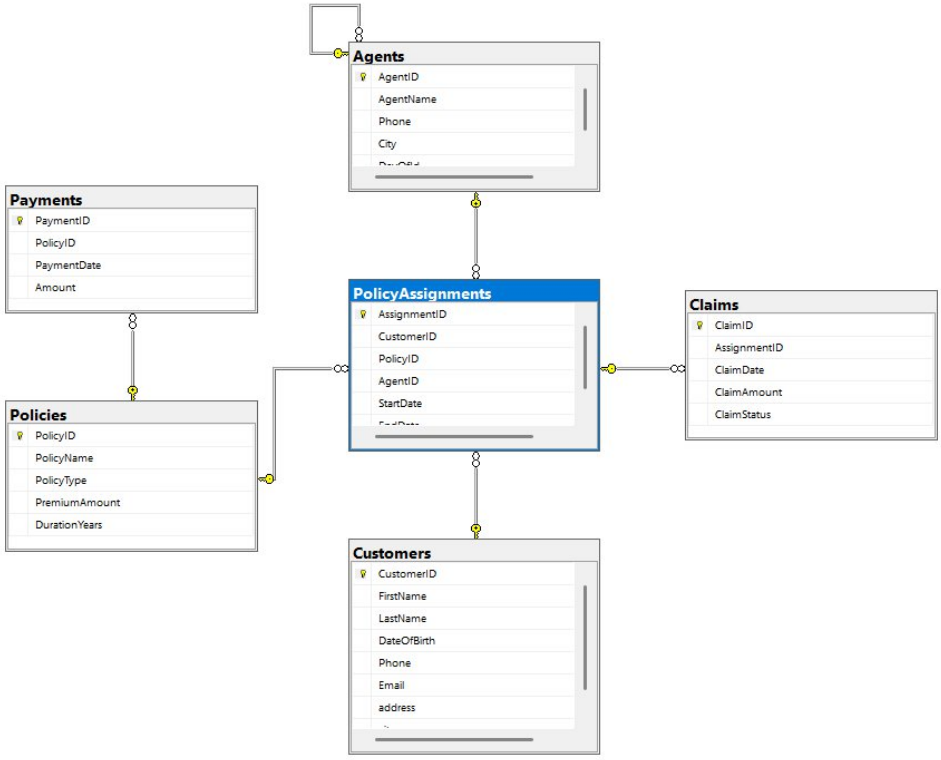
Module 4.4 Practical Project Assignment

1. Insurance Database Creation

```
create database insuranceDB;
```

2. Schema Diagram of Database



3. Create Table commands

```
create table Customers(
CustomerID int primary key identity,
FirstName nchar(50),
LastName nchar(50),
DateOfBirth date,
Phone nchar(50),
Email nchar(50)
);

create table Policies(
PolicyID int primary key identity,
PolicyName nchar(50),
PolicyType nchar(50),
PremiumAmount int,
DurationYears int
);


create table Agents(
AgentID int primary key identity,
AgentName nchar(50),
Phone nchar(10),
City nchar(30)
);
```

```
create table PolicyAssignments(
AssignmentID int primary key identity,
CustomerID int,
PolicyID int,
AgentID int,
StartDate date,
EndDate date
constraint customer_fk foreign key(CustomerID) references Customers(CustomerID),
constraint policy_fk foreign key(PolicyID) references Policies(PolicyID),
constraint agent_fk foreign key(AgentID) references Agents(AgentID),
constraint customer_fk foreign key(customerid) references customers(customerid)
);

create table Claims(
ClaimID int primary key identity,
AssignmentID int,
ClaimDate date,
ClaimAmount money,
ClaimStatus nvarchar(10)
constraint assignmentid_fk foreign key(AssignmentID) references PolicyAssignments(AssignmentID)
);

CREATE TABLE Payments (
PaymentID INT IDENTITY PRIMARY KEY,
PolicyID INT NOT NULL,
PaymentDate DATE NOT NULL,
Amount DECIMAL(10,2) NOT NULL,

CONSTRAINT fk_payments_policy
  FOREIGN KEY (PolicyID)
  REFERENCES Policies(PolicyID)
);
```

