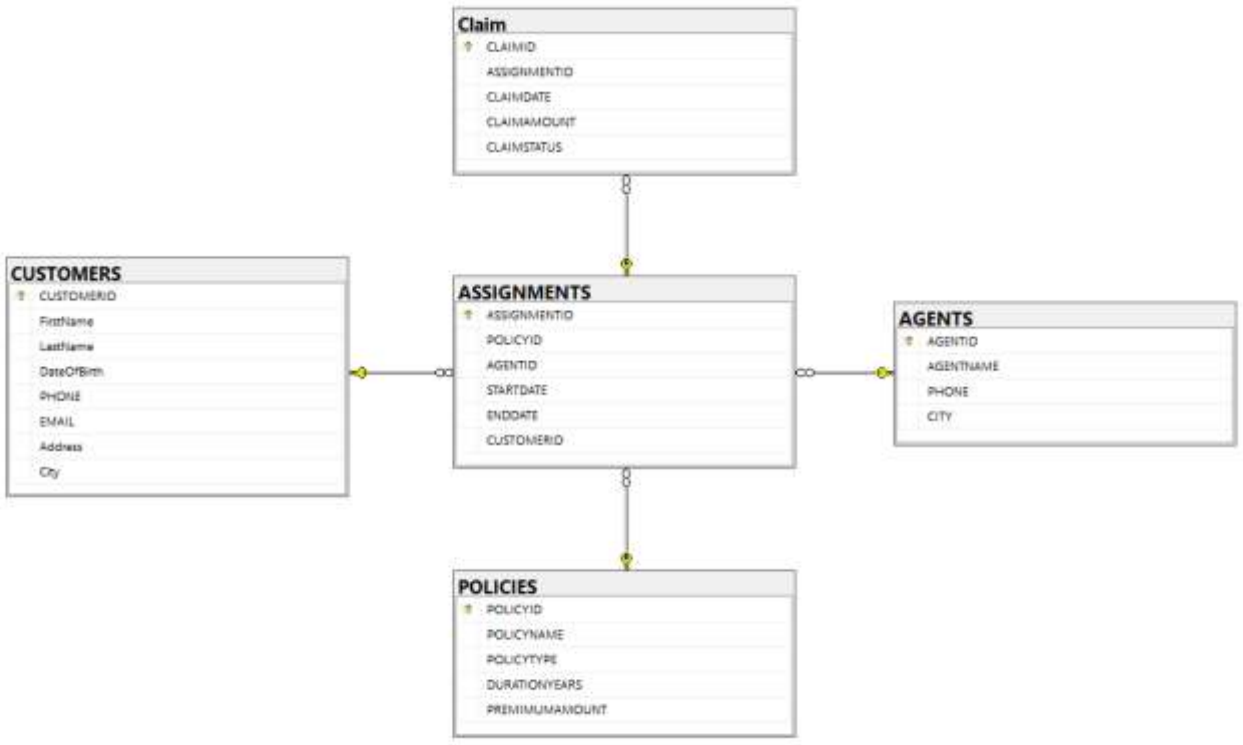
Module 4.4 Practical Project Assignment

1. **Create Database command.**

```
CREATE DATABASE InsuranceDB;
```

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

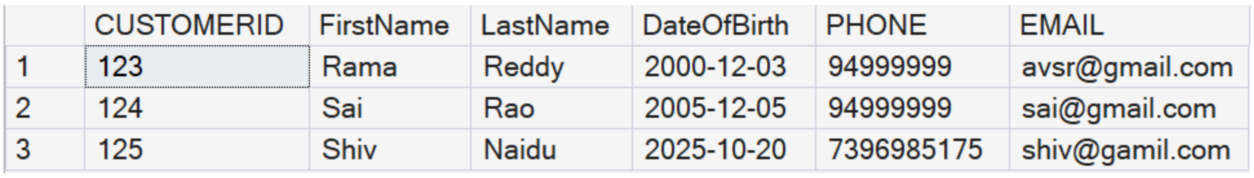
```
CREATE TABLE CUSTOMERS(

CUSTOMERID INT PRIMARY KEY,

FirstName VARCHAR(50),

LastName VARCHAR(50),

DateOfBirth Date,

PHONE INT,

EMAIL VARCHAR(50));


CREATE TABLE POLICIES(

POLICYID INT PRIMARY KEY,

POLICYNAME VARCHAR(50),

POLICYTYPE VARCHAR(50),

PREMIMUMACCOUNT VARCHAR(50),

DURATIONYEARS INT

);


CREATE TABLE AGENTS(

AGENTID INT PRIMARY KEY,

AGENTNAME VARCHAR(50),

PHONE TEXT,
CITY VARCHAR(20)

);



CREATE TABLE CLAIM(

CLAIMID INT PRIMARY KEY,

ASSIGNMENTID INT FOREIGN KEY REFERENCES POLICYASSIGNEMNTS(ASSIGNMENTID) ,

CLAIMDATE DATE,

CLAIMAMOUNT INT,

CLAIMSTATUS BIT

);



CREATE TABLE ASSIGNMENTS(

ASSIGNMENTID INT PRIMARY KEY,

POLICYID INT FOREIGN KEY REFERENCES POLICIES(POLICYID),

AGENTID INT FOREIGN KEY REFERENCES AGENTS(AGENTID),

CUSTOMERID INT FOREIGN KEY REFERENCES CUSTOMERS(CUSTOMERID),

STARTDATE DATE,

ENDDATE DATE

);
```



