**INTERNAL PROJECT**

**CASE STUDY ON LIC**

**DBMS**

**TEAM MEMBERS: -**

1. **Ashlesha Nanaware**
2. **Habiba Ansari**
3. **Riddhi Lokegaonkar**
4. **Pratik More**
5. **Komal Agrahari**
6. **Aditya Jagtap**

**Index**

* **Introduction**
  + LIC Overview
  + Purpose of the Document
* **Case Study: LIC Policy Management Database**
  + Overview
  + Entities and Attributes
  + Relationships
  + ER Diagram
* **Tables**
  + Policyholder Table
  + Policyholder Address Table
  + Policy Table
  + Agent Table
  + Agent Address Table
  + Beneficiary Table
  + Payment Table
  + Relationship Tables
* **Queries**
  1. List Beneficiaries and Their Relationship for a Specific Policyholder
  2. Policy with Highest Premium
  3. Beneficiary Share Calculation
  4. Payment History for a Policy
  5. Retrieve Policyholders and their Policy Details
  6. List Policies with Premium Amount Greater Than a Threshold
  7. List Policyholders in a Specific City
  8. Retrieve Policyholders with Age Greater Than 35
  9. Calculate Total Premium Collection
  10. Find Policies with Expiry Date Within a Year
* **Functions**
  + Get Policyholder Information
* **Procedures**
  + Get Policy Type Summary
* **Triggers**
  + Generate Policy Number Trigger
* **Conclusion**

****

**Case Study: -**

Life Insurance Corporation (LIC) is one of the largest life insurance companies in the world, serving millions of policyholders across various insurance products. To improve operational efficiency, customer service, and data management, LIC has initiated the development of a comprehensive Policy Management Database (PMD). This case study provides an in-depth view of the LIC PMD.

**Policy Purchase: -**

* A policyholder can choose a policy type, provide personal information, and purchase a policy.
* The system generates a unique PolicyNumber and records policy details.
* A policyholder can own multiple policies, but each policy is owned by one policyholder.

**Agent Assignment: -**

* Agents can be assigned to policies they sell.
* Multiple agents can be associated with a single policy, and vice versa.
* An agent can sell multiple policies, and a policy can be sold by multiple agents.

**Beneficiary Management: -**

* Policyholders can add beneficiaries to their policies.
* Policyholders can add, modify, or remove beneficiaries for their policies.
* Beneficiaries can be shared among different policies.
* A policy can have multiple beneficiaries, and a beneficiary can be associated with multiple policies.
* Beneficiary information is tracked, including the percentage share of the policy amount.

**Payment Processing: -**

* Policyholders can make payments for their policies.
* Policyholders can make payments for their policies through various methods, such as bank transfers, credit cards, or automated deductions.
* Payment details are recorded, including PaymentDate, PaymentAmount, and PaymentMethod.
* A policyholder can make multiple payments, but each payment is associated with one policy.

**Policyholder Information: -**

* Policyholders can update their personal information, such as address, email, or phone number, to ensure accurate communication.

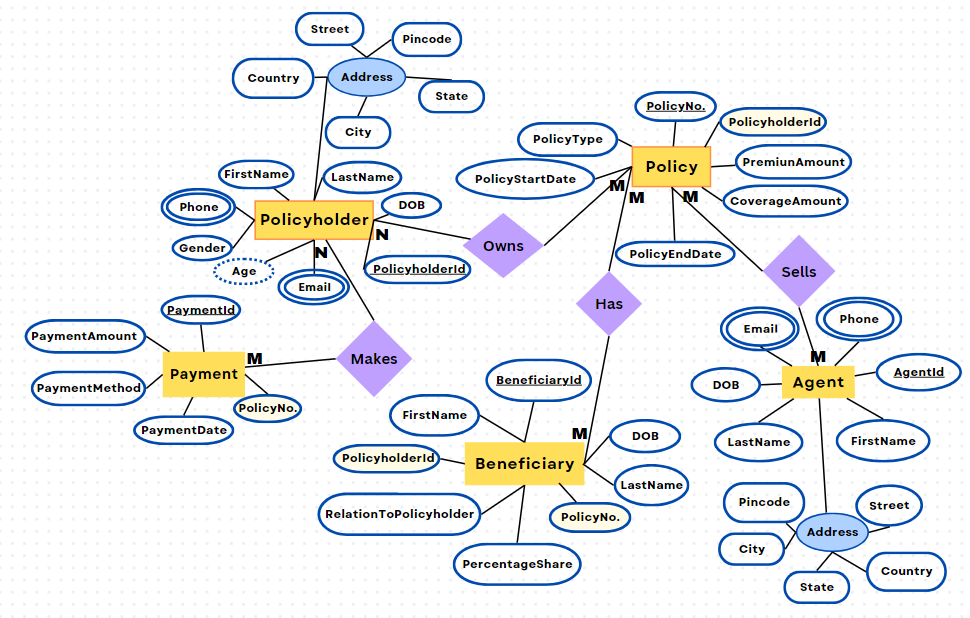
**Entities and Attributes:**

1. **Policyholder:**
   * PolicyholderID (Primary Key)
   * FirstName
   * LastName
   * DateOfBirth
   * Gender
   * Address
   * Email
   * Phone
2. **Policy:**
   * PolicyNumber (Primary Key)
   * PolicyType
   * PremiumAmount
   * CoverageAmount
   * PolicyStartDate
   * PolicyEndDate
   * PolicyholderID (Foreign Key)
3. **Agent:**
   * AgentID (Primary Key)
   * FirstName
   * LastName
   * DateOfBirth
   * Address
   * Email
   * Phone
4. **Beneficiary:**
   * BeneficiaryID (Primary Key)
   * PolicyholderID (Foreign Key)
   * FirstName
   * LastName
   * DateOfBirth
   * RelationshipToPolicyholder
   * PercentageShare
   * PolicyNumber (Foreign Key)
5. **Payment:**
   * PaymentID (Primary Key)
   * PolicyNumber (Foreign Key)
   * PaymentDate
   * PaymentAmount
   * PaymentMethod

**Relationships:**

1. **Policyholder owns Policy:**
   * Relationship Type: One-to-Many
   * Description: Each policyholder can own multiple policies, but each policy is owned by one policyholder.
2. **Agent sells Policy:**
   * Relationship Type: Many-to-Many
   * Description: An agent can sell multiple policies, and a policy can be sold by multiple agents. The relationship captures the history of agent-policy associations.
3. **Policy has Beneficiaries:**
   * Relationship Type: Many-to-Many
   * Description: A policy can have multiple beneficiaries, and a beneficiary can be associated with multiple policies. This allows for flexible beneficiary management.
4. **Policyholder makes Payment:**
   * Relationship Type: One-to-Many
   * Description: Each policyholder can make multiple payments, but each payment is associated with one policy.