4. Insert commands

```
INSERT INTO Customers (FirstName, LastName, DateOfBirth, Phone, Email)
VALUES
('Rahul', 'Sharma', '1995-12-12', '9704873975', 'rahulez@gmail.com'),
('Neha', 'Shetty', '1999-07-12', '9014486579', 'neha69@gmail.com'),
('Amit', 'Patel', '1993-03-25', '9823456789', 'amit.patel@gmail.com'),
('Sneha', 'Reddy', '1997-11-05', '9345678123', 'sneha.reddy@gmail.com'),
('Arjun', 'Reddy', '1997-10-05', '9345678123', 'arjunreddy@gmail.com'),
('Ronin', 'Reddy', '2004-07-01', '9876543210', 'ronin@gmail.com');

INSERT INTO Policies (PolicyName, PolicyType, PremiumAmount, DurationYears)
VALUES
('Life Secure',        'Life Insurance',        15000, 20),
('HealthPlus',         'Health Insurance',      15972, 12),
('CarPlus',            'Car Insurance',         8000,  8),
('BoatPlus',           'Boat Insurance',        12000, 15),
('Two Wheeler Shield', 'Vehicle Insurance',     4000,  3),
('Home Shield',        'Property Insurance',    10000, 15),
('Home Premium',       'Property Insurance',    20000, 25),
('Travel Safe',        'Travel Insurance',      3000,  1),
('Child Future Plan',  'Life Insurance',        22000, 18),
('Retirement Secure',  'Life Insurance',        25000, 30),
('Disability Help',    'Disability Insurance',  12000,  1);
```

```
INSERT INTO Agents (AgentName, Phone, City)
VALUES
('Suresh Kumar', '9876501234', 'Hyderabad'),
('Neha Singh',   '9123409876', 'Bangalore'),
('Virat',        '9871201234', 'Delhi'),
('Vijay',        '9127809876', 'Chennai'),
('Mahesh',       '9812345678', 'Baroda');

INSERT INTO PolicyAssignments (CustomerID, PolicyID, AgentID, StartDate, EndDate)
VALUES
(1, 1, 1, '2022-01-01', '2042-01-01'),
(2, 2, 2, '2023-06-15', '2035-06-15'),
(7, 4, 4, '2025-12-11', '2035-07-08'),
(9, 4, 4, '2025-12-11', '2035-07-08'),
(10, 4, 3, '2024-12-11', '2036-07-08'),
(1, 2, 2, '2024-12-11', '2026-07-08'),
(1, 2, 4, '2022-12-11', '2025-07-08');

INSERT INTO Claims (AssignmentID, ClaimDate, ClaimAmount, ClaimStatus)
VALUES
(1, '2023-05-10', 50000, 'Approved'),
(2, '2024-02-18', 30000, 'Pending'),
(7, '2026-01-25', 75000, 'Approved'),
(8, '2027-08-12', 45000, 'Rejected'),
(1, '2024-11-05', 20000, 'Approved'),
(2, '2025-06-20', 15000, 'Pending');

INSERT INTO Payments (PolicyID, PaymentDate, Amount)
VALUES
(1, '2023-01-10', 15000),
(1, '2024-01-10', 15000),
(2, '2023-06-15', 15972),
(2, '2024-06-15', 15972),
(3, '2023-03-20', 8000),
(4, '2023-09-05', 12000),
(5, '2023-04-12', 4000),
(5, '2024-04-12', 4000),
(6, '2023-08-01', 10000),
(6, '2024-08-01', 10000),
(7, '2023-02-18', 20000),
(8, '2023-12-01', 3000),
(9, '2023-07-25', 22000),
(10, '2023-05-30', 25000),
(11, '2023-11-10', 12000);
```

