Analysis of Health Insurance Database Management

Group: 10

Divyank Harjani - 055010

Priyadarshani Dash - 055033

1. Project Information

This project focuses on analyzing a comprehensive Health Insurance Database using SQL. The database is structured to store customer details, insurance policies, claims, payments, hospitalizations, and treatments. The project was implemented using SQL queries for data manipulation and analysis. It supports efficient data retrieval and relationship management across multiple healthcare-related entities.

2. Data Description

The Health Insurance Database consists of 8 interrelated tables with properly defined primary and foreign keys. The structure and attributes are as follows:

1. **Customers Table**
   - **CustomerID** (INT, Primary Key): Unique identifier for each customer.
   - **FirstName** (VARCHAR): Customer's first name.
   - **LastName** (VARCHAR): Customer's last name.
   - **DOB** (DATE): Date of birth.
   - **PhoneNumber** (VARCHAR): Contact number.
   - **Email** (VARCHAR): Email address.
2. **Policies Table**
   - **PolicyID** (INT, Primary Key): Unique identifier for each policy.
   - **CustomerID** (INT, Foreign Key): References Customers.
   - **PolicyType** (VARCHAR): Type of policy (e.g., Health Insurance).
   - **StartDate** (DATE): Policy start date.
   - **EndDate** (DATE): Policy end date.
   - **PremiumAmount** (DECIMAL): Premium amount charged.
3. **Claims Table**
   - **ClaimID** (INT, Primary Key): Unique identifier for each claim.
   - **PolicyID** (INT, Foreign Key): References Policies.
   - **ClaimDate** (DATE): Date the claim was filed.
   - **ClaimAmount** (DECIMAL): Amount claimed.
   - **ClaimStatus** (VARCHAR): Status (Approved, Pending, Rejected).
4. **Payments Table**
   - **PaymentID** (INT, Primary Key): Unique identifier for each payment.
   - **CustomerID** (INT, Foreign Key): References Customers.
   - **PaymentDate** (DATE): Date of payment.
   - **PaymentAmount** (DECIMAL): Amount paid.
   - **PaymentMethod** (VARCHAR): Method used (Credit Card, UPI, etc.).

5. **Hospitals Table**
    - **HospitalID** (INT, Primary Key): Unique identifier for each hospital.
    - **HospitalName** (VARCHAR): Name of the hospital.
    - **Address** (VARCHAR): Hospital address.
    - **PhoneNumber** (VARCHAR): Contact number.
6. **Hospitalizations Table**
    - **HospitalizationID** (INT, Primary Key): Unique identifier.
    - **CustomerID** (INT, Foreign Key): References Customers.
    - **HospitalID** (INT, Foreign Key): References Hospitals.
    - **AdmissionDate** (DATE): Date of admission.
    - **DischargeDate** (DATE): Date of discharge.
    - **TotalCost** (DECIMAL): Total hospitalization cost.
7. **Doctors Table**
    - **DoctorID** (INT, Primary Key): Unique identifier for each doctor.
    - **DoctorName** (VARCHAR): Name of the doctor.
    - **Specialty** (VARCHAR): Area of specialization.
    - **PhoneNumber** (VARCHAR): Contact number.
    - **HospitalID** (INT, Foreign Key): References Hospitals.
8. **Treatments Table**
    - **TreatmentID** (INT, Primary Key): Unique identifier.
    - **HospitalizationID** (INT, Foreign Key): References Hospitalizations.
    - **DoctorID** (INT, Foreign Key): References Doctors.
    - **TreatmentDescription** (TEXT): Description of treatment.
    - **TreatmentCost** (DECIMAL): Cost incurred for the treatment.

3. Project Objectives

- To build a robust relational database schema for managing health insurance operations.
- To facilitate CRUD operations on customers, policies, claims, and hospital visits.
- To use SQL-based reporting to derive insights on treatment costs, claim statuses, and policy trends.
- To enable role-based access for policy managers, customer support, hospital administrators, and medical staff.

