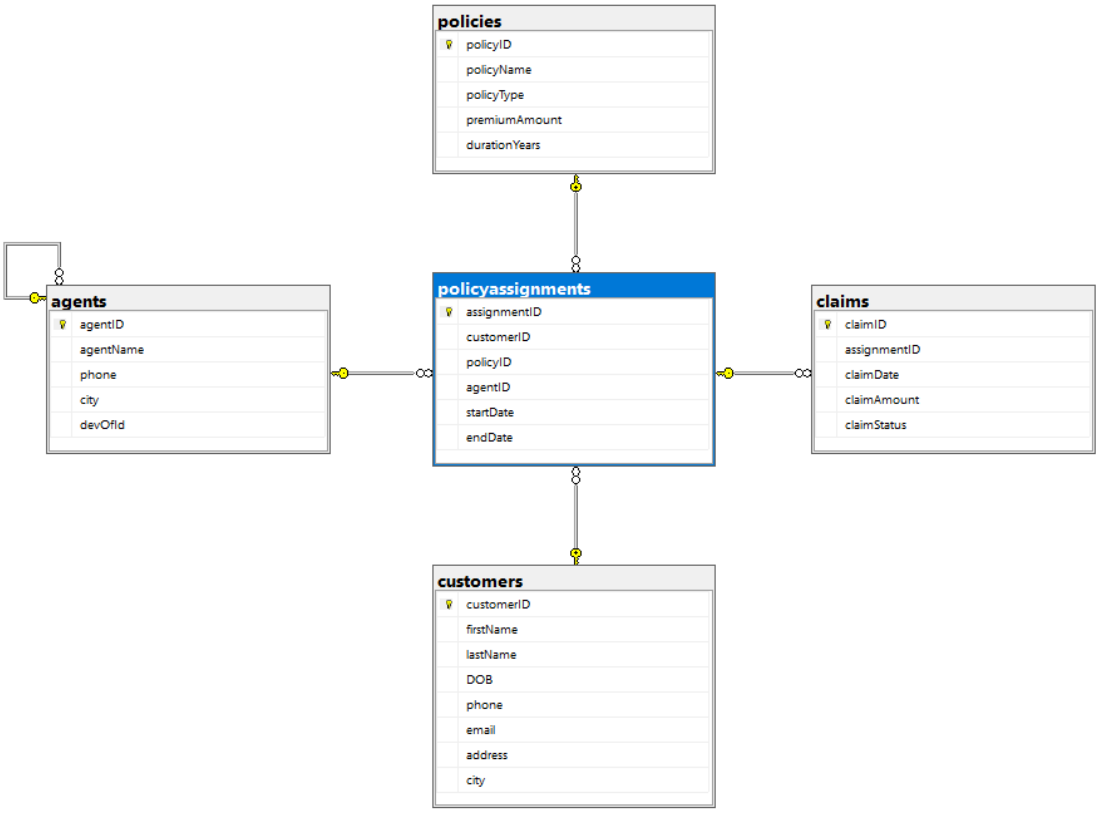
Module 4.4 : practical project assignment



1. Create Database command.

create database insurance;

use insurance;

