SOFE3700U
Database Management Systems Project

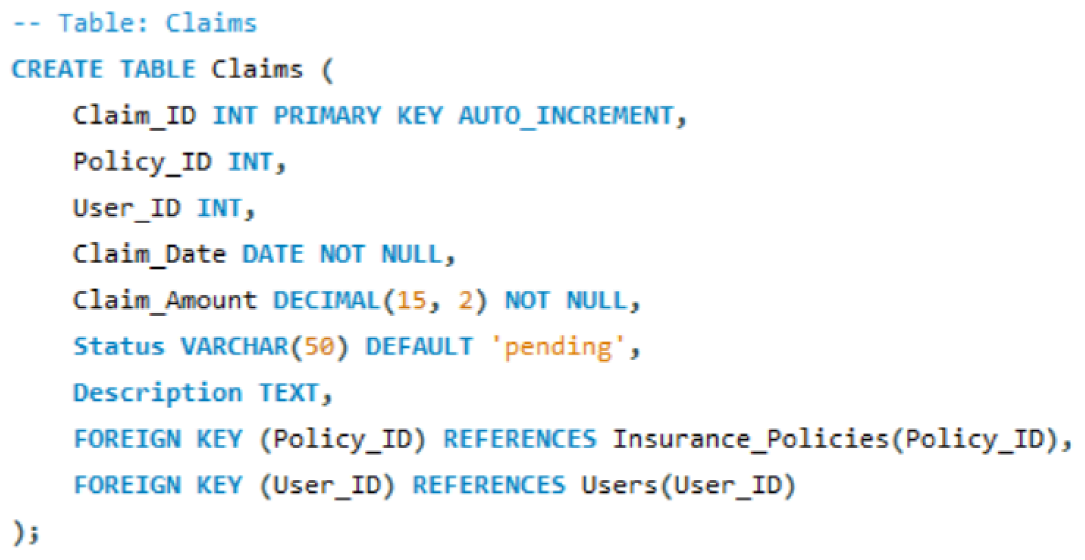
Student	**Student ID**
Sufiya Arab	100876515
Ashka Patel	100871256
Mashal Niazi	100860021
Yordanos Keflinkiel	100867864
Mahnoor Jamal	100822030
Riya Rajesh	100869701

Part A: Relational Schema

The following is a relational database schema for the proposed application area from Phase I:

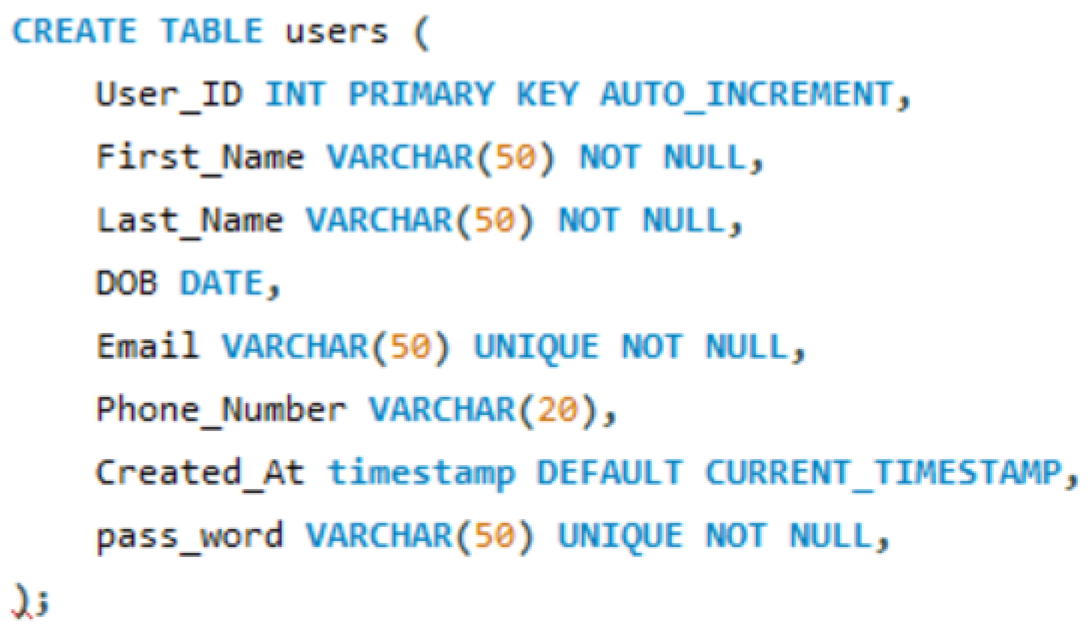
SQL Create Table Commands:.

Claims



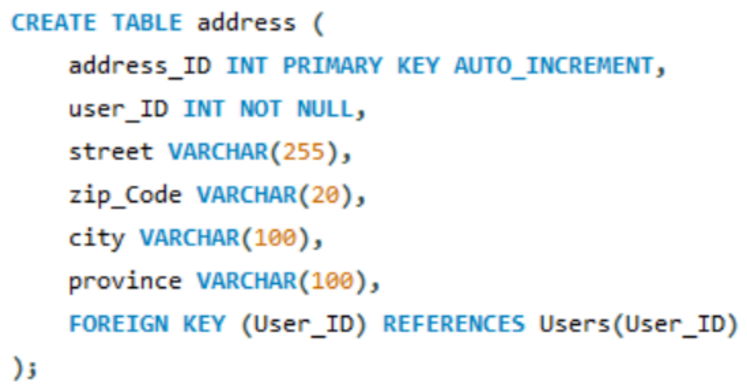
```
-- Table: Claims
CREATE TABLE Claims (
    Claim_ID INT PRIMARY KEY AUTO_INCREMENT,
    Policy_ID INT,
    User_ID INT,
    Claim_Date DATE NOT NULL,
    Claim_Amount DECIMAL(15, 2) NOT NULL,
    Status VARCHAR(50) DEFAULT 'pending',
    Description TEXT,
    FOREIGN KEY (Policy_ID) REFERENCES Insurance_Policies(Policy_ID),
    FOREIGN KEY (User_ID) REFERENCES Users(User_ID)
);
```

Users



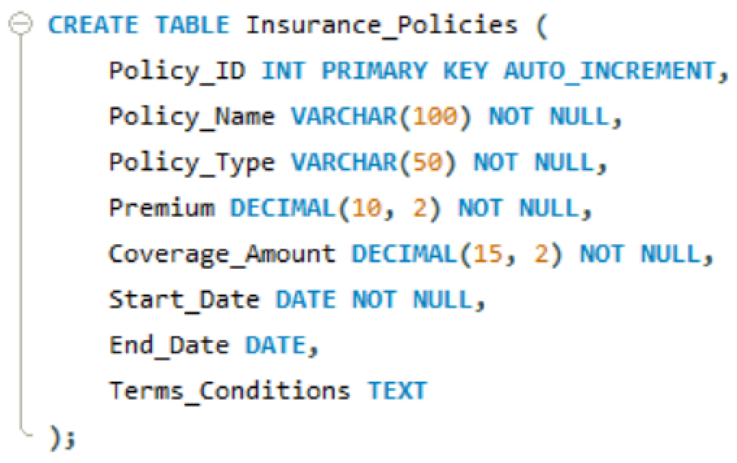
```
CREATE TABLE users (
    User_ID INT PRIMARY KEY AUTO_INCREMENT,
    First_Name VARCHAR(50) NOT NULL,
    Last_Name VARCHAR(50) NOT NULL,
    DOB DATE,
    Email VARCHAR(50) UNIQUE NOT NULL,
    Phone_Number VARCHAR(20),
    Created_At timestamp DEFAULT CURRENT_TIMESTAMP,
    pass_word VARCHAR(50) UNIQUE NOT NULL,
);
```

Addresses



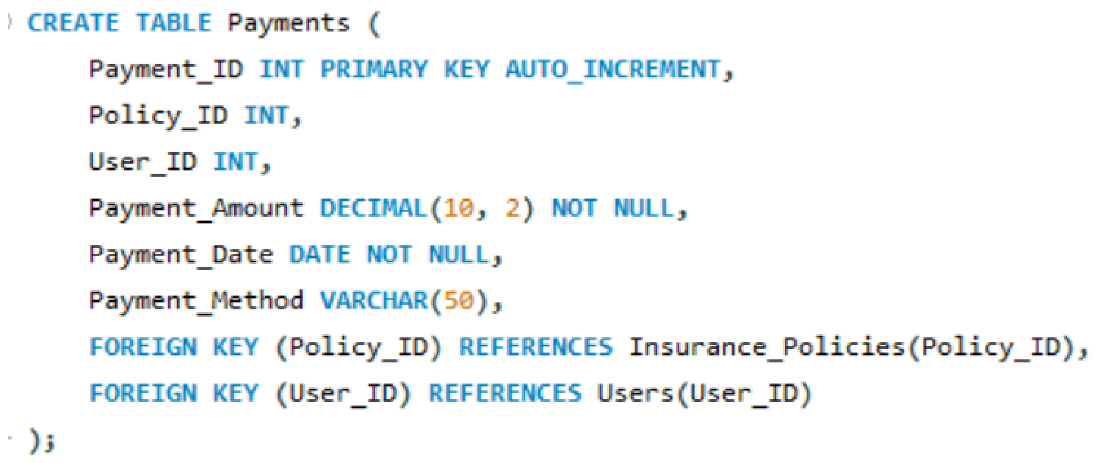
```sql
CREATE TABLE address (
    address_ID INT PRIMARY KEY AUTO_INCREMENT,
    user_ID INT NOT NULL,
    street VARCHAR(255),
    zip_Code VARCHAR(20),
    city VARCHAR(100),
    province VARCHAR(100),
    FOREIGN KEY (User_ID) REFERENCES Users(User_ID)
);
```

