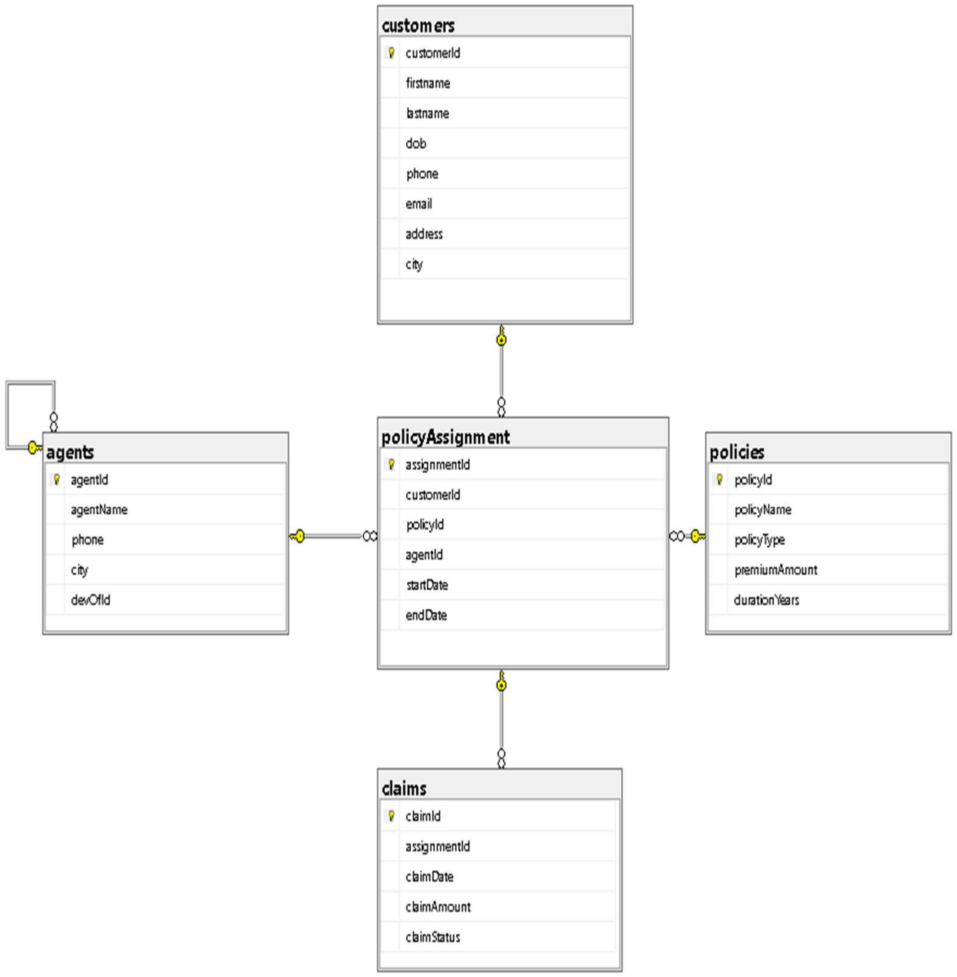
Module -4.4: Practical Project Assignment

Database creation:

1.Create database InsuranceDb;

Schema:



Create tables:

1.Customer Table:

create table customers(customerId int primary key,firstname varchar(50) ,lastname varchar(50),dob date,phone varchar(20),email varchar(50));

2.Policies Table:

create table policies(policyId int primary key,policyName varchar(20) not null,policyType varchar(20),premiumAmount decimal(20,2),durationYears int);

3.Agents:

create table agents(agentId int primary key,agentName varchar(20) not null,phone varchar(20),city varchar(20));

4.PolicyAssignment:

create table policyAssignment(assignmentId int primary key,customerId int references customers(customerId),policyId int references policies(policyId),agentId int references agents(agentId),startDate date,endDate date);

5.Claims:

create table claims(claimId int primary key,assignmentId int references policyAssignment(assignmentId),claimDate date, claimAmount decimal(20,2),claimStatus varchar(10));

Insert commands:

1.Into Customer table:

INSERT INTO customers VALUES

(1,'Ravi','Kumar','1995-06-12','9876543210','ravi@gmail.com'),

(2,'Anita','Sharma','1992-03-25','9123456780','anita@gmail.com'),

(3,'Suresh','Reddy','1988-11-10','9988776655','suresh@gmail.com'),

(4,'Priya','Singh','1996-08-14','9012345678','priya@gmail.com'),

(5,'Amit','Verma','1990-01-30','9090909090','amit@gmail.com'),

(6,'Neha','Gupta','1994-12-05','9888888888','neha@gmail.com'),

(7,'Kiran','Patel','1987-07-19','9777777777','kiran@gmail.com'),

(8,'Pooja','Nair','1993-05-22','9666666666','pooja@gmail.com'),

(9,'Rahul','Das','1991-09-09','9555555555','rahul@gmail.com'),

(10,'Sneha','Iyer','1997-04-03','9444444444','sneha@gmail.com');

