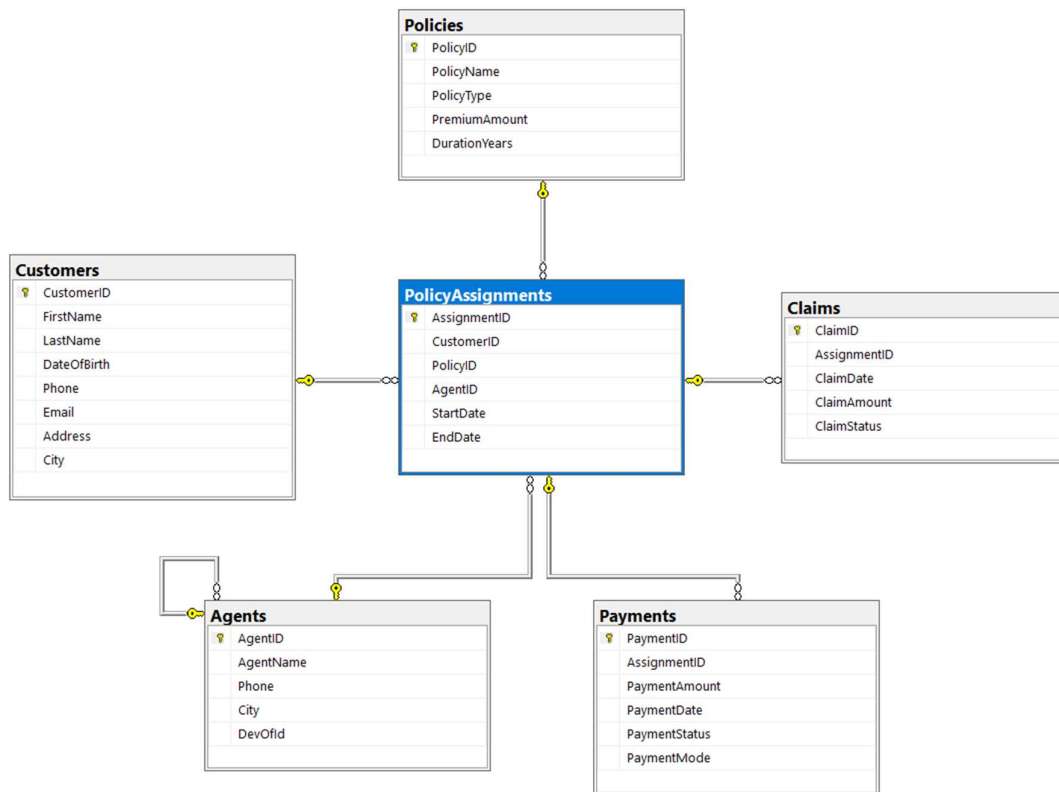


MODULE 4.4 Practical Project Assignment

InsuranceDB:

1. **CREATE DATABASE InsuranceDB;**
USE InsuranceDB;
2. Create table commands for all the tables with constraints, relationships etc.



```
CREATE TABLE Customers(  
CustomerID INT PRIMARY KEY,  
FirstName VARCHAR(50),  
LastName VARCHAR(50),  
DateOfBirth DATE,  
Phone VARCHAR(20),  
Email VARCHAR(50)  
);
```

```
CREATE TABLE Policies(  
PolicyID INT PRIMARY KEY,  
PolicyName VARCHAR(50),  
PolicyType VARCHAR(20),  
PremiumAmount DECIMAL(10,2),  
DurationYears VARCHAR(20)  
);
```

```

CREATE TABLE Agents(
AgentID INT PRIMARY KEY,
AgentName VARCHAR(50),
Phone VARCHAR(20),
City VARCHAR(20)
);

CREATE TABLE PolicyAssignments(
AssignmentID INT PRIMARY KEY,
CustomerID INT,
PolicyID INT,
AgentID INT,
StartDate DATE,
EndDate DATE,
Constraint fk_customer_pa
FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID),
Constraint fk_policy_pa
FOREIGN KEY (PolicyID) REFERENCES Policies(PolicyID),
Constraint fk_agent_pa
FOREIGN KEY (AgentID) REFERENCES Agents(AgentID)
);

CREATE TABLE Claims(
ClaimID INT PRIMARY KEY,
AssignmentID INT,
ClaimDate DATE,
ClaimAmount DECIMAL(10,2),
ClaimStatus VARCHAR(10),
Constraint fk_pa_Claims
FOREIGN KEY (AssignmentID) REFERENCES PolicyAssignments(AssignmentID)
);

CREATE TABLE Payments(
PaymentID INT PRIMARY KEY,
AssignmentID INT NOT NULL,
PaymentAmount DECIMAL(10,2),
PaymentDate DATE,
PaymentStatus VARCHAR(20),
PaymentMode VARCHAR(20),
FOREIGN KEY (AssignmentID) REFERENCES PolicyAssignments(AssignmentID)
);

