ASSIGNMENT-3

DATABASE MANAGEMENT SYSTEM
CSA0593

SATHISH.R
192324204

Insurance Claims Processing System

System Overview:

The Insurance Claims Processing System is designed to efficiently manage the entire lifecycle of an insurance policy and the associated claims process. It includes tables for policyholders, insurance policies, claims, payments, and adjusters. This system ensures that claims are processed accurately, adhering to policy coverage limits, and provides automated features for submission, approval, and payment of claims.

Database Structure

1. Tables:

Policyholders: This table stores information about the policyholders who have insurance policies with the company.

Column Name	Data Type	Description
policyholder_id	INT	Primary key, unique policyholder ID
name	VARCHAR	Name of the policyholder
date_of_birth	DATE	Date of birth of the policyholder
address	VARCHAR	Address of the policyholder
phone	VARCHAR	Contact phone number of the policyholder
email	VARCHAR	Email address of the policyholder

Policies: This table holds information about the insurance policies offered to the policyholders.

Column Name	Data Type	Description
policy_id	INT	Primary key, unique policy ID
policyholder_id	INT	Foreign key, references Policyholders table
policy_type	VARCHAR	Type of insurance policy (e.g., health, auto)
coverage_limit	DECIMAL	Maximum amount the policy covers
premium	DECIMAL	Premium amount paid by the policyholder
start_date	DATE	Policy start date
end_date	DATE	Policy end date

Claims: This table tracks the claims submitted by policyholders.

Column Name	Data Type	Description
claim_id	INT	Primary key, unique claim ID
policy_id	INT	Foreign key, references Policies table
claim_date	DATE	Date the claim was submitted
claim_amount	DECIMAL	Amount requested for the claim
claim_status	VARCHAR	Status of the claim (e.g., submitted, approved, rejected)
adjuster_id	INT	Foreign key, references Adjusters table

Payments: This table stores information about the payments made for approved claims
.

Column Name	Data Type	Description
payment_id	INT	Primary key, unique payment ID
claim_id	INT	Foreign key, references Claims table
payment_date	DATE	Date of the payment
payment_amount	DECIMAL	Amount paid to the policyholder
payment_status	VARCHAR	Payment status (e.g., processed, pending)

Adjusters: This table stores information about the adjusters who evaluate the claims.

Column Name	Data Type	Description
adjuster_id	INT	Primary key, unique adjuster ID
name	VARCHAR	Name of the adjuster
phone	VARCHAR	Contact phone number of the adjuster
email	VARCHAR	Email address of the adjuster

Constraints:

Policy Coverage Validation: A constraint to ensure the claim amount does not exceed the policy's coverage limit.

sql

```
ALTER TABLE ClaimsADD CONSTRAINT check_claim_amountCHECK (claim_amount <= (SELECT coverage_limit FROM Policies WHERE policy_id = Claims.policy_id));
```

Claim Validity: Ensures that claims are only submitted for policies that are active.

sql

```
ALTER TABLE ClaimsADD CONSTRAINT check_policy_validityCHECK (claim_date BETWEEN (SELECT start_date FROM Policies WHERE policy_id = Claims.policy_id)
                AND (SELECT end_date FROM Policies WHERE policy_id = Claims.policy_id));
```

Stored Procedures:

Create Policy: This procedure adds a new insurance policy for a policyholder.

```
Sql

CREATE PROCEDURE CreatePolicy(IN policyholder_id INT, IN policy_type VARCHAR, IN coverage_limit DECIMAL, IN premium DECIMAL, IN start_date DATE, IN end_date DATE)BEGIN
  INSERT INTO Policies(policyholder_id, policy_type, coverage_limit, premium, start_date, end_date)
  VALUES(policyholder_id, policy_type, coverage_limit, premium, start_date, end_date);END;
```

Submit Claim: This procedure allows a policyholder to submit a claim for an insured event.

```
Sql

CREATE PROCEDURE SubmitClaim(IN policy_id INT, IN claim_amount DECIMAL, IN claim_date DATE)BEGIN
  INSERT INTO Claims(policy_id, claim_amount, claim_date, claim_status)
  VALUES(policy_id, claim_amount, claim_date, 'Submitted');END;
```

Process Payment: This procedure processes a payment for an approved claim.

```
Sql

CREATE PROCEDURE ProcessPayment(IN claim_id INT, IN payment_amount DECIMAL)BEGIN
  UPDATE Claims SET claim_status = 'Approved' WHERE claim_id = claim_id;
  INSERT INTO Payments(claim_id, payment_date, payment_amount, payment_status)
  VALUES(claim_id, NOW(), payment_amount, 'Processed');END;
```

Triggers:

Update Claim Status on Submission: This trigger automatically sets the claim status to 'Submitted' when a new claim is added.

```
Sql

CREATE TRIGGER SetClaimStatusOnSubmit
BEFORE INSERT ON ClaimsFOR EACH ROWBEGIN
  SET NEW.claim_status = 'Submitted';END;
```

Update Claim Status on Approval/Rejection: This trigger automatically updates the claim status when the claim is approved or rejected after evaluation.

```
Sql

CREATE TRIGGER UpdateClaimStatusOnApproval
AFTER UPDATE ON ClaimsFOR EACH ROWBEGIN
  IF NEW.claim_status = 'Approved' THEN
    UPDATE Payments SET payment_status = 'Processed' WHERE claim_id = NEW.claim_id;
  END IF;END;
```

SQL Queries for Reports:

Claim Frequencies: This query generates a report on how frequently claims are submitted by policyholders.

```
Sql

SELECT policyholder_id, COUNT(claim_id) AS claim_countFROM ClaimsGROUP BY policyholder_id;
```

Total Payouts: This query calculates the total amount paid out for approved claims.

```
Sql

SELECT SUM(payment_amount) AS total_payoutsFROM PaymentsWHERE payment_status = 'Processed';
```

Revenue by Policy Type: This query provides revenue by policy type based on the premiums paid.

```
Sql

SELECT policy_type, SUM(premium) AS total_revenueFROM PoliciesGROUP BY policy_type;
```

Adjuster Performance: This query analyzes the performance of adjusters based on the number of claims they have processed.

```
Sql

SELECT adjuster_id, COUNT(claim_id) AS processed_claimsFROM ClaimsWHERE adjuster_id IS NOT NULLGROUP BY adjuster_id;
```

Conclusion:

The Insurance Claims Processing System enables efficient management of policies, claims, payments, and adjuster assignments. The system enforces data integrity with constraints to verify coverage limits and claim validity, ensuring compliance. Stored procedures automate the creation of policies, claim submissions, and payment processing, while triggers streamline claim status updates. SQL queries provide useful insights into claim frequencies, payouts, revenue, and adjuster performance. Overall, this system improves operational efficiency, ensures data accuracy, and facilitates quick decision-making for insurance companies.