOIN claim_service ON Provider.ProviderID = claim_service.Provider_ProviderID

GROUP BY Provider.Provider_FName,Provider.Provider_LName ,claim_service.ServiceType;
```

/*

4.Policy Renewal Upcoming

Identifies policies that are due for renewal in the next 30 days.

Value: Essential for the sales team to proactively reach out to customers for policy renewals, enhancing customer retention.

*/

```
SELECT Customer.Cust_FName,Customer.Custom_LName ,Policy.PolicyID, Policy.EndDate

FROM Policy

JOIN Customer ON Policy.Customer_CustomerID = Customer.CustomerID
```

```
WHERE Policy.EndDate BETWEEN CURDATE() AND DATE_ADD(CURDATE(),
INTERVAL 30 DAY);
```

```
/*

5. Claims Processing Status

Lists all claims that are currently being processed.

Value: Critical for claims department to monitor ongoing claims, ensuring timely processing and
improving customer satisfaction.

*/

SELECT Claims.ClaimsID, Customer.Cust_FName,Customer.Custom_LName,
Claims.DateFiled, Claims.Status

FROM Claims

JOIN Policy ON Claims.Policy_PolicyID = Policy.PolicyID

JOIN Customer ON Policy.Customer_CustomerID= Customer.CustomerID

WHERE Claims.Status = 'Processing';
```

```
/*

1. Stored Procedure: CalculatePolicyPremium
```

Purpose: This stored procedure is designed to calculate and update the premium for a policy based on certain factors like policy type,

customer age, and coverage amount.

*/

```
DELIMITER //

CREATE PROCEDURE CalculatePolicyPremium(IN policy_id INT, IN customer_age INT, IN coverage_amount DECIMAL)

BEGIN

    DECLARE premium_rate DECIMAL;

    -- Example calculation logic (simplified)

    IF customer_age < 30 THEN

        SET premium_rate = 0.02;

    ELSE

        SET premium_rate = 0.03;

    END IF;

    UPDATE Policy SET Premium = coverage_amount * premium_rate WHERE PolicyID = policy_id;

END //

DELIMITER ;
```

/*

2. Trigger: AuditClaimUpdates

Purpose: This trigger is intended to create an audit record every time a claim's status is updated.

It helps in tracking the history of claim processing.

*/

```
DELIMITER //

CREATE TRIGGER AuditClaimUpdates

BEFORE UPDATE ON Claims

FOR EACH ROW

BEGIN

    IF OLD.Status <> NEW.Status THEN

        INSERT INTO AuditLog (ClaimsID, OldStatus, NewStatus, ChangeDate)

        VALUES (NEW.ClaimsID, OLD.Status, NEW.Status, NOW());

    END IF;

END //

DELIMITER ;
```