2. Create table commands for all the tables with constraints, relationships etc.

```
CREATE TABLE customers (
    customerID INT PRIMARY KEY,
    firstName VARCHAR(50) NOT NULL,
    lastName VARCHAR(50) NOT NULL,
    DOB DATE NOT NULL,
    phone VARCHAR(15) NOT NULL,
```

```
    email VARCHAR(100) NOT NULL,
    CONSTRAINT UQ_customers_email UNIQUE (email),
    CONSTRAINT CK_customers_DOB CHECK (DOB < GETDATE())
);

CREATE TABLE agents (
    agentID INT PRIMARY KEY,
    agentName VARCHAR(50) NOT NULL,
    phone VARCHAR(15) NOT NULL,
    city VARCHAR(50) NOT NULL,

    CONSTRAINT UQ_agents_phone UNIQUE (phone)
);

CREATE TABLE policies (
    policyID INT PRIMARY KEY,
    policyName VARCHAR(50) NOT NULL,
    policyType VARCHAR(50) NOT NULL,
    premiumAmount MONEY NOT NULL,
    durationYears INT NOT NULL,

    CONSTRAINT CK_policies_premium CHECK (premiumAmount > 0),
    CONSTRAINT CK_policies_duration CHECK (durationYears > 0)
);

CREATE TABLE policyassignments (
    assignmentID INT PRIMARY KEY,
    customerID INT NOT NULL,
```

```
    policyID INT NOT NULL,
    agentID INT NOT NULL,
    startDate DATE NOT NULL,
    endDate DATE NOT NULL,

    CONSTRAINT FK_policyassignments_customers
        FOREIGN KEY (customerID)
        REFERENCES customers(customerID),

    CONSTRAINT FK_policyassignments_policies
        FOREIGN KEY (policyID)
        REFERENCES policies(policyID),

    CONSTRAINT FK_policyassignments_agents
        FOREIGN KEY (agentID)
        REFERENCES agents(agentID),

    CONSTRAINT CK_policyassignments_dates
        CHECK (endDate > startDate)
);

CREATE TABLE policyassignments (
    assignmentID INT PRIMARY KEY,
    customerID INT NOT NULL,
    policyID INT NOT NULL,
    agentID INT NOT NULL,
    startDate DATE NOT NULL,
    endDate DATE NOT NULL,
```

```
    CONSTRAINT FK_policyassignments_customers
        FOREIGN KEY (customerID)
        REFERENCES customers(customerID),

    CONSTRAINT FK_policyassignments_policies
        FOREIGN KEY (policyID)
        REFERENCES policies(policyID),

    CONSTRAINT FK_policyassignments_agents
        FOREIGN KEY (agentID)
        REFERENCES agents(agentID),
);

CREATE TABLE claims (
    claimID INT PRIMARY KEY,
    assignmentID INT NOT NULL,
    claimDate DATE NOT NULL,
    claimAmount MONEY NOT NULL,
    claimStatus VARCHAR(20) NOT NULL,

    CONSTRAINT FK_claims_policyassignments
        FOREIGN KEY (assignmentID)
        REFERENCES policyassignments(assignmentID),
);
```

3. Insert commands for all tables.

```
INSERT INTO customers VALUES
(1, 'Ravi',  'Kumar', '1998-05-10', '9876543210', 'ravi@gmail.com'),
(2, 'Anita', 'Sharma','2003-03-15', '9876543211', 'anita@gmail.com'),
(3, 'Suresh','Reddy', '2010-07-20', '9876543212', 'suresh@gmail.com'),
(4, 'Priya', 'Verma', '1995-11-25', '9876543213', 'priya@gmail.com'),
(5, 'Amit',  'Patel', '2005-01-05', '9876543214', 'amit@gmail.com');

INSERT INTO agents VALUES
(1, 'Ramesh Rao', '9123456780', 'Hyderabad'),
(2, 'Sunita Das', '9123456781', 'Nagpur'),
(3, 'Vikram Singh','9123456782', 'Jaipur');

INSERT INTO policies VALUES
(1, 'Life Secure',   'Life',   15000, 1),
(2, 'Health Plus',   'Health', 12000, 1),
(3, 'Motor Shield',  'Motor',   8000,  1),
(4, 'Health Gold',   'Health', 20000, 2);

INSERT INTO policyassignments VALUES
(1, 5, 1, 1, '2023-01-01', '2024-01-01'),
(2, 2, 2, 2, '2024-02-01', '2025-02-01'),
(3, 1, 3, 3, '2023-06-01', '2024-06-01'),
(4, 4, 4, 1, '2024-01-01', '2026-01-01');
```

```
INSERT INTO claims VALUES
(1, 1, '2023-05-10', 30000, 'Approved'),
(2, 2, '2024-06-15', 20000, 'Rejected'),
(3, 4, '2024-08-01', 60000, 'Approved');
```

SELECT (BASIC)

1. View all records from Customers table.

```
select * from customers;
```

2. View customerID, policyID, startDate and endDate from PolicyAssignments table.

```
select customerID, policyID, startDate, endDate
from policyassignments;
```

3. Display all policies of Health type.

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

4. Display unique city names from where agents belong.

```
select distinct city
from agents;
```

5. Display claims where claim status is Rejected.

```
select * from claims
where claimStatus = 'Rejected';
```

6. Display policyName and premiumAmount from Policies table.

```
select policyName, premiumAmount
from policies;
```

UPDATE, ALTER AND DELETE

1. Increase premium by 10% for Health policies.

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

2. Delete expired policy assignments.

```
delete from policyassignments
where endDate < cast(getdate() as date);
```

3. Add address column to customers.

```
alter table customers
add address varchar(100);
```

4. Add city column to customers.

```
alter table customers
add city varchar(50);
```

5. Add DevOfId column to agents.

```
alter table agents
add devOfId int;
```

6. Make DevOfId a recursive foreign key.

```
alter table agents
add constraint FK_agents_dev
foreign key (devOfId)
references agents(agentID);
```