5. Practice Queries
    a. Comparison Operators

i. Write a query to find all customers who live in 'Delhi'.

```
select * from Agents
where City='Delhi';
```

ii. Display policies with PremiumAmount greater than 25,000.

```
select PolicyName from policies
where PremiumAmount>25000;
```

iii. List claims where ClaimAmount is less than 50,000.

```
select * from Claims
where ClaimAmount<50000;
```

iv. **Find customers whose State is not 'Maharashtra'.**

```
select * from Customers
where state != 'Maharashtra';
```

v. **Display policies that expire on or before '2025-12-31'.**

```
select p.PolicyName from Policies as p
join PolicyAssignments as a on p.PolicyID=a.PolicyID
where a.EndDate <='2025-12-31';
```

vi. **Retrieve agents who joined after 1st Jan 2020.**

```
select * from claims
where ClaimDate>='2020-01-01';
```

b. Logical Operators

i. **Find policies where PolicyType is 'Health' AND PremiumAmount > 20,000.**

```
select * from Policies
where PolicyType='Health' and PremiumAmount >20000;
```

ii. **List customers who live in Mumbai OR Pune.**

```
select * from Agents
where City IN('Mumbai','Pune');
```

iii. **Display claims where ClaimStatus is NOT 'Approved'.**

```
select * from Claims
where ClaimStatus!='Approved';
```

iv. **Find policies that are Active AND PremiumAmount < 15,000.**

```
select * from Policies
where PremiumAmount <15000;
```

v. **Retrieve customers who are Male AND live in 'Bangalore'.**

```
select * from customers
where gender=Male and City ='Bangalore';
```

vi. **Show agents who joined before 2019 OR live in 'Chennai'.**

```
select * from agents
where StartDate<='2019-01-01' or city='Chennai';
```

c. In Operator

i. **Display customers whose City is IN ('Delhi', 'Mumbai', 'Chennai').**

```
select * from customers
where City In ('Delhi', 'Mumbai', 'Chennai');
```

ii. **Find policies with PolicyType IN ('Life', 'Health').**

```
select PolicyName from Policies
where PolicyType IN ('Life', 'Health');
```

iii. **Retrieve claims where ClaimStatus IN ('Rejected', 'Pending').**

```
select * from Claims
where ClaimStatus IN ('Rejected', 'Pending');
```

iv. **List payments made using PaymentMode IN ('UPI', 'Credit Card').**

```
select * from Payments
where PaymentMode IN  ('UPI', 'Credit Card');
```

d. Between Operator

i. **Find policies with PremiumAmount BETWEEN 10,000 AND 30,000.**

```
select * from policies
where PremiumAmount Between 10000 and 30000;
```

ii. **Retrieve claims made between '2024-01-01' and '2024-12-31'.**

```
select * from claims
where ClaimDate between '2024-01-01' and '2024-12-31';
```

iii. **List customers born between '1985-01-01' and '1995-12-31'.**

```
select * from customers
where DateOfBirth between '1985-01-01' and '1995-12-31';
```

iv. **Display payments where PaymentAmount BETWEEN 5,000 AND 20,000.**

```
select * from Payments
where Amount BETWEEN 5000 AND 20000;
```

e. Like Operator

i. **Find customers whose FirstName starts with 'A'.**

```
select * from Customers
where FirstName like 'A%';
```

ii. **Display agents whose AgentName ends with 'Sharma'.**

```
select * from Agents
where AgentName like '%sharma';
```

iii. **Retrieve customers whose LastName contains 'Singh'.**

```
select * from customers
where LastName like '%Singh%';
```

iv. **List cites from customers where City starts with 'B'.**

```
select * from Agents
where City like 'B%';
```

v. **Find policies where PolicyType contains the word 'Vehicle'.**

```
select * from Policies
where PolicyName like '%Vehicle%';
```

f. Is Null / Is Not Null

i. **Display policies where EndDate IS NULL.**

```
select PolicyName from Policies as p
join PolicyAssignments as a on p.PolicyID=a.PolicyID
where a.EndDate is NULL;
```

ii. **Find claims where ClaimStatus IS NOT NULL.**

```
select * from Claims
where ClaimStatus is NOT NULL;
```

iii. **Retrieve customers where State IS NULL.**

```
select * from Customers
where State is NULL;
```

g. Mixed/Tricky Scenarios

i. **Find policies where PolicyType = 'Health' AND PremiumAmount BETWEEN 15,000 AND 40,000.**

```
select * from policies
where policytype like 'Health%' and premiumamount between 15000 and 40000;
```

ii. **Display claims where ClaimAmount > 1,00,000 AND ClaimStatus <> 'Rejected'.**

```
select * from claims
where claimamount > 100000 and claimstatus <> 'Rejected';
```