```

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

```

INSERT INTO Customers (CustomerID, FirstName, LastName, DateOfBirth,
Phone, Email)
VALUES
(1, 'Ravi', 'Kumar', '1990-05-12', '9876543210',
'ravi.kumar@example.com'),
(2, 'Priya', 'Sharma', '1988-11-22', '9876543211',
'priya.sharma@example.com'),
(3, 'Amit', 'Verma', '1995-03-15', '9876543212',
'amit.verma@example.com'),
(4, 'Sneha', 'Reddy', '1992-07-09', '9876543213',
'sneha.reddy@example.com');

INSERT INTO Customers (CustomerID, FirstName, LastName, DateOfBirth,
Phone, Email)
VALUES

```

```
(5, 'Chaitanya', 'Talasila', '2005-04-15', '9876543210',  
'chaitanya@example.com');
```

```
INSERT INTO Policies (PolicyID, PolicyName, PolicyType, PremiumAmount,  
DurationYears)
```

```
VALUES
```

```
(101, 'Health Secure', 'Health', 12000.00, '3 Years'),  
(102, 'Life Shield', 'Life', 25000.00, '10 Years'),  
(103, 'Motor Protect', 'Vehicle', 8000.00, '1 Year'),  
(104, 'Senior Care', 'Health', 18000.00, '5 Years');
```

```
INSERT INTO Agents (AgentID, AgentName, Phone, City)
```

```
VALUES
```

```
(201, 'Anil Mehta', '9000011111', 'Hyderabad'),  
(202, 'Kavya Rao', '9000011112', 'Mumbai'),  
(203, 'Rohit Singh', '9000011113', 'Delhi');
```

```
INSERT INTO Agents (AgentID, AgentName, Phone, City)
```

```
VALUES
```

```
(204, 'Harshadh', '9000041111', 'Calicut');
```

```
INSERT INTO PolicyAssignments
```

```
(AssignmentID, CustomerID, PolicyID, AgentID, StartDate, EndDate)
```

```
VALUES
```

```
(301, 1, 101, 201, '2023-01-01', '2026-01-01'),  
(302, 1, 102, 202, '2020-05-10', '2030-05-10'),  
(303, 2, 103, 201, '2024-01-01', '2025-01-01'),  
(304, 3, 104, 203, '2022-06-15', '2027-06-15'),  
(305, 4, 101, 202, '2021-02-02', '2024-02-02');
```

```
SELECT * FROM PolicyAssignments;
```

```
INSERT INTO Claims
```

```
(ClaimID, AssignmentID, ClaimDate, ClaimAmount, ClaimStatus)
```

```
VALUES
```

```
(401, 301, '2024-03-12', 5000.00, 'Approved'),  
(402, 303, '2024-04-15', 12000.00, 'Pending'),  
(403, 305, '2023-10-10', 8000.00, 'Rejected'),  
(404, 304, '2024-01-25', 15000.00, 'Approved'),  
(405, 302, '2024-06-30', 30000.00, 'Pending');
```

```
INSERT INTO Payments
```

```
(PaymentID, AssignmentID, PaymentAmount, PaymentDate, PaymentStatus,  
PaymentMode)
```

```
VALUES
```

```
(501, 301, 6000, '2024-02-10', 'Paid', 'Online'),  
(502, 301, 6000, '2024-05-10', 'Paid', 'Cash'),  
(503, 302, 25000, '2024-03-05', 'Pending', 'Cheque'),  
(504, 303, 8000, '2024-01-15', 'Paid', 'Online'),  
(505, 303, 4000, '2024-04-20', 'Failed', 'Online'),  
(506, 304, 9000, '2023-12-30', 'Paid', 'Card'),  
(507, 304, 9000, '2024-07-10', 'Refunded', 'Card'),  
(508, 305, 12000, '2023-08-10', 'Paid', 'Cash');
```

