Module 4.4 practical project assignment

Database insurance creation

Create database Insurancedb;

Creation commands

1.Customer table creation:

create table Customers(CustomerID int identity(1,1) primary key not null,FirstName varchar(50),LastName varchar(50),DateOfBirth date,Phone varchar(50),Email varchar(50));

2. Policies table creation:

create table Policies(PolicyID int identity(1,1) primary key,policyName varchar(50),PolicyType varchar(50),PremiumAmount decimal(10,2),DurationYears int);

3.Agents table creation:

create table Agents(AgentID int identity(1,1) primary key ,AgentName varchar(50),phone varchar(50),city varchar(50));

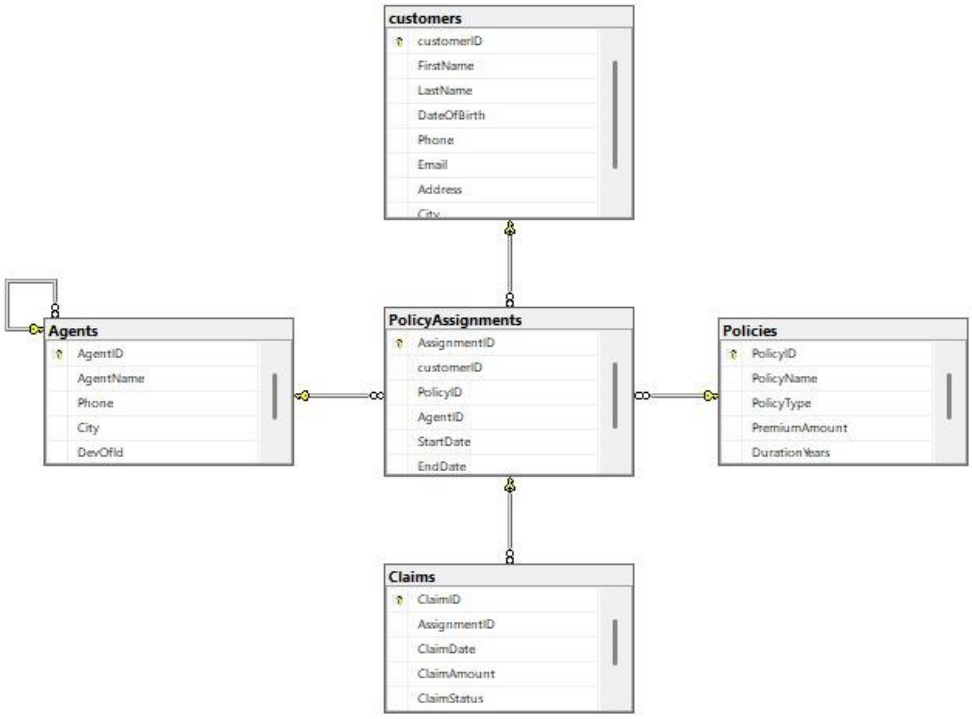
4. PolicyAssignment table creation:

create table PolicyAssignments(AssignmentID int PRIMARY KEY, CustomerID INT, FOREIGN KEY(CustomerID) REFERENCES Customers(CustomerID),PolicyID INT, FOREIGN KEY(PolicyID) REFERENCES Policies(PolicyID),AgentID INT, FOREIGN KEY(AgentID) REFERENCES Agents(AgentID),StartDate date,EndDate date);

5. Claims table creation:

create table Claims(ClaimId int identity(1,1) primary key,AssignmentID int,foreign key(AssignmentID) REFERENCES PolicyAssignments(AssignmentID),ClaimDate date,ClaimAmount decimal(10,2),ClaimStatus varchar(50));

Schema diagram



Insert into commands:

1.Customer table Insertion:

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

('Anand', 'Chedapangu', '2002-05-14', '9876543210', 'anand@gmail.com'),

('Ravi', 'Kumar', '1998-11-22', '9123456780', 'ravi@gmail.com'),

('Sita', 'Reddy', '1995-03-10', '9988776655', 'sita@gmail.com'),

('Amit', 'Sharma', '1988-07-01', '9012345678', 'amit@gmail.com'),

('Priya', 'Verma', '1999-12-19', '9345678901', 'priya@gmail.com');

2.Agents table Insertion:

INSERT INTO Agents (AgentName, Phone, City) VALUES

('Suresh Rao', '8888888888', 'Hyderabad'),

('Neha Singh', '7777777777', 'Bangalore'),

('Vikram Patel', '6666666666', 'Mumbai');

