Module 4.4. Practical Project Assignment

INSURANCE DATABASE

Database Creation:

Create database insurance

Use insurance;

Tables Creation:

CUSTOMERS TABLE:

Create table customers(

CustomerID int identity primary key,

FirstName varchar(20),

LastName varchar(20),

DateOfBirth date,

Phone varchar(20),

Email varchar(100) unique

);

POLICIES:

create table policies(

policyid int identity primary key,

policyname varchar(50),

policytype varchar(50),

premiumamount decimal(10,2),

durationyears int

);

AGENTS:

```
create table agents(
agentid int identity primary key,
agentname varchar(50),
phone varchar(20),
city varchar(20)
);
```

PolicyAssignments:

```
create table policyassignments(
assignmentid int identity primary key,
customerid int,
policyid int,
agentid int,
startdate date,
enddate date
constraint fk_customers_assignment
foreign key (customerid) references customers(customerid),
constraint fk_policies_assignment
foreign key (policyid) references policies(policyid),
constraint fk_agents_assignment
foreign key (agentid) references agents(agentid)
);
```

CLAIMS:

```
create table claims(
claimid int identity primary key,
assignmentid int,
```

claimdate date,

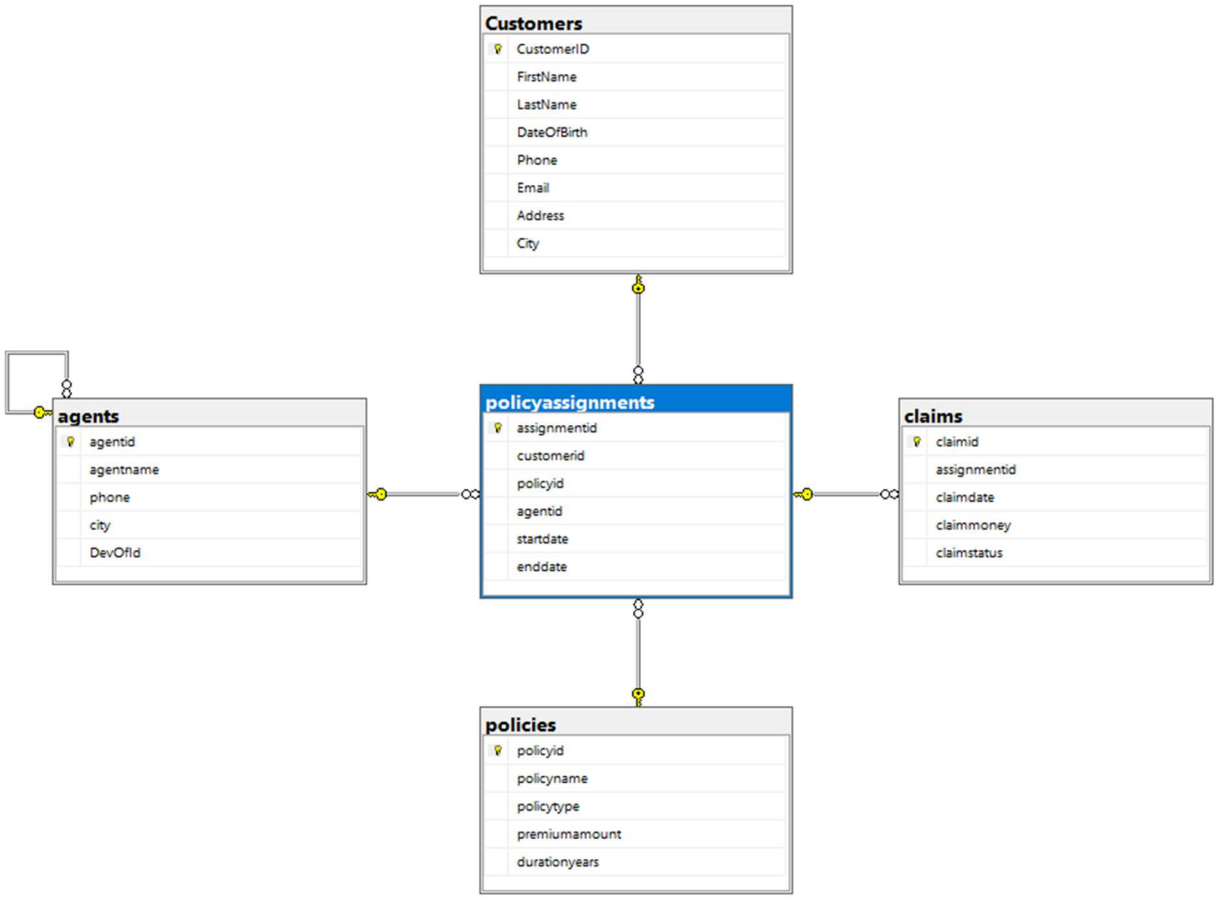
claimmoney decimal(10,2),

claimstatus varchar(20)

constraint fk_assignment_claims

foreign key (assignmentid) references policyassignments(assignmentid)

);



Insertion commands:

Customers:

INSERT INTO customers (FirstName, LastName, DateOfBirth, Phone, Email) VALUES

('Amit', 'Sharma', '1992-05-14', '9876543210', 'amit.sharma@gmail.com'),

('Priya', 'Reddy', '1995-08-22', '9123456780', 'priya.reddy@gmail.com'),

('Rahul', 'Verma', '1988-12-03', '9988776655', 'rahul.verma@gmail.com'),

('Sneha', 'Patel', '1999-03-17', '9090909090', 'sneha.patel@gmail.com'),