- SELECT COMMANDS

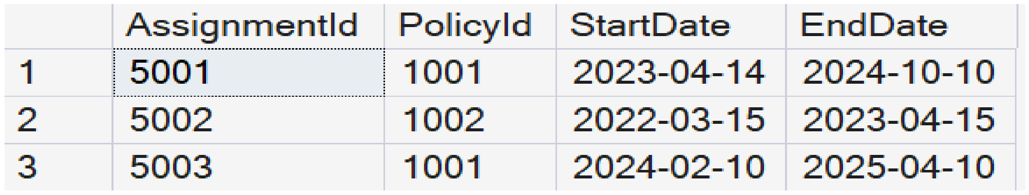
3. **View all records Customers table.**

   SELECT * FROM CUSTOMERS;



	CUSTOMERID	FirstName	LastName	DateOfBirth	PHONE	EMAIL
1	123	Rama	Reddy	2000-12-03	94999999	avsr@gmail.com
2	124	Sai	Rao	2005-12-05	94999999	sai@gmail.com
3	125	Shiv	Naidu	2025-10-20	7396985175	shiv@gamil.com

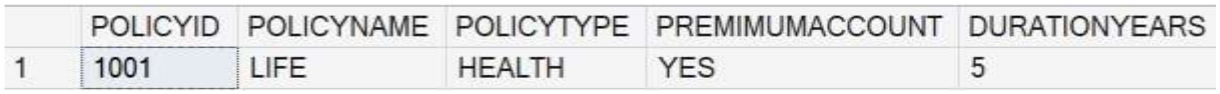
4. **View all records of PolicyAssignment table with AssignmentId, PolicyId, StartDate and EndDate columns only.**



	AssignmentId	PolicyId	StartDate	EndDate
1	5001	1001	2023-04-14	2024-10-10
2	5002	1002	2022-03-15	2023-04-15
3	5003	1001	2024-02-10	2025-04-10

5. **Display all policies of Health type.**

   SELECT * FROM policies where POLICYTYPE='HEALTH';



	POLICYID	POLICYNAME	POLICYTYPE	PREMIMUMACCOUNT	DURATIONYEARS
1	1001	LIFE	HEALTH	YES	5

6. **Display policies having premium amount more than 10000 and DurationYears is 1.**
   SELECT * FROM POLICIES WHERE PREMIUMAMOUNT>10000 AND DURATIONYEARS=1;
7. **List policies of type Life, Health, Motor use IN operator.**
   SELECT * FROM POLICIES WHERE POLICYTYPE IN ('LIFE','HEALTH','MOTOR');

STRING FUNCTIONS

1. Display customer full name by concatenating FirstName and LastName.
SELECT CONCAT(FirstName, ' ', LastName) AS FullName FROM CUSTOMERS;

2. Find customers whose FirstName starts with 'A'.
SELECT * FROM CUSTOMERS WHERE FirstName LIKE 'A%';

3. Display LastName in uppercase.
SELECT UPPER(LastName) FROM CUSTOMERS;

4. Find customers whose LastName length is more than 6 characters.
SELECT * FROM CUSTOMERS WHERE LEN(LastName) > 6;

5. Display the first 3 characters of FirstName.
SELECT SUBSTRING(FirstName,1,3) FROM CUSTOMERS;

NUMERIC FUNCTIONS

6. Display ClaimAmount rounded to nearest integer.
SELECT ROUND(CLAIMAMOUNT,0) FROM CLAIMS;

7. Display absolute value of ClaimAmount.

SELECT ABS(CLAIMAMOUNT) FROM CLAIMS;

