

Day 4 - Practical Project Assignment

Database Creation

```
--Database Creation  
create database insuranceDB;
```

Tables Creation

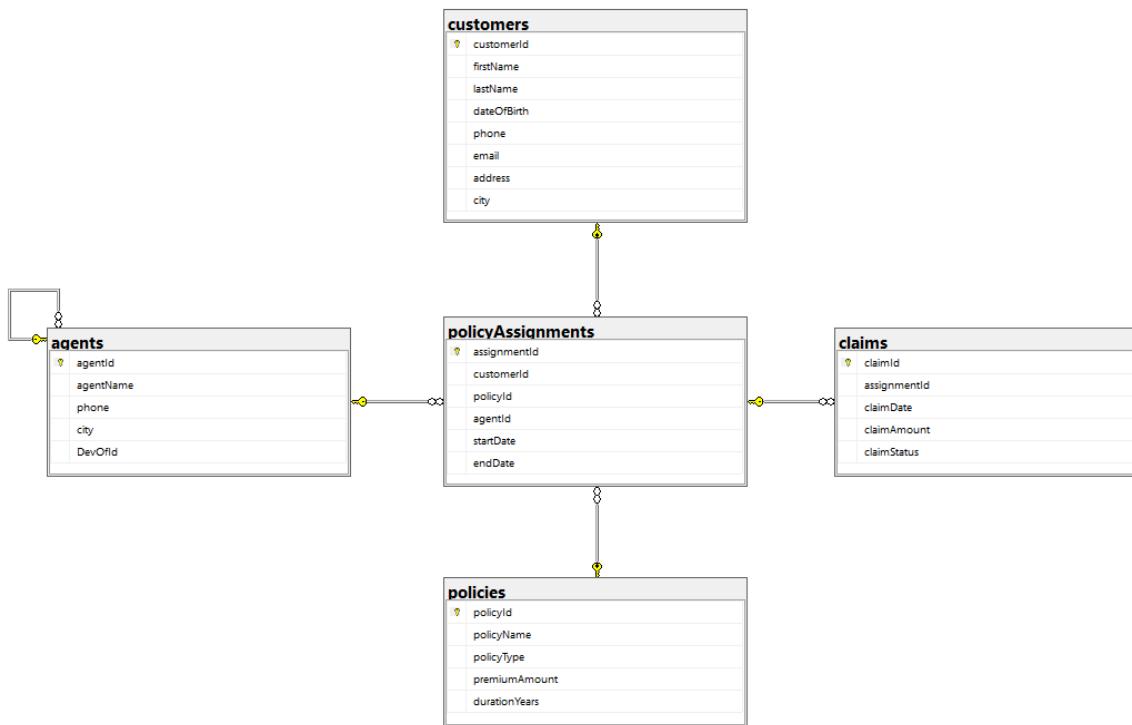
```
--policy assignments table  
create table policyAssignments(  
assignmentId int primary key,  
customerId int,--f  
policyId int,--f  
agentId int,--f  
startDate date,  
endDate date  
foreign key (customerId) references customers(customerId),  
foreign key (policyId) references policies(policyId),  
foreign key (agentId) references agents(agentId)  
)
```

```
--policies table  
create table policies(  
policyId int primary key,  
policyName varchar(20),  
policyType varchar(20),  
premiumAmount decimal(10,2),  
durationYears int  
)
```

```
--customers table  
create table customers (  
customerId int primary key,  
firstName varchar(10) null,  
lastName varchar(50) not null,  
dateOfBirth date not null,  
phone varchar(12) not null,  
email varchar(50) not null  
);
```

```
--claims table
create table claims(
claimId int primary key,
assignmentId int,
claimDate date,
claimAmount decimal(10,2),
claimStatus varchar(5),
foreign key(assignmentId) references policyAssignments(assignmentId)
)
--agents table
create table agents(
agentId int primary key,
agentName varchar(50),
phone varchar(12),
city varchar(50)
)
```

Database diagram



Insert values

--insert policies

```
insert into policies (policyId, policyName, policyType, premiumAmount, durationYears)
values
(101, 'Aman', 'casuality', 35000, 4),
(102, 'LIC', 'personal', 25000, 2),
(103, 'Bhima', 'Health', 22000, 3),
(104, 'Life', 'casuality', 15000, 1);
```