InsurancePolicies



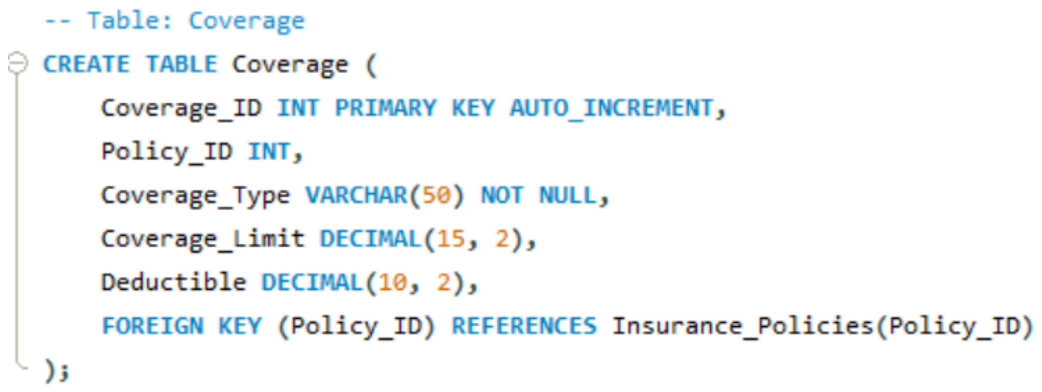
```sql
CREATE TABLE Insurance_Policies (
    Policy_ID INT PRIMARY KEY AUTO_INCREMENT,
    Policy_Name VARCHAR(100) NOT NULL,
    Policy_Type VARCHAR(50) NOT NULL,
    Premium DECIMAL(10, 2) NOT NULL,
    Coverage_Amount DECIMAL(15, 2) NOT NULL,
    Start_Date DATE NOT NULL,
    End_Date DATE,
    Terms_Conditions TEXT
);
```

Payments



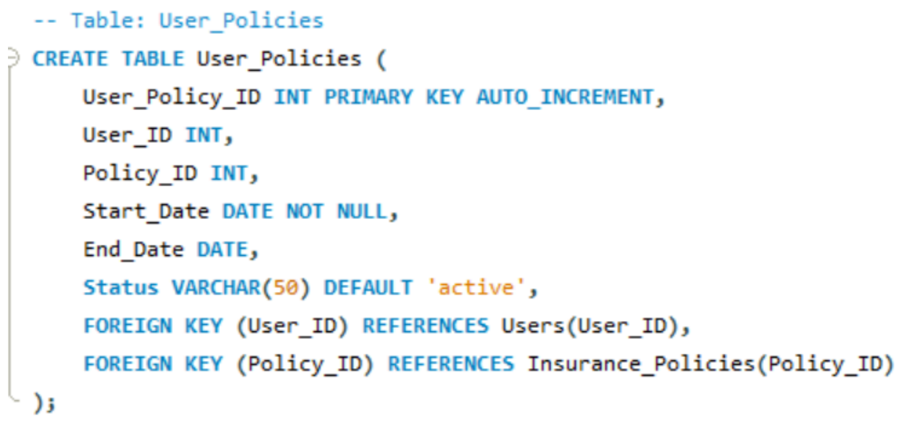
```sql
CREATE TABLE Payments (
    Payment_ID INT PRIMARY KEY AUTO_INCREMENT,
    Policy_ID INT,
    User_ID INT,
    Payment_Amount DECIMAL(10, 2) NOT NULL,
    Payment_Date DATE NOT NULL,
    Payment_Method VARCHAR(50),
    FOREIGN KEY (Policy_ID) REFERENCES Insurance_Policies(Policy_ID),
    FOREIGN KEY (User_ID) REFERENCES Users(User_ID)
);
```

Coverage



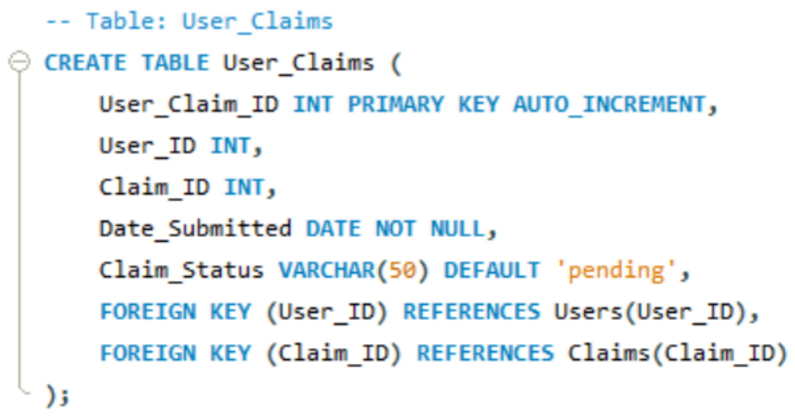
```sql
-- Table: Coverage
CREATE TABLE Coverage (
    Coverage_ID INT PRIMARY KEY AUTO_INCREMENT,
    Policy_ID INT,
    Coverage_Type VARCHAR(50) NOT NULL,
    Coverage_Limit DECIMAL(15, 2),
    Deductible DECIMAL(10, 2),
    FOREIGN KEY (Policy_ID) REFERENCES Insurance_Policies(Policy_ID)
);
```

User_policies



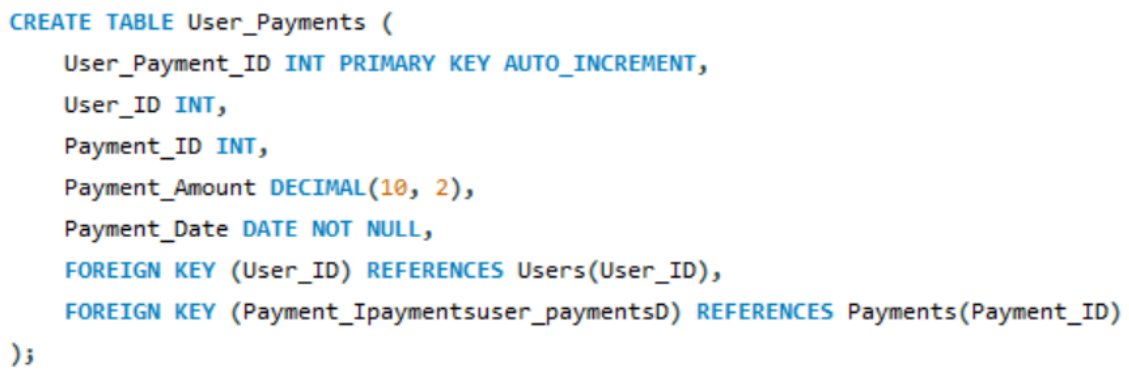
```sql
-- Table: User_Policies
CREATE TABLE User_Policies (
    User_Policy_ID INT PRIMARY KEY AUTO_INCREMENT,
    User_ID INT,
    Policy_ID INT,
    Start_Date DATE NOT NULL,
    End_Date DATE,
    Status VARCHAR(50) DEFAULT 'active',
    FOREIGN KEY (User_ID) REFERENCES Users(User_ID),
    FOREIGN KEY (Policy_ID) REFERENCES Insurance_Policies(Policy_ID)
);
```