4. Select commands

-- 1. View all records Customers table.

```
SELECT * FROM Customers;
```

-- 2. View all records of PolicyAssignment table with CustomerId, PolicyId, StartDate and EndDate columns only.

```
SELECT CustomerId, PolicyId, StartDate, EndDate FROM PolicyAssignments;
```

-- 3. Display all policies of Health type.

```
SELECT PolicyName FROM Policies;
```

-- 4. Display policies having premium amount more than 10000 and DurationYears is 1 Year

```
SELECT PolicyName, PremiumAmount, DurationYears FROM Policies WHERE  
PremiumAmount>1000 and DurationYears ='1 Year';
```

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

```
SELECT DISTINCT(city) FROM Agents;
```

-- 6. List policies of type Life, Health, Motor using OR clause.

```
SELECT PolicyName FROM Policies WHERE PolicyType='Life' OR PolicyType='Health' OR  
PolicyType='Vehicle';
```

-- 7. List policies of type Life, Health, Motor using IN operator.

```
SELECT PolicyName FROM Policies WHERE PolicyType IN ('Life', 'Health',  
'Vehicle');
```

-- 8. Display list of customers born after January 1st, 2001 and before December 31st, 2020 using >= and <= operators.

```
SELECT CustomerID, FirstName, LastName, DateOfBirth FROM Customers WHERE  
DateOfBirth>='2001-01-01' AND DateOfBirth<='2020-12-31';
```

-- 9. Display list of customers born after January 1st, 2001 and before December 31st, 2020 using BETWEEN operator.

```
SELECT CustomerID, FirstName, LastName, DateOfBirth FROM Customers WHERE  
DateOfBirth BETWEEN '2001-01-01' AND '2020-12-31';
```

-- 10. Display claims data where claim status is Rejected.

```
SELECT * FROM Claims WHERE ClaimStatus='Rejected';
```

-- 11. Display records of Agents who stay in a city whose second letter is 'a'.

```
SELECT * FROM Agents WHERE city LIKE '_a%';
```

```
-- 12. Display highest and lowest ClaimAmount from Claims table.
SELECT MIN(ClaimAmount) as Lowest_claim, MAX(ClaimAmount) AS Highest_claim FROM
Claims;
```

```
-- 13. Display latest claim record.
SELECT * FROM Claims ORDER BY ClaimDate DESC OFFSET 0 ROWS FETCH NEXT 1 ROWS
ONLY;
```

5. Update, Delete, Alter commands

```
-- Increase premium amount by 10% for all health insurance policies.
```

```
UPDATE Policies SET PremiumAmount=1.1*PremiumAmount
SELECT * FROM Policies;
```

```
-- Delete the record of PolicyAssignments whose EndDate is before today's date.
```

```
DELETE FROM Claims
WHERE AssignmentID IN (
    SELECT AssignmentID
    FROM PolicyAssignments
    WHERE EndDate < GETDATE()
);
DELETE FROM PolicyAssignments WHERE EndDate<GetDate();
SELECT * FROM PolicyAssignments;
```

```
-- Write a command to add Address and City Columns in the Customers table.
```

```
ALTER TABLE Customers
ADD Address VARCHAR(50), City VARCHAR(20)
SELECT * FROM Customers
```

```
-- Write a command to add a new column named DevOfId (DevelopmentOfficerId) in an
existing Agents table.
```

```
ALTER TABLE Agents
ADD DevOfId INT;
```

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

```
ALTER TABLE Agents
ADD CONSTRAINT fk_agent_devof
FOREIGN KEY (DevOfId) REFERENCES Agents(AgentID);
```

6. LIKE operator + Wildcards (%) and (_) pattern matching.

```
-- Display customers whose email ends with '@example.com'.
SELECT * FROM Customers
WHERE Email LIKE '%@example.com';

-- Retrieve customers whose email contains the text 'example' anywhere.
SELECT * FROM Customers
WHERE Email LIKE '%example%';

-- Find customers whose first name is exactly 4 characters long.
SELECT * FROM Customers
WHERE FirstName LIKE '____';

-- Show customers whose first name starts with 'S' and ends with 'a'.
SELECT * FROM Customers
WHERE FirstName LIKE 'S#a';

-- Display customers whose last name starts with any 3 characters
-- followed by 'ma' (Example: Sharma, Verma, Kumar etc.)
SELECT * FROM Customers
WHERE LastName LIKE '___ma';
```