3.Policies table Insertion:

INSERT INTO Policies (PolicyName, PolicyType, PremiumAmount, DurationYears) VALUES

('Health Plus', 'Health', 12000, 5),

('Life Secure', 'Life', 18000, 10),

('Car Protect', 'Vehicle', 8000, 3),

('Home Shield', 'Property', 15000, 7);

4. PolicyAssignments table Insertion:

INSERT INTO PolicyAssignments (AssignmentID, CustomerID, PolicyID, AgentID, StartDate, EndDate) VALUES

(1, 1, 1, 1, '2023-01-01', '2028-01-01'),

(2, 1, 2, 2, '2024-06-01', '2034-06-01'),

(3, 2, 3, 1, '2022-03-15', '2025-03-15'),

(4, 3, 1, 3, '2021-09-10', '2026-09-10'),

(5, 4, 4, 2, '2020-12-01', '2027-12-01'),

(6, 5, 2, 3, '2023-07-20', '2033-07-20');

5. Claims table Insertion:

INSERT INTO Claims (AssignmentID, ClaimDate, ClaimAmount, ClaimStatus) VALUES

(1, '2024-02-10', 25000, 'Approved'),

(2, '2024-08-05', 50000, 'Pending'),

(3, '2023-11-18', 15000, 'Rejected'),

(4, '2022-12-01', 30000, 'Approved'),

(6, '2024-09-09', 45000, 'Pending');

SQL COMMANDS:

BASIC SELECT COMMANDS:

1. fetch all customers

select * from customers;

2. get customer names and email ids

select FirstName, LastName, Email from customers;

3. list policies with premium greater than 10000

```
select * from policies where PremiumAmount > 10000;
```

4. find agents working in hyderabad

```
select * from agents where City = 'Hyderabad';
```

5. get all approved claims

```
select * from claims where ClaimStatus = 'Approved';
```

6. find customers born after 1998

```
select * from customers where DateOfBirth > '1998-01-01';
```

DATE FUNCTIONS:

1. get current system date

```
select getdate();
```

2. calculate age of each customer

```
select FirstName, datediff(year, DateOfBirth, getdate()) as Age from customers;
```

3. get claims raised in the last one year

```
select * from claims where ClaimDate >= dateadd(year, -1, getdate());
```

4. calculate policy duration in days

```
select AssignmentID, datediff(day, StartDate, EndDate) as PolicyDays from
policyassignments;
```

5. extract month from claim date

```
select ClaimID, month(ClaimDate) as ClaimMonth from claims;
```

6. find policies started in the last 3 years

```
select * from policyassignments where StartDate >= dateadd(year, -3, getdate());
```

STRING FUNCTIONS:

1. convert customer first names to uppercase

```
select upper(FirstName) as FirstNameUpper from customers;
```

2. convert agent names to lowercase

select lower(AgentName) as AgentNameLower from agents;

3. display full name of customers (firstname + lastname)

select concat(FirstName, ' ', LastName) as FullName from customers;

4. find customers whose email contains 'gmail'

select * from customers where Email like '%gmail%';

5. extract first 4 characters of policy name

select PolicyName, left(PolicyName, 4) as PolicyPrefix from policies;

6. get length of agent names

select AgentName, len(AgentName) as NameLength from agents;

JOINS COMMANDS:

1. find customers who never raised a claim

select distinct c.FirstName from customers c left join policyassignments pa on c.CustomerID= pa.CustomerID left join claims cl on pa.AssignmentID = cl.AssignmentID where cl.ClaimID is null;

2. View all customers with their policies.

select Concat(c.FirstName,c.LastName) as Fullname,p.PolicyID,p.policyName,p.PolicyType,p.PremiumAmount,pa.AgentID, a.AgentName,pa.StartDate,pa.EndDate from Policies AS p join PolicyAssignments as pa on p.PolicyID=pa.PolicyID join Customers as c on c.CustomerID=pa.CustomerID join Agents as a on a.AgentID=pa.AgentID;

3. Display FirstName, PolicyName, AgentName, StartDate and EndDate from their respective tables.

select c.FirstName,p.policyName,a.AgentName,pa.StartDate,pa.EndDate from Policies AS p join PolicyAssignments as pa on p.PolicyID=pa.PolicyID join CH.ANAND Customers as c on c.CustomerID=pa.CustomerID join Agents as a on a.AgentID=pa.AgentID;

4.Display records of Customers with or without Policies.