User_claims



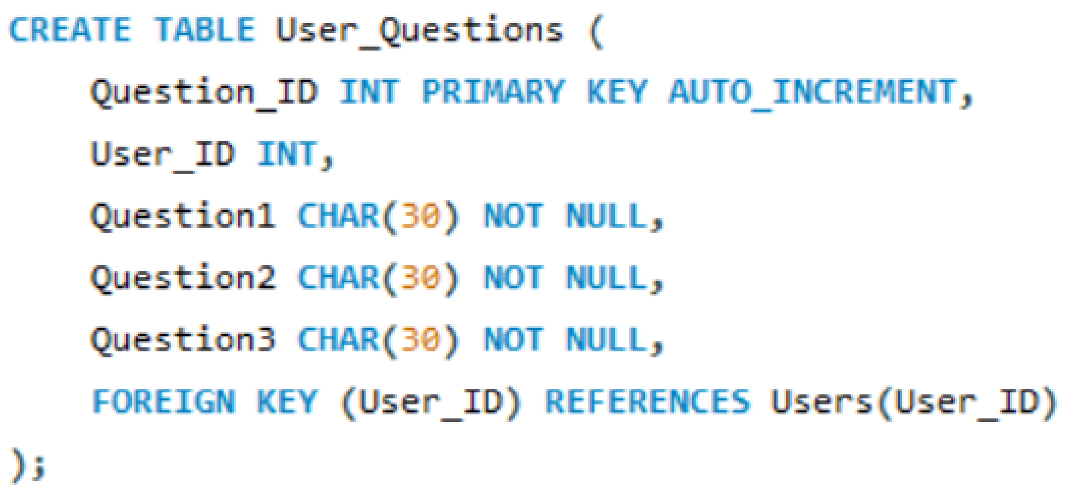
```sql
-- Table: User_Claims
CREATE TABLE User_Claims (
    User_Claim_ID INT PRIMARY KEY AUTO_INCREMENT,
    User_ID INT,
    Claim_ID INT,
    Date_Submitted DATE NOT NULL,
    Claim_Status VARCHAR(50) DEFAULT 'pending',
    FOREIGN KEY (User_ID) REFERENCES Users(User_ID),
    FOREIGN KEY (Claim_ID) REFERENCES Claims(Claim_ID)
);
```

User_payments



```sql
CREATE TABLE User_Payments (
    User_Payment_ID INT PRIMARY KEY AUTO_INCREMENT,
    User_ID INT,
    Payment_ID INT,
    Payment_Amount DECIMAL(10, 2),
    Payment_Date DATE NOT NULL,
    FOREIGN KEY (User_ID) REFERENCES Users(User_ID),
    FOREIGN KEY (Payment_Ipaymentsuser_paymentsD) REFERENCES Payments(Payment_ID)
);
```

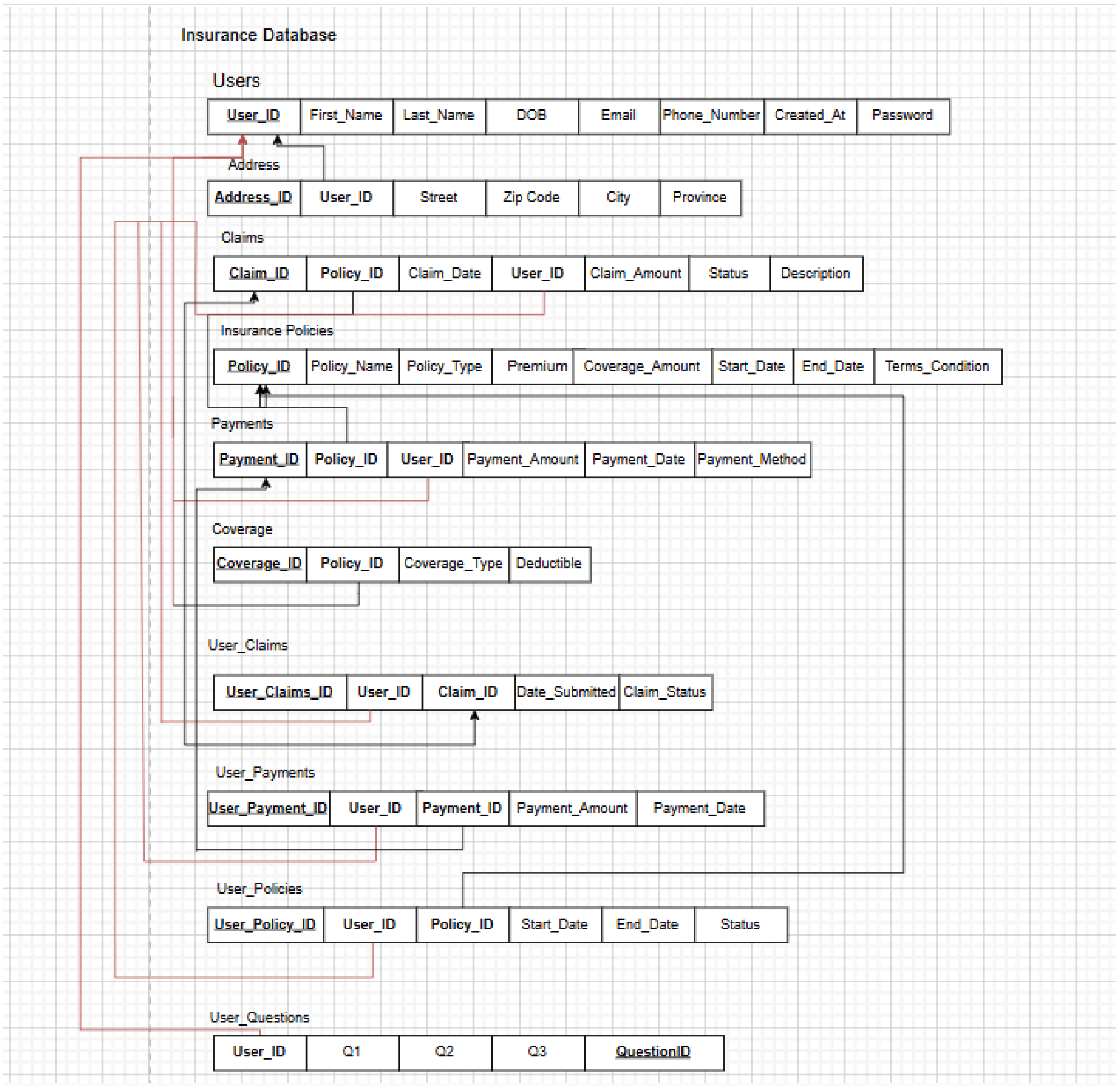
User_questions



```sql
CREATE TABLE User_Questions (
    Question_ID INT PRIMARY KEY AUTO_INCREMENT,
    User_ID INT,
    Question1 CHAR(30) NOT NULL,
    Question2 CHAR(30) NOT NULL,
    Question3 CHAR(30) NOT NULL,
    FOREIGN KEY (User_ID) REFERENCES Users(User_ID)
);
```

Relational Database Graphic:

https://drive.google.com/file/d/16bc5RMH5XLnajjhE8CJZS9ZaZjz-LLsL/view?usp=sharing

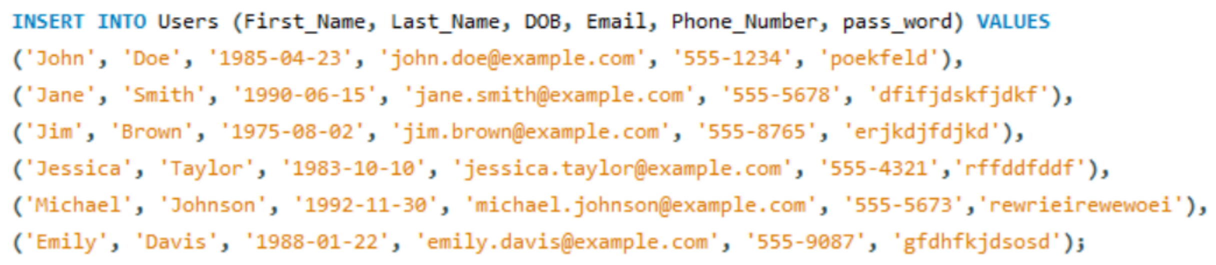


Insurance Database
Users
User_ID | First_Name | Last_Name | DOB | Email | Phone_Number | Created_At | Password
Address
Address_ID | User_ID | Street | Zip Code | City | Province
Claims
Claim_ID | Policy_ID | Claim_Date | User_ID | Claim_Amount | Status | Description
Insurance Policies
Policy_ID | Policy_Name | Policy_Type | Premium | Coverage_Amount | Start_Date | End_Date | Terms_Condition
Payments
Payment_ID | Policy_ID | User_ID | Payment_Amount | Payment_Date | Payment_Method
Coverage
Coverage_ID | Policy_ID | Coverage_Type | Deductible
User_Claims
User_Claims_ID | User_ID | Claim_ID | Date_Submitted | Claim_Status
User_Payments
User_Payment_ID | User_ID | Payment_ID | Payment_Amount | Payment_Date
User_Policies
User_Policy_ID | User_ID | Policy_ID | Start_Date | End_Date | Status
User_Questions
User_ID | Q1 | Q2 | Q3 | QuestionID

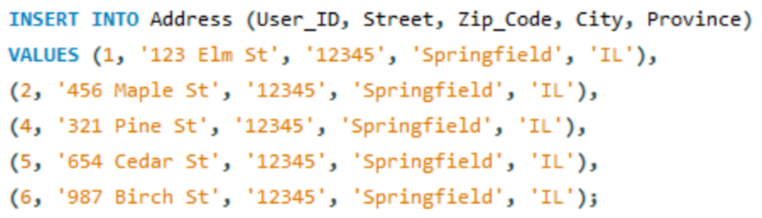
Part B: Sample Data

The following is the database populated with sample data. Each relation contains at least 6 tuples and the populated data is suitable for the type of queries in Part C.