7. ORDER BY, OFFSET-FETCH, and TOP

```
-- Display the latest claim record using ORDER BY with TOP.
SELECT TOP 1 * FROM Claims
ORDER BY ClaimDate DESC;

-- Display the latest claim record using OFFSET - FETCH (like you did).
SELECT * FROM Claims
ORDER BY ClaimDate DESC
OFFSET 0 ROWS
FETCH NEXT 1 ROWS ONLY;

-- Display the latest policy assignment (most recent StartDate).
SELECT TOP 1 * FROM PolicyAssignments
ORDER BY StartDate DESC;

-- Display the latest claim raised by each policy assignment
-- (Latest claim per AssignmentID using TOP WITH TIES).
SELECT TOP 1 WITH TIES * FROM Claims
ORDER BY ROW_NUMBER() OVER(PARTITION BY AssignmentID ORDER BY ClaimDate DESC);

-- Display the latest approved claim.
SELECT TOP 1 * FROM Claims
WHERE ClaimStatus = 'Approved'
ORDER BY ClaimDate DESC;
```

8. STRING FUNCTIONS in SQL Server

```
-- Display the latest approved claim.
SELECT TOP 1 * FROM Claims
WHERE ClaimStatus = 'Approved'
ORDER BY ClaimDate DESC;

-- Display Customer Full Name in UPPERCASE.
SELECT CustomerID, UPPER(FirstName + ' ' + LastName) AS FullNameUpper FROM
Customers;

-- Display Customer First Name in lowercase.
SELECT CustomerID, LOWER(FirstName) AS FirstNameLower FROM Customers;

-- Display first 3 characters of each customer's first name using SUBSTRING.
SELECT CustomerID, SUBSTRING(FirstName, 1, 3) AS First3Letters FROM Customers;

-- Show length of each customer's email using LEN().
SELECT CustomerID, Email, LEN(Email) AS EmailLength FROM Customers;

-- Replace domain 'example.com' with 'insurance.com' in email using REPLACE()
SELECT CustomerID, Email AS OldEmail, REPLACE(Email, 'example.com',
'insurance.com') AS UpdatedEmail
FROM Customers;
```

9. Queries using Joins, Group By, Having.

```
-- Display today's date using GETDATE().
SELECT GETDATE() AS TodayDate;

-- Display year, month and day of each claim date.
SELECT ClaimID, ClaimDate, YEAR(ClaimDate) AS ClaimYear, MONTH(ClaimDate) AS
ClaimMonth,
DAY(ClaimDate) AS ClaimDay
FROM Claims;

-- Display policy assignments whose EndDate has already expired
-- (EndDate earlier than today).
SELECT AssignmentID, CustomerID, PolicyID, EndDate FROM PolicyAssignments
WHERE EndDate < CAST(GETDATE() AS DATE);

-- Display number of days between StartDate and EndDate of each policy
-- using DATEDIFF().
SELECT AssignmentID, StartDate, EndDate, DATEDIFF(DAY, StartDate, EndDate) AS
PolicyDurationDays
FROM PolicyAssignments;

-- Display each claim date and its month-end date using EOMONTH().
SELECT ClaimID, ClaimDate, EOMONTH(ClaimDate) AS MonthEndDate FROM Claims;
```

10. Queries using Joins, Group By, Having.

-- 1. List all Policies for a CustomerId 4.

```
SELECT a.CustomerID,c.PolicyName
FROM Customers a JOIN PolicyAssignments b
ON a.CustomerID=b.CustomerID
JOIN Policies c
ON c.PolicyID=B.PolicyID
WHERE a.customerid=4;
```

-- 2. View all customers with their policies.

```
SELECT a.CustomerID,c.PolicyName
FROM Customers a JOIN PolicyAssignments b
ON a.CustomerID=b.CustomerID
JOIN Policies c
ON c.PolicyID=B.PolicyID;
```

-- 3. View claims with customer name.

```
SELECT a.FirstName,a.LastName,c.ClaimID, c.ClaimDate, c.claimamount,c.claimstatus
FROM Customers a JOIN PolicyAssignments b
ON a.CustomerID=b.CustomerID
JOIN Claims c
ON b.AssignmentID=c.AssignmentID;
```