**ER Diagram:**

****

**Tables: -**

**Policyholder table**

CREATE TABLE Policyholder (

PolicyholderID INT PRIMARY KEY,

FirstName VARCHAR(50),

LastName VARCHAR(50),

DateOfBirth DATE,

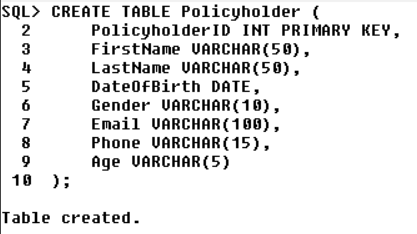
Gender VARCHAR(10),

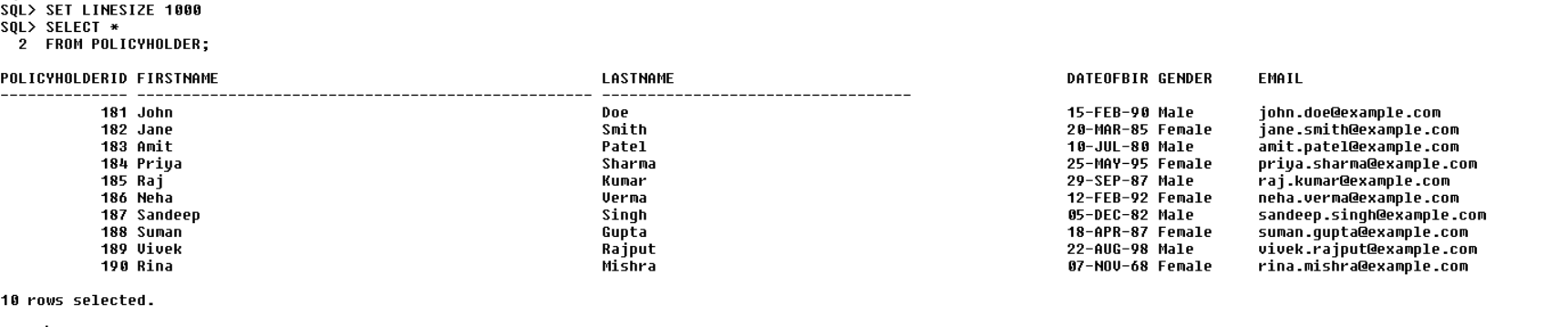
Email VARCHAR(100),

Phone VARCHAR(15),

Age VARCHAR(5)

);





**Policyholder Address Table**

CREATE TABLE PolicyholderAddress (

AddressID INT PRIMARY KEY,

PolicyholderID INT,

StreetAddress VARCHAR(255),

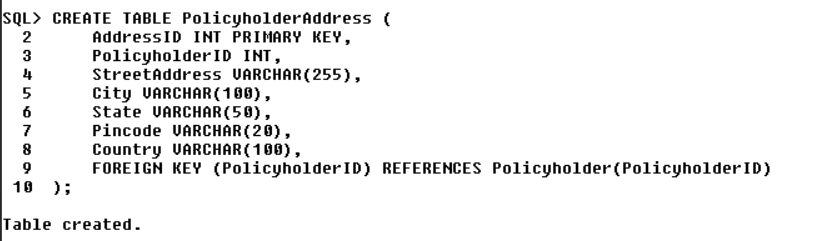
City VARCHAR(100),

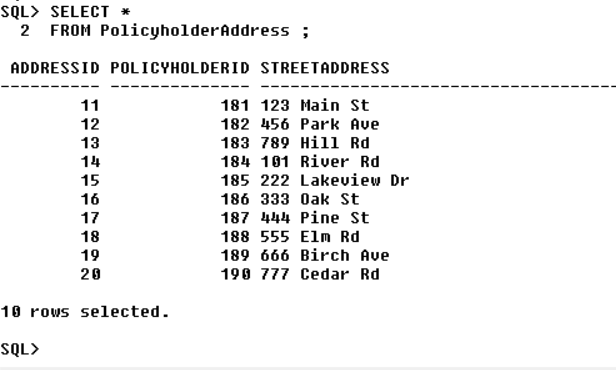
State VARCHAR(50),

Pincode VARCHAR(20),

Country VARCHAR(100),

FOREIGN KEY (PolicyholderID) REFERENCES Policyholder(PolicyholderID));