SELECT WITH AGGREGATE FUNCTIONS

1. Display total number of claims.

```
select count(*) as totalClaims
from claims;
```

2. Display total claim amount.

```
select sum(claimAmount) as totalClaimAmount
from claims;
```

3. Display average claim amount.

```
select avg(claimAmount) as averageClaimAmount
from claims;
```

4. Display highest claim amount.

```
select max(claimAmount) as highestClaim
from claims;
```

5. Display lowest claim amount.

```
select min(claimAmount) as lowestClaim
from claims;
```

6. Display number of policies.

```
select count(*) as totalPolicies
from policies;
```

SELECT WITH DATE FUNCTIONS (T-SQL)

1. Display current system date.

```
select getdate() as currentDate;
```

2. Display claims from the current year.

```
select * from claims
where year(claimDate) = year(getdate());
```

3. Display claims from the current month.

```
select * from claims
where month(claimDate) = month(getdate())
and year(claimDate) = year(getdate());
```

4. Display claims from last 30 days.

```
select * from claims
where claimDate >= dateadd(day,-30,getdate());
```

5. Display claims created today.

```
select * from claims
where cast(claimDate as date) = cast(getdate() as date);
```

6. Display year of each claim.

```
select claimID, year(claimDate) as claimYear
from claims;
```

SELECT WITH WHERE , AND OPERATORS

1. Display policies with premiumAmount > 10000.

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

2. Display policies with durationYears = 1.

```
select * from policies
where durationYears = 1;
```

3. Display policies of Life, Health using OR.

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

4. Display policies using IN operator.

```
select * from policies
where policyType in ('Life','Health','Motor');
```

5. Display customers born after 2001.

```
select * from customers
where DOB >= '2001-01-01';
```

6. Display agents whose city second letter is ‘a’.

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

SELECT WITH GROUP BY AND HAVING

1. Display number of claims per claim status.

```
select claimStatus, count(*) as totalClaims
from claims
group by claimStatus;
```

2. Display total claim amount per claim status.

```
select claimStatus, sum(claimAmount) as totalAmount
from claims
group by claimStatus;
```

3. Display average claim amount per claim status.

```
select claimStatus, avg(claimAmount) as avgAmount
from claims
group by claimStatus;
```

4. Display number of claims per assignment.

```
select assignmentID, count(*) as totalClaims
from claims
group by assignmentID;
```

5. Display claim statuses having more than 1 claim.

```
select claimStatus, count(*) as totalClaims
from claims
group by claimStatus
having count(*) > 1;
```

6. Display assignments with total claim amount > 50000.

```
select assignmentID, sum(claimAmount) as totalAmount
from claims
group by assignmentID
having sum(claimAmount) > 50000;
```

7. Display policy types having more than 1 policy.

```
select policyType, count(*) as totalPolicies
from policies
group by policyType
having count(*) > 1;
```

8. Display assignments with highest claim > 40000.

```
select assignmentID, max(claimAmount) as maxClaim
from claims
group by assignmentID
having max(claimAmount) > 40000;
```

SELECT WITH ORDER BY

1. Display latest claim record.

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

2. Display claims ordered by amount descending.

```
select * from claims
order by claimAmount desc;
```

3. Display customers ordered by DOB.

```
select * from customers
order by DOB;
```

4. Display policies ordered by premiumAmount.

```
select * from policies
order by premiumAmount desc;
```

5. Display assignments ordered by endDate.

```
select * from policyassignments
order by endDate;
```

6. Display agents ordered by city.

```
select * from agents
order by city;
```

JOINS

1. List all policies for customerID = 5.

```
select p.policyName
from policies p
join policyassignments pa on p.policyID = pa.policyID
where pa.customerID = 5;
```

2. View all customers with their policies.

```
select c.firstName, p.policyName
from customers c
join policyassignments pa on c.customerID = pa.customerID
join policies p on pa.policyID = p.policyID;
```

3. View claims with customer name.

```
select c.firstName, cl.claimAmount
from customers c
join policyassignments pa on c.customerID = pa.customerID
join claims cl on pa.assignmentID = cl.assignmentID;
```

4. Display FirstName, PolicyName, AgentName.

```
select c.firstName, p.policyName, a.agentName
from policyassignments pa
join customers c on pa.customerID = c.customerID
join policies p on pa.policyID = p.policyID
join agents a on pa.agentID = a.agentID;
```

