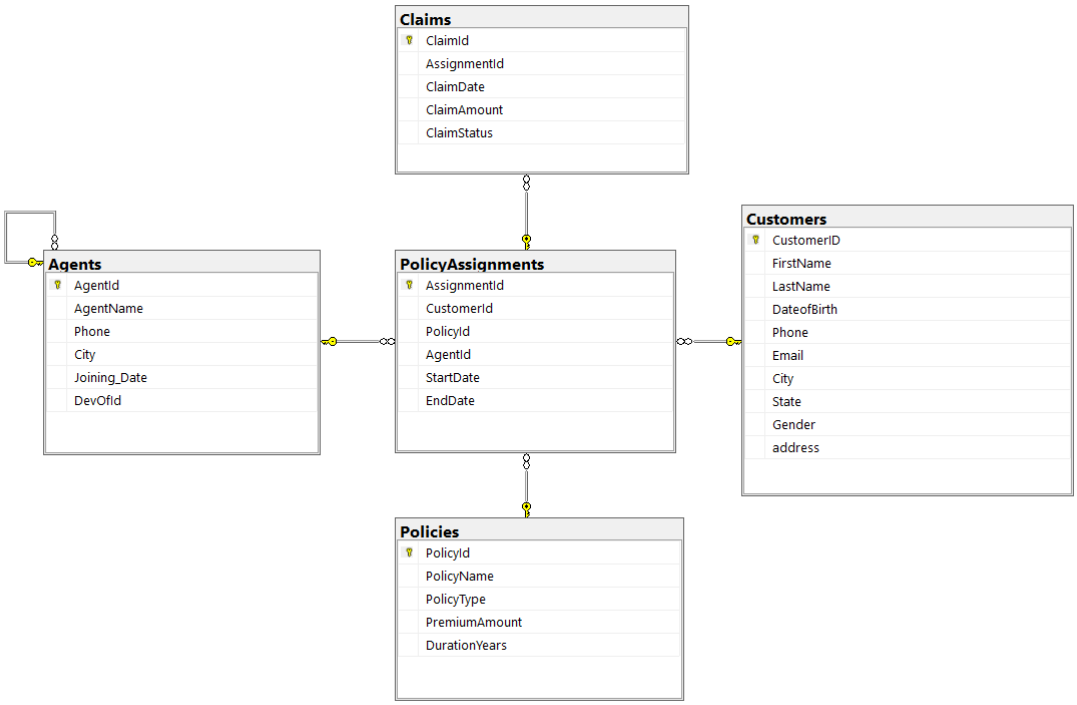
MODULE 4.4 PRACTICAL PROJECT ASSIGNMENT

- CREATE DATABASE InsuranceDB;
- USE InsuranceDB;

SCHEMA DIAGRAM---



- **CREATION OF TABLES**

```
--1 create table customers(
 customerid int primary key,
 firstname varchar(50),
 lastname varchar(50) not null,
 dateofbirth date,
 phone varchar(50),
 email varchar(50) unique
);
```

```
--2 create table policies(
 policyid int primary key,
 policyname varchar(50),
 policytype varchar(50),
 premiumaccount varchar(50),
 durationyears int
);
--3 create table agents(
 agentid int primary key,
 agentname varchar(50),
 phone varchar(50),
 city varchar(50)
);
--4 create table policyassignments(
 assignmentid int primary key,
 customerid int,
 policyid int,
 agentid int,
 startdate date,
 enddate date,
 foreign key(customerid) references customers(customerid),
 foreign key(policyid) references policies(policyid),
 foreign key(agentid) references agents(agentid)
);
--5 create table claims(
 claimid int primary key,
 assignmentid int,
 claimdate date,
```

```
 claimamount money,
 claimstatus varchar(50),
 foreign key (assignmentid) references policyassignments(assignmentid)
);
```

■ **Changing one column name**

```
EXEC sp_rename 'Policies.PremiumAccount', 'PremiumAmount', 'COLUMN';
```

