Module 4.4 Practical Project Assignment

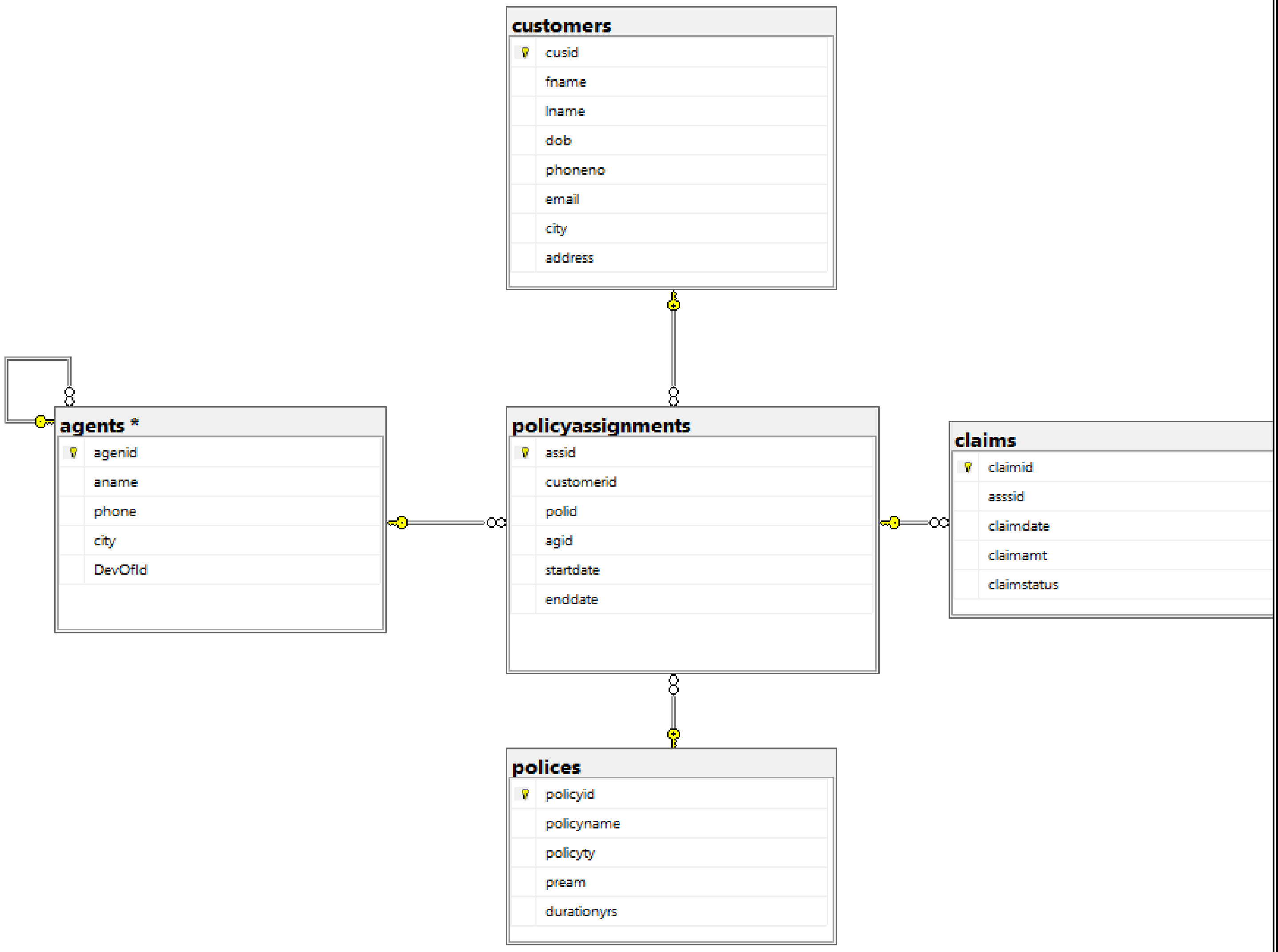
Create the database insurance

- create database insdb;

Use the database

- use insdb;