6. Functions in Querying

a. String Function

i. **Display customer full name by concatenating FirstName and LastName.**

```
select RTRIM (firstname)+' '+RTRIM (lastname) as names_of_customers
from customers;
```

ii. **Find customers whose FirstName starts with 'A'.**

```
select * from customers
where firstname like 'A%';
```

iii. **Display LastName in UPPERCASE.**

```
select upper(lastname) from customers;
```

iv. **Find customers whose LastName length is more than 6 characters.**

```
select firstname,lastname from customers
where len(lastname) >= 6;
```

v. **Display the first 3 characters of FirstName.**

```
select left(firstname,3) from customers;
```

vi. **Replace the word 'Life' with 'Term Life' in PolicyType.**

```
update policies
set PolicyType=replace ('Term Life Insurance','Term Life','Life')
where PolicyType='Term Life Insurance';
```

vii. **Find customers whose city contains 'del' (case-insensitive).**

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

viii. **Remove leading and trailing spaces from customer FirstName.**

```
select LTRIM (firstname) from customers;
```

ix. **Display PolicyType and its character length.**

```
select distinct policytype,len(policytype) as length_of_string from policies;
```

x. **Extract last 4 characters from PolicyType.**

```
select right (RTRIM(PolicyType),4) from policies;
```

b. Numeric Function

i. **Display PremiumAmount rounded to nearest integer.**

```
select round(premiumamount,0) as roundedpremium from policies;
```

ii. **Find policies where PremiumAmount is greater than 5000 after rounding.**

```
select * from policies
where round(premiumamount,0) > 5000;
```

iii. **Display absolute value of ClaimAmount.**

```
select abs(claimamount) as absoluteclaimamount from claims;
```

iv. **Calculate 10% tax on PremiumAmount.**

```
select premiumamount,(premiumamount*0.10) as taxamount
from policies;
```

v. **Display PremiumAmount rounded to 2 decimal places.**

```
select round(premiumamount,2) as premiumrounded2decimals from policies;
```

vi. **Find the square root of PremiumAmount.**

```
select sqrt(premiumamount) as squarerootpremium from policies;
```

vii. **Display the ceiling value of ClaimAmount.**

```
select ceiling(claimamount) as ceilingclaimamount from claims;
```

viii. **Display the floor value of PremiumAmount.**

```
select floor(premiumamount) as floorpremiumamount from policies;
```

ix. **Calculate total premium including 18% GST.**

```
select premiumamount,(premiumamount+(premiumamount*0.18)) as totalpremiumwithgst from policies;
```

x. **Find policies where ClaimAmount modulo 2 = 0.**

```
select * from claims where claimamount % 2 = 0;
```

c. Date Function

i. **Display current system date.**

```
select GETDATE();
```

ii. **Find policies that started in the year 2023.**

```
select * from PolicyAssignments
where year (StartDate) = '2023';
```

iii. **Calculate policy duration in days.**

```
select policyname, (durationyears*365) as duration_as_days from policies;
```

iv. **Find customers whose birthday is in the current month.**

```
select * from customers
where month(dateofbirth)=month(getdate());
```

v. **Display ClaimDate in DD-MM-YYYY format.**

```
select format(startdate,'dd/MM/yyyy'),format(enddate,'dd/MM/yyyy') from policyassignments;
```

vi. **Find policies that expire within the next 30 days.**

```
select * from policyassignments
where year(getdate())+10 >= year(enddate);
```

vii. **Calculate customer age from DOB.**

```
select firstname,lastname,year(getdate())-year(dateofbirth) as age from customers;
```