--insert customers

```
insert into customers (customerId, firstName, lastName, dateOfBirth, phone, email)
values
(10, 'Thanmai', 'Danda', '2005-06-01', '6281332238', 'thanmai@gmail.com'),
(13, 'Pranay', 'Danda', '2007-11-05', '6489209384', 'pranay@yahoo.com'),
(16, 'Triveni', 'sripati', '2005-04-03', '7463829209', 'triveni@gmail.com'),
(19, 'Murali', 'Koppa', '2006-12-05', '8393029337', 'mur@gmail.com');
```

--insert claims

```
insert into claims (claimId, assignmentId, claimDate, claimAmount, claimStatus) values
(150, 1, '2016-06-01', 15000, 'yes'),
(152, 1, '2016-06-05', 20000, 'yes'),
(153, 2, '2015-05-03', 25000, 'no'),
(154, 1, '2013-02-01', 20000, 'no');
```

--insert agents

```
insert into agents (agentId, agentName, phone, city) values
(200, 'suman', '9440132609', 'hyd'),
(201, 'amit', '9441124719', 'bng'),
(202, 'ram', '9441132609', 'bng'),
(203, 'syam', '6389202918', 'hyd');
```

--insert policyAssignments

```
insert into policyAssignments
(assignmentId, customerId, policyId, agentId, startDate, endDate) values
(1, 10, 101, 200, '2010-01-01', '2020-01-01'),
(2, 10, 102, 203, '2007-02-01', '2007-05-01'),
(3, 13, 101, 202, '2008-03-01', '2009-01-01'),
(4, 10, 103, 200, '2010-03-01', '2026-02-02');
```

Select Commands

1. View all records Customers table.

```
select * from customers;
```

2. Display all policies of Health type.

```
select * from policies where policyType='Health';
```

3. List policies of type Life, Health, Motor use OR clause.

```
select * from policies where policyType='casualty' or policyType='health';
```

4. List policies of type Life, Health, Motor use IN operator.

```
select * from policies where policyType in ('casualty','health');
```

5. Display unique city names from where agents belong to.

```
select city from agents group by city;
```

6. Display records of Agents who stay in a city whose second letter is n.

```
select * from agents where city like '_n%';
```

7. Display latest claim record.

```
select top 1 * from claims order by ClaimDate desc;
```

8. Increase premium amount to 10% for all health insurance policies.

```
update policies set premiumAmount=premiumAmount*1.10 where policyType='Health';
```

```
select * from policies where policyType='Health';
```

9.. Display PolicyId, PolicyName, PremiumAmount along with computed fields not in table a 6% LocalTaxes, PremiumAmountWithTax and MonthlyPremiumAmount consideringPremiumAmount is Annual.

```
select policyId,policyName,premiumAmount,premiumAmount*1.06 as  
premiumAmountWithTax,premiumAmount/12 as MonthlyPremiumAmount from policies;
```

10. Write command to make the above DevOfId as a recursive foreign key to AgentId as Parent.

```
alter table agents
```

```
add constraint FK_DevOfId_agents
```

```
foreign key (DevOfId) references agents(agentId);
```

```
select * from agents;
```

Functions

a.String functions

-concat()

```
select concat(lastname,firstname) as Name from customers;
```

-upper()

```
select Upper(agentName) from agents;
```

-reverse()

```
select reverse(startDate) from policyAssignments
```

-substring()

```
select policyType, substring(policyType, 1, 3) from policies;
```

-length()

```
select customerName, len(customerName) from customers;
```

b.Date functions

-datepart()

```
select datepart(yyyy,getdate());
```

-dateadd()

```
select dateadd(m,02,getdate())
```

-year()

```
select * from customers where year(dateOfBirth);
```

-datediff()

```
select policyId, datediff(month, startDate, getdate()) as months_active  
from policies;
```

-isdate()

```
select isdate(claimDate) from claims;
```

Joins and group by

1.View claims with customer name.

```
select c.claimId,c.claimDate,c.claimAmount,cu.firstName from claims c join policyAssignments  
pa on c.assignmentId=pa.assignmentId join customers cu on pa.customerId=cu.customerId;
```

2.Display records of Customers with or without Policies.

```
select c.firstName,p.policyName,pa.startDate,pa.endDate  
from customers c  
left join policyAssignments pa on pa.customerId=c.customerId  
left join policies p on pa.policyId=p.policyId;
```

3.Show CustomerName with Total Claim Amount per Customer.

```
select c.firstName,  
       sum(cl.claimAmount) as totalClaimAmount
```

```
from customers c
join policyAssignments pa on pa.customerId=c.customerId
join claims cl on cl.assignmentId=pa.assignmentId
group by c.firstName;
```