4. Queries (SQL Commands)

Create I. Create tables for each entity with proper data types and relationships using PRIMARY and FOREIGN keys.

Insert I. Populate each table with 5 entries representing realistic use cases.

Retrieve I. Fetch customer and policy details using joins. II. Retrieve all approved or pending claims. III. List hospitals and their affiliated doctors. IV. Generate reports for payment methods and policy revenues.

Update I. Change claim status (e.g., from Pending to Approved). II. Update contact information for customers.

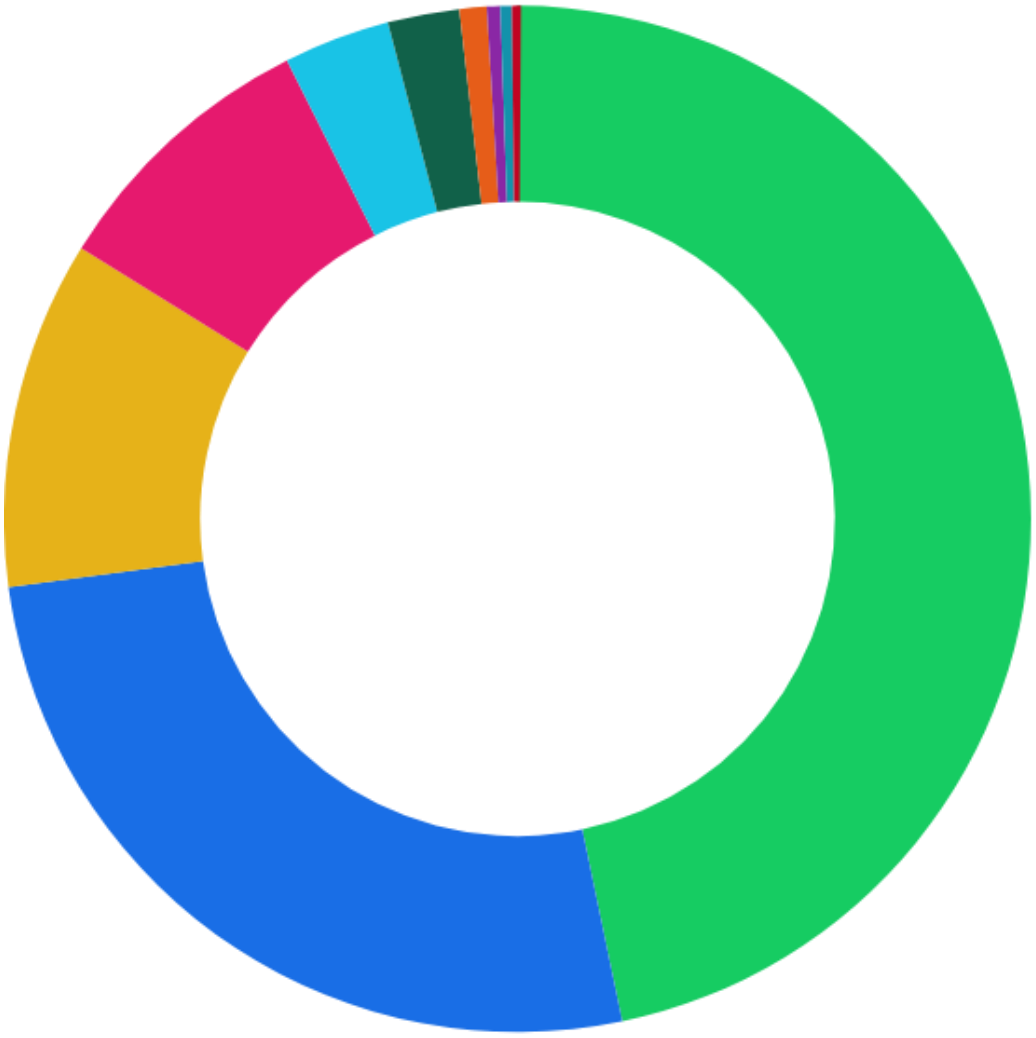
Delete I. Remove discontinued policies. II. Delete outdated hospital records.