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

```
SELECT a.FirstName,c.PolicyName,d.AgentName,b.StartDate,b.EndDate
FROM Customers a JOIN PolicyAssignments b
ON a.CustomerID=b.CustomerID
JOIN Policies c
ON c.PolicyID=B.PolicyID
JOIN Agents d
ON d.AgentID=b.AgentID;
```

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

```
SELECT a.FirstName,c.PolicyName,d.ClaimAmount,d.ClaimStatus,d.ClaimDate
FROM Customers a JOIN PolicyAssignments b
ON a.CustomerID=b.CustomerID
JOIN Policies c
ON c.PolicyID=B.PolicyID
JOIN Claims d
ON b.AssignmentID=d.AssignmentID;
```

-- 6. Display records of Customers with or without Policies.

```
SELECT a.CustomerID,a.FirstName,a.LastName,b.AssignmentID,b.PolicyID
FROM Customers a LEFT JOIN PolicyAssignments b
ON a.CustomerID=b.CustomerID;
```

```

-- 7. Display all Customers with NO Claims.
SELECT a.CustomerID,a.FirstName,a.LastName
FROM Customers a LEFT JOIN PolicyAssignments b
ON a.CustomerID=b.CustomerID
WHERE b.AssignmentID IS NULL;

-- 8. Show CustomerName with Total Claim Amount per Customer.
SELECT a.customerid,a.FirstName,a.lastname,SUM(c.claimamount)
FROM Customers a JOIN PolicyAssignments b
ON a.CustomerID=b.CustomerID
JOIN Claims c
ON b.AssignmentID=c.AssignmentID
GROUP BY a.customerid,a.FirstName,a.lastname;

-- 9. Show names and total claim amount of Customers With Claim Amount > 50000
(Use HAVING Clause).
SELECT a.customerid,a.FirstName,a.lastname,SUM(c.claimamount)
FROM Customers a JOIN PolicyAssignments b
ON a.CustomerID=b.CustomerID
JOIN Claims c
ON b.AssignmentID=c.AssignmentID
GROUP BY a.customerid,a.FirstName,a.lastname
HAVING SUM(c.claimamount)>10000;

-- 10. Display list with Agent Wise Policy Count.
SELECT a.agentid,COUNT(*)
FROM agents a join policyassignments b
on a.agentid=b.agentid
GROUP BY a.agentid;

```

11. SUBQUERIES using EXISTS, ANY, ALL

```

-- 1. List customers who have at least one claim (EXISTS)
SELECT c.CustomerID, c.FirstName, c.LastName FROM Customers c
WHERE EXISTS (
    SELECT 1
    FROM PolicyAssignments pa
    JOIN Claims cl ON pa.AssignmentID = cl.AssignmentID
    WHERE pa.CustomerID = c.CustomerID
);

-- 2. List customers who do NOT have any payments (NOT EXISTS)
SELECT c.CustomerID, c.FirstName, c.LastName FROM Customers c
WHERE NOT EXISTS (
    SELECT 1
    FROM PolicyAssignments pa
    JOIN Payments p ON pa.AssignmentID = p.AssignmentID
    WHERE pa.CustomerID = c.CustomerID
);

```

```

-- 3. Display payments that are greater than ANY payment made in 2023 (ANY)
SELECT PaymentID, AssignmentID, PaymentAmount FROM Payments
WHERE PaymentAmount > ANY (
    SELECT PaymentAmount
    FROM Payments
    WHERE YEAR(PaymentDate) = 2023
);

-- 4. List policies whose PremiumAmount is greater than ALL payment amounts (ALL)
SELECT PolicyID, PolicyName, PremiumAmount FROM Policies
WHERE PremiumAmount > ALL (
    SELECT PaymentAmount
    FROM Payments
);

-- 5. Display customers who have more than one claim (Correlated Subquery)
SELECT c.CustomerID, c.FirstName, c.LastName FROM Customers c
WHERE (
    SELECT COUNT(*)
    FROM PolicyAssignments pa
    JOIN Claims cl ON pa.AssignmentID = cl.AssignmentID
    WHERE pa.CustomerID = c.CustomerID
) > 1;