Database Diagram



Creating Tables

```
-- 1. Customers table

CREATE TABLE customers (

    cusid INT CONSTRAINT c_id PRIMARY KEY,

    fname NCHAR(10),

    lname NCHAR(10),

    dob DATE,

    phoneno BIGINT,

    email VARCHAR(20)

);


-- 2. Agents table

CREATE TABLE agents (

    agenid INT PRIMARY KEY,

    aname NCHAR(10),

    phone BIGINT,

    city NCHAR(10)

);


-- 3. Policies table

CREATE TABLE polices (

    policyid INT CONSTRAINT p_id PRIMARY KEY,

    policyname NCHAR(10),

    policyty NCHAR(10),

    pream INT,

    durationyrs INT

);
```

```sql
-- 4. Policy Assignments table

CREATE TABLE policyassignments (

    assid INT PRIMARY KEY,

    customerid INT,

    polid INT,

    agid INT,

    startdate DATE,

    enddate DATE,

    CONSTRAINT fk_cust FOREIGN KEY (customerid) REFERENCES customers(cusid),

    CONSTRAINT fk_p FOREIGN KEY (polid) REFERENCES polices(policyid),

    CONSTRAINT fk_agent FOREIGN KEY (agid) REFERENCES agents(agenid)

);


-- 5. Claims table

CREATE TABLE claims (

    claimid INT PRIMARY KEY,

    assid INT,

    claimdate DATE,

    claimamt INT,

    claimstatus NCHAR(10),

    CONSTRAINT fk_ass FOREIGN KEY (assid) REFERENCES policyassignments(assid)

);
```

Inserting values into the tables

```sql
INSERT INTO customers VALUES

(1, 'Ravi',   'Kumar',  '1998-05-10', 9876543210, 'ravi@gmail.com'),

(2, 'Anita',  'Sharma', '1999-08-15', 9876543221, 'anita@gmail.com'),

(3, 'Rahul',  'Verma',  '1997-03-20', 9876543232, 'rahul@gmail.com'),
(4, 'Sneha',  'Reddy',  '2000-11-02', 9876543243, 'sneha@gmail.com'),
(5, 'Arjun',  'Mehta',  '1996-01-25', 9876543254, 'arjun@gmail.com');
INSERT INTO agents VALUES
(101, 'Suresh', 9876500011, 'Chennai'),
(102, 'Mahesh', 9876500022, 'Hyderabad'),
(103, 'Kiran',  9876500033, 'Bangalore'),
(104, 'Neha',   9876500044, 'Pune'),
(105, 'Amit',   9876500055, 'Mumbai');
INSERT INTO polices VALUES
(201, 'LifePlus', 'Life',   12000, 20),
(202, 'HealthMax','Health', 15000, 10),
(203, 'CarSafe',  'Vehicle',8000,  5),
(204, 'HomeCare', 'Home',   10000, 15),
(205, 'TravelGo', 'Travel', 5000,  2);
INSERT INTO policyassignments VALUES
(301, 1, 201, 101, '2023-01-01', '2043-01-01'),
(302, 2, 202, 102, '2022-06-15', '2032-06-15'),
(303, 3, 203, 103, '2021-09-10', '2026-09-10'),
(304, 4, 204, 104, '2020-03-05', '2035-03-05'),
(305, 5, 205, 105, '2024-02-01', '2026-02-01');
INSERT INTO claims VALUES
(401, 301, '2024-01-10', 50000, 'Approved'),
(402, 302, '2023-11-05', 30000, 'Pending'),
(403, 303, '2022-08-20', 15000, 'Rejected'),
(404, 304, '2024-03-18', 45000, 'Approved'),
(405, 305, '2024-06-25', 20000, 'Pending');
```

❖ **Queries**

Display

select * from customers;

select * from polices

select * from claims

select * from policyassignments

select * from agents

Queries

Select Queries

1. SELECT * FROM polices WHERE policyty = 'Health' AND pream > 20000;
2. SELECT * FROM agents WHERE city = 'Mumbai' OR city = 'Pune';
3. SELECT * FROM claims WHERE claimstatus <> 'Approved';
4. SELECT * FROM policyassignments WHERE enddate IS NULL;
5. SELECT * FROM claims WHERE claimstatus IS NOT NULL;
6. SELECT * FROM customers WHERE fname LIKE 'A_';
7. SELECT * FROM polices WHERE policyname LIKE '%Care%';
8. SELECT TOP 3 * FROM polices ORDER BY pream DESC;
9. SELECT * FROM polices ORDER BY pream DESC LIMIT 3;
10. SELECT * FROM polices WHERE policyty NOT IN ('Travel', 'Home');
11. SELECT * FROM claims WHERE claimamt BETWEEN 20000 AND 50000;

Update , Alter Commands

1.Update phone number of customer with cusid 101

UPDATE customers SET phoneno = 9998887770 WHERE cusid = 101;

2.Update claim status to 'Approved' for claimid 402

UPDATE claims SET claimstatus = 'Approved' WHERE claimid = 402;