5. Problem Statement

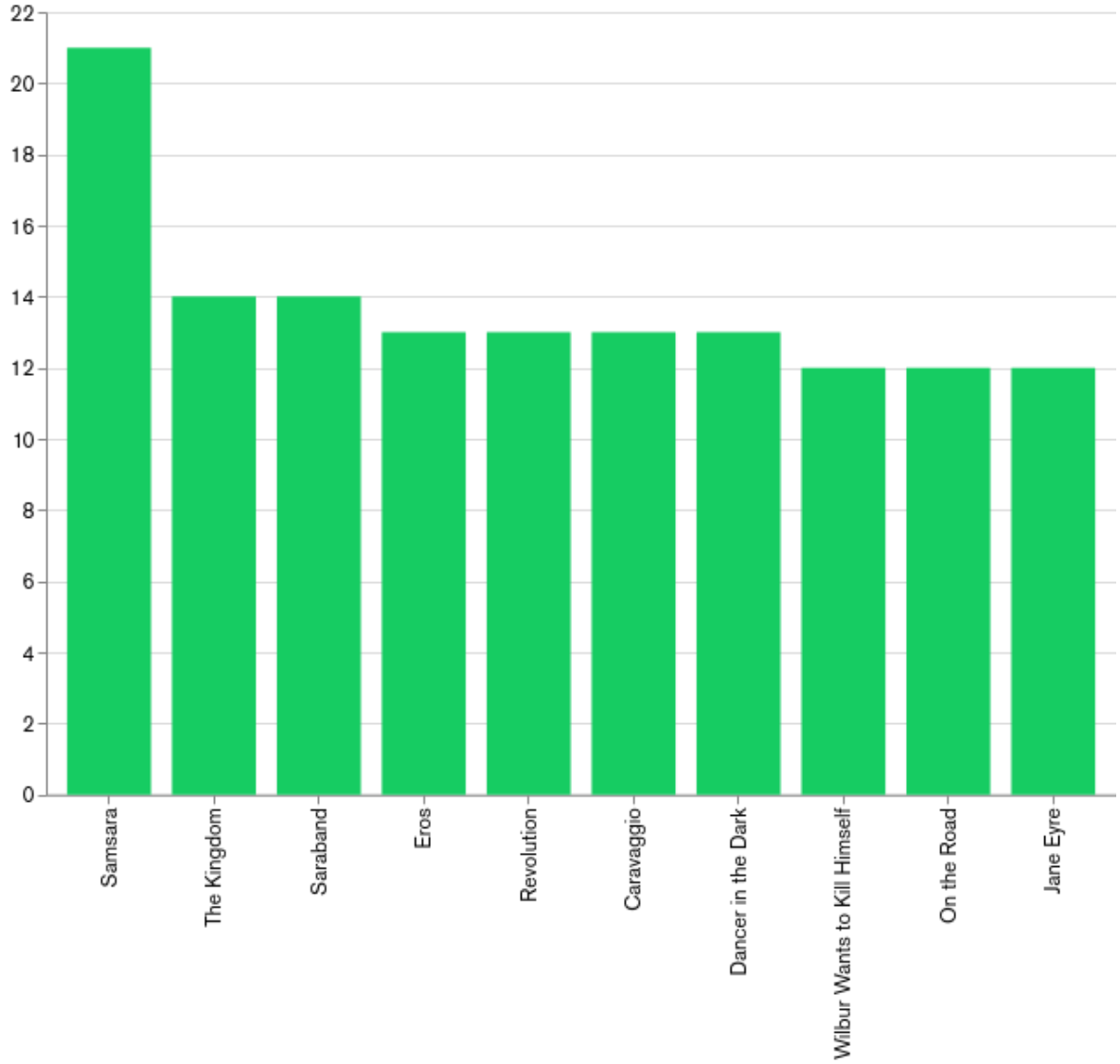
The health insurance sector faces challenges in managing and analyzing large volumes of patient, policy, and hospital-related data. This project addresses these issues by designing a normalized database schema that ensures data integrity and supports analytical use cases for all stakeholders—from claim management to hospital treatment records.