viii. **Find claims filed in the last 6 months.**

```
select * from claims
where claimdate >= dateadd(month,-6,getdate());
```

ix. **Display month name from Policy StartDate.**

```
select datename(month,startdate) as monthname from policyassignments;
```

x. **Find policies where StartDate is a Monday.**

```
select * from policyassignments
where datename(weekday,startdate) = 'Monday';
```

d. Aggregate Function

i. **Find the total number of customers.**

```
select count(*) as totalcustomers from customers;
```

ii. **Calculate total premium collected.**

```
select sum(premiumamount) as totalpremiumcollected from policies;
```

iii. **Find average premium amount.**

```
select avg(premiumamount) as averagepremium from policies;
```

iv. **Find the maximum claim amount.**

```
select max(claimamount) as maximumclaimamount from claims;
```

v. **Find the minimum premium amount.**

```
select min(premiumamount) as minimumpremiumamount from policies;
```

vi. **Count number of policies per PolicyType.**

```
select policytype,count(*) as policycount from policies
group by policytype;
```

vii. **Find total ClaimAmount per ClaimStatus.**

```
select claimstatus,sum(claimamount) as totalclaimamount from claims
group by claimstatus;
```

viii. **Find PolicyTypes where average premium > 6000.**

```
select policytype from policies
group by policytype
having avg(premiumamount) > 6000;
```

ix. **Count number of claims per year**.

```
select year(claimdate) as claimyear,count(*) as claimcount from claims
group by year(claimdate);
```

x. **Find customers having more than 2 policies.**

```
select customerid from policyassignments
group by customerid
having count(policyid) > 2;
```

7. Queries Using Joins

a. List all Policies for a CustomerId 5.

```
select * from policyassignments p
join customers c on
p.customerid=c.customerid
where c.customerid=1;
```

b. View all customers with their policies.

```
select c.customerid,c.firstname,c.lastname,po.policyname,po.policytype from customers c
left join policyassignments p
on c.customerid=p.customerid
left join policies po on
p.policyid=po.policyid;
```

c. View claims with customer name.

```
select c.firstname,c.lastname,cl.claimDate,cl.claimamount from claims cl
left join policyassignments po
on cl.assignmentid=po.assignmentid
left join customers c
on po.customerid=c.customerid;
```

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

```
select c.firstname,p.policyname,a.agentname,po.startdate,po.enddate from policyassignments po
join customers c on
c.customerid=po.customerid
join agents a on
po.agentid=a.agentid
join policies p on
po.policyid=p.policyid;
```

e. **Display claims report with FirstName, PolicyName, ClaimAmount, ClaimStatus, and ClaimDate** from their respective tables.

```
select c.firstname,p.policyname,cl.claimamount,cl.claimstatus,cl.claimdate from claims cl
join policyassignments po on
cl.assignmentid=po.assignmentid
join customers c on
po.customerid=c.customerid
join policies p on
po.policyid=p.policyid;
```

f. Display records of Customers with or without Policies.

```
select c.firstname,p.policyname from customers c
```

```
left join policyassignments po on
c.customerid=po.customerid
left join policies p on
po.policyid=p.policyid
where po.policyid is not null;
```

g. Display all Customers with NO Claims.

```
select c.firstname,c.lastname,claimid,po.assignmentid from customers c
left join policyassignments po on
po.customerid=c.customerid
left join claims cl on
po.assignmentid=cl.assignmentid
where claimid is null and po.assignmentid is not null;
```

h. Show CustomerName with Total Claim Amount per Customer.

```
select c.firstname,c.customerid,sum(cl.claimamount) from claims cl
join policyassignments po on
po.assignmentid=cl.assignmentid
join customers c on
po.customerid=c.customerid
group by c.customerid,c.firstname;
```