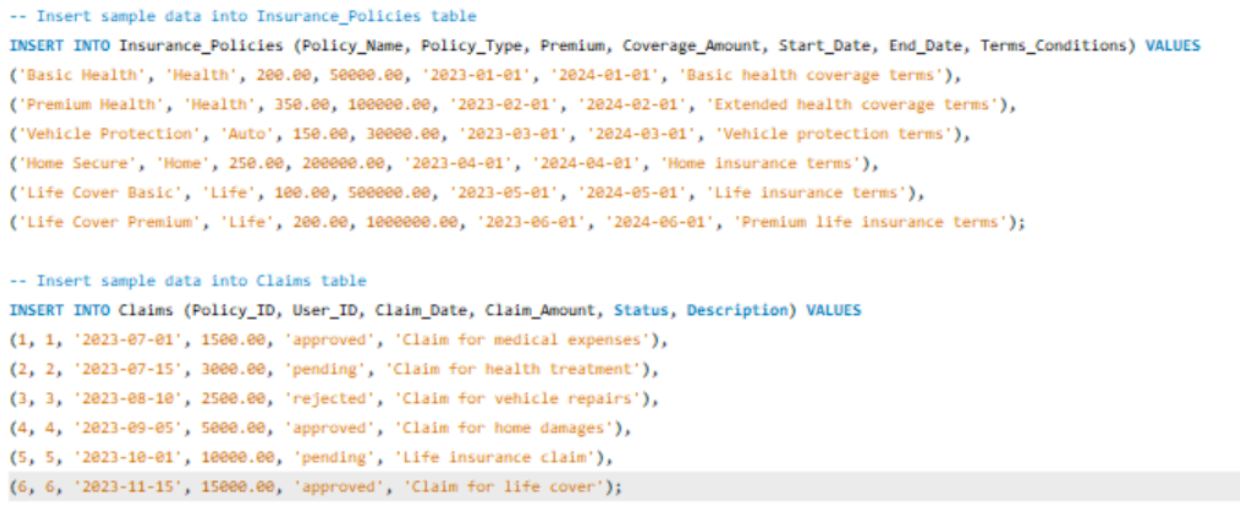
CSV Files



```sql
INSERT INTO Users (First_Name, Last_Name, DOB, Email, Phone_Number, pass_word) VALUES
('John', 'Doe', '1985-04-23', 'john.doe@example.com', '555-1234', 'poekfeld'),
('Jane', 'Smith', '1990-06-15', 'jane.smith@example.com', '555-5678', 'dfifjdskfjdkf'),
('Jim', 'Brown', '1975-08-02', 'jim.brown@example.com', '555-8765', 'erjkdjfdjkd'),
('Jessica', 'Taylor', '1983-10-10', 'jessica.taylor@example.com', '555-4321','rffddfddf'),
('Michael', 'Johnson', '1992-11-30', 'michael.johnson@example.com', '555-5673','rewrieirewewoei'),
('Emily', 'Davis', '1988-01-22', 'emily.davis@example.com', '555-9087', 'gfdhfkjdsosd');
```



```sql
INSERT INTO Address (User_ID, Street, Zip_Code, City, Province)
VALUES (1, '123 Elm St', '12345', 'Springfield', 'IL'),
(2, '456 Maple St', '12345', 'Springfield', 'IL'),
(4, '321 Pine St', '12345', 'Springfield', 'IL'),
(5, '654 Cedar St', '12345', 'Springfield', 'IL'),
(6, '987 Birch St', '12345', 'Springfield', 'IL');
```



```sql
-- Insert sample data into Insurance_Policies table
INSERT INTO Insurance_Policies (Policy_Name, Policy_Type, Premium, Coverage_Amount, Start_Date, End_Date, Terms_Conditions) VALUES
('Basic Health', 'Health', 200.00, 50000.00, '2023-01-01', '2024-01-01', 'Basic health coverage terms'),
('Premium Health', 'Health', 350.00, 100000.00, '2023-02-01', '2024-02-01', 'Extended health coverage terms'),
('Vehicle Protection', 'Auto', 150.00, 30000.00, '2023-03-01', '2024-03-01', 'Vehicle protection terms'),
('Home Secure', 'Home', 250.00, 200000.00, '2023-04-01', '2024-04-01', 'Home insurance terms'),
('Life Cover Basic', 'Life', 100.00, 500000.00, '2023-05-01', '2024-05-01', 'Life insurance terms'),
('Life Cover Premium', 'Life', 200.00, 1000000.00, '2023-06-01', '2024-06-01', 'Premium life insurance terms');

-- Insert sample data into Claims table
INSERT INTO Claims (Policy_ID, User_ID, Claim_Date, Claim_Amount, Status, Description) VALUES
(1, 1, '2023-07-01', 1500.00, 'approved', 'Claim for medical expenses'),
(2, 2, '2023-07-15', 3000.00, 'pending', 'Claim for health treatment'),
(3, 3, '2023-08-10', 2500.00, 'rejected', 'Claim for vehicle repairs'),
(4, 4, '2023-09-05', 5000.00, 'approved', 'Claim for home damages'),
(5, 5, '2023-10-01', 10000.00, 'pending', 'Life insurance claim'),
(6, 6, '2023-11-15', 15000.00, 'approved', 'Claim for life cover');
```

```sql
INSERT INTO Coverage (Policy_ID, Coverage_Type, Coverage_Limit, Deductible) VALUES
(1, 'Hospitalization', 20000.00, 500.00),
(2, 'Extended Hospitalization', 50000.00, 1000.00),
(3, 'Vehicle Damage', 30000.00, 750.00),
(4, 'Natural Disaster', 150000.00, 2500.00),
(5, 'Accidental Death', 500000.00, 0.00),
(6, 'Critical Illness', 1000000.00, 0.00);

-- Insert sample data into Payments table
INSERT INTO Payments (Policy_ID, User_ID, Payment_Amount, Payment_Date, Payment_Method) VALUES
(1, 1, 200.00, '2023-01-05', 'Credit Card'),
(2, 2, 350.00, '2023-02-05', 'Bank Transfer'),
(3, 3, 150.00, '2023-03-05', 'Credit Card'),
(4, 4, 250.00, '2023-04-05', 'Debit Card'),
(5, 5, 100.00, '2023-05-05', 'PayPal'),
(6, 6, 200.00, '2023-06-05', 'Credit Card');
```

```sql
INSERT INTO User_Claims (User_ID, Claim_ID, Date_Submitted, Claim_Status) VALUES
(1, 1, '2023-07-01', 'approved'),
(2, 2, '2023-07-15', 'pending'),
(3, 3, '2023-08-10', 'rejected'),
(4, 4, '2023-09-05', 'approved'),
(5, 5, '2023-10-01', 'pending'),
(6, 6, '2023-11-15', 'approved');

-- Insert sample data into User_Payments table
INSERT INTO User_Payments (User_ID, Payment_ID, Payment_Amount, Payment_Date) VALUES
(1, 1, 200.00, '2023-01-05'),
(2, 2, 350.00, '2023-02-05'),
(3, 3, 150.00, '2023-03-05'),
(4, 4, 250.00, '2023-04-05'),
(5, 5, 100.00, '2023-05-05'),
(6, 6, 200.00, '2023-06-05');

-- Insert sample data into User_Questions table
INSERT INTO User_Questions (User_ID, Question1, Question2, Question3) VALUES
(1, 'Pet name?', 'Mother’s maiden name?', 'First school?'),
(2, 'Pet name?', 'Favorite teacher?', 'Birth city?'),
(3, 'Mother’s maiden name?', 'Pet name?', 'Favorite color?'),
(4, 'Favorite teacher?', 'Pet name?', 'First school?'),
(5, 'Pet name?', 'First school?', 'Favorite food?'),
(6, 'First school?', 'Mother’s maiden name?', 'Favorite color?');
```

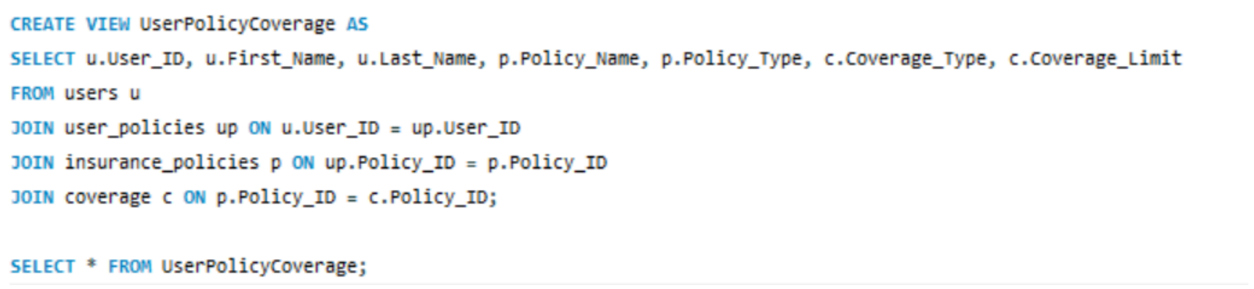
```sql
-- Insert sample data into User_Policies table
INSERT INTO User_Policies (User_ID, Policy_ID, Start_Date, End_Date, Status) VALUES
(1, 1, '2023-01-01', '2024-01-01', 'active'),
(2, 2, '2023-02-01', '2024-02-01', 'active'),
(3, 3, '2023-03-01', '2024-03-01', 'active'),
(4, 4, '2023-04-01', '2024-04-01', 'active'),
(5, 5, '2023-05-01', '2024-05-01', 'expired'),
(6, 6, '2023-06-01', '2024-06-01', 'active');
```