****

**Policy table**

CREATE TABLE Policyy (

PolicyNumber INT PRIMARY KEY,

PolicyholderID INT,

PolicyType VARCHAR(50),

PremiumAmount DECIMAL(10, 2),

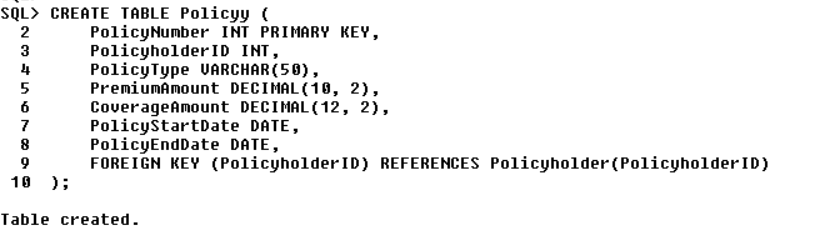
CoverageAmount DECIMAL(12, 2),

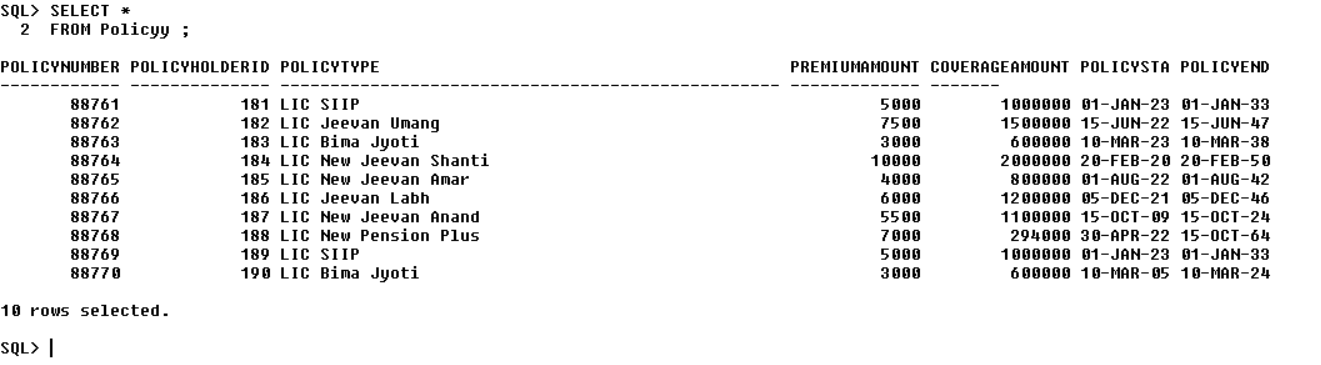
PolicyStartDate DATE,

PolicyEndDate DATE,

FOREIGN KEY (PolicyholderID) REFERENCES Policyholder(PolicyholderID)

);

****

****

**Agent table**

CREATE TABLE Agent (

AgentID INT PRIMARY KEY,

FirstName VARCHAR(50),

LastName VARCHAR(50),

DateOfBirth DATE,

Email VARCHAR(100),

Phone VARCHAR(15)

);





**Agent Address Table**

CREATE TABLE AgentAddress (

AddressID INT PRIMARY KEY,

AgentID INT,

StreetAddress VARCHAR(255),

City VARCHAR(100),

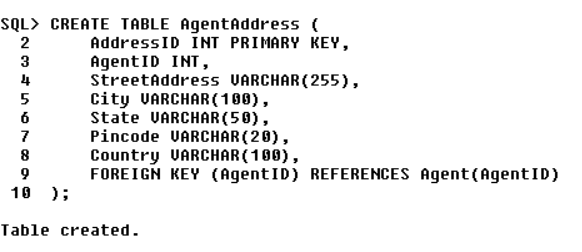
State VARCHAR(50),

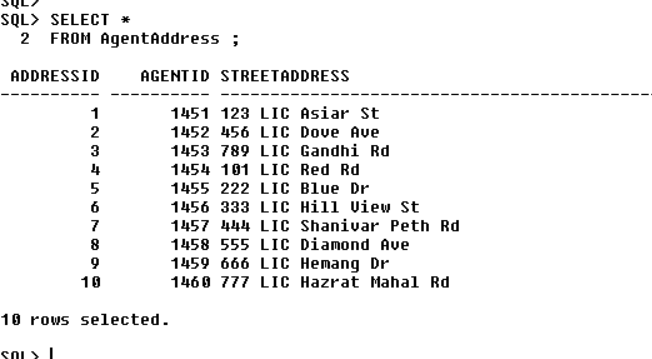
Pincode VARCHAR(20),

Country VARCHAR(100),

FOREIGN KEY (AgentID) REFERENCES Agent(AgentID)

);





**Beneficiary table**

CREATE TABLE Beneficiary (

BeneficiaryID INT PRIMARY KEY,

PolicyholderID INT,

FirstName VARCHAR(50),

LastName VARCHAR(50),

DateOfBirth DATE,

RelationshipToPolicyholder VARCHAR(50),

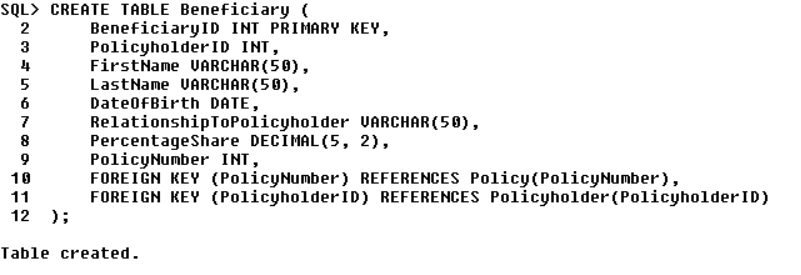
PercentageShare DECIMAL(5, 2),

PolicyNumber INT,

FOREIGN KEY (PolicyNumber) REFERENCES Policy(PolicyNumber),

FOREIGN KEY (PolicyholderID) REFERENCES Policyholder(PolicyholderID)

);

****

**Payment table**

CREATE TABLE Payment (

PaymentID INT PRIMARY KEY,

PolicyNumber INT,

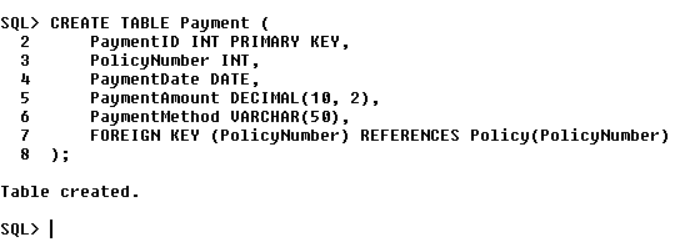
PaymentDate DATE,

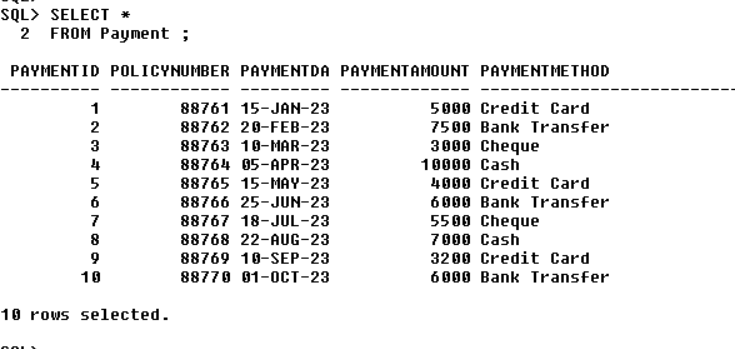
PaymentAmount DECIMAL(10, 2),

PaymentMethod VARCHAR(50),

FOREIGN KEY (PolicyNumber) REFERENCES Policy(PolicyNumber)