('Kiran', 'Naik', '1990-07-09', '9445566778', 'kiran.naik@gmail.com');

Policies:
INSERT INTO policies (policyname, policytype, premiumamount, durationyears) VALUES

('Life Secure Plus', 'Life Insurance', 15000.00, 20),

('Health Shield', 'Health Insurance', 12000.00, 5),

('Car Protect', 'Vehicle Insurance', 8000.00, 3),

('Home Safe', 'Property Insurance', 10000.00, 10),

('Child Future Plan', 'Education Insurance', 18000.00, 15);

Agents:
INSERT INTO agents (agentname, phone, city) VALUES

('Ramesh Kumar', '9012345678', 'Hyderabad'),

('Sunita Rao', '9345678123', 'Bangalore'),

('Anil Mehta', '9876123450', 'Mumbai'),

('Pooja Singh', '9123987654', 'Delhi'),

('Meghana Shetty','9878276789','Hyderabad');

Policyassignments:
INSERT INTO policyassignments (customerid, policyid, agentid, startdate, enddate)
VALUES

(2, 1, 1, '2022-01-01', '2042-01-01'),

(6, 2, 2, '2023-06-15', '2028-06-15'),

(3, 3, 3, '2021-09-10', '2024-09-10'),

(4, 4, 4, '2020-03-20', '2030-03-20'),

(5, 5, 5, '2024-02-01', '2039-02-01');

Claims:
INSERT INTO claims (assignmentid, claimdate, claimmoney, claimstatus) VALUES

(10, '2024-01-10', 45000.00, 'Approved'),

(12, '2023-11-05', 25000.00, 'Rejected'),

(9, '2023-08-18', 100000.00, 'Approved'),

(11, '2022-06-30', 60000.00, 'Pending'),

(8, '2021-07-08', 79600.00, 'Approved');

Basic Select Queries:

1. SELECT * FROM customers;

2. SELECT customerid, policyid, startdate, enddate
   FROM policyassignments;

3. SELECT * FROM Policies
   WHERE PolicyType = 'Life Insurance' OR PolicyType = 'Health Insurance' OR
   PolicyType = 'Vehicle Insurance';

4. SELECT *FROM Policies
   WHERE PolicyType IN ('Life Insurance', 'Health Insurance', 'Vehicle Insurance');

5. SELECT MAX(ClaimMoney) AS HighestClaimAmount,
   MIN(ClaimMoney) AS LowestClaimAmount
   FROM Claims;

Aggregation Functions:

1. SELECT COUNT(*) AS total_customers

```
FROM customers;
```

2. ```
   SELECT AVG(premiumamount) AS avg_premium
   FROM policies;
   ```

3. ```
   SELECT
     MAX(premiumamount) AS max_premium,
     MIN(premiumamount) AS min_premium
   FROM policies;
   ```

4. ```
   SELECT SUM(premiumamount) AS total_premium
   FROM policies;
   ```

5. ```
   SELECT claimstatus, COUNT(*) AS total_claims
   FROM claims
   GROUP BY claimstatus;
   ```

Date and Time Functions:

1. ```
   SELECT GETDATE() AS current_datetime;
   ```

2. ```
   SELECT CAST(GETDATE() AS DATE) AS current_date;
   ```

3. ```
   SELECT CAST(GETDATE() AS TIME) AS current_time;
   ```

4. ```
   SELECT *
   FROM policyassignments
   WHERE YEAR(startdate) = 2022;
   ```

5. ```
   SELECT *
   FROM claims
   WHERE claimdate >= DATEADD(YEAR, -2, GETDATE());
   ```
   ```

String Functions:

1. SELECT UPPER(FirstName) AS FirstName_Upper
   FROM customers;

2. SELECT
     FirstName + ' ' + LastName AS FullName
   FROM customers;

3. SELECT
     Email,
     LEN(Email) AS email_length
   FROM customers;

4. SELECT *
   FROM customers
   WHERE Email LIKE '%@gmail.com';

5. SELECT phone, RIGHT(phone, 4) AS last_four_digits
   FROM customers;

Joins:

1. **INNER JOIN – Customer with Policy Details**
   SELECT c.FirstName, c.LastName, p.policyname, p.policytype
   FROM customers c
   INNER JOIN policyassignments pa
   ON c.CustomerID = pa.customerid
   INNER JOIN policies p
   ON pa.policyid = p.policyid;

2. **LEFT JOIN – Customers with NO Policies**
   SELECT c.FirstName, c.LastName
   FROM customers c
   LEFT JOIN policyassignments pa
   ON c.CustomerID = pa.customerid
   WHERE pa.assignmentid IS NULL;

3. **LEFT JOIN – Policy-wise Customer Count**

```
SELECT  p.policyname,  COUNT(pa.assignmentid) AS total_customers
FROM policies p
LEFT JOIN policyassignments pa
ON p.policyid = pa.policyid
GROUP BY p.policyname;
```

4. **RIGHT JOIN – All Policies Even If Not Assigned**

```
SELECT  p.policyname,  c.FirstName
FROM customers c
RIGHT JOIN policyassignments pa
ON c.CustomerID = pa.customerid
RIGHT JOIN policies p
ON pa.policyid = p.policyid;
```

5. **SELF JOIN – Agents Working in Same City**

```
SELECT  a1.agentname AS Agent1, a2.agentname AS Agent2,  a1.city
FROM agents a1
JOIN agents a2
ON a1.city = a2.city AND a1.agentid < a2.agentid;
```

Sub-Queries:

1. **Customers who have taken at least one policy**

```
SELECT *
FROM customers
WHERE CustomerID IN (
SELECT customerid
FROM policyassignments
);
```

2. **Customers who have NOT taken any policy**

```
SELECT *
FROM customers
WHERE CustomerID NOT IN (
SELECT customerid
FROM policyassignments
);
```

3. **Policy with the highest premium amount**

```
SELECT *
FROM policies
WHERE premiumamount = (
 SELECT MAX(premiumamount)
 FROM policies
);
```

4. **Customers whose total claim amount is greater than 50,000**

```
SELECT c.FirstName, c.LastName
FROM customers c
WHERE c.CustomerID IN (
SELECT pa.customerid
 FROM policyassignments pa
 JOIN claims cl
 ON pa.assignmentid = cl.assignmentid
 GROUP BY pa.customerid
 HAVING SUM(cl.claimmoney) > 50000
);
```