Part C: Views

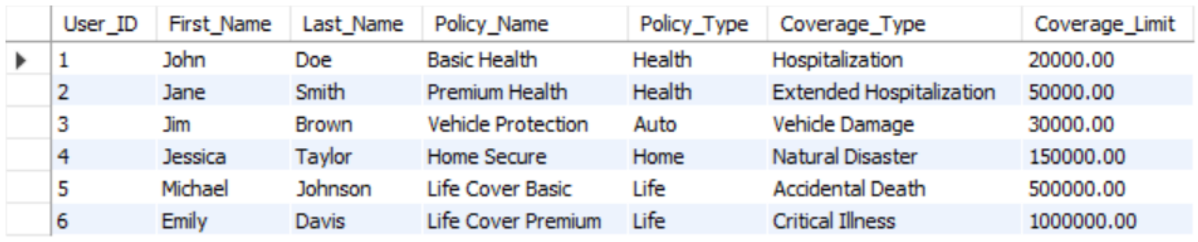
The following are 10 views, each with their SQL syntax and English description, that a user of the database system would find useful.Note, from these 10 views, the first 5 are common for all groups.

- **View 1: Computes a join of at least three tables**

This view will show user details, their policies, and related coverage information.



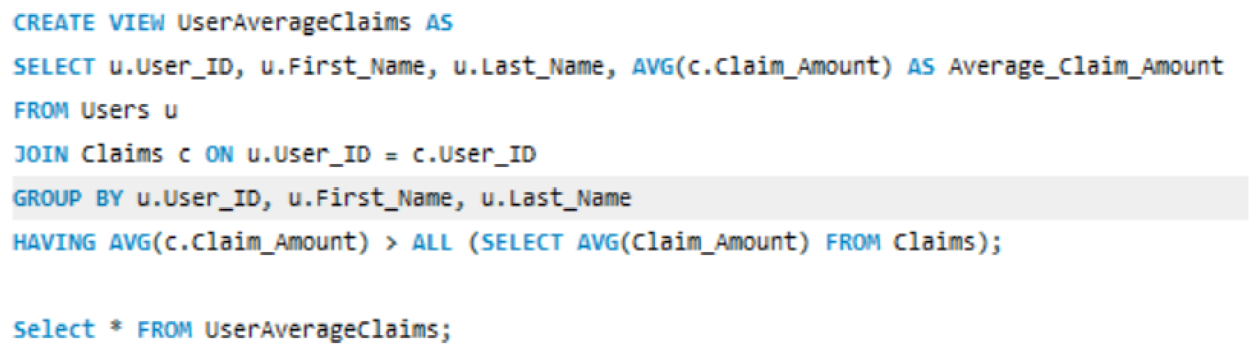
```
CREATE VIEW UserPolicyCoverage AS
SELECT u.User_ID, u.First_Name, u.Last_Name, p.Policy_Name, p.Policy_Type, c.Coverage_Type, c.Coverage_Limit
FROM users u
JOIN user_policies up ON u.User_ID = up.User_ID
JOIN insurance_policies p ON up.Policy_ID = p.Policy_ID
JOIN coverage c ON p.Policy_ID = c.Policy_ID;

SELECT * FROM UserPolicyCoverage;
```



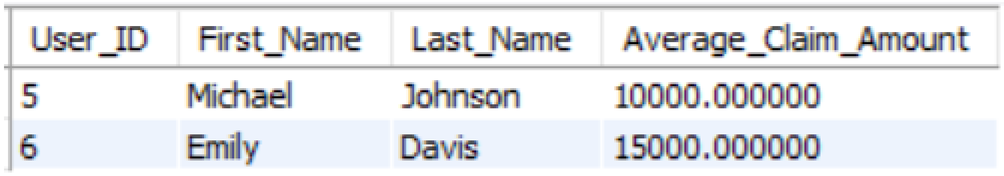
User_ID	First_Name	Last_Name	Policy_Name	Policy_Type	Coverage_Type	Coverage_Limit
1	John	Doe	Basic Health	Health	Hospitalization	20000.00
2	Jane	Smith	Premium Health	Health	Extended Hospitalization	50000.00
3	Jim	Brown	Vehicle Protection	Auto	Vehicle Damage	30000.00
4	Jessica	Taylor	Home Secure	Home	Natural Disaster	150000.00
5	Michael	Johnson	Life Cover Basic	Life	Accidental Death	500000.00
6	Emily	Davis	Life Cover Premium	Life	Critical Illness	1000000.00

- **View 2: Uses nested queries with the ANY or ALL operator and uses a GROUP BY clause**

This view will return the average claim amount for each user and filter out users who have made claims above the average of all users' claim amounts.



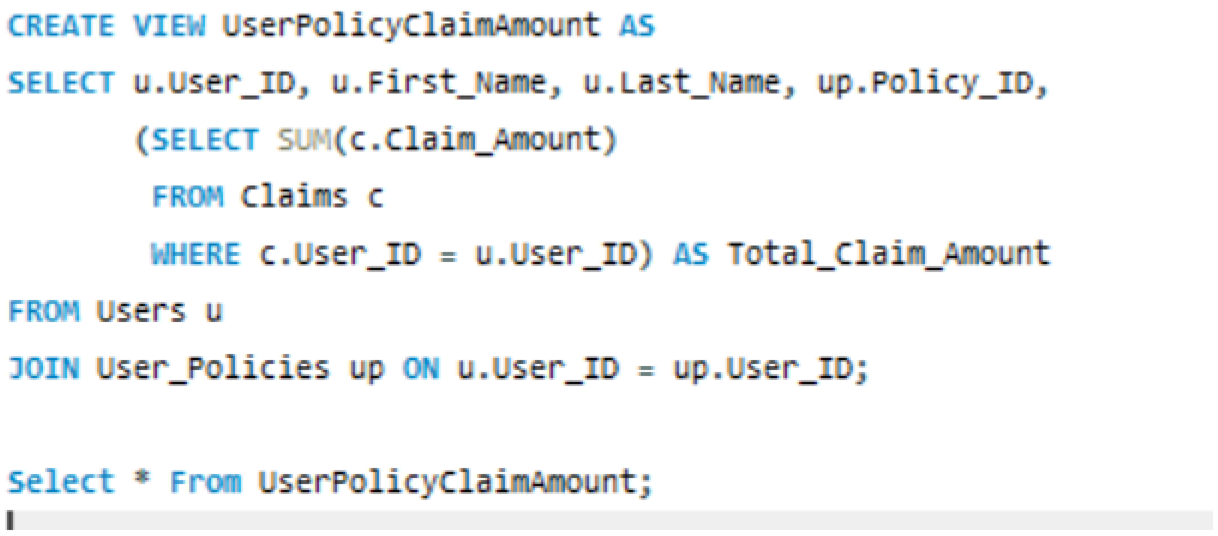
```
CREATE VIEW UserAverageClaims AS
SELECT u.User_ID, u.First_Name, u.Last_Name, AVG(c.Claim_Amount) AS Average_Claim_Amount
FROM Users u
JOIN Claims c ON u.User_ID = c.User_ID
GROUP BY u.User_ID, u.First_Name, u.Last_Name
HAVING AVG(c.Claim_Amount) > ALL (SELECT AVG(Claim_Amount) FROM Claims);

Select * FROM UserAverageClaims;
```



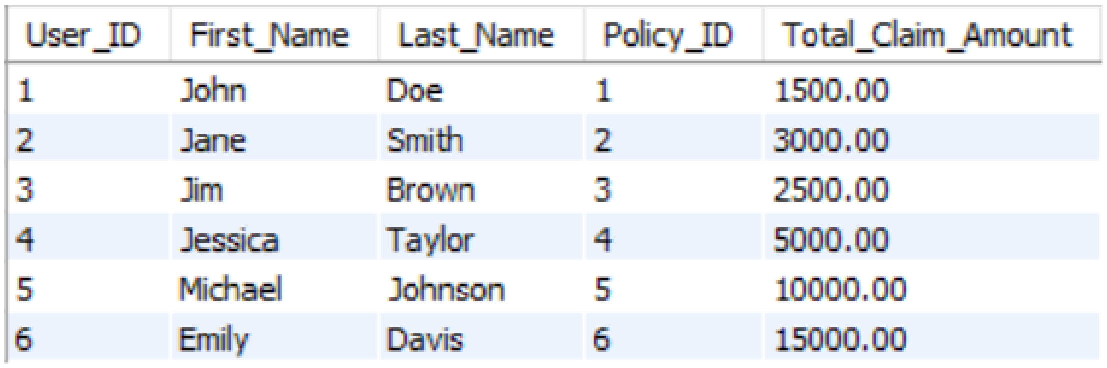
User_ID	First_Name	Last_Name	Average_Claim_Amount
5	Michael	Johnson	10000.000000
6	Emily	Davis	15000.000000

- **View 3: A correlated nested query**

This view will list each user and their policies, along with the total claim amount for each user where claims exist.