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

```
select c.firstname,c.customerid,sum(cl.claimamount) as total_claim from claims cl
join policyassignments po on
po.assignmentid=cl.assignmentid
join customers c on
po.customerid=c.customerid
group by c.customerid,c.firstname
having sum(cl.claimamount) > 50000;
```

j. Display list with Agent Wise Policy Count.

```
select a.agentid,a.agentname,count(customerid) as total_policies_sold from
policyassignments p
right join agents a on
a.agentid=p.agentid
group by a.agentid,a.agentname
order by total_policies_sold desc;
```

8. Queries Using Subquery
   a. Subquery using exists
      i. Find customers who have at least one policy.

```
select * from customers c
where exists (select 1 from policyassignments pa
```

```
                    where pa.customerid = c.customerid);
```

ii. List customers who have made at least one claim.

```
select * from customers c
where exists (select 1 from policyassignments pa
                    join claims cl on pa.assignmentid = cl.assignmentid
                    where pa.customerid = c.customerid);
```

iii. Find policies for which at least one claim exists.

```
select * from policies p
where exists (select 1 from policyassignments pa
                    join claims cl on pa.assignmentid = cl.assignmentid
                    where pa.policyid = p.policyid);
```

iv. Display customers who do not have any claims.

```
select * from customers c
where not exists (select 1 from policyassignments pa
                        join claims cl on pa.assignmentid = cl.assignmentid
                        where pa.customerid = c.customerid);
```

v. Find customers who have more than one policy (using EXISTS with correlated subquery).

```
select * from customers c
where exists (select 1 from policyassignments pa1
                        where pa1.customerid = c.customerid and exists (select 1 from
                        policyassignments pa2
                        where pa2.customerid = pa1.customerid and pa2.assignmentid <>
                        pa1.assignmentid));
```

b. Subqueries using any

i. Find policies with premium greater than ANY premium of policies held by CustomerID = 101.

```
select * from policies
where premiumamount > any (select p.premiumamount from policies p
                        join policyassignments pa on p.policyid = pa.policyid where pa.customerid =
                        101);
```

ii. Find claims where claim amount is greater than ANY claim made in 2024.

```
select * from claims
where claimamount > any (select claimamount from claims
                                where year(claimdate) = 2024);
```

iii. List agents who handle policies whose premium is greater than ANY premium of Life policies.

```
select distinct a.* from agents a
join policyassignments pa on a.agentid = pa.agentid
join policies p on pa.policyid = p.policyid
where p.premiumamount > any (select premiumamount from policies
```

```
                                        where policytype like 'Life%');
```

iv. Find claims whose amount is less than ANY rejected claim.

```
select * from claims
where claimamount < any (select claimamount from claims
                                        where claimstatus = 'Rejected');
```

v. Find customers whose policy premium is greater than ANY premium of customers from Mumbai.

```
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
                                        = 'Mumbai');
```

c. Subqueries using All

i. Find policies with premium greater than ALL premiums of Motor policies.

```
select * from policies
where premiumamount > all (select premiumamount from policies
                                        where policytype like 'Motor%');
```

ii. Find claims where claim amount is greater than ALL claims made in 2023.

```
select * from claims
where claimamount > all (select claimamount from claims
                                        where year(claimdate) = 2023);
```

iii. Find customers whose policy premium is greater than ALL premiums of customers from Chennai.

```
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
                                        join policyassignments pa2 on p2.policyid = pa2.policyid
                                        join customers c2 on pa2.customerid = c2.customerid where c2.city
                                        = 'Chennai');
```

iv. Display customers whose claim amount is less than ALL rejected claim amounts.

```
select distinct c.* from customers c
join policyassignments pa on c.customerid = pa.customerid
join claims cl on pa.assignmentid = cl.assignmentid
where cl.claimamount < all (select claimamount from claims
                                        where claimstatus = 'Rejected');
```

v. Display policies whose premium is less than ALL premiums of policies having claims.

```
select * from policies
```

```
where premiumamount < all (select p.premiumamount from policies p
                           join policyassignments pa on p.policyid = pa.policyid
                           join claims c on pa.assignmentid = c.assignmentid);
```