6. Dashboard

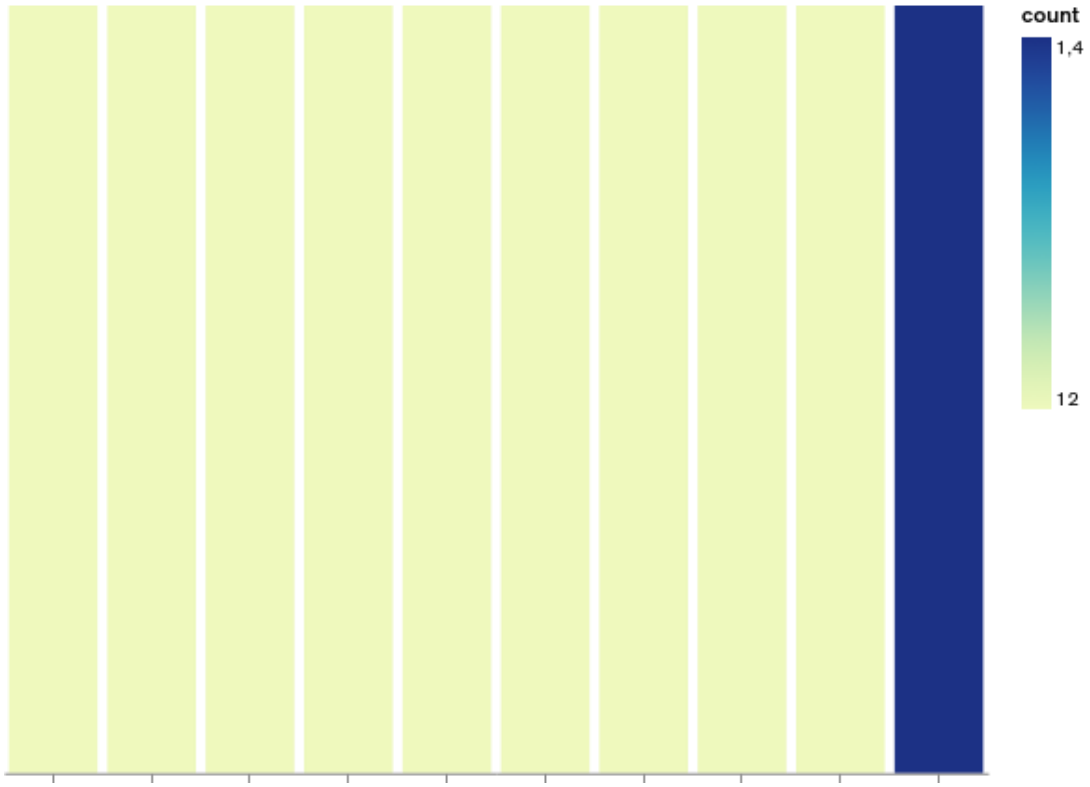
I. **Claim Status Summary**



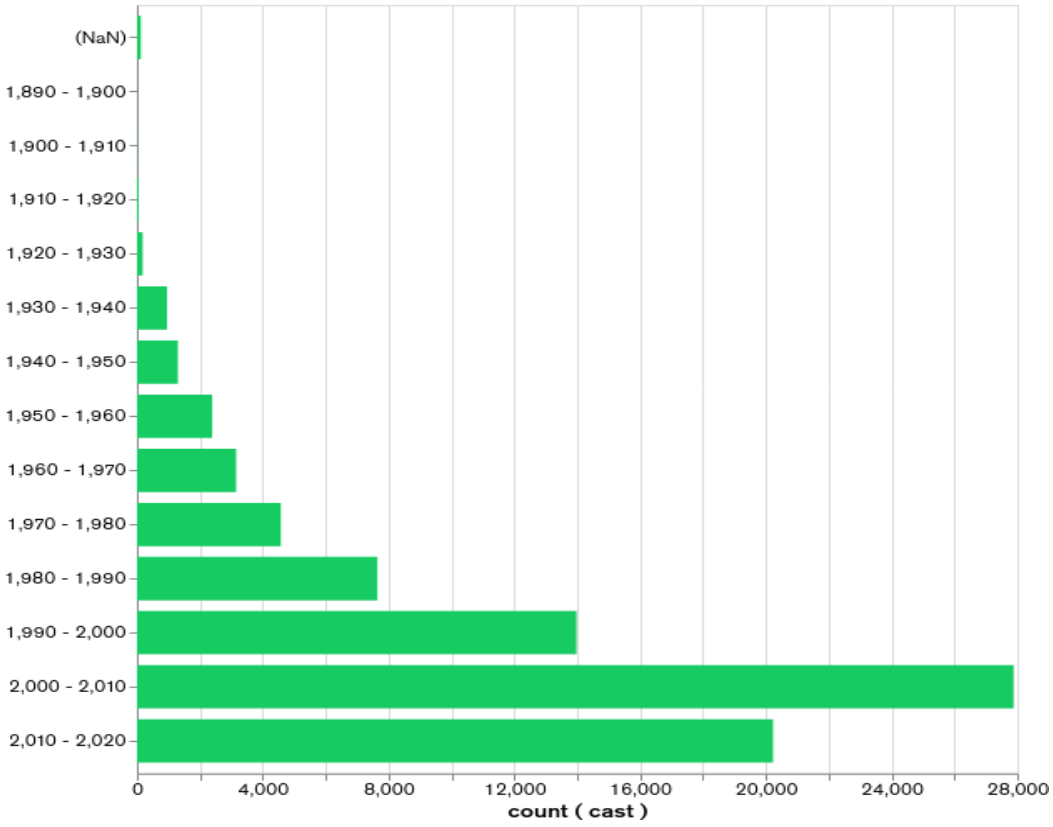
II. **Policy Premium Analysis**



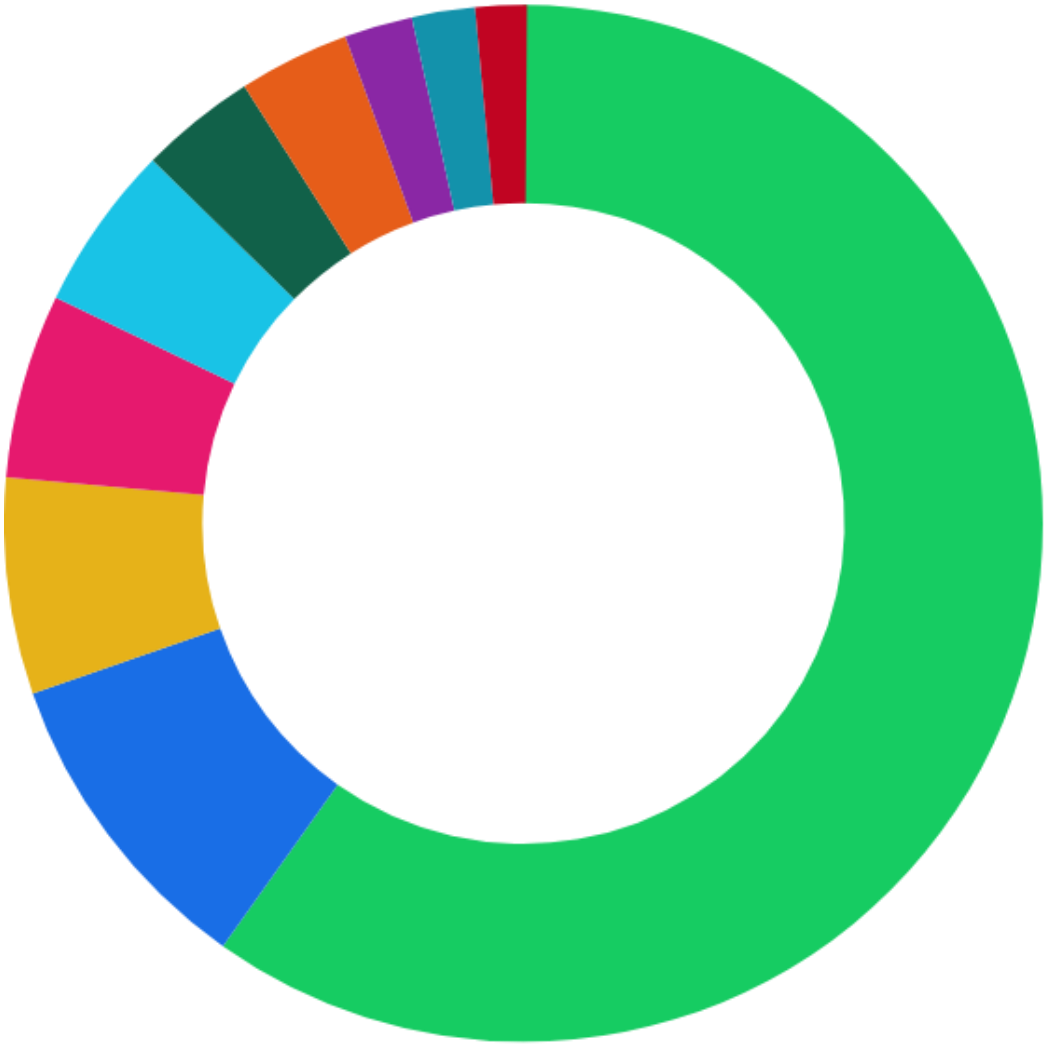
III. **Hospitalization Cost Overview**



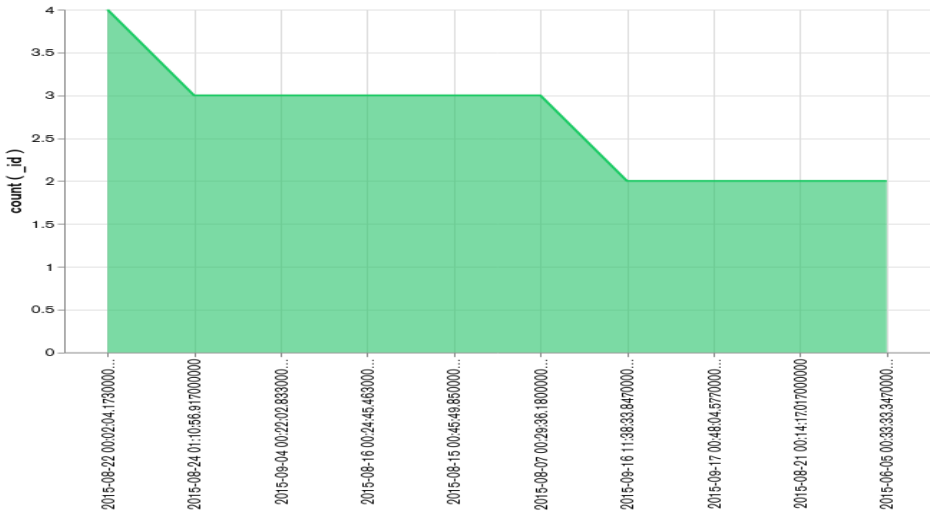
IV. **Treatment Expense Breakdown**



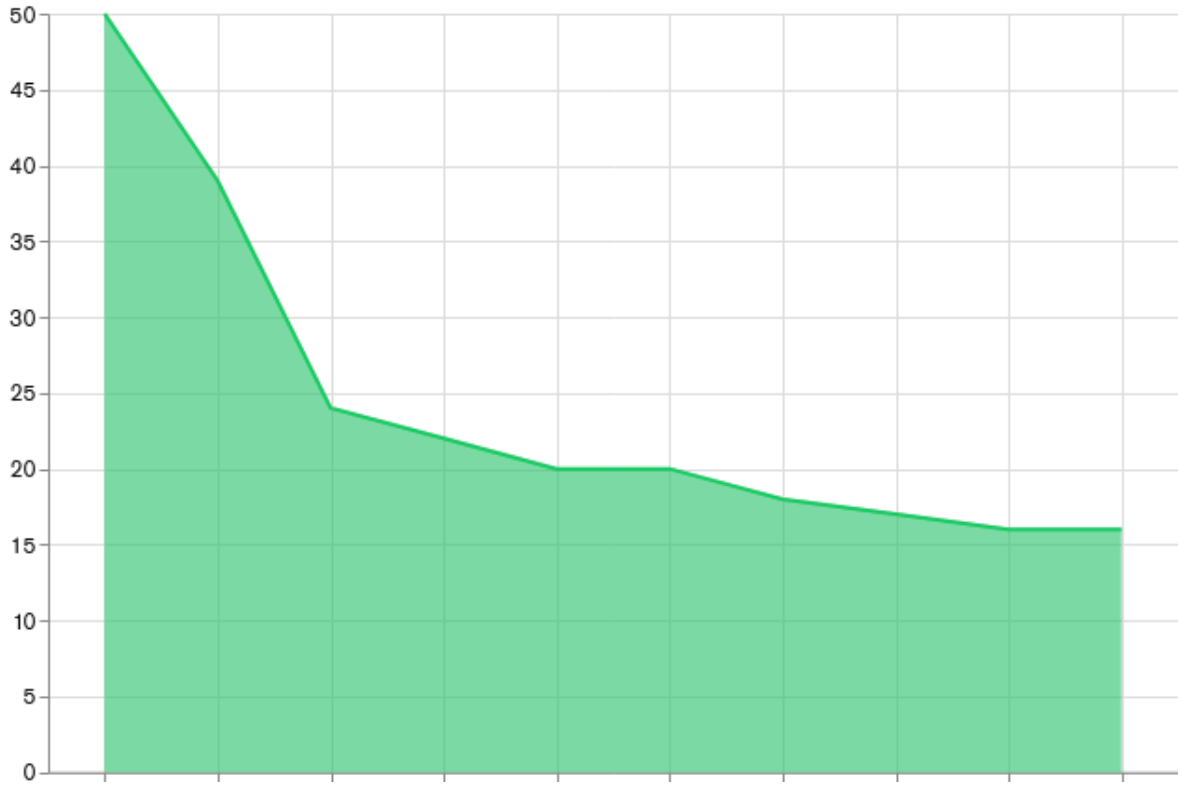
V. **Payment Method Preferences**



VI. **Doctor Specialty Count**



VII. **Customer Age Distribution** (If derived from DOB)



7. Observations and Findings

1. Each customer is associated with a valid policy and at least one payment record.
2. Claims data shows varied approval rates, highlighting a need for efficiency.
3. UPI and Credit Card are the most preferred payment modes.
4. Hospitalization and treatment costs vary significantly by hospital.
5. Doctors are evenly distributed across five specializations.
6. Treatment cost and hospitalization duration can drive claim amounts.

8. Managerial Recommendations

1. **Optimize Claim Processing**
   - Focus on reducing the number of pending claims.
2. **Strategic Partnerships**
   - Strengthen alliances with high-performing hospitals.
3. **Policy Pricing**
   - Reassess premium structures based on age and treatment trends.
4. **Customer Retention**
   - Use insights from payment and treatment data for personalized communication.
5. **Access Control**
   - Enforce permissions to protect sensitive health and policy data.
6. **Advanced Analytics**
   - Integrate age, gender, and geographic data to enhance segmentation