2.Into Policies table:

```
INSERT INTO policies VALUES
(101,'Life Secure','Life',25000.00,20),
(102,'Health Plus','Health',18000.00,10),
(103,'Car Protect','Vehicle',12000.00,5),
(104,'Home Shield','Property',15000.00,15),
(105,'Travel Safe','Travel',8000.00,2),
(106,'Child Future','Life',20000.00,18),
(107,'Senior Care','Health',22000.00,8),
(108,'Bike Guard','Vehicle',6000.00,3),
(109,'Term Life','Life',30000.00,25),
(110,'Family Health','Health',28000.00,12);
```

3.Into Agents table:

```
INSERT INTO agents VALUES
(201,'Arjun','9000011111','Hyderabad'),
(202,'Meena','9000022222','Bangalore'),
(203,'Rahul','9000033333','Chennai'),
(204,'Suman','9000044444','Delhi'),
(205,'Rohit','9000055555','Mumbai'),
(206,'Anjali','9000066666','Pune'),
(207,'Vikas','9000077777','Kolkata'),
(208,'Divya','9000088888','Kochi'),
(209,'Nitin','9000099999','Jaipur'),
(210,'Swathi','9000000000','Vizag');
```

4.Into PolicyAssignment table:

```
INSERT INTO policyAssignment VALUES
(301,1,101,201,'2022-01-01','2042-01-01'),
(302,2,102,202,'2023-05-10','2033-05-10'),
(303,3,103,203,'2024-02-15','2029-02-15'),
(304,4,104,204,'2021-07-01','2036-07-01'),
(305,5,105,205,'2023-11-20','2025-11-20'),
(306,6,106,206,'2020-03-18','2038-03-18'),
(307,7,107,207,'2022-09-25','2030-09-25'),
(308,8,108,208,'2024-01-10','2027-01-10'),
(309,9,109,209,'2019-06-05','2044-06-05'),
(310,10,110,210,'2023-04-12','2035-04-12');
```

5.Into Claims table:

```
INSERT INTO claims VALUES
(401,301,'2023-02-15',40000.00,'Approved'),
(402,302,'2024-01-20',50000.00,'Approved'),
(403,303,'2024-08-05',15000.00,'Pending'),
(404,304,'2022-12-10',30000.00,'Rejected'),
(405,305,'2024-05-18',10000.00,'Approved'),
(406,306,'2021-06-25',45000.00,'Approved'),
(407,307,'2023-11-02',25000.00,'Pending'),
(408,308,'2024-03-14',8000.00,'Approved'),
(409,309,'2020-10-30',60000.00,'Rejected'),
(410,310,'2024-09-01',35000.00,'Pending');
```

Basic to Advanced Queries:

Select Queries:

1. Find all customer details

```
SELECT * FROM customers;
```

2. Find customer first name, last name, and email

```
SELECT firstname, lastname, email FROM customers;
```

3. Find policy names and their premium amounts

```
SELECT policyName, premiumAmount FROM policies;
```

4. Find agent names and their cities

```
SELECT agentName, city FROM agents;
```

5. Find assignment ID with policy start and end dates

```
SELECT assignmentId, startDate, endDate FROM policyAssignment;
```

Update Queries:

1. Increase premium by 10% for Health policies

```
UPDATE policies

SET premiumAmount = premiumAmount * 1.10

WHERE policyType = 'Health';
```

2. Update phone number of a specific customer

```
UPDATE customers

SET phone = '9999999999'

WHERE customerId = 1;
```

3. Approve a specific claim

```
UPDATE claims

SET claimStatus = 'Approved'

WHERE claimId = 403;
```

4. Change city of an agent

```
UPDATE agents
```

```
SET city = 'Delhi'
WHERE agentId = 205;
```

5. Extend policy end date

```
UPDATE policyAssignment
SET endDate = '2030-12-31'
WHERE assignmentId = 301;
```

Group By:

1. Find number of policies per policy type

```
SELECT policyType, COUNT(*)
FROM policies
GROUP BY policyType;
```

2. Find number of policies sold by each agent

```
SELECT agentId, COUNT(*)
FROM policyAssignment
GROUP BY agentId;
```

3. Find number of policies per customer

```
SELECT customerId, COUNT(*)
FROM policyAssignment
GROUP BY customerId;
```

4. Find claim count per status

```
SELECT claimStatus, COUNT(*)
FROM claims
GROUP BY claimStatus;
```

5. Find agent count per city

```
SELECT city, COUNT(*)
FROM agents
GROUP BY city;
```

Alter:

1. Add address column to customers table

```
ALTER TABLE customers

ADD address VARCHAR(100);
```

2. Add city column to customers table

```
ALTER TABLE customers

ADD city VARCHAR(50);
```

3. Modify premiumAmount datatype in policies table

```
ALTER TABLE policies

ALTER COLUMN premiumAmount DECIMAL(15,2);
```

4. Drop phone column from agents table

```
ALTER TABLE agents

DROP COLUMN phone;
```

5. Add check constraint to ensure claimAmount is positive

```
ALTER TABLE claims

ADD CONSTRAINT chk_claimAmount

CHECK (claimAmount > 0);
```

Set operations:

1. Find all unique customer IDs who either bought a policy or made a claim

```
SELECT customerId FROM policyAssignment

UNION

SELECT pa.customerId

FROM policyAssignment pa

JOIN claims c ON pa.assignmentId = c.assignmentId;
```

2. Find customers who bought a policy AND made a claim

```
SELECT customerId FROM policyAssignment

INTERSECT

SELECT pa.customerId

FROM policyAssignment pa

JOIN claims c ON pa.assignmentId = c.assignmentId;
```

3. Find customers who bought policies but NEVER made a claim

```
SELECT customerId FROM policyAssignment

EXCEPT

SELECT pa.customerId

FROM policyAssignment pa

JOIN claims c ON pa.assignmentId = c.assignmentId;
```

4. Find all cities from agents and customers (after adding city to customers)

```
SELECT city FROM agents

UNION

SELECT city FROM customers;
```

5. Find agents who never handled any policy assignment

```
SELECT agentId FROM agents
```

```
EXCEPT

SELECT agentId FROM policyAssignment;
```

Aggregate Functions:

16. Find total number of customers

```
SELECT COUNT(*) FROM customers;
```

17. Find average premium amount

```
SELECT AVG(premiumAmount) FROM policies;
```

18. Find highest and lowest claim amount

```
SELECT MAX(claimAmount), MIN(claimAmount) FROM claims;
```

19. Find total approved claim amount

```
SELECT SUM(claimAmount)

FROM claims

WHERE claimStatus = 'Approved';
```

20. Find number of unique agent cities

```
SELECT COUNT(DISTINCT city) FROM agents;
```

Date Functions:

1. Find year of birth of customers

```
SELECT customerId, YEAR(dob) FROM customers;
```

2. Find duration (in years) of each policy assignment

```
SELECT assignmentId,

DATEDIFF(YEAR, startDate, endDate)

FROM policyAssignment;
```

3. Find claims made after Jan 1, 2024

```
SELECT *

FROM claims

WHERE claimDate >= '2024-01-01';
```

4. Find today's date

SELECT GETDATE();

5. Find expired policy assignments

SELECT assignmentId

FROM policyAssignment

WHERE endDate < GETDATE();

String Functions:

1. Convert customer names to uppercase

SELECT UPPER(firstname) FROM customers;

2. Find full name of customers

SELECT CONCAT(firstname,' ',lastname) AS FullName

FROM customers;

3. Find first 3 letters of agent cities

SELECT LEFT(city,3) FROM agents;

4. Find length of each policy name

SELECT LEN(policyName) FROM policies;

5. Find customers whose name starts with 'A'

SELECT firstname

FROM customers

WHERE firstname LIKE 'A%';

Numeric Functions:

1. Round premium amount

SELECT ROUND(premiumAmount,0) FROM policies;

2. Find absolute difference from 50,000 claim amount

SELECT ABS(claimAmount - 50000) FROM claims;

3. Find monthly premium (rounded up)

SELECT CEILING(premiumAmount/12) FROM policies;

4. Find monthly premium (rounded down)

```
SELECT FLOOR(premiumAmount/12) FROM policies;
```

5. Find square of policy duration

```
SELECT POWER(durationYears,2) FROM policies;
```

Case When:

1. Categorize policies as High or Low premium

```
SELECT policyName,
CASE WHEN premiumAmount > 20000 THEN 'High' ELSE 'Low' END
FROM policies;
```

2. Mark claim approval symbol

```
SELECT claimId,
CASE claimStatus WHEN 'Approved' THEN 'Yes' ELSE 'No' END
FROM claims;
```

3. Categorize agents by city

```
SELECT agentName,
CASE WHEN city='Delhi' THEN 'North' ELSE 'Other' END
FROM agents;
```