9. Queries using SET Operator

a. Customers who bought Life OR Health policies.

```
select c.customerid,c.firstname,c.lastname from customers c
join policyassignments pa on c.customerid=pa.customerid
join policies p on pa.policyid=p.policyid
where p.policytype like 'Life%'
union
select c.customerid,c.firstname,c.lastname from customers c
join policyassignments pa on c.customerid=pa.customerid
join policies p on pa.policyid=p.policyid
where p.policytype like 'Health%';
```

b. Customers who bought both Life AND Health policies.

```
select c.customerid,c.firstname,c.lastname from customers c
join policyassignments pa on c.customerid=pa.customerid
join policies p on pa.policyid=p.policyid
where p.policytype like 'Life%'
intersect
select c.customerid,c.firstname,c.lastname from customers c
join policyassignments pa on c.customerid=pa.customerid
join policies p on pa.policyid=p.policyid
where p.policytype like 'Health%';
```

c. Customers who bought Life but NOT Health policies.

```
select c.customerid,c.firstname,c.lastname from customers c
join policyassignments pa on c.customerid=pa.customerid
join policies p on pa.policyid=p.policyid
where p.policytype like 'Life%'
except
select c.customerid,c.firstname,c.lastname from customers c
join policyassignments pa on c.customerid=pa.customerid
join policies p on pa.policyid=p.policyid where p.policytype like 'Health%';
```

d. Distinct customers with policies.

```
select p.policyid,p.policyname from policies p
join policyassignments pa on p.policyid=pa.policyid
union
select p.policyid,p.policyname from policies p
join policyassignments pa on p.policyid=pa.policyid
join claims c on pa.assignmentid=c.assignmentid;
```

e. Customers with policies but no claims.

```
select c.customerid,c.firstname,c.lastname from customers c
```

```
join policyassignments pa on c.customerid=pa.customerid
except
select c.customerid,c.firstname,c.lastname from customers c
join policyassignments pa on c.customerid=pa.customerid
join claims cl on pa.assignmentid=cl.assignmentid;
```

10. CASE…ELSE

Classify policies based on their PremiumAmount as:

'Low Premium' if PremiumAmount < 10,000

'Medium Premium' if PremiumAmount is between 10,000 and 20,000

'High Premium' if PremiumAmount > 20,000

```
select policyname,premiumamount,
case
        when premiumamount < 10000 then 'Low Premium'
        when premiumamount between 10000 and 20000 then 'Medium Premium'
else 'High Premium'
end as premiumcategory
from policies;
```

11. MERGE in SQL Server

Synchronize agent city updates from a temporary update table into the main agents table.

```
-- create staging table for agent updates
create table agent_updates (
   agentid int,
   agentname varchar(50),
   phone varchar(10),
   city varchar(30)
);

-- insert data into staging table
insert into agent_updates (agentid, agentname, phone, city) values
(1, 'suresh kumar', '9876501234', 'chennai'),
(3, 'virat', '9871201234', 'pune'),
(7, 'rohit sharma', '9991112223', 'mumbai');

-- merge update data into main agents table
merge agents as target
using agent_updates as source
on target.agentid = source.agentid
when matched then
   update set
      target.city = source.city
when not matched then
   insert (agentname, phone, city)
```

```
values (source.agentname, source.phone, source.city);
```

12. ROLLUP IN SQL SERVER

Show the total premium amount collected per PolicyType and PolicyName, including, subtotal for each PolicyType, grand total across all policies

```
select policytype, policyname, sum(premiumamount) as total_premium
from policies
group by rollup (policytype, policyname);
```

13. CUBE IN SQL SERVER

Generate a report showing the total premium amount based on PolicyType and PolicyName, including total premium for each PolicyType, PolicyName, PolicyType–PolicyName combination, grand total of all policies

```
select policytype, policyname, sum(premiumamount) as total_premium
from policies
group by cube (policytype, policyname);
```

14. GROUPING SETS IN SQL SERVER

Generate a report that shows, total premium amount by PolicyType, DurationYears, the grand total premium amount across all policies

```
select policytype, durationyears, sum(premiumamount) as total_premium
from policies
group by grouping sets ((policytype),(durationyears),());
```