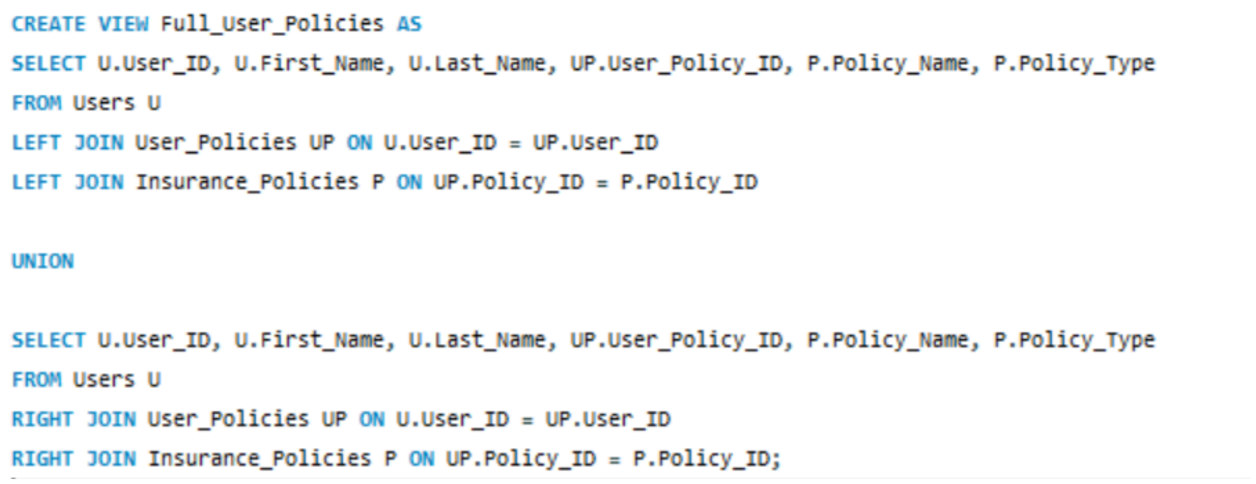
```
CREATE VIEW UserPolicyClaimAmount AS
SELECT u.User_ID, u.First_Name, u.Last_Name, up.Policy_ID,
       (SELECT SUM(c.Claim_Amount)
        FROM Claims c
        WHERE c.User_ID = u.User_ID) AS Total_Claim_Amount
FROM Users u
JOIN User_Policies up ON u.User_ID = up.User_ID;

Select * From UserPolicyClaimAmount;
```



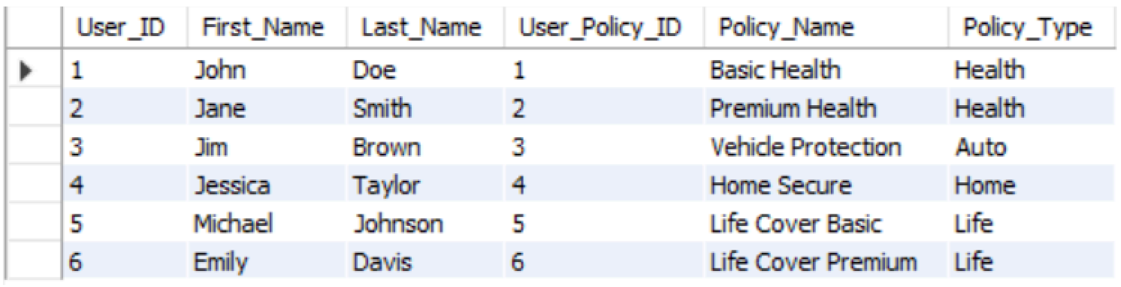
User_ID	First_Name	Last_Name	Policy_ID	Total_Claim_Amount
1	John	Doe	1	1500.00
2	Jane	Smith	2	3000.00
3	Jim	Brown	3	2500.00
4	Jessica	Taylor	4	5000.00
5	Michael	Johnson	5	10000.00
6	Emily	Davis	6	15000.00

- **View 4: Uses a FULL JOIN**

The first part uses a `LEFT JOIN` *to get all users with their policies. The second part uses a* `RIGHT JOIN` *to ensure that all policies are included, even if they have no associated users. The* `UNION` *combines both results, simulating a full outer join.*



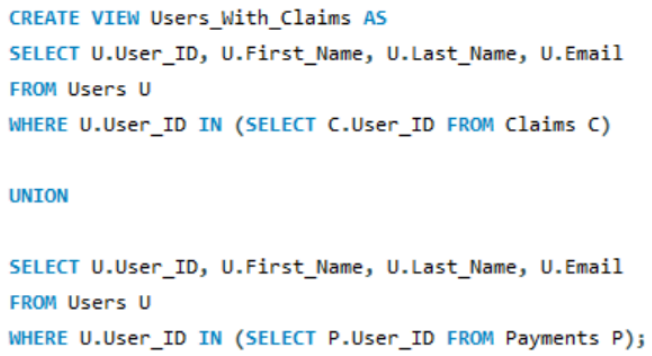
```
CREATE VIEW Full_User_Policies AS
SELECT U.User_ID, U.First_Name, U.Last_Name, UP.User_Policy_ID, P.Policy_Name, P.Policy_Type
FROM Users U
LEFT JOIN User_Policies UP ON U.User_ID = UP.User_ID
LEFT JOIN Insurance_Policies P ON UP.Policy_ID = P.Policy_ID

UNION

SELECT U.User_ID, U.First_Name, U.Last_Name, UP.User_Policy_ID, P.Policy_Name, P.Policy_Type
FROM Users U
RIGHT JOIN User_Policies UP ON U.User_ID = UP.User_ID
RIGHT JOIN Insurance_Policies P ON UP.Policy_ID = P.Policy_ID;
```



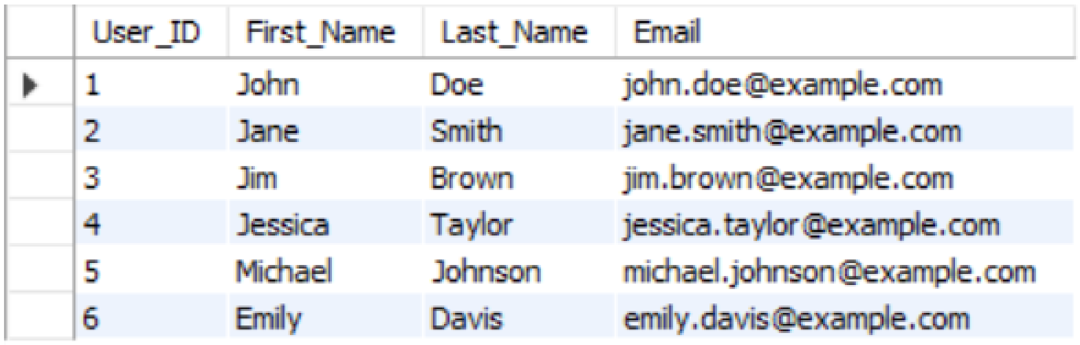
	User_ID	First_Name	Last_Name	User_Policy_ID	Policy_Name	Policy_Type
▶	1	John	Doe	1	Basic Health	Health
	2	Jane	Smith	2	Premium Health	Health
	3	Jim	Brown	3	Vehicle Protection	Auto
	4	Jessica	Taylor	4	Home Secure	Home
	5	Michael	Johnson	5	Life Cover Basic	Life
	6	Emily	Davis	6	Life Cover Premium	Life

- **View 5: Uses nested queries with any of the set operations UNION, EXCEPT, or INTERSECT**

This view uses `UNION` to combine two sets of users: those who have claims and those who have made payments.



```
CREATE VIEW Users_With_Claims AS
SELECT U.User_ID, U.First_Name, U.Last_Name, U.Email
FROM Users U
WHERE U.User_ID IN (SELECT C.User_ID FROM Claims C)

UNION

SELECT U.User_ID, U.First_Name, U.Last_Name, U.Email
FROM Users U
WHERE U.User_ID IN (SELECT P.User_ID FROM Payments P);
```

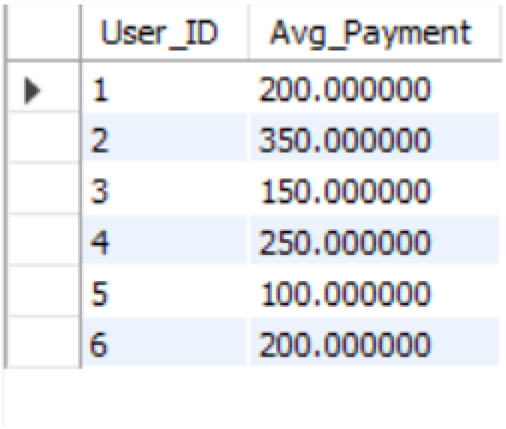


User_ID	First_Name	Last_Name	Email
1	John	Doe	john.doe@example.com
2	Jane	Smith	jane.smith@example.com
3	Jim	Brown	jim.brown@example.com
4	Jessica	Taylor	jessica.taylor@example.com
5	Michael	Johnson	michael.johnson@example.com
6	Emily	Davis	emily.davis@example.com

- **View 6: Windows Function**

This view uses the AVG function to calculate the average payment amount for each user in the payments table. It provides insights into individual user spending behavior.