■ **INSERTION OF VALUES INTO TABLE**

```
-- Customers
insert into customers values
(1, 'rohan', 'sharma', '2002-05-12', '9876543210', 'rohan@gmail.com'),
(2, 'anita', 'verma', '1999-03-22', '9988776655', 'anita@hotmail.com'),
(3, 'karthik', 'reddy', '2005-11-30', '8877665544', 'karthik@yahoo.com'),
(4, 'sneha', 'patel', '2003-01-15', '9090909090', 'sneha@gmail.com'),
(5, 'vishnu', 'mohan', '2004-07-19', '8008008008', 'vishnu@yahoo.com'),
(6, 'meera', 'nair', '2000-08-09', '9123456780', 'meera@mail.com'),
(7, 'akash', 'gupta', '2006-12-11', '7012345678', 'akash@gmail.com'),
(8, 'priya', 'desai', '1998-02-05', '6201234567', 'priya@gmail.com');
--policies
insert into policies values
(101, 'health secure', 'health', '15000', 1),
(102, 'life shield', 'life', '45000', 10),
(103, 'motor protect', 'motor', '8000', 1),
(104, 'health plus', 'health', '20000', 1),
(105, 'life premium', 'life', '60000', 15),
(106, 'motor basic', 'motor', '7000', 1),
(107, 'travel guard', 'travel', '5000', 1),
(108, 'accident cover', 'accident', '12000', 2),
--agents
```

```
insert into agents values
(201, 'suresh kumar', '9999988888', 'bangalore'),
(202, 'lakshmi rao', '8888877777', 'hyderabad'),
(203, 'manoj singh', '7777766666', 'chennai'),
(204, 'deepak das', '6666655555', 'mumbai'),
(205, 'rita pandey', '5555544444', 'kolkata'),
(206, 'sanjay mehta', '4444433333', 'vadodara'),
(109, 'health supercare', 'health', '18000', 1);
--policyassignments
insert into policyassignments values
(301, 1, 101, 201, '2023-01-01', '2024-01-01'),
(302, 2, 102, 202, '2022-06-10', '2027-06-10'),
(303, 3, 103, 203, '2024-02-15', '2025-02-15'),
(304, 4, 104, 204, '2023-05-20', '2024-05-20'),
(305, 5, 105, 205, '2021-12-01', '2036-12-01'),
(306, 6, 106, 206, '2024-01-25', '2025-01-25'),
(307, 7, 108, 201, '2023-07-01', '2025-07-01'),
(308, 3, 109, 202, '2024-03-10', '2025-03-10'),
(309, 5, 101, 203, '2024-09-01', '2025-09-01');
--claims
insert into claims values
(401, 301, '2023-12-30', 8000, 'approved'),
(402, 302, '2024-01-15', 50000, 'rejected'),
(403, 303, '2024-03-20', 15000, 'approved'),
(404, 304, '2023-06-30', 7000, 'pending'),
(405, 305, '2024-02-10', 90000, 'approved'),
(406, 306, '2024-01-30', 12000, 'rejected'),
(407, 307, '2024-04-05', 25000, 'approved'),
```

(408, 308, '2024-05-01', 3000, 'rejected');

- **DISPLAY**

```
select * from Customers;
select * from Policies;
select * from Agents;
select * from PolicyAssignments;
select * from Claims;
```

- **QUERIES(select,alter,update)**

1. select * from policies where policytype='health' and premiumamount>20000;
2. select * from customers where city in('mumbai','pune');
3. select * from claims where claimstatus!='approved';
4. select * from policies where premiumamount<15000 and policyid in (select policyid from policyassignments where getdate() between startdate and enddate);
5. alter table customers add gender varchar(20);
6. update customers set gender='female' where customerid=1;
7. update customers set gender='male' where customerid=2;
8. update customers set gender='male' where customerid=3;
9. select * from customers where gender='male' or city='banglore';
10. select * from agents where city='chennai' or year(joining_date)<2019;
11. select * from policies where policytype in('life insurance','health');
12. select * from agents where agentname like '% _____';
13. select * from policies where policyname like '%guard';
14. alter table customers add address varchar(100);
15. alter table policies alter column premiumamount decimal(20,2);
16. alter table agents add constraint fk_agents_devofid foreign key (devofid) references agents(agentid);