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

```
select c.firstName,
       sum(cl.claimAmount) as totalClaimAmount
from customers c
join policyAssignments pa on pa.customerId=c.customerId
join claims cl on cl.assignmentId=pa.assignmentId
group by c.firstName
having sum(cl.claimAmount) > 50000;
```

5. Display list with Agent Wise Policy Count.

```
select a.agentName,
       count(pa.policyId) as policyCount
from agents a
join policyAssignments pa on pa.agentId=a.agentId
group by a.agentName;
```

Subqueries

1. Customers who have at least one policy

```
select *
from customers c
where exists (
    select *
    from policyAssignments pa
    where pa.customerId = c.customerId
);
```

2. Customers who do not have any claims

```
select *
from customers c
where not exists (
    select *
    from policyAssignments pa
    join claims cl on pa.assignmentId = cl.assignmentId
    where pa.customerId = c.customerId
);
```

3.Find policies with premium greater than ANY premium of policies held by CustomerID = 10

```
select *
from policies p
where p.premiumAmount > any (
    select p2.premiumAmount
    from policies p2
    join policyAssignments pa2 on p2.policyId = pa2.policyId
    where pa2.customerId = 10
);
```

4.Find customers whose policy premium equals ANY premium of policies in city 'Hyd'

```
select distinct c.*
from customers c
join policyAssignments pa on c.customerId = pa.customerId
join policies p on pa.policyId = p.policyId
where p.premiumAmount = any (
    select p2.premiumAmount
    from policies p2
    join policyAssignments pa2 on p2.policyId = pa2.policyId
    join customers c2 on pa2.customerId = c2.customerId
    where c2.city = 'Hyd'
);
```

5.customers whose policy premium is less than all premiums of Life policies

```
select distinct c.*
from customers c
join policyAssignments pa on c.customerId = pa.customerId
join policies p on pa.policyId = p.policyId
where p.premiumAmount < all (
    select p2.premiumAmount
    from policies p2
    where p2.policyType = 'Life'
);
```

Set Operators

1.Union

List the names of all people who directly interact with the insurance company

```
select firstName as personName from customers
```

```
union
```

```
select agentName from agents;
```

2.Union all

Find every city mentioned in the system,including repeated appearances from both customers and agents.

```
select city from customers
```

```
union all
```

```
select city from agents;
```

3.Except

Find customers who HOLD policies but have NEVER filed any claims

```
select distinct pa.customerId from policyAssignments pa
```

```
except
```

```
select distinct pa.customerId from policyAssignments pa
```

```
join claims cl on pa.assignmentId = cl.assignmentId;
```

4.Intersect

Find cities where customers AND agents both live

```
select city from customers
```

```
intersect
```

```
select city from agents;
```

Case Else

Show claims with a custom message

```
select
```

```
    cl.claimId,
```

```
    cl.claimAmount,
```

```
case
```

```
    when cl.claimStatus = 'Approved' then 'Payment Released'
```

```
    when cl.claimStatus = 'Pending' then 'Under Review'
```

```
    else 'Not Approved'
```

```
end as claimResult
```

```
from claims cl;
```

Merge

Merge agents

```
merge into agents a
using newAgents na
on a.agentId = na.agentId
```

when matched then

```
    update set
        a.agentName = na.agentName,
        a.city = na.city
```

when not matched then

```
    insert (agentId, agentName, city)
        values (na.agentId, na.agentName, na.city);
```

Group By Rollup

Write an SQL query to display the total premium amount collected.

```
select
    c.city,
    p.policyType,
    sum(p.premiumAmount) as totalPremium
from customers c
join policyAssignments pa on c.customerId = pa.customerId
join policies p on pa.policyId = p.policyId
group by rollup (c.city, p.policyType);
```

Group By Cube

Premium totals by city AND policy type

```
select
    c.city,
    p.policyType,
    sum(p.premiumAmount) as totalPremium
from customers c
join policyAssignments pa on c.customerId = pa.customerId
join policies p on pa.policyId = p.policyId
group by cube (c.city, p.policyType);
```

Grouping Sets

Get totals by city&policy type in one query

```
select
    c.city,
    p.policyType,
    sum(p.premiumAmount) as totalPremium
from customers c
join policyAssignments pa on c.customerId = pa.customerId
join policies p on pa.policyId = p.policyId
group by grouping sets (
    (c.city),
    (p.policyType),
    ()
);
```