```

12. CASE-WHEN-ELSE, MERGE, ROLLUP, CUBE, GROUPING SETS

```

-- Display claim status in readable format
-- (Approved → 'Successful', Pending → 'In Progress', Rejected → 'Failed')
SELECT ClaimID, ClaimStatus,
    CASE
        WHEN ClaimStatus = 'Approved' THEN 'Successful'
        WHEN ClaimStatus = 'Pending' THEN 'In Progress'
        ELSE 'Failed'
    END AS ClaimResult
FROM Claims;

-- Categorize claim amount as 'High', 'Medium', 'Low'
SELECT ClaimID, ClaimAmount,
    CASE
        WHEN ClaimAmount >= 20000 THEN 'High'
        WHEN ClaimAmount BETWEEN 10000 AND 19999 THEN 'Medium'
        ELSE 'Low'
    END AS ClaimCategory
FROM Claims;

-- Merge new agent data:
-- If Agent exists update city, else insert as new agent
MERGE Agents AS Target
USING (
    SELECT 204 AS AgentID, 'Harshadh' AS AgentName, '9000041111' AS Phone,
    'Kozhikode' AS City
) AS Source
ON Target.AgentID = Source.AgentID
WHEN MATCHED THEN

```

```

    UPDATE SET Target.City = Source.City
WHEN NOT MATCHED THEN
    INSERT (AgentID, AgentName, Phone, City)
    VALUES (Source.AgentID, Source.AgentName, Source.Phone, Source.City);

```

```

-- Display total claim amount per ClaimStatus
-- and overall total using ROLLUP
SELECT ClaimStatus, (ClaimAmount) AS TotalAmount
FROM Claims
GROUP BY ROLLUP (ClaimStatus);

```

```

-- Show total claim amount by Customer and ClaimStatus,
-- including subtotals per Customer, per Status, and Grand Total
SELECT PA.CustomerID, C.ClaimStatus, SUM(C.ClaimAmount) AS TotalAmount
FROM Claims C
JOIN PolicyAssignments PA
ON C.AssignmentID = PA.AssignmentID
GROUP BY CUBE (PA.CustomerID, C.ClaimStatus);

```

```

-- Show total claims by
-- 1 Customer
-- 2 Status
-- 3 Overall total
SELECT PA.CustomerID, C.ClaimStatus, SUM(C.ClaimAmount) AS TotalAmount
FROM Claims C
JOIN PolicyAssignments PA
ON C.AssignmentID = PA.AssignmentID
GROUP BY GROUPING SETS
(
    (PA.CustomerID, C.ClaimStatus),
    (PA.CustomerID),
    (C.ClaimStatus),
    ()
);

```

13. UNION, UNION ALL, INTERSECT, EXCEPT

```

-- Display list of all CustomerIDs and CustomerIDs
-- who have made claims (UNION removes duplicates)
SELECT CustomerID
FROM Customers
UNION
SELECT PA.CustomerID
FROM PolicyAssignments PA
JOIN Claims C
ON PA.AssignmentID = C.AssignmentID;

```

```
-- Display list of all CustomerIDs and
-- CustomerIDs who have made claims using UNION ALL
-- (UNION ALL keeps duplicates)
```

```
SELECT CustomerID
FROM Customers
UNION ALL
SELECT PA.CustomerID
FROM PolicyAssignments PA
JOIN Claims C
ON PA.AssignmentID = C.AssignmentID;
```

```
-- Display customers who have policies assigned
-- AND have made at least one claim
-- (INTERSECT → common results)
```

```
SELECT CustomerID
FROM PolicyAssignments
INTERSECT
SELECT PA.CustomerID
FROM Claims C
JOIN PolicyAssignments PA
ON C.AssignmentID = PA.AssignmentID;
```

```
-- Display customers who have policies
-- but NEVER made any claim
-- (EXCEPT → present in first, not in second)
```

```
SELECT CustomerID
FROM PolicyAssignments
EXCEPT
SELECT PA.CustomerID
FROM Claims C
JOIN PolicyAssignments PA
ON C.AssignmentID = PA.AssignmentID;
```

```
-- Display PolicyIDs which are created in Policies table
-- but NOT assigned to any customer
```

```
SELECT PolicyID
FROM Policies
EXCEPT
SELECT PolicyID
FROM PolicyAssignments;
```

```
-- Display AgentIDs who are assigned policies
-- but have NOT handled any claim
```

```
SELECT AgentID
FROM PolicyAssignments
EXCEPT
SELECT PA.AgentID
FROM Claims C
JOIN PolicyAssignments PA
ON C.AssignmentID = PA.AssignmentID;
```