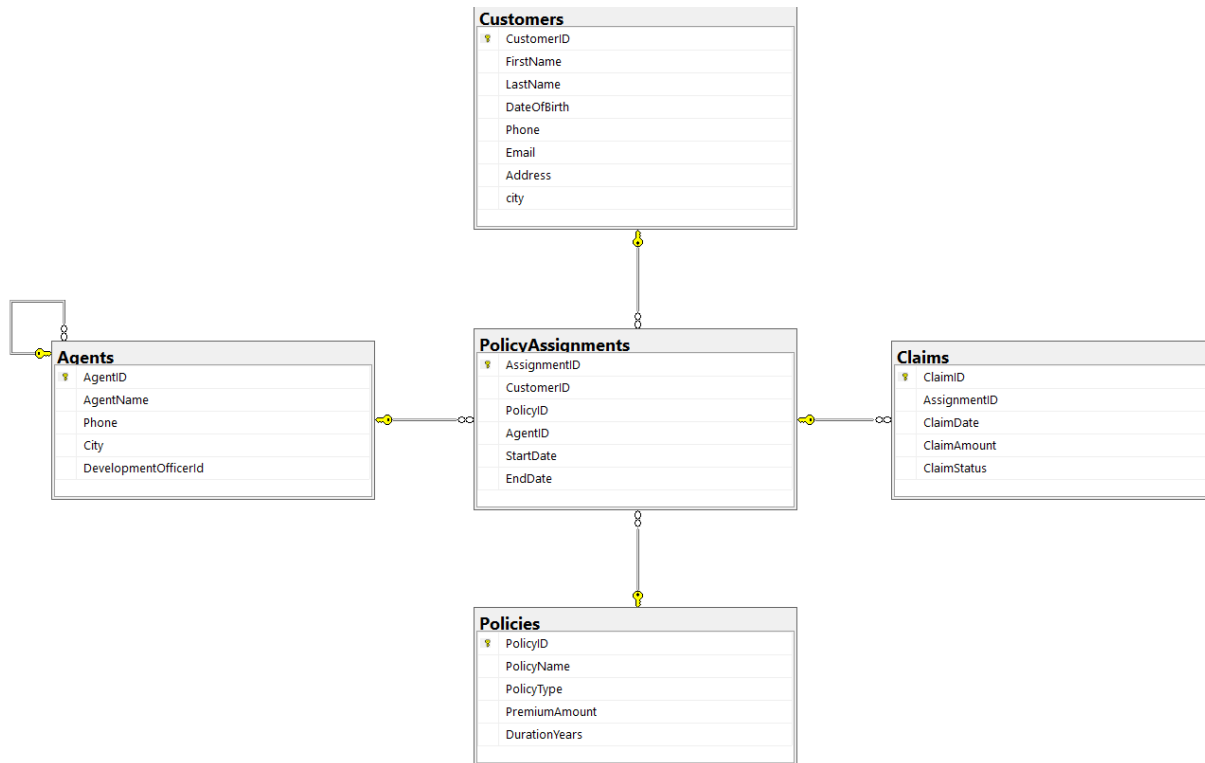


Practical Assignment on SQL-Server

Schema for Insurance Database:



Queries :

Create Database:

```
create database InsuranceDB;
```

use Database:

```
use InsuranceDB;
```

Creation of tables of Customers, Agents, Policies, Claims, PolicyAssignments:

Customers table:

```
create table Customers (CustomerID int primary key, FirstName text, LastName text, DateOfBirth date, Phone varchar(20), Email varchar(20));
```

Policies table:

```
create table Policies (PolicyID int primary key,
```

```

        PolicyName varchar(50) not null,
        PolicyType varchar(20) not null,
        PremiumAmount decimal(10,2) not null,
        DurationYears int check (DurationYears > 0)
    );

```

Agents table:

```

create table Agents (
    AgentID int primary key,
    AgentName varchar(50) not null,
    Phone varchar(20) not null,
    City varchar(50) not null
);

```

PolicyAssignments table:

```

create table PolicyAssignments (
    AssignmentID int primary key,
    CustomerID int,
    PolicyID int,
    AgentID int,
    StartDate date,
    EndDate date,

    constraint fk_customer
        foreign key (CustomerID)
        references Customers (CustomerID),

    constraint fk_policy
        foreign key (PolicyID)
        references Policies (PolicyID),
    constraint fk_agentid
        foreign key (AgentID)
        references Agents (AgentID)
);

```

Claims table:

```

create table Claims (
    ClaimID int primary key,
    AssignmentID int,
    ClaimDate date,
    ClaimAmount money,
    ClaimStatus varchar(50),

    constraint fk_assignments
        foreign key (AssignmentID)
        references PolicyAssignments (AssignmentID)
);