8. Calculate 10% tax on ClaimAmount.
SELECT CLAIMAMOUNT*0.10 AS Tax FROM CLAIMS;

9. Display ceiling value of ClaimAmount.
SELECT CEILING(CLAIMAMOUNT) FROM CLAIMS;

10. Display floor value of ClaimAmount.
SELECT FLOOR(CLAIMAMOUNT) FROM CLAIMS;

DATE FUNCTIONS

11. Display current system date.
SELECT GETDATE();

12. Find claims filed in last 6 months.
SELECT * FROM CLAIMS WHERE CLAIMDATE >= DATEADD(MONTH,-6,GETDATE());

13. Display ClaimDate in DD-MM-YYYY format.
SELECT FORMAT(CLAIMDATE,'dd-MM-yyyy') FROM CLAIMS;

14. Find year of ClaimDate.
SELECT YEAR(CLAIMDATE) FROM CLAIMS;

15. Find month name from ClaimDate.
SELECT DATENAME(MONTH,CLAIMDATE) FROM CLAIMS;

AGGREGATE FUNCTIONS

16. Find total number of customers.
SELECT COUNT(*) FROM CUSTOMERS;

17. Find maximum ClaimAmount.
SELECT MAX(CLAIMAMOUNT) FROM CLAIMS;

18. Find minimum ClaimAmount.
SELECT MIN(CLAIMAMOUNT) FROM CLAIMS;

19. Find average ClaimAmount.
SELECT AVG(CLAIMAMOUNT) FROM CLAIMS;

20. Count number of claims.
SELECT COUNT(CLAIMID) FROM CLAIMS;

21. List all Policies for a CustomerId = 5
SELECT *
FROM POLICIES
WHERE CUSTOMERID = 5;

22. View all customers with their policies
SELECT c.CUSTOMERID, c.FirstName,c.LastName,p.POLICYID,p.POLICYNAME,p.POLICYTYPE
FROM CUSTOMERS c JOIN POLICIES p ON c.CUSTOMERID = p.CUSTOMERID;

23. View claims with customer name
SELECT c.FirstName,c.LastName,cl.CLAIMID,cl.CLAIMAMOUNT,cl.CLAIMDATE,cl.CLAIMSTATUS
FROM CUSTOMERS c JOIN POLICIES p ON c.CUSTOMERID = p.CUSTOMERIDJOIN CLAIMS cl ON
p.POLICYID = cl.POLICYID;

24. Display FirstName, PolicyName, AgentName, StartDate and EndDate
SELECT
  c.FirstName, p.POLICYNAME, a.AGENTNAME, p.STARTDATE, p.ENDDATE FROM CUSTOMERS c
JOIN POLICIES p ON c.CUSTOMERID = p.CUSTOMERID JOIN AGENTS ON p.AGENTID =
a.AGENTID;

25. Display claims report with FirstName, PolicyName, ClaimAmount, ClaimStatus, ClaimDate
SELECT c.FirstName, p.POLICYNAME, cl.CLAIMAMOUNT, cl.CLAIMSTATUS, cl.CLAIMDATE FROM
CUSTOMERS c JOIN POLICIES p ON c.CUSTOMERID = p.CUSTOMERID JOIN CLAIMS cl ON
p.POLICYID = cl.POLICYID;

SUBQUERIES

26. Find customers who have policies (using IN)

```
SELECT *
FROM CUSTOMERS
WHERE CUSTOMERID IN (
   SELECT CUSTOMERID
   FROM POLICIES
);
```

27. Find customers who do NOT have any policies (using NOT IN)

```
SELECT *
FROM CUSTOMERS
WHERE CUSTOMERID NOT IN (
   SELECT CUSTOMERID
   FROM POLICIES
);
```

28. Find policies that have at least one claim (using EXISTS)

```
SELECT *
FROM POLICIES p
WHERE EXISTS (
   SELECT 1
   FROM CLAIMS c
   WHERE c.POLICYID = p.POLICYID
);
```

29. Find policies that have no claims (using NOT EXISTS)

```
SELECT *
FROM POLICIES p
WHERE NOT EXISTS (
   SELECT 1
   FROM CLAIMS c
   WHERE c.POLICYID = p.POLICYID
);
```

30. Find claims whose amount is greater than ANY claim amount of policy ID = 1

```
SELECT *
FROM CLAIMS
WHERE CLAIMAMOUNT > ANY (
   SELECT CLAIMAMOUNT
   FROM CLAIMS
   WHERE POLICYID = 1
);
```

31. Find claims whose amount is greater than ALL claim amounts of policy ID = 1

```
SELECT *
FROM CLAIMS
WHERE CLAIMAMOUNT > ALL (
   SELECT CLAIMAMOUNT
   FROM CLAIMS
   WHERE POLICYID = 1
);
```

32. Find policies whose premium is greater than ANY other policy premium

```
SELECT *
FROM POLICIES
WHERE PREMIMUMACCOUNT > ANY (
   SELECT PREMIMUMACCOUNT
   FROM POLICIES
);
```