- **FUNCTIONS**

A. STRING FUNCTIONS:

```
-- 1. upper()
select upper(firstname) as uppername from customers;
-- 2. lower()
select lower(email) as loweremail from customers;
-- 3. len()
select len(firstname) as namelength from customers;
-- 4. substring()
select substring(firstname, 1, 3) as first3letters from customers;
```

```
-- 5. concat()
select concat(firstname, ' ', lastname) as fullname from customers;
```

B. NUMBER FUNCTIONS

```
-- 1. round()
select round(premiumamount, 2) as roundedpremium from policies;
-- 2. ceiling()
select ceiling(premiumamount / 12) as monthlyceil from policies;
-- 3. floor()
select floor (premiumamount / 12) as monthlyfloor from policies;
-- 4. abs()
select abs(claimamount - 50000) as difference from claims;
-- 5. power()
select power(premiumamount, 2) as squarepremium from policies;
```

C. DATE FUNCTIONS

```
-- 1. getdate()
select getdate() as today;
-- 2. year()
select year(dateofbirth) as birthyear from customers;
-- 3. month()
select month(dateofbirth) as birthmonth from customers;
-- 4. datediff()
select datediff(year, dateofbirth, getdate()) as age from customers;
-- 5. dateadd()
select dateadd(year, durationyears, startdate) as newenddate from
policyassignments;
```

D. AGGREGATE FUNCTIONS

```
-- 1. sum()
select sum(claimamount) as totalclaims from claims;
-- 2. avg()
select avg(premiumamount) as averagepremium from policies;
-- 3. min()
select min(premiumamount) as lowestpremium from policies;
-- 4. max()
select max(premiumamount) as highestpremium from policies;
```

-- 5. count()
select count(*) as totalpolicies from policies;

E.OTHER

--1. top()
select top 5 * from policies;
--2. limit()
select * from policies limit 5;

■ SET OPERATORS

--UNION
select customerid from customers union select customerid from policyassignments;
--UNION ALL
select policyid from policies union all select policyid from policyassignments;
--INTERSECT
select city from customers intersect select city from agents;
--EXCEPT
select firstname from customers except select agentname from agents;
--INTERSECT
select policytype from policies intersect select policytype from policies where premiumamount > 15000;

■ JOINS

--inner join – shows each customer's total claim amount with policy and agent matched

select c.firstname, p.policyname, a.agentname, sum(cl.claimamount) as total_claim from customers c
inner join policyassignments pa on c.customerid = pa.customerid
inner join claims cl on pa.assignmentid = cl.assignmentid
inner join policies p on pa.policyid = p.policyid
inner join agents a on pa.agentid = a.agentid group by c.firstname, p.policyname, a.agentname;

-- left join – shows every agent with count of claims handled even if some agents handled none

select a.agentname, count(cl.claimid) as claim_count from agents a

```
left join policyassignments pa on a.agentid = pa.agentid
left join policies p on pa.policyid = p.policyid
left join claims cl on pa.assignmentid = cl.assignmentid
group by a.agentname;
```

--right join – shows all policies and how many claims they generated even if some policies have zero claims

```
select p.policyname, count(cl.claimid) as total_claims from policies p
right join policyassignments pa on p.policyid = pa.policyid
right join claims cl on pa.assignmentid = cl.assignmentid
group by p.policyname;
```

--full outer join – shows customer, policy, and claim details even when any table has missing records

```
select c.firstname, p.policyname, sum(cl.claimamount) as total_claim
from customers c
full join policyassignments pa on c.customerid = pa.customerid
full join policies p on pa.policyid = p.policyid
full join claims cl on pa.assignmentid = cl.assignmentid
group by c.firstname, p.policyname;
```

■ **SUBQUERIES**

```
-- 1) show policies for which there exists at least one claim above 1,00,000

select p.policyid, p.policyname from policies p
where exists (
    select 1 from policyassignments pa
    join claims cl on cl.assignmentid = pa.assignmentid
    where pa.policyid = p.policyid and cl.claimamount > 100000
);


-- 2) list agents who handle any policy whose premium is greater than any life policy premium

select distinct a.agentid, a.agentname from agents a
join policyassignments pa on pa.agentid = a.agentid
join policies p on p.policyid = pa.policyid
where p.premiumamount > any (
    select premiumamount from policies where policytype = 'life'
```

```
);
```