SELECT c.CustomerID,c.FirstName,c.LastName,pa.PolicyID,pa.AssignmentID,pa.StartDat e,pa.EndDate FROM Customers c LEFT JOIN PolicyAssignments pa ON c.CustomerID = pa.CustomerID;

5. get total number of claims per policy

select p.PolicyName, count(cl.ClaimID) as ClaimCount from policies p join policyassignments pa on p.PolicyID = pa.PolicyID left join claims cl on pa.AssignmentID = cl.AssignmentID group by p.PolicyName;

6. find how many policies each agent handles

select a.AgentName, count(pa.AssignmentID) as PoliciesHandled from agents a

left join policyassignments pa on a.AgentID = pa.AgentID group by a.AgentName;

SUB-QUERIES COMMANDS:

1. find customers who have raised at least one claim

select * from customers where CustomerID in (select CustomerID from policyassignments where AssignmentID in (select AssignmentID from claims));

2. get policy with the highest premium

select * from policies where PremiumAmount = (select max(PremiumAmount) from policies);

3. find customers having more than one policy

select * from customers where CustomerID in (select CustomerID from policyassignments group by CustomerID having count(*) > 1);

4. find agents who handled claims

select * from agents where AgentID in (select AgentID from policyassignments where AssignmentID in (select AssignmentID from claims));

5. find policies for which no claim was raised

select * from policies where PolicyID not in (select PolicyID from policyassignments where AssignmentID in (select AssignmentID from claims));

6. find customer who raised the highest claim amount

select * from customers where CustomerID = (select pa.CustomerID from claims cl join policyassignments pa on cl.AssignmentID = pa.AssignmentID where cl.ClaimAmount = (select max(ClaimAmount) from claims));

AGGREGATE FUNCTIONS:

1. find total number of customers

select count(*) as TotalCustomers from customers;

2. calculate average policy premium

select avg(PremiumAmount) as AvgPremium from policies;

3. find maximum claim amount

select max(ClaimAmount) as MaxClaim from claims;

4. calculate total claim amount

select sum(ClaimAmount) as TotalClaims from claims;

5. count claims by status

select ClaimStatus, count(*) as Count from claims group by ClaimStatus;

6. count number of policies per customer

select CustomerID, count(*) as PolicyCount from policyassignments group by CustomerID;

DELETE AND UPDADE COMMANDS:

1. update phone number of a customer

update customers set Phone = '9999999999' where CustomerID = 1;

2. increase all policy premiums by 10 percent

update policies set PremiumAmount = PremiumAmount * 1.10;

3. approve a pending claim

update claims set ClaimStatus = 'Approved' where ClaimID = 2;

4. delete policies not assigned to any customer

delete from policies where PolicyID not in (select PolicyID from policyassignments);

5. delete agents not handling any policy

delete from agents where AgentID not in (select AgentID from policyassignments);

6. delete claims with very low amount

delete from claims where ClaimAmount < 10000;

OTHER COMMANDS:

case:

classify claims based on claim amount

```
select ClaimID,ClaimAmount,
    case
      when ClaimAmount >= 50000 then 'high'
      when ClaimAmount >= 20000 then 'medium'
      else 'low'
    end as ClaimCategory
from claims;
```

merge:

update or insert customer email details from a temporary source table

```
merge customers as target using tempcustomers as source on target.CustomerID = source.CustomerID when matched then  update set target.Email = source.Email when not matched then insert (FirstName, LastName, DateOfBirth, Phone, Email) values (source.FirstName, source.LastName, source.DateOfBirth, source.Phone, source.Email);
```

rollup:

get total number of policies per agent including grand total

```
select AgentID,count(AssignmentID) as PolicyCount from PolicyAssignments group by rollup (AgentID);
```

cube:

get policy count by agent and policy including all subtotals

```
select AgentID, PolicyID, count(AssignmentID) as PolicyCount from PolicyAssignments group by cube (AgentID, PolicyID);
```

SET OPERATIONS:

union: get all unique customer and agent phone numbers

```
select Phone from customers union select Phone from agents;
```

union all: get all customer and agent phone numbers including duplicates

```
select Phone from customers union all select Phone from agents;
```

intersect: find phone numbers common to both customers and agents

select Phone from customers intersect select Phone from agents;

except: find phone numbers present in customers but not in agents

select Phone from customers except select Phone from agents;