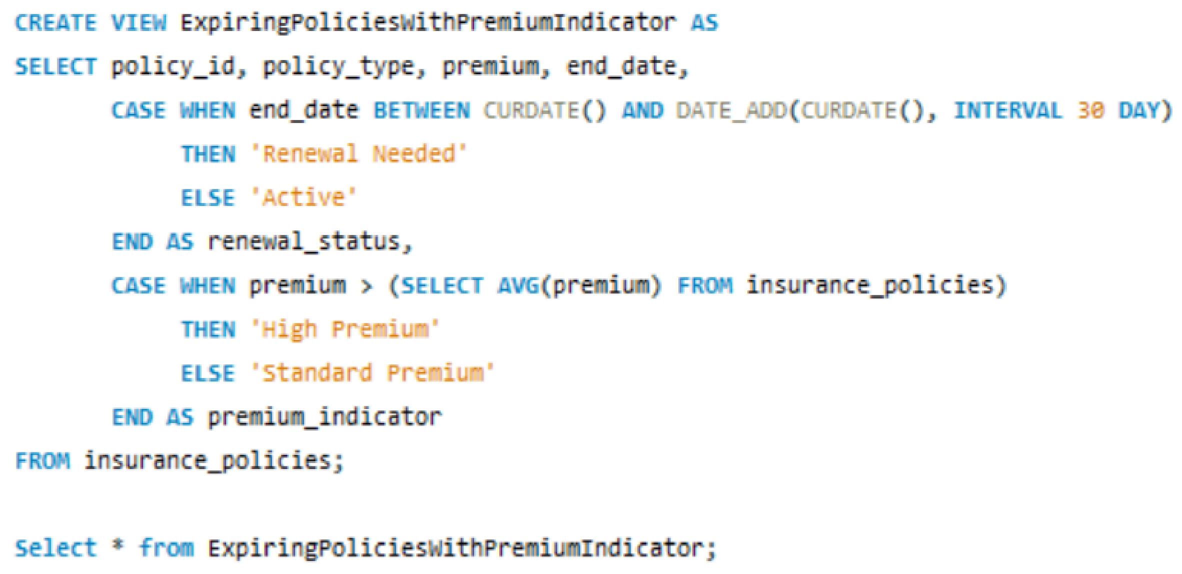
```
CREATE VIEW AvgPaymentPerUser AS
SELECT
    U.User_ID,
    AVG(P.Payment_Amount) AS Avg_Payment
FROM
    Users U
JOIN
    Payments P ON U.User_ID = P.User_ID
GROUP BY
    U.User_ID;

select * FROM AvgPaymentPerUser;
```



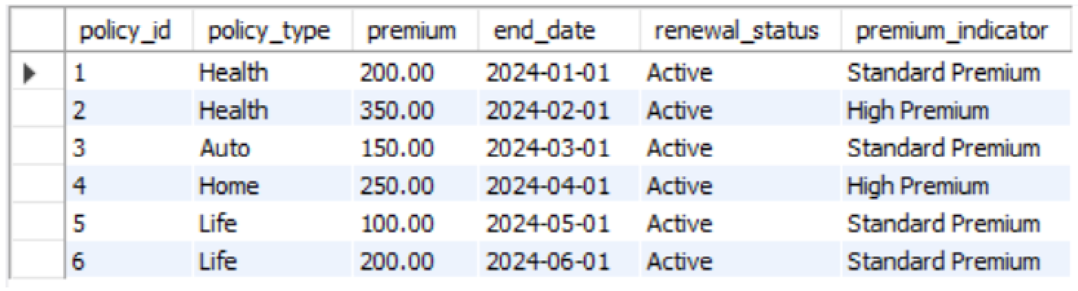
User_ID	Avg_Payment
1	200.000000
2	350.000000
3	150.000000
4	250.000000
5	100.000000
6	200.000000

- **View 7: Case statement**

This view uses conditional logic with the CASE *statement to mark policies expiring within 30 days as "Renewal Needed" and those with high premiums as "High Premium." This is useful for prioritizing renewals.*



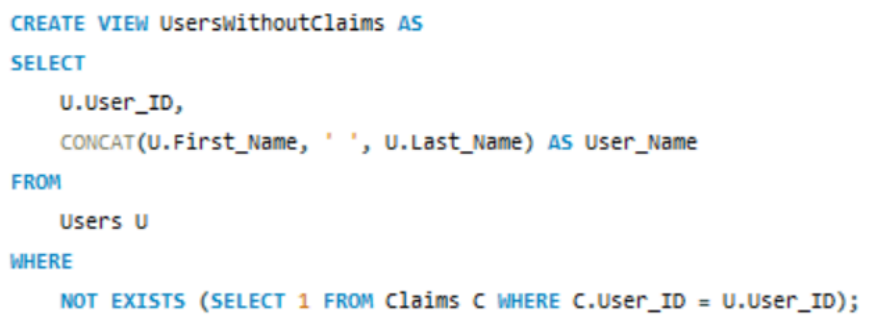
```
CREATE VIEW ExpiringPoliciesWithPremiumIndicator AS
SELECT policy_id, policy_type, premium, end_date,
       CASE WHEN end_date BETWEEN CURDATE() AND DATE_ADD(CURDATE(), INTERVAL 30 DAY)
            THEN 'Renewal Needed'
            ELSE 'Active'
       END AS renewal_status,
       CASE WHEN premium > (SELECT AVG(premium) FROM insurance_policies)
            THEN 'High Premium'
            ELSE 'Standard Premium'
       END AS premium_indicator
FROM insurance_policies;

Select * from ExpiringPoliciesWithPremiumIndicator;
```



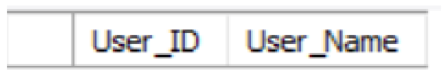
	policy_id	policy_type	premium	end_date	renewal_status	premium_indicator
▶	1	Health	200.00	2024-01-01	Active	Standard Premium
	2	Health	350.00	2024-02-01	Active	High Premium
	3	Auto	150.00	2024-03-01	Active	Standard Premium
	4	Home	250.00	2024-04-01	Active	High Premium
	5	Life	100.00	2024-05-01	Active	Standard Premium
	6	Life	200.00	2024-06-01	Active	Standard Premium

- **View 8: Exists function**

This view lists all users who have no entries in the claims table, helping identify users who haven't filed any claims since registration.



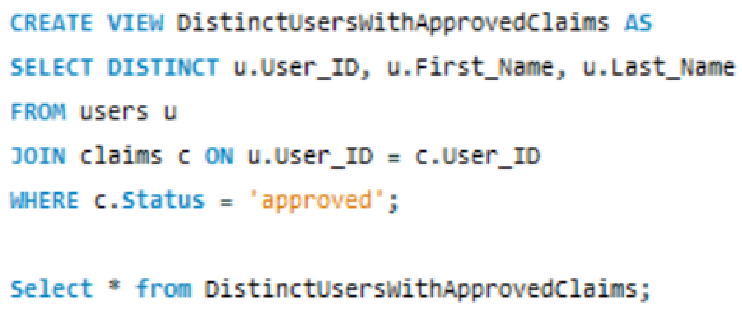
```
CREATE VIEW UsersWithoutClaims AS
SELECT
    U.User_ID,
    CONCAT(U.First_Name, ' ', U.Last_Name) AS User_Name
FROM
    Users U
WHERE
    NOT EXISTS (SELECT 1 FROM Claims C WHERE C.User_ID = U.User_ID);
```



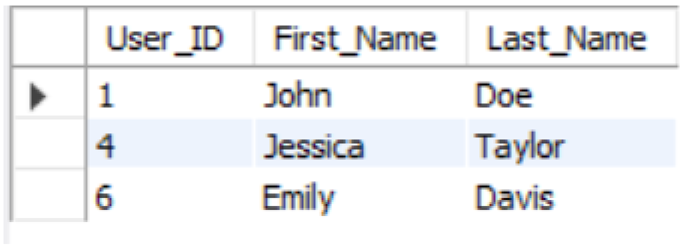
	User_ID	User_Name

- **View 9: Distinct clause**

This view displays a distinct list of users who have at least one approved claim, including their first and last names. The DISTINCT ensures each user appears only once.