33. Find policies whose premium is greater than ALL other policy premiums

```
SELECT *
FROM POLICIES
WHERE PREMIMUMACCOUNT > ALL (
   SELECT PREMIMUMACCOUNT
   FROM POLICIES
);
```

CASE STATEMENT

34. Display claim status as text (Approved / Rejected)

```
SELECT
    CLAIMID,
    CLAIMAMOUNT,
    CASE
        WHEN CLAIMSTATUS = 1 THEN 'Approved'
        ELSE 'Rejected'
    END AS ClaimStatusText
FROM CLAIMS;
```

35. Categorize policies based on duration

```
SELECT
    POLICYID,
    POLICYNAME,
    CASE
        WHEN DURATIONYEARS >= 10 THEN 'Long Term'
        WHEN DURATIONYEARS BETWEEN 5 AND 9 THEN 'Medium Term'
        ELSE 'Short Term'
    END AS PolicyCategory
FROM POLICIES;
```

MERGE STATEMENT

36. Merge data from TEMP_CUSTOMERS into CUSTOMERS

```
MERGE INTO CUSTOMERS AS target
USING TEMP_CUSTOMERS AS source
ON target.CUSTOMERID = source.CUSTOMERID
WHEN MATCHED THEN
```

```
    UPDATE SET
        target.FirstName = source.FirstName,
        target.LastName = source.LastName
WHEN NOT MATCHED THEN
    INSERT (CUSTOMERID, FirstName, LastName, DateOfBirth, PHONE, EMAIL)
    VALUES (source.CUSTOMERID, source.FirstName, source.LastName, source.DateOfBirth,
source.PHONE, source.EMAIL);
```

37. Merge policy updates from TEMP_POLICIES

```
MERGE INTO POLICIES AS target
USING TEMP_POLICIES AS source
ON target.POLICYID = source.POLICYID
WHEN MATCHED THEN
    UPDATE SET
        target.POLICYNAME = source.POLICYNAME,
        target.DURATIONYEARS = source.DURATIONYEARS
WHEN NOT MATCHED THEN
    INSERT (POLICYID, POLICYNAME, POLICYTYPE, PREMIMUMACCOUNT, DURATIONYEARS)
    VALUES (source.POLICYID, source.POLICYNAME, source.POLICYTYPE,
source.PREMIMUMACCOUNT, source.DURATIONYEARS);
```

QUERIES USING ROLLUP

38. Total claim amount per year with grand total

```
SELECT
    YEAR(CLAIMDATE) AS ClaimYear,
    SUM(CLAIMAMOUNT) AS TotalAmount
FROM CLAIMS
GROUP BY ROLLUP (YEAR(CLAIMDATE));
```

39. Total claim amount by status with grand total

```
SELECT
   CLAIMSTATUS,
   SUM(CLAIMAMOUNT) AS TotalAmount
FROM CLAIMS
GROUP BY ROLLUP (CLAIMSTATUS);
```

QUERIES USING CUBE

40. Total claim amount by year and status (all combinations)

```
SELECT
   YEAR(CLAIMDATE) AS ClaimYear,
   CLAIMSTATUS,
   SUM(CLAIMAMOUNT) AS TotalAmount
FROM CLAIMS
GROUP BY CUBE (YEAR(CLAIMDATE), CLAIMSTATUS);
```

QUERIES USING SET OPERATORS

41. UNION

```
SELECT CUSTOMERID AS ID FROM CUSTOMERS
UNION
SELECT AGENTID FROM AGENTS;
```

42. UNION ALL

```
SELECT CUSTOMERID AS ID FROM CUSTOMERS
UNION ALL
SELECT AGENTID FROM AGENTS;
```

43. INTERSECT

```
SELECT CUSTOMERID AS ID FROM CUSTOMERS
INTERSECT
SELECT AGENTID FROM AGENTS;
```

44. EXCEPT / MINUS

```
SELECT CUSTOMERID AS ID FROM CUSTOMERS
EXCEPT
SELECT AGENTID FROM AGENTS;
```

45. INTERSECT (Policies having Claims)

```
SELECT POLICYID FROM POLICIES
INTERSECT
SELECT POLICYID FROM CLAIMS;
```

46. Total ClaimAmount by PolicyID and Grand Total

```
SELECT POLICYID, SUM(CLAIMAMOUNT) AS TotalClaimAmount FROM CLAIMS GROUP BY
GROUPING SETS ((POLICYID), ());
```

47. Number of Customers by City and Overall Total

```
SELECT CITY, COUNT(*) AS TotalCustomers FROM CUSTOMERS GROUP BY GROUPING SETS
((CITY), ());
```