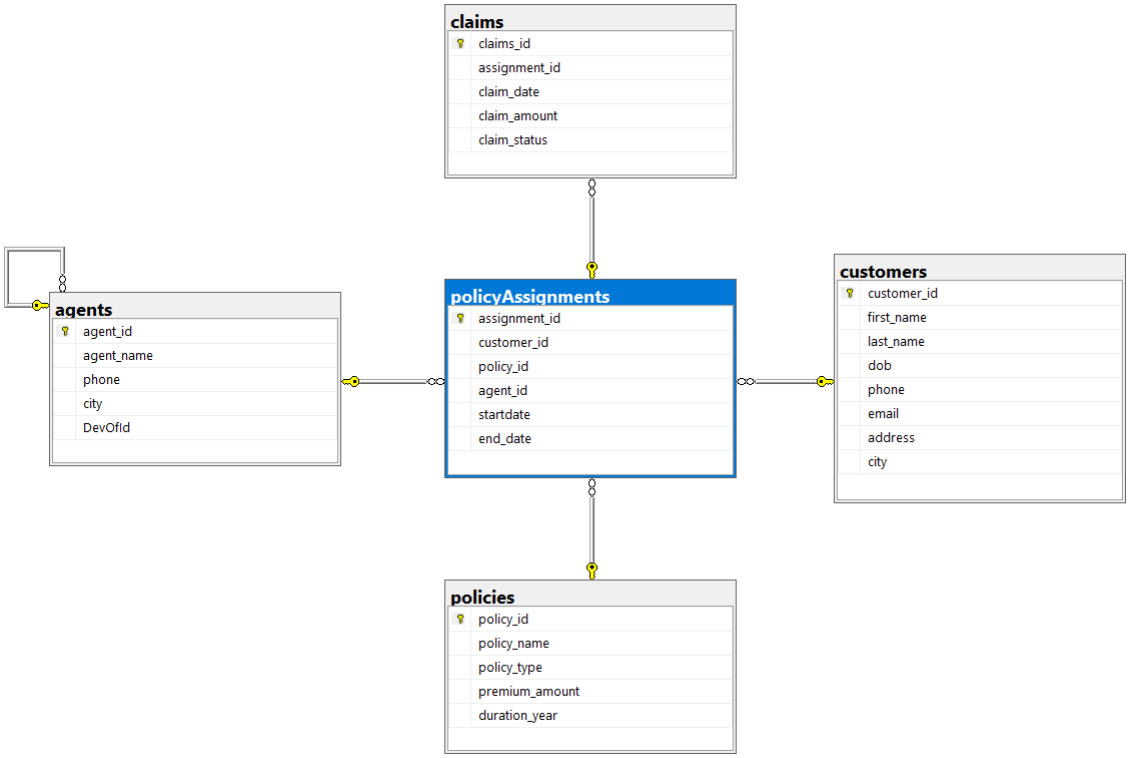
MODULE 4.4 PRACTICAL PROJECT ASSIGNMENT

```
Create database InsuranceDB;
use InsuranceDB;
```

SCHEMA



CREATE TABLE COMMANDS

```
create table customers(
customer_id int identity(1,1) primary key,
first_name varchar(50),
last_name varchar(50),
dob date,
phone varchar(10),
email varchar(50) unique
);

create table policies(
policy_id int identity(1,1) primary key,
policy_name varchar(50),
policy_type varchar(50),
premium_amount decimal(10,2),
duration_year int
);


create table agents(
agent_id int identity primary key,
agent_name varchar(50),
phone varchar(10),
city varchar(100)
);
```

```sql
create table policyAssignments(
assignment_id int identity(1,1) primary key,
customer_id int,
policy_id int,
agent_id int,
startdate date,
end_date date
constraint custumer_fk foreign key(customer_id) references
customers(customer_id),
constraint policy_fk foreign key(policy_id) references policies(policy_id),
constraint agent_fk foreign key(agent_id) references agents(agent_id)
);


create table claims(
claims_id int  identity(1,1) primary key,
assignment_id int,
claim_date date,
claim_amount decimal(10,2),
claim_status varchar(50),
constraint assign_fk foreign key(assignment_id) references
policyAssignments(assignment_id)
);
```