3.Add a new column 'city' to customers table

ALTER TABLE customers ADD city NVARCHAR(20);

4.Change datatype of 'pream' column in polices table to BIGINT

ALTER TABLE polices ALTER COLUMN pream BIGINT;

Date

1.Claims made in the current month

```
SELECT * FROM claims WHERE MONTH(claimdate) = MONTH(GETDATE());
```

2.Customers born before 1-Jan-1999

```
SELECT * FROM customers WHERE dob < '1999-01-01';
```

3.Policy assignments lasting more than 10 years

```
SELECT * FROM policyassignments WHERE DATEDIFF(YEAR, startdate, enddate) > 10;
```

4.Claims made in the last 365 days

```
SELECT * FROM claims WHERE claimdate >= DATEADD(DAY, -365, GETDATE());
```

Aggregate Functions

1.Count total customers-

```
SELECT COUNT(*) AS TotalCustomers FROM customers;
```

2.Sum of all claim amounts-

```
SELECT SUM(claimamt) AS TotalClaims FROM claims;
```

3.Average premium of policies-

```
SELECT AVG(pream) AS AvgPremium FROM polices;
```

4.Minimum premium-

```
SELECT MIN(pream) AS MinPremium FROM polices;
```

5.Maximum premium-

```
SELECT MAX(pream) AS MaxPremium FROM polices;
```

String Functions

1.Uppercase first name of customers-

```
SELECT UPPER(fname) AS UpperName FROM customers;
```

2. Lowercase last name-

```
SELECT LOWER(lname) AS LowerName FROM customers;
```

3.Length of email-

```
SELECT LEN(email) AS EmailLength FROM customers;
```

4.Concatenate first and last name-

SELECT fname + ' ' + lname AS FullName FROM customers;

5.Substring of first 3 letters of policy name-

SELECT SUBSTRING(policyname, 1, 3) AS PolicyShort FROM polices;

Numeric Functions

1. Round premium to nearest 1000-

SELECT ROUND(pream, -3) AS RoundedPremium FROM polices;

2.Ceiling of premium-

SELECT CEILING(pream/1000.0) AS PremiumCeil FROM polices;

3.Floor of premium-

SELECT FLOOR(pream/1000.0) AS PremiumFloor FROM polices;

4.Absolute difference between two premiums-

SELECT ABS(15000 - 12000) AS PremiumDiff;

JOINS

1.Left join-List all customers with their policy types

```
select c.fname, p.policyty
from customers c
left join policyassignments pa on c.cusid = pa.customerid
left join polices p on pa.polid = p.policyid;
```

2.Inner Join-Total claim amount per customer (only customers with claims > 50000)

```
select c.fname, sum(cl.claimamt) as totalclaimamount
from customers c
join policyassignments pa on c.cusid = pa.customerid
join claims cl on pa.assid = cl.asssid
group by c.fname
having sum(cl.claimamt) > 50000
```

3.Right Join-List all policy assignments with policy info

```
SELECT p.policyid, p.policyname, pa.customerid

FROM polices p

RIGHT JOIN policyassignments pa ON p.policyid = pa.polid;
```

4.OUTER JOIN-Customers and their policy assignments

```
SELECT c.cusid, c.fname, pa.polid

FROM customers c

FULL OUTER JOIN policyassignments pa ON c.cusid = pa.customerid;
```

Corelated and Nested Subquery

1.Policies with start date later than ALL policies of CustomerID = 102

```
SELECT *  FROM policyassignments WHERE startdate > ALL (SELECT startdate FROM
policyassignments WHERE customerid = 102);
```

2.Customers whose assigned policy is the most expensive

```
SELECT fname, lname FROM customers WHERE cusid IN ( SELECT customerid

FROM policyassignments WHERE polid = (SELECT policyid FROM polices ORDER BY
pream DESC LIMIT 1)

);
```

MERGE, ROLLUP,CUBE,GROUPING

1.MERGE: Update or Insert claim

```
MERGE INTO claims AS target

USING (SELECT 305 AS assid, '2025-12-31' AS claimdate, 25000 AS claimamt,
'Pending' AS claimstatus) AS source

ON target.assid = source.assid

WHEN MATCHED THEN UPDATE SET claimstatus = source.claimstatus, claimdate =
source.claimdate, claimamt = source.claimamt

WHEN NOT MATCHED THEN INSERT (assid, claimdate, claimamt, claimstatus) VALUES
(source.assid, source.claimdate, source.claimamt, source.claimstatus);
```