4. Categorize customers by age group

```
SELECT customerId,
CASE WHEN YEAR(dob)<1990 THEN 'Senior' ELSE 'Young' END
FROM customers;
```

5. Categorize policy duration

```
SELECT policyId,
CASE WHEN durationYears>=10 THEN 'Long Term' ELSE 'Short Term' END
FROM policies;
```

Sub Queries:

1. Find customers who bought at least one policy

```
SELECT *
FROM customers
WHERE customerId IN (
  SELECT customerId FROM policyAssignment
);
```

2. Find policies never assigned

```
SELECT *
FROM policies
WHERE policyId NOT IN (
  SELECT policyId FROM policyAssignment
);
```

3. Find claims above average amount

```
SELECT *
FROM claims
WHERE claimAmount > (
  SELECT AVG(claimAmount) FROM claims
);
```

4. Find agents who sold more than one policy

```
SELECT *
FROM agents
WHERE agentId IN (
  SELECT agentId
  FROM policyAssignment
  GROUP BY agentId
  HAVING COUNT(*)>1
);
```

5. Find first customer who bought a policy

```
SELECT *
FROM customers
WHERE customerId = (
  SELECT TOP 1 customerId
  FROM policyAssignment-
  ORDER BY startDate
);
```

Joins:

1. Find customers and their policies

```
SELECT c.firstname, p.policyName
FROM customers c
JOIN policyAssignment pa ON c.customerId=pa.customerId
JOIN policies p ON pa.policyId=p.policyId;
```

2. Find agents and their customers

```
SELECT a.agentName, c.firstname
FROM agents a
JOIN policyAssignment pa ON a.agentId=pa.agentId
JOIN customers c ON pa.customerId=c.customerId;
```

3. Find customers and claim amounts

```
SELECT c.firstname, cl.claimAmount
FROM customers c
JOIN policyAssignment pa ON c.customerId=pa.customerId
JOIN claims cl ON pa.assignmentId=cl.assignmentId;
```

4. Find customers with or without policies

```
SELECT c.firstname, p.policyName
FROM customers c
LEFT JOIN policyAssignment pa ON c.customerId=pa.customerId
LEFT JOIN policies p ON pa.policyId=p.policyId;
```

5. Find policies with or without assignments

```
SELECT p.policyName, pa.assignmentId

FROM policies p

LEFT JOIN policyAssignment pa ON p.policyId=pa.policyId;
```

Advanced Grouping:

1. Policy count using ROLLUP

```
SELECT policyType, COUNT(*)

FROM policies

GROUP BY ROLLUP(policyType);
```

2. Policy count using CUBE

```
SELECT policyType, durationYears, COUNT(*)

FROM policies

GROUP BY CUBE(policyType, durationYears);
```

3. Policy count using GROUPING SETS

```
SELECT policyType, durationYears, COUNT(*)

FROM policies

GROUP BY GROUPING SETS

(

 (policyType),

 (durationYears),

 (policyType, durationYears)

);
```

Views:

1. Create a view to see customer policy details

```
CREATE VIEW vw_CustomerPolicies AS
SELECT c.firstname, c.lastname, p.policyName, pa.startDate, pa.endDate
FROM customers c
JOIN policyAssignment pa ON c.customerId = pa.customerId
JOIN policies p ON pa.policyId = p.policyId;
```

2. Display data from Customer Policies view

```
SELECT * FROM vw_CustomerPolicies;
```

3. Create a view for claims report

```
CREATE VIEW vw_ClaimsReport AS
SELECT c.firstname, p.policyName, cl.claimAmount, cl.claimStatus, cl.claimDate
FROM claims cl
JOIN policyAssignment pa ON cl.assignmentId = pa.assignmentId
JOIN customers c ON pa.customerId = c.customerId
JOIN policies p ON pa.policyId = p.policyId;
```

4. Display approved claims using the view

```
SELECT *
FROM vw_ClaimsReport
WHERE claimStatus = 'Approved';
```

5. Create a view showing agent-wise policy count

```
CREATE VIEW vw_AgentPolicyCount AS
SELECT a.agentName, COUNT(pa.policyId) AS policyCount
FROM agents a
JOIN policyAssignment pa ON a.agentId = pa.agentId
GROUP BY a.agentName;
```

Indexing:

70. Create an index on customer email for faster search

(Used when searching customers by email)

```
CREATE INDEX idx_customers_email
ON customers(email);
```

71. Create an index on policy type for filtering policies

(Used in WHERE policyType = 'Health')

```
CREATE INDEX idx_policies_policyType
ON policies(policyType);
```

72. Create an index on agent city

(Used to quickly find agents by city)

```
CREATE INDEX idx_agents_city
ON agents(city);
```