);





**Values: -**

**Insert data into the Policyholder table**

INSERT INTO Policyholder VALUES (181, 'John', 'Doe','15-Feb-1990', 'Male', 'john.doe@example.com', '+91-1234567890', '33');

INSERT INTO Policyholder VALUES (182, 'Jane', 'Smith', '20-March-1985', 'Female', 'jane.smith@example.com', 9876543210, 38);

INSERT INTO Policyholder VALUES (183, 'Amit', 'Patel', '10-Jul-1980', 'Male', 'amit.patel@example.com', 7890123456, 43);

INSERT INTO Policyholder VALUES (184, 'Priya', 'Sharma', '25-May-1995', 'Female', 'priya.sharma@example.com', 8765432109, 28);

INSERT INTO Policyholder VALUES (185, 'Raj', 'Kumar', '29-Sep -1987', 'Male', 'raj.kumar@example.com', 9988776655, 33);

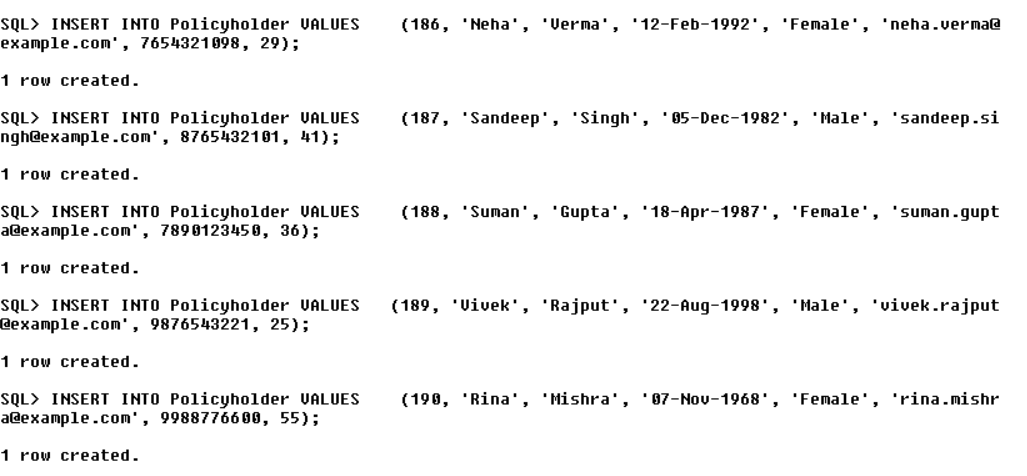
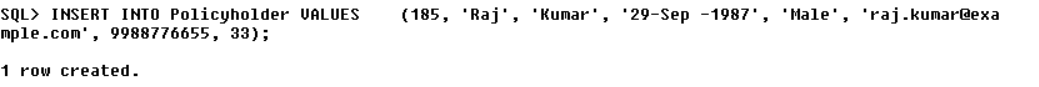
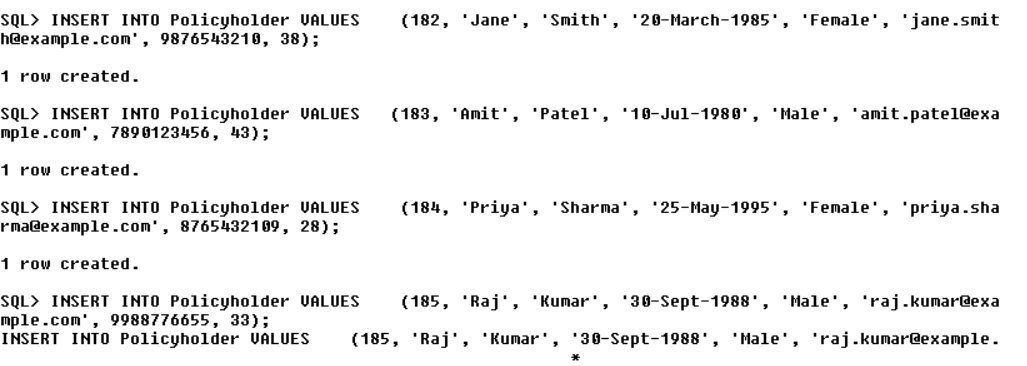
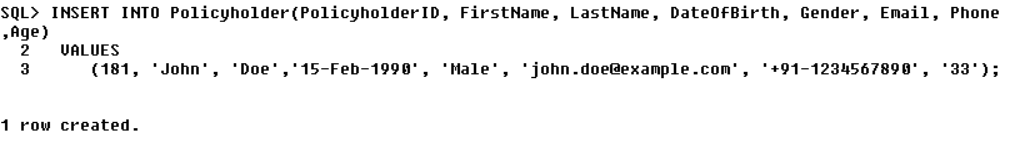
INSERT INTO Policyholder VALUES (186, 'Neha', 'Verma', '12-Feb-1992', 'Female', 'neha.verma@example.com', 7654321098, 29);

INSERT INTO Policyholder VALUES (187, 'Sandeep', 'Singh', '05-Dec-1982', 'Male', 'sandeep.singh@example.com', 8765432101, 41);

INSERT INTO Policyholder VALUES (188, 'Suman', 'Gupta', '18-Apr-1987', 'Female', 'suman.gupta@example.com', 7890123450, 36);

INSERT INTO Policyholder VALUES (189, 'Vivek', 'Rajput', '22-Aug-1998', 'Male', 'vivek.rajput@example.com', 9876543221, 25);