```

Insertions into tables:

```

insert into Customers (customerID, FirstName, LastName, DateOfBirth, phone, Email)
values
(1, 'Ravi', 'Kumar', '1995-04-12', '9876543210', 'ravi@gmail.com'),
(2, 'Anita', 'Sharma', '1992-09-23', '9123456780', 'anita@yahoo.com'),
(3, 'Suresh', 'Reddy', '1988-01-15', '9988776655', 'suresh@outlook'),
(4, 'Priya', 'Verma', '1997-06-30', '9090909090', 'priya@gmail.com'),
(5, 'Amit', 'Patel', '1990-11-05', '9567891234', 'amit@gmail.com');

```

```

insert into policies (policyid, policyname, policytype, premiumamount,
durationyears)
values
(201, 'LifeSecure', 'Life', 50000, 20),
(202, 'HealthPlus', 'Health', 30000, 10),
(203, 'CarProtect', 'Vehicle', 15000, 5),
(204, 'HomeShield', 'Property', 40000, 15),
(205, 'TravelSafe', 'Travel', 10000, 2);

insert into agents (agentid, agentname, phone, city)
values
(1, 'RahulSharma', '9876543210', 'Delhi'),
(2, 'AnitaVerma', '9123456789', 'Mumbai'),
(3, 'SureshReddy', '9988776655', 'Hyderabad'),
(4, 'PriyaPatel', '9090909090', 'Ahmedabad'),
(5, 'AmitKumar', '9567891234', 'Bangalore');

insert into policyassignments
(assignmentid, customerid, policyid, agentid, startdate, enddate)
values
(1, 1, 201, 1, '2023-01-01', '2043-01-01'),
(2, 2, 202, 2, '2022-06-15', '2032-06-15'),
(3, 3, 203, 3, '2021-03-10', '2026-03-10'),
(4, 4, 204, 4, '2020-09-05', '2035-09-05'),
(5, 5, 205, 5, '2024-01-20', '2026-01-20');

insert into claims
(claimid, AssignmentID, claimdate, claimamount, claimstatus)
values
(1, 1, '2024-02-10', 15000, 'Approved'),
(2, 2, '2023-08-05', 20000, 'Pending'),
(3, 3, '2022-11-18', 12000, 'Rejected'),
(4, 4, '2024-01-25', 30000, 'Approved'),
(5, 5, '2024-06-12', 8000, 'Pending');

```

Queries on select commands:

1: Display all policies of Health type

Command: `select * from Policies where PolicyType = 'Health';`

2. Display unique city names where agents belong

Command: `select distinct City from Agents;`

3. Display customers born between Jan 1, 2001 and Dec 31, 2020 (using BETWEEN)

Command: `select * from Customers where DateOfBirth between '2001-01-01' and '2020-12-31';`

4. Display latest claim record

Command: `select top 1 * from Claims order by ClaimDate desc;`

5. Display highest and lowest claim amount from Claims table

Command: `select max(ClaimAmount) as HighestClaimAmount, min(ClaimAmount) as LowestClaimAmount from Claims;`

Queries using Joins, Group by, Having:

1. List all Policies for a CustomerId 5.

Command: `select p.PolicyID,p.PolicyName,pa.StartDate,pa.EndDate
from PolicyAssignments pa join Policies p on pa.PolicyID = p.PolicyID
where pa.CustomerID = 5;
select
c.CustomerID,c.FirstName,c.LastName,p.PolicyID,p.PolicyName,p.PolicyType,p.Premium
Amount
from Customers c
join PolicyAssignments pa
on c.CustomerID = pa.CustomerID
join Policies p
on pa.PolicyID = p.PolicyID;`

2. View all customers with their policies.

Command: `select cl.ClaimID,
cl.ClaimDate,cl.ClaimAmount,cl.ClaimStatus,c.CustomerID,c.FirstName,c.LastName
from Claims cl
join PolicyAssignments pa
on cl.AssignmentID = pa.AssignmentID
join Customers c
on pa.CustomerID = c.CustomerID;`

3. Show names and total claim amount of Customers With Claim Amount > 50000 (Use
HAVING Clause).

Command: `select c.CustomerID,cast(c.FirstName as varchar(50)) + ' ' +
cast(c.LastName as varchar(50)) as CustomerName,
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.CustomerID,
cast(c.FirstName as varchar(50)),
cast(c.LastName as varchar(50))
having sum(cl.ClaimAmount) > 50000;`

4.Display list with Agent Wise Policy Count.