INSERTION

```sql
--Insertion of customers data
insert into customers values('Sneha', 'Reddy', '1998-08-15', '9123456789',
'sneha@gmail.com');
insert into customers values('Ajay','Kumar',
'1996-04-24','9857458745','ajay07@gmail.com');
insert into customers (first_name, last_name, dob, phone, email)
values('Rahul', 'Verma', '1997-11-12', '9012345678', 'rahul.verma@gmail.com'),
('Priya', 'Sharma', '1999-02-05', '9098765432', 'priya.sharma@gmail.com'),
('Amit', 'Singh', '1995-06-18', '9887766554', 'amit.singh@gmail.com');


--Insertion of policies data
INSERT INTO policies (policy_name, policy_type, premium_amount, duration_year)
VALUES ('Health Secure', 'Health', 15000.00, 5),
('Life Shield', 'Life', 25000.00, 10),
('Family Health Plus', 'Health', 18000.00, 7),
('Term Life Pro', 'Life', 30000.00, 15),
('Vehicle Secure', 'Motor', 12000.00, 3);


-- Insertion of agents data
INSERT INTO agents (agent_name, phone, city)
VALUES ('Ramesh Rao', '9876543210', 'Hyderabad'),
('Anita Sharma', '9988776655', 'Bangalore'),
('Suresh Patel', '9011223344', 'Ahmedabad'),
('Kavya Nair', '9090909090', 'Kochi'),
('Vikram Singh', '9888899999', 'Delhi');
```

```sql
--Insertion of policyAssignments data
INSERT INTO policyAssignments (customer_id, policy_id, agent_id,
startdate,end_date)
VALUES(1, 6, 6, '2023-01-01', '2028-01-01'),
(2, 7, 7, '2022-06-15', '2032-06-15'),
(3, 8, 8, '2023-03-01', '2028-03-01'),
(4, 9, 9, '2023-07-10', '2028-07-10'),
(5,10,10,'2024-01-15', '2029-01-15');



SELECT assignment_id FROM policyAssignments;

--Insertion of claims data
INSERT INTO claims (assignment_id, claim_date, claim_amount, claim_status)
VALUES(19, '2024-02-10', 50000.00, 'Approved'),
(20, '2024-05-20', 30000.00, 'Pending'),
(21, '2024-03-15', 20000.00, 'Approved'),
(22, '2024-04-18', 12000.00, 'Pending'),
(23, '2024-06-05', 18000.00, 'Rejected');
```

BASIC SELECT QUERIES

```sql
select * from customers;

select * from policies;

select * from policyAssignments;

select * from claims;

select * from agents;

--Filtering using where
select * from claims
where claim_date='2024-02-10';

select * from claims
where assignment_id=21;

--Display Policies with Premium Amount Greater than 10000 and Duration More than
1 Year
select * from policies
where premium_amount>10000 and duration_year>1;

--Display Unique Cities Where Agents Are Located
select distinct city from agents;

--Display Health, Life, or Motor Policies (Using OR Operator)
select * from policies where policy_type='Health' or policy_type='life' or
policy_type='motor';

--Display Health, Life, or Motor Policies (Using IN Operator)
select * from policies where policy_type in ('Health','life','motor');


--Display Customers Born Between 2001 and 2020 (Using >= and <= Operators)
select * from customers where dob>='2001-01-01' and dob<='2020-12-31';


--Patterns
```

```sql
select * from customers
where first_name like '_a%';

--Trimming

select RTrim(agent_name) + '(' +city+ ')'
from agents order by agent_name;



                              --Aggregation functions
select count(*) as total_customers
from customers;


select sum(claim_amount) as total_claim_amount
from claims;

select avg(premium_amount) as avg_premium
from policies;

select max(claim_amount) as highest_claim
from claims;

select min(premium_amount) as lowest_premium
from policies;
```

Queries on DATE & TIME functions

```sql
select claim_amount,datepart(WEEKDAY,claim_date) from claims;

--Getdate() Function To Get Current Date And Time
select getdate() as today;

--CAST() to convert into specific type like time,integer etc
--Extracts only the date (removes time part).
select cast(getdate() as date) as today;

--Extract year from DOB
select first_name ,year(dob) from customers;

--Extract day from DOB
select first_name ,day(dob) from customers;

select year(dob) as year,
       month(dob) as month,
       day(dob) as day
from customers;
```