2. ROLLUP: Total claim amount by customer and overall

```
SELECT c.fname, c.lname, SUM(cl.claimamt) AS TotalClaim
FROM customers c
JOIN policyassignments pa ON c.cusid = pa.customerid
JOIN claims cl ON pa.assid = cl.assid
GROUP BY ROLLUP(c.fname, c.lname);
```

3. CUBE: Total claims by customer and policy type

```
SELECT c.fname, p.policyty, SUM(cl.claimamt) AS TotalClaim
FROM customers c
JOIN policyassignments pa ON c.cusid = pa.customerid
JOIN polices p ON pa.polid = p.policyid
 JOIN claims cl ON pa.assid = cl.assid
GROUP BY CUBE(c.fname, p.policyty);
```

4. CASE…ELSE: Categorize claims based on amount

```
SELECT cl.claimid, cl.claimamt,
    CASE
        WHEN claimamt > 40000 THEN 'High'
        WHEN claimamt BETWEEN 20000 AND 40000 THEN 'Medium'
        ELSE 'Low'
    END AS ClaimCategory
FROM claims cl;
```

5.ROLLUP with GROUPING – Total claims by Customer and Policy

```
SELECT
  c.fname AS CustomerName,
  p.policyty AS PolicyType,
  SUM(cl.claimamt) AS TotalClaim,
  GROUPING(c.fname) AS IsCustomerTotal,
  GROUPING(p.policyty) AS IsPolicyTotal
```

```
FROM customers c
JOIN policyassignments pa ON c.cusid = pa.customerid
JOIN polices p ON pa.polid = p.policyid
JOIN claims cl ON pa.assid = cl.assid
GROUP BY ROLLUP(c.fname, p.policyty);
```

6.CUBE with GROUPING – Total claims by Customer and Policy

```
SELECT
   c.fname AS CustomerName,
   p.policyty AS PolicyType,
   SUM(cl.claimamt) AS TotalClaim,
   GROUPING(c.fname) AS IsCustomerTotal,
   GROUPING(p.policyty) AS IsPolicyTotal
FROM customers c
JOIN policyassignments pa ON c.cusid = pa.customerid
JOIN polices p ON pa.polid = p.policyid
JOIN claims cl ON pa.assid = cl.assid
GROUP BY CUBE(c.fname, p.policyty);
```

SET OPERATORS

1. **UNION: Combine two queries and remove duplicates**

```
SELECT cusid, fname FROM customers WHERE cusid < 103
UNION
SELECT cusid, fname FROM customers WHERE cusid > 101;
```

2. **UNION ALL: Combine two queries and keep duplicates**

```
SELECT cusid, fname FROM customers WHERE cusid < 103
UNION ALL
SELECT cusid, fname FROM customers WHERE cusid > 101;
```

3. **INTERSECT: Return only rows common to both queries**

```
SELECT cusid, fname FROM customers WHERE cusid < 104
INTERSECT
SELECT cusid, fname FROM customers WHERE cusid > 101;
```

4. **EXCEPT: Return rows in first query not in second**

```
SELECT cusid, fname FROM customers WHERE cusid < 104
EXCEPT
SELECT cusid, fname FROM customers WHERE cusid = 102;
```

5. **MINUS: Return rows in first query not in second (Oracle)**

```
SELECT cusid, fname FROM customers WHERE cusid < 104
MINUS
SELECT cusid, fname FROM customers WHERE cusid = 102;
```

INDEXING

1: Clustered index on primary key

```
CREATE CLUSTERED INDEX idx_customers_cusid
ON customers(cusid);
--Query using clustered index
SELECT * FROM customers
WHERE cusid = 101;
```

2: Non-clustered index on frequently searched column

```
CREATE NONCLUSTERED INDEX idx_claims_claimdate
ON claims(claimdate);
-- Query using non-clustered index
SELECT * FROM claims
WHERE claimdate >= '2024-01-01';
```

VIEWS

1.Customer Policy Details

CREATE VIEW vw_customer_policies AS SELECT c.cusid, c.fname, c.lname, p.policyname, pa.startdate, pa.enddate FROM customers c JOIN policyassignments pa ON c.cusid = pa.cusid JOIN policies p ON pa.policyid = p.policyid;

2. Claim Details with Customer Info

CREATE VIEW vw_claim_details AS SELECT cl.claimid, c.fname, c.lname, cl.claimamount, cl.claimdate, cl.claimstatus FROM claims cl JOIN policyassignments pa ON cl.assid = pa.assid JOIN customers c ON pa.cusid = c.cusid;