MODULE 4.4 PRACTICAL PROJECT ASSIGNMENT

4.4.1 Creating Database:

create database insdb;

use insdb;

4.4.2 Creating Tables:

```
-- agents table
create table agents(
agentID INT,
AgentName varchar(30),
phone varchar(12),
city varchar(20)
primary key(agentID)
)
-- customer table
create table customers(
customerID int primary key,
firstname varchar(30),
lastname varchar(30),
dob date,
phone varchar(10),
email varchar(30)
)
-- policies table
create table policies(
policyId int primary key,
policyName varchar(30),
policyType varchar(30),
PremiumAmount money,
durationyears int
)
```

```
-- claims table
create table claims(
claimId int primary key,
AssignemntId INT,
cliamdate date,
cliamAmount money,
claimStatus varchar(30)
)

-- policyAssignments table
create table PolicyAssignments(
assignmentId int primary key,
customerid int,
policyid int,
agentid int,
startdate date,
enddate date
foreign key(customerid) references customers(customerID),
foreign key(policyid) references policies(policyID),
foreign key(agentid) references agents(agentID)

)

ALTER TABLE claims
ADD CONSTRAINT FK_Claims_Assignment
FOREIGN KEY (assignemntId)
REFERENCES PolicyAssignments(assignmentId);
```