Results | Messages

	year	month	day
1	1996	4	24
2	1998	8	15
3	1997	11	12
4	1999	2	5
5	1995	6	18

```sql
--Difference Between Two Dates
select assignment_id,
       datediff(day, startdate, end_date) as policydurationdays
from policyassignments;

select first_name,
datediff(year, dob, getdate()) as age
from customers;

--Get Day Name from Date
select claim_date,
       datename(weekday, claim_date) as dayname
from claims;
```

STRING FUNCTIONS IN SQL

```sql
--Extract the characters from string
select first_name,
       substring(first_name, 1, 3) as first_3_chars
from customers;


--Extract from left side of the string
select first_name,
       left(first_name, 3) as first_3_chars
from customers;


--Extract from left side of the string
select first_name,
       right(first_name, 3) as first_3_chars
from customers;


--Convert Text to Uppercase

select upper(first_name) as uppername
from customers;


--Convert Text to Lowercase

select lower(first_name) as lowername
from customers;

--Replace Letters in a String

select replace(first_name, 'a', '@') as replacedname
from customers;


--Concatenation
select agent_name + '(' + city + ')' as agent_data
from agents order by agent_name;
```

JOINS

```
select c.first_name,c.dob,pa.policy_id,pa.startdate
from customers c,policyAssignments pa
where c.customer_id=pa.customer_id;  --(FK=PK)


--joins on customers and policies
select c.customer_id, p.policy_name
from customers c
JOIN policyAssignments pa on c.customer_id = pa.customer_id
JOIN policies p on pa.policy_id = p.policy_id
where c.customer_id=5;


--joins on customers and agents
select c.first_name,a.agent_name,a.phone
from customers c
JOIN policyAssignments pa on pa.customer_id=c.customer_id
JOIN agents a on pa.agent_id=a.agent_id;


--join on customers and claims
select c.first_name,c.dob,cl.claim_amount,cl.claim_status
from customers c
JOIN policyAssignments p on p.customer_id=c.customer_id
JOIN claims cl on cl.assignment_id=p.assignment_id;


--Display Claims Report with Customer and Policy Details
select c.first_name,p.policy_name,cl.claim_amount,
   cl.claim_status,cl.claim_date
   from customers c
   JOIN policyAssignments pa on pa.customer_id=c.customer_id
   JOIN policies p on pa.policy_id=p.policy_id
   JOIN claims cl on cl.assignment_id = pa.assignment_id;


--Display All Customers with or Without Policies
select c.customer_id, c.first_name + ' ' + c.last_name as
   CustomerName,p.policy_name,p.policy_type, p.premium_amount
   from customers c
   LEFT JOIN policyAssignments pa on  c.customer_id = pa.customer_id
   LEFT JOIN policies p on pa.policy_id = p.policy_id;
```

SUBQUERIES

```
--Find Customers Who Have Made at Least One Claim
select customer_id, first_name, last_name
from customers
where customer_id in (
    select pa.customer_id
    from policyassignments pa
    JOIN claims cl ON pa.assignment_id = cl.assignment_id
);
```

```sql
--Find Policies with Premium Higher Than the Average Premium
select policy_id, policy_name, premium_amount
from policies
where premium_amount >
(
    select avg(premium_amount)
    from policies
);

--Find Customers Who Purchased the Costliest Policy
select distinct c.first_name, c.last_name
from customers c
JOIN policyassignments pa on c.customer_id = pa.customer_id
JOIN policies p on pa.policy_id = p.policy_id
where p.premium_amount =
(
    select max(premium_amount)
    from policies
);



--Find Agents Handling More Than One Policy
select agent_id, agent_name
from agents
where agent_id in (
    select agent_id
    from policyassignments
    group by agent_id
    having count(policy_id) > 1
);



--Find Policies That Have Never Been Claimed
select policy_id, policy_name
from policies
where policy_id not in (
    select pa.policy_id
    from policyAssignments pa
    JOIN claims cl ON pa.assignment_id = cl.assignment_id
);
```

