Task-12

Mini-Project

2) ER Diagram

2.1 Normalization steps:-

first Normal form :- Ensure atomic attribute values for each table

Second Normal form :- Remove partial dependencies; each non-key attributes must depend on the whole primary key

Third Normal form :- Remove transithe dependences; non-key attrib : depend only on the primary key

Candidate keys examples

- Patient :- patientid (unique)
- policy :- policy Number (unique)
- claim :- claimid (unique)
- Hospital :- Hostapitalid (unique)

2.2 SQL Queries to process, approve and track insurance claims

- Insert new claim :-

```
INSERT INTO claim[claimid, Policyid, Patienid,
Hospital id, claim Number, amount)
VALUES [1001, 'PN123', 1, 101, 22, 2000
```

- Approve a claim

```
update claim SET Status = 'Approval',
Approval Date = '2023-02-01' WHERE Claimid = 1001,
```

- Track claims for a given policy.

```
SELECT "FROM claim' WHERE policy Number = 'PN123';
```

- Calculate total approved claims per patient :

```
SELECT Patientid, Sum(Amount) AS Total Approved
FROM claim
WHERE Status = 'Approved'
GROUP By Patientid;
```

3 Transaction Management during concurrent claim approvals

1. optimistic concurrency control:-

```
UPDATE claim
  SET status = 'Approved', version = version + 1
WHERE claim id = 'uuid-123u' AND status = 'In-Review'
 AND version = 3;
```

2. pessimistic locking

```
BEGIN;
SELECT status FROM claim WHERE claim-id = 'unid-123u'
FOR UPDATE; UPDATE claim SET status = 'Approved' WHERE
claim_id = 'uuid-123u'; COMMIT;
```

3. Idempotency & business-level compensation

Make approval operations idempotent

SUBMITTED → IN-Review → Approved.

4. isolation Levels

Use read commited typically, or REPEATABLE READ

SERMLIZE if strict serializibity is required

4 Performing CRUD operations in MongoDB to Manage claims records.

- Basic MongoDB CRUD operations for claim records:-

- Create:-

```
db.claims.insertOne({
    claimid: 1001,
  Policy Number: 'PN123',
    Patient id: 1,
```

```
"timestamp" : "2025-01-15T08:30:00Z",
  "sensors" : {
  "Temperature" : 23.5,
    "humidity" : 45.5
  };
    "status" : "advice",
    "metadata" : {
    "manufacture" : "AcmeDevices",
    "firmware version" : "1.2.5"
  }
}
```

How indexing steps

```
db.devicesData.createIndex({"deviceId":1});
db.deviceData.CreateIndex({"location.LocationId":1});
db.devicesData.create.index({"devisid":1,"location.
                              .LocationId":1});
```

Benefits

- Fetch all records of a device in different Location
- Query all devices in a location
- Retrive temperature readings across all thermostats

Result

Thus the user case is successfully verified and executed successfully