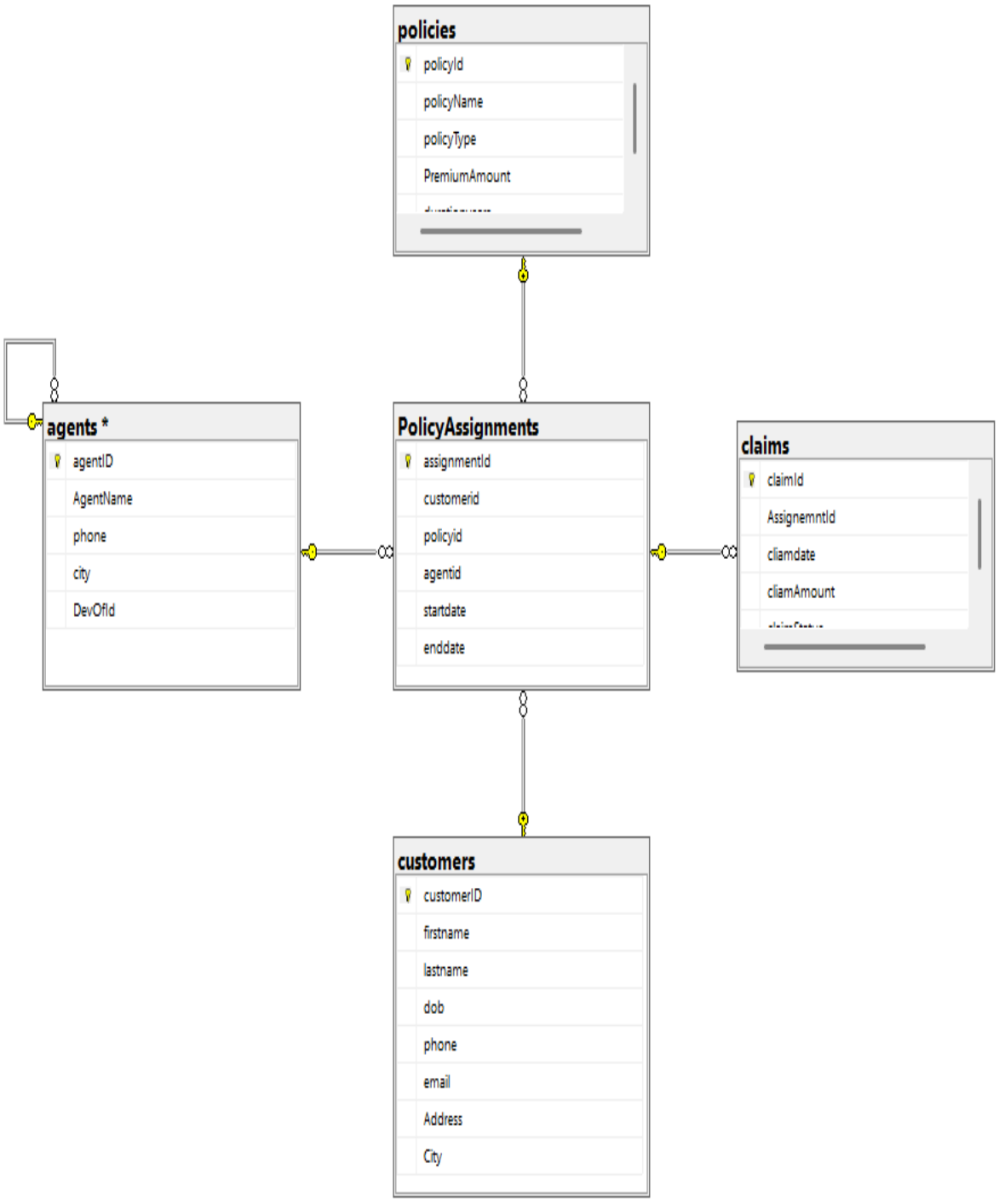

Fig 4.4.1 a Insurance DB Schema

4.4.3 Inserting Values into all the Tables

```
-- inserting values into tables
insert into customers values(101,'amit','sharma','1990-05-10','7659831938','amit@gmail.com'),
(102,'shiva','kumar','2005-10-19','9999999999','shiva@gmail.com');

INSERT INTO agents
VALUES
(1, 'Ramesh Kumar', '9876543210', 'Hyderabad'),
(2, 'Suresh Rao',  '9123456789', 'Bangalore'),
(3, 'Anita Singh', '9012345678', 'Mumbai');
INSERT INTO customers
VALUES

(103, 'Neha',  'Verma',  '1998-03-22', '7766554433', 'neha@gmail.com');

INSERT INTO policies VALUES
(201, 'Life Secure',  'Life',   25000.00, 20),
(202, 'Health Plus',  'Health', 15000.00, 10),
(203, 'Car Protect',  'Vehicle',12000.00, 5);

Insert Into policies values (204, 'Bike Protect','Vehicle',12000,1);

INSERT INTO PolicyAssignments VALUES
(301, 101, 201, 1, '2022-01-01', '2042-01-01'),
(302, 102, 202, 2, '2023-06-01', '2033-06-01'),
(303, 103, 203, 3, '2024-03-15', '2029-03-15');

INSERT INTO claims VALUES
(401, 301, '2023-08-10', 50000.00, 'Approved'),
(402, 302, '2024-01-05', 20000.00, 'Pending'),
(403, 303, '2024-06-20', 15000.00, 'Rejected');
```

4.4.4 Select commands:

```
-- 1. View all customers
SELECT * FROM Customers;

-- 2. View policy assignments
SELECT CustomerID, PolicyID, StartDate, EndDate
FROM PolicyAssignments;

-- 3. Health policies
SELECT * FROM Policies
WHERE PolicyType = 'Health';

-- 4. Premium > 10000 and duration = 1
SELECT * FROM Policies
WHERE PremiumAmount > 10000 AND DurationYears = 1;

-- 5. Distinct agent cities
SELECT DISTINCT City FROM Agents;
```

4.4.5. WHERE, IN, BETWEEN, LIKE:

```
-- Using IN
SELECT PolicyID, PolicyName
FROM Policies
WHERE PolicyType IN ('Health', 'Life', 'Vehicle');

-- Using BETWEEN
SELECT * FROM Customers
WHERE DOB BETWEEN '2001-01-01' AND '2020-12-31';

-- Using LIKE
SELECT * FROM Agents
WHERE City LIKE '_a%';
```

4.4.6 AGGREGATE FUNCTIONS:

```
-- Max and Min claim amount
SELECT
   MAX(ClaimAmount) AS MaxClaim,
   MIN(ClaimAmount) AS MinClaim
FROM Claims;


-- Latest claim
SELECT TOP 1 *
FROM Claims
ORDER BY ClaimDate DESC;
```

4.4. 7. UPDATE & DELETE:

```
-- Increase health policy premium by 10%
UPDATE Policies
SET PremiumAmount = PremiumAmount * 1.10
WHERE PolicyType = 'Health';


-- Delete expired policies
DELETE FROM PolicyAssignments
WHERE EndDate < GETDATE();
```

4.4.8 COMPUTED COLUMNS:

```
SELECT PolicyID,
      PolicyName,
      PremiumAmount,
      PremiumAmount * 0.06 AS LocalTax,
      PremiumAmount * 1.06 AS PremiumWithTax,
      (PremiumAmount * 1.06) / 12 AS MonthlyPremium
FROM Policies;
```

4.4.9 Offset & Fetch:

```
SELECT *
FROM Customers
ORDER BY CustomerID
OFFSET 1 ROWS
FETCH NEXT 2 ROWS ONLY;
```

4.4.10 JOINS:

```
-- Customers with policies
SELECT c.FirstName, p.PolicyName
FROM Customers c
JOIN PolicyAssignments pa ON c.CustomerID = pa.CustomerID
JOIN Policies p ON pa.PolicyID = p.PolicyID;

-- Claims with customer details
SELECT c.FirstName, cl.ClaimAmount, cl.ClaimStatus
FROM Claims cl
JOIN PolicyAssignments pa ON cl.AssignmentID = pa.AssignmentID
JOIN Customers c ON pa.CustomerID = c.CustomerID;
```

4.4.11 GROUPING AND HAVING:

```
-- Total claim amount per customer
SELECT c.FirstName,
    SUM(cl.ClaimAmount) AS TotalClaimAmount
FROM Customers c
JOIN PolicyAssignments pa ON c.CustomerID = pa.CustomerID
JOIN Claims cl ON pa.AssignmentID = cl.AssignmentID
GROUP BY c.FirstName;
```

```
-- Customers with claims > 50000
SELECT c.FirstName,
    SUM(cl.ClaimAmount) AS TotalClaimAmount
FROM Customers c
JOIN PolicyAssignments pa ON c.CustomerID = pa.CustomerID
JOIN Claims cl ON pa.AssignmentID = cl.AssignmentID
GROUP BY c.FirstName
HAVING SUM(cl.ClaimAmount) > 50000;
```