Command: `select a.AgentID,a.AgentName,count(pa.PolicyID) as PolicyCount
from Agents a
left join PolicyAssignments pa
on a.AgentID = pa.AgentID
group by a.AgentID,a.AgentName;`

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

Command: `select c.FirstName, p.PolicyName, a.AgentName, pa.StartDate, pa.EndDate
from Customers c
join PolicyAssignments pa
on c.CustomerID = pa.CustomerID
join Policies p
on pa.PolicyID = p.PolicyID
join Agents a
on pa.AgentID = a.AgentID;`

Queries on Date Functions:

1. Retrieve all claims that were raised today.

```
Command: select
        claimid,
        assignmentid,
        claimdate,
        claimamount
from claims
where cast(claimdate as date) = cast(getdate() as date);
```

2. Find the number of years each policy has been active since its start date.

```
Command: select
        assignmentid,
        policyid,
        startdate,
        enddate,
        datediff(year, startdate, isnull(enddate, getdate())) as years_active
from policyassignments;
```

3. Display the expected policy expiry date for each policy assignment.

```
Command: select
        pa.assignmentid,
        pa.startdate,
        dateadd(year, p.durationyears, pa.startdate) as expected_expiry_date
from policyassignments pa
join policies p
on pa.policyid = p.policyid;
```

4. Display claim details along with the month name in which the claim was raised.

```
Command: select
        claimid,
        datename(month, claimdate) as claim_month,
        claimamount,
        claimstatus
from claims;
```

5. Retrieve all claims that were raised in the year 2024

```
Command: select
        claimid,
        assignmentid,
        claimdate,
        claimstatus
from claims
where year(claimdate) = 2024;
```

SUBQUERY QUERIES:

1. Retrieve customers who have raised at least one claim.

Command: `select`

```
    firstname,
    lastname
from customers
where customerid in (
    select customerid
    from policyassignments
    where assignmentid in (
        select assignmentid
        from claims
    )
);
```

2. Find policies that have at least one claim associated with them.

Command:

```
select
    policyname
from policies
where policyid in (
    select policyid
    from policyassignments
    where assignmentid in (
        select assignmentid
        from claims
    )
);
```

3. Display agents who are handling more than one policy assignment.

Command: `select`

```
    agentname
from agents
where agentid in (
    select agentid
    from policyassignments
    group by agentid
    having count(*) > 1
);
```

4. Retrieve customers who do not have any active claims

Command: `select`

```
    firstname,
    lastname
from customers
where customerid not in (
    select pa.customerid
    from policyassignments pa
    join claims cl
    on pa.assignmentid = cl.assignmentid
);
```

5. Display the policy with the highest premium amount.

Command:

```
select
    policyname,
    premiumamount
from policies
```

```
where premiumamount = (  
    select max(premiumamount)  
    from policies  
);
```

SET OPERATION QUERIES:

1. Retrieve the list of all cities where either customers or agents are located.

Command: `select city
from customers`

`union`

`select city
from agents;`

2. Retrieve all cities including duplicates where customers and agents are located.

Command: `select city
from customers`

`union all
select city
from agents;`

3. Retrieve cities where both customers and agents are located

Command: `select city
from customers`

`intersect
select city
from agents;`

4. Retrieve cities where customers are located but agents are not located.

Command: `select city
from customers except
select city
from agents;`

5. Retrieve customer IDs who have policy assignments and customer IDs who have raised claims (combined without duplicates)

Command:

```
select customerid  
from policyassignments  
union  
select customerid  
from policyassignments  
where assignmentid in (  
    select assignmentid  
    from claims  
);
```

CASE-ELSE:

1. Display claim status as "High Amount" if claim amount is greater than 50,000, otherwise display "Normal Amount"

Command: `select
 claimid,
 claimamount,`

```

        case
        when claimamount > 50000 then 'high amount'
        else 'normal amount'
        end as claim_category
from claims;

```

MERGE:

1. Synchronize agent data into a backup table by inserting new agents and updating existing ones.

Command: `merge agents_backup as target
using agents as source
on target.agentid = source.agentid`

```

when matched then
    update set
        target.agentname = source.agentname,
        target.phone = source.phone,
        target.city = source.city

when not matched then
    insert (agentid, agentname, phone, city)
    values (source.agentid, source.agentname, source.phone, source.city);

```

ROLLUP:

1. Display total claim amount per claim status along with grand total.

Command: `select claimstatus, sum(claimamount) as total_claim_amount from claims
group by rollup (claimstatus);`

CUBE:

1. Display total claim amount by claim status and year including all subtotals

Command: `select claimstatus, year(claimdate) as claim_year, sum(claimamount) as
total_claim_amount from claims group by cube (claimstatus, year(claimdate));`

GROUPING:

1. Identify subtotal and grand total rows in claim summary report.

Command: `select
 claimstatus,
 sum(claimamount) as total_claim_amount,
 grouping(claimstatus) as grouping_flag
from claims
group by rollup (claimstatus);`