5. **Agents who are handling more than one policy**

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

Case-Else :

1. **Categorize Policies Based on Premium Amount**

```
SELECT policyname, premiumamount,
  CASE
    WHEN premiumamount >= 15000 THEN 'High Premium'
    WHEN premiumamount BETWEEN 10000 AND 14999 THEN 'Medium Premium'
    ELSE 'Low Premium'
  END AS premium_category
```

```
FROM policies;
```

2. **Display Claim Status Message**

```
SELECT claimid, claimmoney,
  CASE
    WHEN claimstatus = 'Approved' THEN 'Payment Released'
    WHEN claimstatus = 'Rejected' THEN 'Claim Denied'
    ELSE 'Under Processing'
  END AS claim_message
FROM claims;
```

3. **Check Policy Status (Active / Expired)**

```
SELECT assignmentid, startdate, enddate,
  CASE
    WHEN enddate >= CAST(GETDATE() AS DATE) THEN 'Active'
    ELSE 'Expired'
  END AS policy_status
FROM policyassignments;
```

4. **Customer Age Group Classification**

```
SELECT  FirstName, LastName,
  CASE
    WHEN DATEDIFF(YEAR, DateOfBirth, GETDATE()) < 25 THEN 'Young'
    WHEN DATEDIFF(YEAR, DateOfBirth, GETDATE()) BETWEEN 25 AND 40 THEN
'Adult'
    ELSE 'Senior'
  END AS age_group
FROM customers;
```

5. **Claim Amount Category**

```
SELECT claimid, claimmoney,
  CASE
    WHEN claimmoney >= 80000 THEN 'High Claim'
    WHEN claimmoney BETWEEN 30000 AND 79999 THEN 'Medium Claim'
    ELSE 'Low Claim'
  END AS claim_category
FROM claims;
```

Merge Commands:

1. **Insert or Update Customers**

```
MERGE customers AS target
USING (
  SELECT
    'Ravi' AS FirstName,
    'Kumar' AS LastName,
    '1994-06-12' AS DateOfBirth,
    '9876541111' AS Phone,
    'ravi.kumar@gmail.com' AS Email
) AS source
ON target.Email = source.Email

WHEN MATCHED THEN
  UPDATE SET
    Phone = source.Phone

WHEN NOT MATCHED THEN
  INSERT (FirstName, LastName, DateOfBirth, Phone, Email)
  VALUES (source.FirstName, source.LastName, source.DateOfBirth,
source.Phone, source.Email);
```

2. **Update Policy Premium or Insert New Policy**

```
MERGE policies AS target
USING (
  SELECT
    'Travel Secure' AS policyname,
    'Travel Insurance' AS policytype,
    9000.00 AS premiumamount,
    2 AS durationyears
) AS source
ON target.policyname = source.policyname

WHEN MATCHED THEN
  UPDATE SET
    premiumamount = source.premiumamount,
```

```
    durationyears = source.durationyears

WHEN NOT MATCHED THEN
  INSERT (policyname, policytype, premiumamount, durationyears)
  VALUES (source.policyname, source.policytype, source.premiumamount,
source.durationyears);
```

Roll-Up Commands:

1. **Agent-wise Claim Amount with Grand Total**

```
SELECT
  pa.agentid,
  SUM(cl.claimmoney) AS total_claim_amount
FROM policyassignments pa
JOIN claims cl
  ON pa.assignmentid = cl.assignmentid
GROUP BY ROLLUP(pa.agentid);
```

2. **Policy-wise Claim Amount with Subtotal & Grand Total**

```
SELECT
  p.policyname,
  SUM(cl.claimmoney) AS total_claim_amount
FROM policies p
JOIN policyassignments pa
  ON p.policyid = pa.policyid
JOIN claims cl
  ON pa.assignmentid = cl.assignmentid
GROUP BY ROLLUP(p.policyname);
```