4.4.12 SQL FUNCTIONS:

12.1 DATE FUNCTIONS

```
-- 1. Get current system date
SELECT GETDATE() AS CurrentDate;

-- 2. Calculate age of customers
SELECT CustomerID,
    FirstName,
    DOB,
    DATEDIFF(YEAR, DOB, GETDATE()) AS Age
FROM Customers;

-- 3. Add policy duration to start date
SELECT AssignmentID,
    StartDate,
    DATEADD(YEAR, 1, StartDate) AS OneYearLater
FROM PolicyAssignments;

-- 4. Extract year and month from claim date
SELECT ClaimID,
    ClaimDate,
    YEAR(ClaimDate) AS ClaimYear,
    MONTH(ClaimDate) AS ClaimMonth FROM Claims;
```

12.2 STRING FUNCTIONS

```
-- 1. Convert customer names to uppercase
SELECT CustomerID,
    UPPER(FirstName) AS UpperName
FROM Customers;

-- 2. Combine first name and last name
SELECT CustomerID,
    CONCAT(FirstName, ' ', LastName) AS FullName
FROM Customers;

-- 3. Length of policy name
SELECT PolicyID,
    PolicyName,
    LEN(PolicyName) AS NameLength
FROM Policies;

-- 4. Extract first 4 characters of policy type
SELECT PolicyID,
    PolicyType,
    LEFT(PolicyType, 4) AS ShortType
FROM Policies;
```

12.3 NUMERIC FUNCTIONS

```
-- 1. Round premium amount
SELECT PolicyID,
    PremiumAmount,
    ROUND(PremiumAmount, 0) AS RoundedPremium
FROM Policies;
```

```
-- 2. Absolute claim amount
SELECT ClaimID,
    ABS(ClaimAmount) AS AbsoluteClaim
FROM Claims;

-- 3. Average premium amount
SELECT AVG(PremiumAmount) AS AveragePremium
FROM Policies;

-- 4. Total premium collection
SELECT SUM(PremiumAmount) AS TotalPremium
FROM Policies;
```

4.4.13 CASE – ELSE STATEMENT

```
-- Categorize policies based on premium amount
SELECT PolicyID,
    PolicyName,
    PremiumAmount,
    CASE
        WHEN PremiumAmount >= 20000 THEN 'High Premium'
        WHEN PremiumAmount BETWEEN 12000 AND 19999 THEN 'Medium Premium'
        ELSE 'Low Premium'
    END AS PremiumCategory
FROM Policies;

-- Display claim decision message using CASE
SELECT ClaimID,
    ClaimAmount,
    ClaimStatus,
    CASE
        WHEN ClaimStatus = 'Approved' THEN 'Claim Accepted'
        WHEN ClaimStatus = 'Pending'  THEN 'Under Review'
```

```
        ELSE 'Claim Rejected'
    END AS ClaimRemark
FROM Claims;
```

4.4.14 MERGE COMMAND

```
-- Create a dummy table to demonstrate MERGE
CREATE TABLE Claims_Backup (
    ClaimID INT PRIMARY KEY,
    AssignmentID INT,
    ClaimDate DATE,
    ClaimAmount MONEY,
    ClaimStatus VARCHAR(30)
);

-- MERGE: Insert new records or update existing records
MERGE Claims_Backup AS Target
USING Claims AS Source
ON Target.ClaimID = Source.ClaimID

WHEN MATCHED THEN
    UPDATE SET
        Target.ClaimAmount = Source.ClaimAmount,
        Target.ClaimStatus = Source.ClaimStatus

WHEN NOT MATCHED THEN
    INSERT (ClaimID, AssignmentID, ClaimDate, ClaimAmount, ClaimStatus)
    VALUES (Source.ClaimID, Source.AssignmentID, Source.ClaimDate,
            Source.ClaimAmount, Source.ClaimStatus);

-- View merged data
SELECT * FROM Claims_Backup;
```

4.4.15 GROUP BY WITH ROLLUP

```
-- Total claim amount per customer with grand total
SELECT c.FirstName,
    SUM(cl.ClaimAmount) AS TotalClaimAmount
FROM Customers c
JOIN PolicyAssignments pa ON c.CustomerID = pa.CustomerID
JOIN Claims cl ON pa.AssignmentID = cl.AssignmentID
GROUP BY ROLLUP (c.FirstName);

-- Count policies handled by agents with grand total
SELECT a.AgentName,
    COUNT(pa.PolicyID) AS TotalPolicies
FROM Agents a
JOIN PolicyAssignments pa ON a.AgentID = pa.AgentID
GROUP BY ROLLUP (a.AgentName);
```