INSERT INTO Policyholder VALUES (190, 'Rina', 'Mishra', '07-Nov-1968', 'Female', 'rina.mishra@example.com', 9988776600, 55);



**Insert data into the PolicyholderAddress table**

INSERT INTO PolicyholderAddress VALUES (11, 181, '123 Main St', 'Mumbai', 'Maharashtra', 400001, 'India');

INSERT INTO PolicyholderAddress VALUES (12, 182, '456 Park Ave', 'New Delhi', 'Delhi', 110001, 'India');

INSERT INTO PolicyholderAddress VALUES (13, 183, '789 Hill Rd', 'Bangalore', 'Karnataka', 560001, 'India');

INSERT INTO PolicyholderAddress VALUES (14, 184, '101 River Rd', 'Kolkata', 'West Bengal', 700001, 'India');

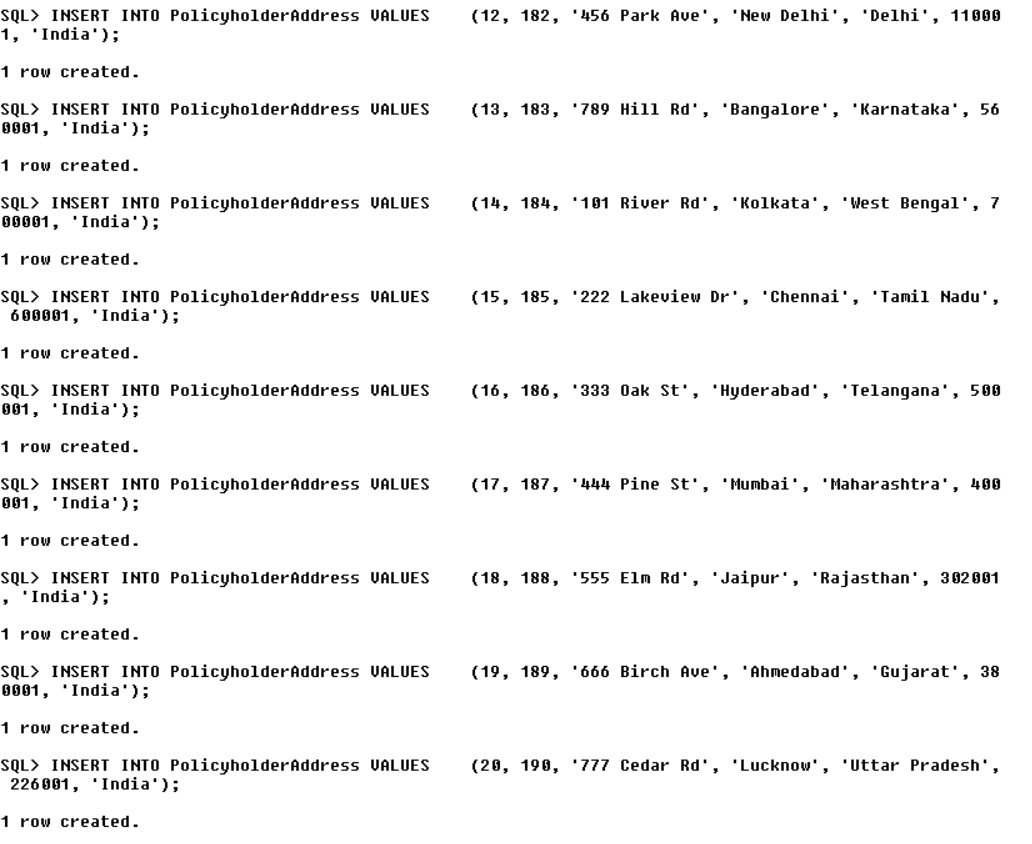
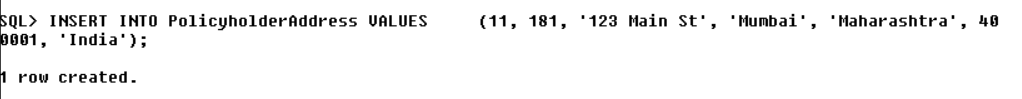
INSERT INTO PolicyholderAddress VALUES (15, 185, '222 Lakeview Dr', 'Chennai', 'Tamil Nadu', 600001, 'India');

INSERT INTO PolicyholderAddress VALUES (16, 186, '333 Oak St', 'Hyderabad', 'Telangana', 500001, 'India');

INSERT INTO PolicyholderAddress VALUES (17, 187, '444 Pine St', 'Mumbai', 'Maharashtra', 400001, 'India');

INSERT INTO PolicyholderAddress VALUES (18, 188, '555 Elm Rd', 'Jaipur', 'Rajasthan', 302001, 'India');

INSERT INTO PolicyholderAddress VALUES (19, 189, '666 Birch Ave', 'Ahmedabad', 'Gujarat', 380001, 'India');

INSERT INTO PolicyholderAddress VALUES (20, 190, '777 Cedar Rd', 'Lucknow', 'Uttar Pradesh', 226001, 'India');

**Insert data into the Policy table**

INSERT INTO Policyy VALUES (88761, 181, 'LIC SIIP', 5000.00, 1000000.00, '01-Jan-2023', '01-Jan-2033');

INSERT INTO Policyy VALUES (88762, 182, 'LIC Jeevan Umang', 7500.00, 1500000.00, '15-Jun-2022', '15-Jun-2047');

INSERT INTO Policyy VALUES (88763, 183, 'LIC Bima Jyoti', 3000.00, 600000.00, '10-March-2023', '10-March-2038');

INSERT INTO Policyy VALUES (88764, 184, 'LIC New Jeevan Shanti', 10000.00, 2000000.00, '20-Feb-2020', '20-Feb-2050');

INSERT INTO Policyy VALUES (88765, 185, 'LIC New Jeevan Amar', 4000.00, 800000.00, '01-Aug-2022', '01-Aug-2042');