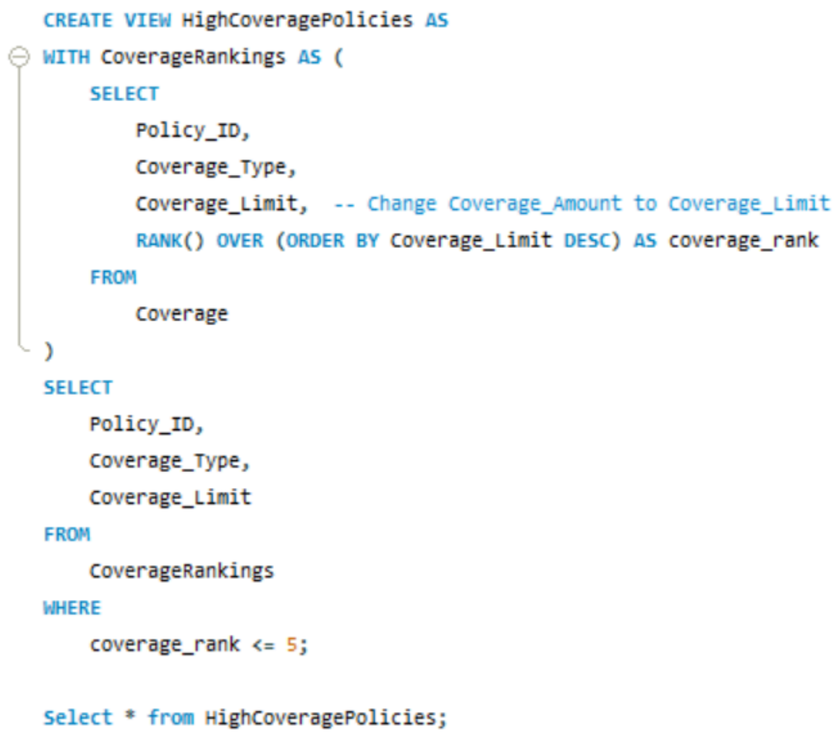
```
CREATE VIEW DistinctUsersWithApprovedClaims AS
SELECT DISTINCT u.User_ID, u.First_Name, u.Last_Name
FROM users u
JOIN claims c ON u.User_ID = c.User_ID
WHERE c.Status = 'approved';

Select * from DistinctUsersWithApprovedClaims;
```



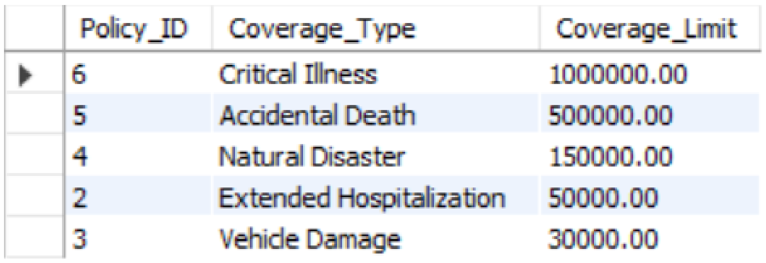
User_ID	First_Name	Last_Name
1	John	Doe
4	Jessica	Taylor
6	Emily	Davis

- **View 10: With clause**

This view ranks policies based on their Coverage_Amount, allowing you to see which policies offer the highest coverage. The WITH clause creates a temporary table (CoverageRankings) that ranks each policy by its Coverage_Amount using a window function (RANK()). The main query then filters the CTE to show only the top 5 policies by coverage amount.



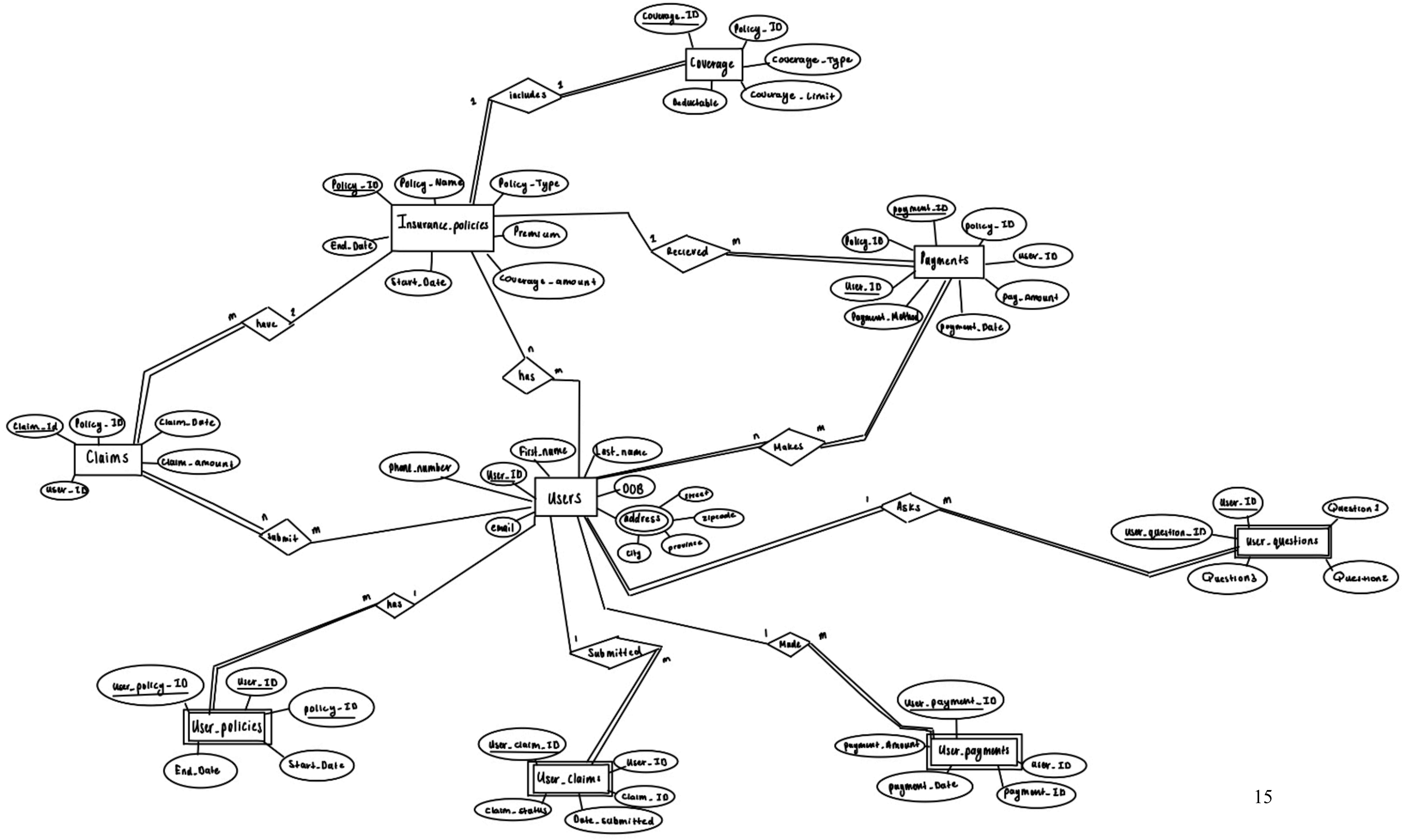
```
CREATE VIEW HighCoveragePolicies AS
WITH CoverageRankings AS (
    SELECT
        Policy_ID,
        Coverage_Type,
        Coverage_Limit,  -- Change Coverage_Amount to Coverage_Limit
        RANK() OVER (ORDER BY Coverage_Limit DESC) AS coverage_rank
    FROM
        Coverage
)
SELECT
    Policy_ID,
    Coverage_Type,
    Coverage_Limit
FROM
    CoverageRankings
WHERE
    coverage_rank <= 5;

Select * from HighCoveragePolicies;
```



	Policy_ID	Coverage_Type	Coverage_Limit
▶	6	Critical Illness	1000000.00
	5	Accidental Death	500000.00
	4	Natural Disaster	150000.00
	2	Extended Hospitalization	50000.00
	3	Vehicle Damage	30000.00

Part D: E-R diagram:

The following is an E-R schema diagram for the database of the project:.


Coverage_ID
Policy_ID
Coverage
Coverage_Type
Deductable
Coverage_Limit
includes
1
1
Policy_ID
Policy_Name
Policy_Type
Insurance_policies
Premium
End_Date
Start_Date
Coverage_amount
1
Recieved
m
payment_ID
policy_ID
Policy_ID
Payments
user_ID
User_ID
pay_Amount
Payment_Method
payment_Date
m
have
1
n
has
m
Claim_Id
Policy_ID
Claim_Date
Claims
Claim_amount
user_ID
First_name
Last_name
phone_number
User_ID
Users
DOB
Address
street
zipcode
province
city
email
n
Makes
m
n
submit
m
1
Asks
m
User_ID
Question 1
User_question_ID
User_questions
Question3
Question2
m
has
1
1
Submitted
m
1
Made
m
user_policy_ID
User_ID
policy_ID
User_policies
End_Date
Start_Date
User_claim_ID
User_claims
User_ID
Claim_ID
claim_status
Date_submitted
User_payment_ID
payment_Amount
User_payments
user_ID
payment_Date
payment_ID