5. Display customers with or without policies.

```
select c.firstName, p.policyName
from customers c
left join policyassignments pa on c.customerID = pa.customerID
left join policies p on pa.policyID = p.policyID;
```

6. Display customers with no claims.

```
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;
```

SUBQUERIES

1. Display claims greater than average claim amount.

```
select * from claims
where claimAmount > (select avg(claimAmount) from claims);
```

2. Display highest claim.

```
select * from claims
where claimAmount = (select max(claimAmount) from claims);
```

3. Display policies with premium higher than average.

```
select * from policies
where premiumAmount > (select avg(premiumAmount) from policies);
```

4. Display customers having at least one policy.

```
select * from customers c
where exists (
   select 1 from policyassignments pa
   where pa.customerID = c.customerID
);
```

5. Display customers with no policies.

```
select * from customers c
where not exists (
   select 1 from policyassignments pa
   where pa.customerID = c.customerID
);
```

SET OPERATORS

1. Customers who bought Life or Health policies.

```
select customerID from policyassignments pa
join policies p on pa.policyID = p.policyID
where p.policyType = 'Life'
union
select customerID from policyassignments pa
join policies p on pa.policyID = p.policyID
where p.policyType = 'Health';
```

2. Customers who bought both Life and Health.

```
select customerID from policyassignments pa
join policies p on pa.policyID = p.policyID
where p.policyType = 'Life'
intersect
select customerID from policyassignments pa
join policies p on pa.policyID = p.policyID
where p.policyType = 'Health';
```

3. Customers who bought Life but not Health.

```
select customerID from policyassignments pa
join policies p on pa.policyID = p.policyID
where p.policyType = 'Life'
except
select customerID from policyassignments pa
join policies p on pa.policyID = p.policyID
where p.policyType = 'Health';
```

4. Customers with policies (duplicates allowed).

```
select customerID from policyassignments
union all
select customerID from policyassignments;
```

5. Distinct customers with policies.

```
select customerID from policyassignments
union
select customerID from policyassignments;
```

6. Customers with policies but no claims.

```
select customerID from policyassignments
except
select pa.customerID
from claims c
join policyassignments pa on c.assignmentID = pa.assignmentID;
```

CASE WHEN ELSE:

1 .Display claim amount category (Low / Medium / High) based on ClaimAmount.

```
select  claimID, claimAmount,
case
   when claimAmount < 25000 then 'Low'
   when claimAmount between 25000 and 75000 then 'Medium'
   else 'High'
end as claimCategory
from claims;
```

2 Display claim status description using CASE WHEN ELSE.

```
select  claimID, claimStatus,
case
   when claimStatus = 'Approved' then 'Claim Approved'
   when claimStatus = 'Rejected' then 'Claim Rejected'
   else 'Pending Review'
end as statusDescription
from claims;
```

1. MERGE

Update policy data from an external system. Synchronize the policies table with newPolicies.

```
MERGE policies AS T
USING newPolicies AS S
ON T.policyID = S.policyID

WHEN MATCHED THEN
   UPDATE SET
      T.policyType = S.policyType,
      T.premiumAmount = S.premiumAmount

WHEN NOT MATCHED BY TARGET THEN
   INSERT (policyID, policyType, premiumAmount)
   VALUES (S.policyID, S.policyType, S.premiumAmount)

WHEN NOT MATCHED BY SOURCE THEN
   DELETE;
```

ROLLUP

1 Display total claim amount per customer along with the grand total.

```
select
c.firstName,
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 rollup (c.firstName);
```

CUBE

1 Display total claim amount by customer and policy type, including all possible subtotals.

```
select
c.firstName,
p.policyType,
sum(cl.claimAmount) as totalClaimAmount
from customers c
join policyassignments pa on c.customerID = pa.customerID
join policies p on pa.policyID = p.policyID
join claims cl on pa.assignmentID = cl.assignmentID
group by cube (c.firstName, p.policyType);
```

GROUPING SETS

Display total claim amount per customer, per policy type, and the overall total.

```
select
c.firstName,
p.policyType,
sum(cl.claimAmount) as totalClaimAmount
from customers c
join policyassignments pa on c.customerID = pa.customerID
join policies p on pa.policyID = p.policyID
join claims cl on pa.assignmentID = cl.assignmentID
group by grouping sets (
   (c.firstName),
   (p.policyType),
   ()
);
```