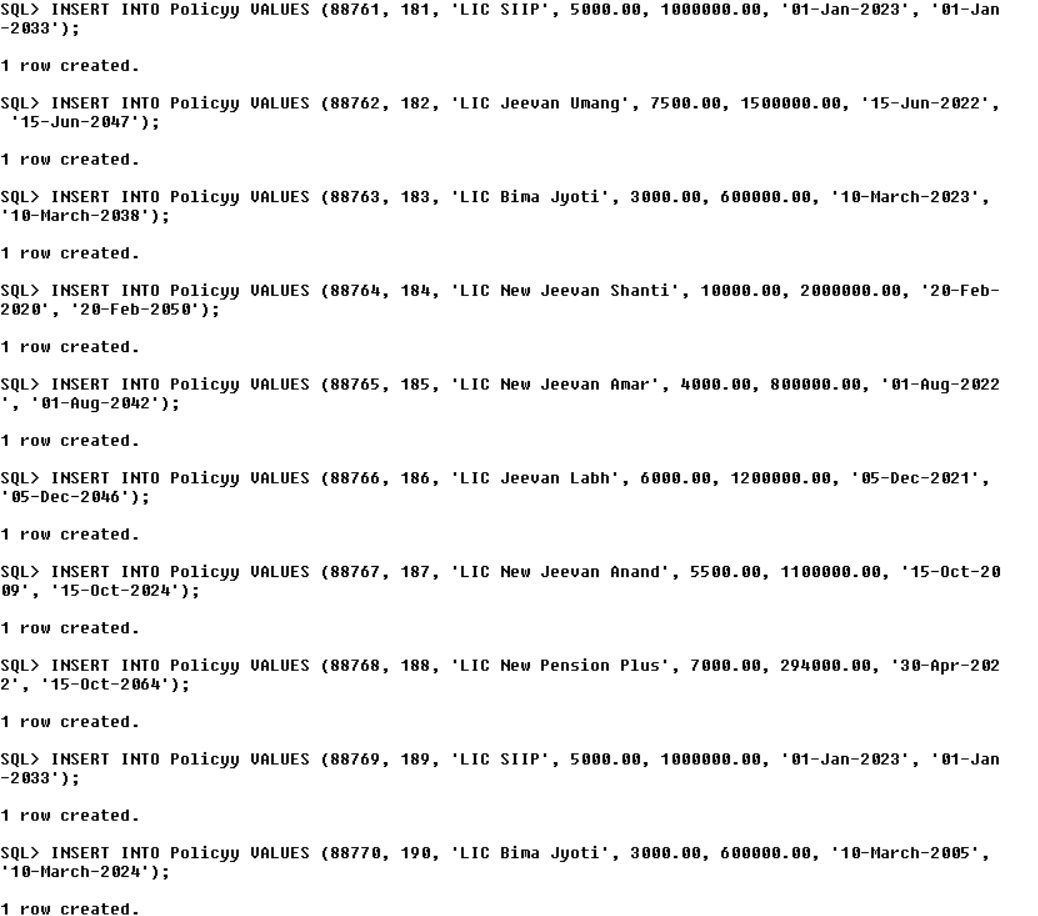
INSERT INTO Policyy VALUES (88766, 186, 'LIC Jeevan Labh', 6000.00, 1200000.00, '05-Dec-2021', '05-Dec-2046');

INSERT INTO Policyy VALUES (88767, 187, 'LIC New Jeevan Anand', 5500.00, 1100000.00, '15-Oct-2009', '15-Oct-2024');

INSERT INTO Policyy VALUES (88768, 188, 'LIC New Pension Plus', 7000.00, 294000.00, '30-Apr-2022', '15-Oct-2064');

INSERT INTO Policyy VALUES (88769, 189, 'LIC SIIP', 5000.00, 1000000.00, '01-Jan-2023', '01-Jan-2033');

INSERT INTO Policyy VALUES (88770, 190, 'LIC Bima Jyoti', 3000.00, 600000.00, '10-March-2005', '10-March-2024');

****

**Insert data into the Agent table**

INSERT INTO Agent VALUES (1451, 'Amit', 'Kumar', '10-Feb-1980', 'amit.kumar@lic.com', '+91-9876543210');

INSERT INTO Agent VALUES (1452, 'Sneha', 'Verma', '15-May-1985', 'sneha.verma@lic.com', '+91-8765432109');

INSERT INTO Agent VALUES (1453, 'Rajesh', 'Sharma', '20-Aug-1990', 'rajesh.sharma@lic.com', '+91-7890123456');

INSERT INTO Agent VALUES (1454, 'Neha', 'Singh', '25-Nov-1982', 'neha.singh@lic.com', '+91-9988776655');

INSERT INTO Agent VALUES (1455, 'Anil', 'Gupta', '30-Sep-1978', 'anil.gupta@lic.com', '+91-7654321098');