CASE(IF-ELSE)

```sql
--Display Claim Status Description
select claims_id,
       claim_status,
       CASE
           WHEN claim_status = 'Approved' THEN 'Claim Accepted'
           WHEN claim_status = 'Rejected' THEN 'Claim Denied'
           ELSE 'Claim Under Review'
       END AS StatusDescription
from claims;


--Show Policy Duration Type
select policy_id,
       policy_name,
       duration_year,
```

```
       CASE
           WHEN duration_year <= 5 THEN 'Short Term'
           WHEN duration_year BETWEEN 6 AND 10 THEN 'Mid Term'
           ELSE 'Long Term'
       END AS PolicyDurationType
from policies;

--Check Whether Policy Is Expired or Active
select assignment_id,
       end_date,
       CASE
           WHEN end_date < cast(getdate() as date) THEN 'Expired'
           ELSE 'Active'
       END AS PolicyStatus
from policyAssignments;


--Convert NULL City Values to Default Text
select first_name,
       CASE
           WHEN city IS NULL THEN 'City Not Provided'
           ELSE city
       END AS City
from customers;
```

GROUP BY ROLLUP

```
--Total Premium Amount by Policy Type (with Grand Total)
select
    policy_type,
    sum(premium_amount) as totalpremium
from policies
group by rollup (policy_type);


--Agent-wise policy count with overall total
select
    a.agent_name,
    count(pa.policy_id) as policycount
from agents a
LEFT JOIN policyassignments pa
    on a.agent_id = pa.agent_id
group by rollup (a.agent_name);
```

GROUP BY CUBE

```
--Total Premium by Policy Type and Duration (All Combinations)
select
    policy_type,
    duration_year,
    sum(premium_amount) as totalpremium
from policies
group by cube (policy_type, duration_year);


--Agent-wise and Policy-wise Assignment Count
select
    a.agent_name,
    p.policy_type,
```

```
    count(pa.policy_id) as policycount
from policyassignments pa
JOIN agents a on pa.agent_id = a.agent_id
JOIN policies p on pa.policy_id = p.policy_id
group by cube (a.agent_name, p.policy_type);
```

MERGE IN SQL

```
--Update Existing Policies or Insert New Policies
MERGE policies as p
using new_policies as np
on p.policy_id = np.policy_id

when MATCHED then
    update set
        p.premium_amount = np.premium_amount,
        p.duration_year = np.duration_year

when NOT MATCHED then
    insert (policy_id, policy_name, policy_type, premium_amount, duration_year)
    values (np.policy_id, np.policy_name, np.policy_type,
            np.premium_amount, np.duration_year);


--Merge Customer Data (Insert or Update Customers)
MERGE customers as c
using new_customers as nc
on c.customer_id = nc.customer_id

when MATCHED then
    update set
        c.first_name = nc.first_name,
        c.last_name = nc.last_name,
        c.email = nc.email

when NOT MATCHED then
    insert (customer_id, first_name, last_name, dob, phone, email)
    values (nc.customer_id, nc.first_name, nc.last_name,
            nc.dob, nc.phone, nc.email);

          ---------------------------------
MERGE policies as p
using new_policies as np
on p.policy_id = np.policy_id

-- 1. update existing records
when MATCHED then
    update set
        p.policy_name    = np.policy_name,
        p.policy_type    = np.policy_type,
        p.premium_amount = np.premium_amount,
        p.duration_year  = np.duration_year

-- 2. insert new records
when NOT MATCHED by target then
    insert (policy_id, policy_name, policy_type, premium_amount, duration_year)
    values (np.policy_id, np.policy_name, np.policy_type,
            np.premium_amount, np.duration_year)

-- 3. DELETE records not present in source
when NOT MATCHED by source then      delete;
```

SET OPERATIONS

```sql
--Display Cities Where Customers and Agents Are Located (Including Duplicates)
select city from customers
UNION ALL
select city from agents;


--Display Cities Where Both Customers and Agents Belong
select city from customers
INTERSECT
select city from agents;


--Display Customers Who Have Policies OR Have Made Claims
select customer_id from policyassignments
UNION
select pa.customer_id
from policyassignments pa
join claims cl
on pa.assignment_id = cl.assignment_id;


--Display Customers Who Purchased Policies but Never Made Any Claims

select customer_id from policyassignments
EXCEPT
select pa.customer_id
from policyassignments pa
join claims cl
on pa.assignment_id = cl.assignment_id;
```