-- 3) find customers whose policy premium is higher than all premiums of customers from chennai

```
select distinct c.customerid, c.firstname, c.lastname from customers c
join policyassignments pa on pa.customerid = c.customerid
join policies p on p.policyid = pa.policyid
where p.premiumamount > all (
    select p2.premiumamount from customers c2
    join policyassignments pa2 on pa2.customerid = c2.customerid
    join policies p2 on p2.policyid = pa2.policyid where c2.city like 'chennai'
);
```

■ **CORRELATED SUBQUERY**

-- show customers whose claim amount is higher than the average claim amount of that same customer

```
select c.customerid, c.firstname, c.lastname, cl.claimamount from customers c
join policyassignments pa on pa.customerid = c.customerid
join claims cl on cl.assignmentid = pa.assignmentid
where cl.claimamount >(
      select avg(c2.claimamount) from claims c2
      join policyassignments pa2 on pa2.assignmentid = c2.assignmentid
      where pa2.customerid = c.customerid   -- correlated condition
 );
```

■ **MERGE,ROLLUP,CUBE,CASEELSE,GROUPING**

--merge – inserts a new policy if not exists or updates premium if it exists */

```
merge policies as tgt
using (
    select 999 as policyid,
        'special cover' as policyname,
        'health' as policytype,
        25000 as premiumamount,
        1 as durationyears
) as src
on tgt.policyid = src.policyid
when matched then
```

```
    update set tgt.premiumamount = src.premiumamount
when not matched then
    insert (policyid, policyname, policytype, premiumamount, durationyears)
    values (src.policyid, src.policyname, src.policytype, src.premiumamount,
src.durationyears);

-- rollup – shows total premium per policy type and the overall total
select policytype, sum(premiumamount) as total_premium from policies
group by rollup(policytype);

-- cube – shows claim totals for every combination of claimstatus and year including
all totals
select claimstatus,
     year(claimdate) as claim_year,
     sum(claimamount) as total_claim
from claims
group by cube(claimstatus, year(claimdate));


-- case – categorizes claim amounts into low, medium and high */
select claimid,
     claimamount,
     case
       when claimamount < 20000 then 'low'
       when claimamount between 20000 and 70000 then 'medium'
       else 'high'
     end as claim_category
from claims;


--rollup with grouping – shows premium totals and identifies subtotal & grand total
rows
select
    case when grouping(policytype) = 1 then 'all policy types total'
       else policytype end as policy_type,
    sum(premiumamount) as total_premium from policies
group by rollup(policytype);
```

-- cube with grouping – shows claim totals for every combination and identifies rolled-up rows

```
select
    case when grouping(claimstatus) = 1 then 'all statuses'
       else claimstatus end as claim_status,
    case when grouping(year(claimdate)) = 1 then 'all years'
       else cast(year(claimdate) as varchar(10)) end as claim_year,
    sum(claimamount) as total_claim
from claims
group by cube(claimstatus, year(claimdate));
```

■ VIEWS, INDEX

```
-- index queries
create index idx_customers_firstname on customers(firstname);
create index idx_policies_premium on policies(premiumamount);

-- 1: clustered index on primary key
create clustered index idx_customers_customerid
on customers(customerid);

-- query using clustered index
select firstname, lastname, email
from customers
where customerid = 5;


-- 2: nonclustered index on frequently searched column
create nonclustered index idx_policies_policytype
on policies(policytype);

-- query using nonclustered index
select policyid, policyname, premiumamount
from policies
where policytype = 'health';
```

-- view queries
1)create view vw_customer_details as select firstname, lastname, email from customers;

2)create view vw_agent_policy_summary as select a.agentname, count(pa.policyid) as total_policies from agents a left join policyassignments pa on a.agentid = pa.agentid group by a.agentname;

-- view to show agent with the policies they handle
1) create view vw_agent_policy_info as select a.agentid, a.agentname, p.policyname, pa.startdate from agents a join policyassignments pa on a.agentid = pa.agentid join policies p on p.policyid = pa.policyid;

-- view to show customer along with their claim details
2) create view vw_customer_claim_summary as select c.cusid, c.fname, c.lname, cl.claimamount, cl.claimstatus, cl.claimdate from customers c join policyassignments pa on c.cusid = pa.cusid join claims cl on cl.assid = pa.assid;