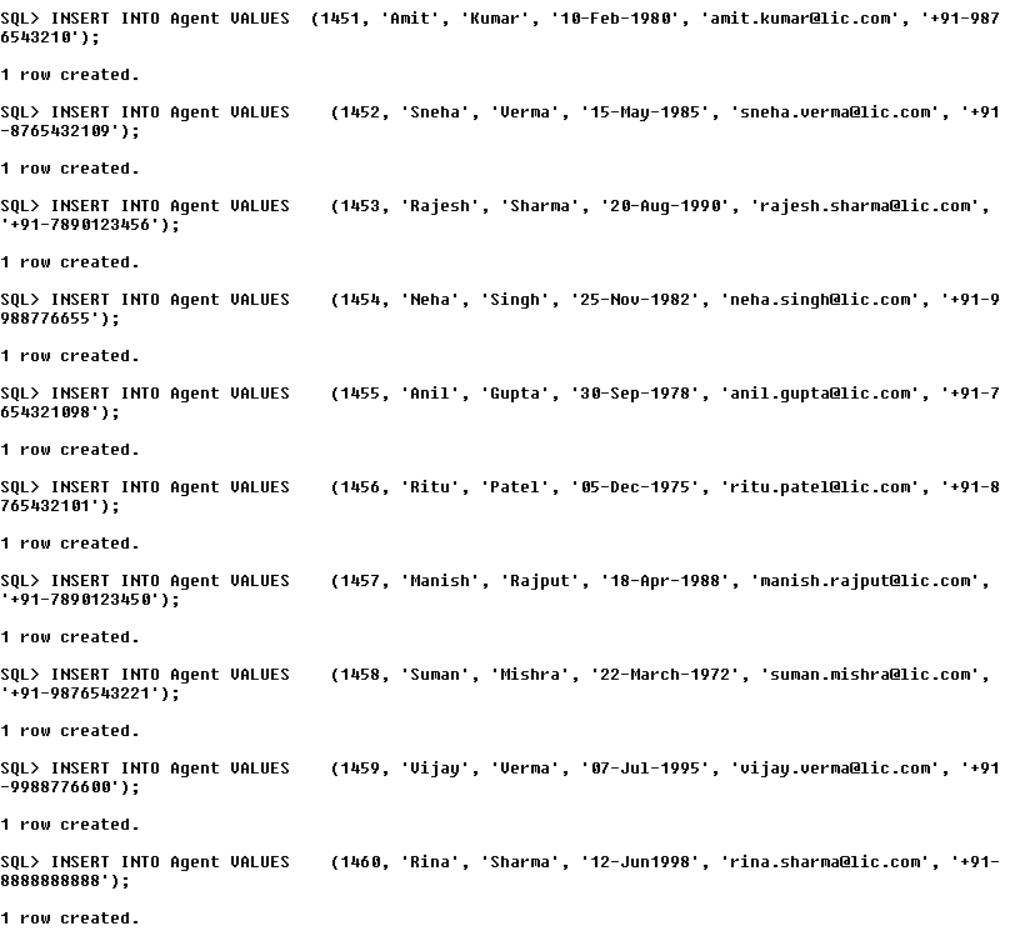
INSERT INTO Agent VALUES (1456, 'Ritu', 'Patel', '05-Dec-1975', 'ritu.patel@lic.com', '+91-8765432101');

INSERT INTO Agent VALUES (1457, 'Manish', 'Rajput', '18-Apr-1988', 'manish.rajput@lic.com', '+91-7890123450');

INSERT INTO Agent VALUES (1458, 'Suman', 'Mishra', '22-March-1972', 'suman.mishra@lic.com', '+91-9876543221');

INSERT INTO Agent VALUES (1459, 'Vijay', 'Verma', '07-Jul-1995', 'vijay.verma@lic.com', '+91-9988776600');

INSERT INTO Agent VALUES (1460, 'Rina', 'Sharma', '12-Jun1998', 'rina.sharma@lic.com', '+91-8888888888');



**Insert data into the AgentAddress table**

INSERT INTO AgentAddress VALUES (1, 1451, '123 LIC Asiar St', 'Mumbai', 'Maharashtra', 400001, 'India');

INSERT INTO AgentAddress VALUES (2, 1452, '456 LIC Dove Ave', 'New Delhi', 'Delhi', 110001, 'India');

INSERT INTO AgentAddress VALUES (3, 1453, '789 LIC Gandhi Rd', 'Bangalore', 'Karnataka', 560001, 'India');

INSERT INTO AgentAddress VALUES (4, 1454, '101 LIC Red Rd', 'Kolkata', 'West Bengal', 700001, 'India');

INSERT INTO AgentAddress VALUES (5, 1455, '222 LIC Blue Dr', 'Chennai', 'Tamil Nadu', 600001, 'India');

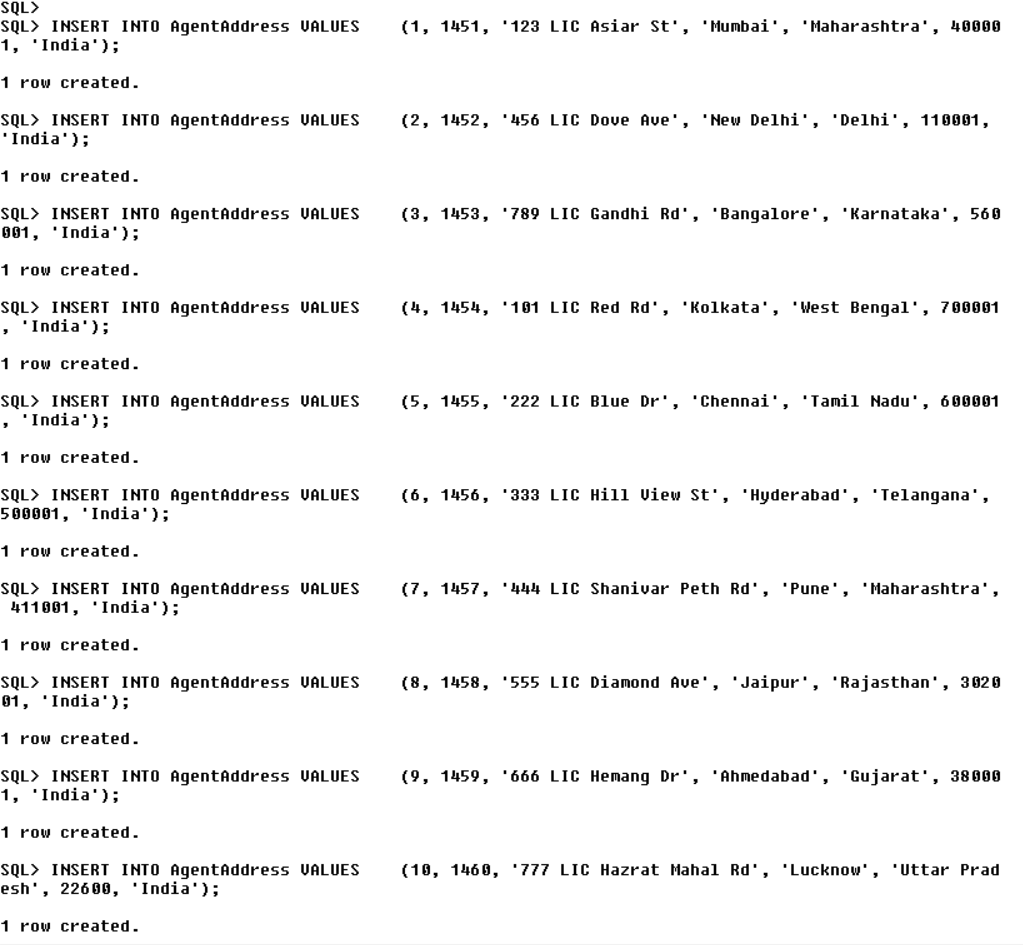
INSERT INTO AgentAddress VALUES (6, 1456, '333 LIC Hill View St', 'Hyderabad', 'Telangana', 500001, 'India');

INSERT INTO AgentAddress VALUES (7, 1457, '444 LIC Shanivar Peth Rd', 'Pune', 'Maharashtra', 411001, 'India');

INSERT INTO AgentAddress VALUES (8, 1458, '555 LIC Diamond Ave', 'Jaipur', 'Rajasthan', 302001, 'India');

INSERT INTO AgentAddress VALUES (9, 1459, '666 LIC Hemang Dr', 'Ahmedabad', 'Gujarat', 380001, 'India');

INSERT INTO AgentAddress VALUES (10, 1460, '777 LIC Hazrat Mahal Rd', 'Lucknow', 'Uttar Pradesh', 22600, 'India');



**Insert data into the Beneficiaryy table for LIC company**

INSERT INTO Beneficiary VALUES (8841, 181, 'Aarav', 'Sharma', '15-Feb-1995', 'Spouse', 50.00, 88761);

INSERT INTO Beneficiary VALUES (8842, 182, 'Aditi', 'Verma', '20-May-1990', 'Child', 25.00, 88762);

INSERT INTO Beneficiary VALUES (8843, 183, 'Ramesh', 'Patel', '10-Aug-1982', 'Parent', 10.00, 88763);

INSERT INTO Beneficiary VALUES (8844, 184, 'Suman', 'Gupta', '25-Nov-1978', 'Sibling', 5.00, 88764);

INSERT INTO Beneficiary VALUES (8845, 185, 'Priya', 'Singh', '30-Sep-1993', 'Child', 5.00, 88765);

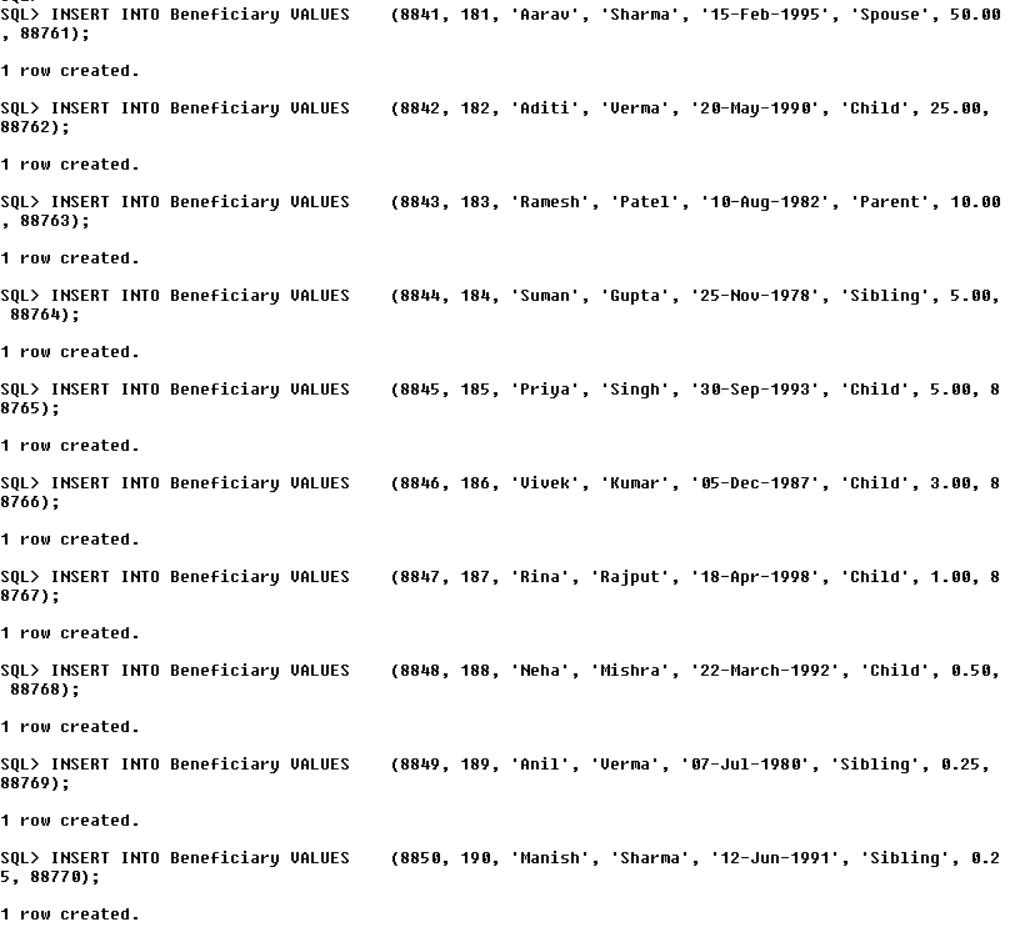
INSERT INTO Beneficiary VALUES (8846, 186, 'Vivek', 'Kumar', '05-Dec-1987', 'Child', 3.00, 88766);

INSERT INTO Beneficiary VALUES (8847, 187, 'Rina', 'Rajput', '18-Apr-1998', 'Child', 1.00, 88767);

INSERT INTO Beneficiary VALUES (8848, 188, 'Neha', 'Mishra', '22-March-1992', 'Child', 0.50, 88768);

INSERT INTO Beneficiary VALUES (8849, 189, 'Anil', 'Verma', '07-Jul-1980', 'Sibling', 0.25, 88769);

INSERT INTO Beneficiary VALUES (8850, 190, 'Manish', 'Sharma', '12-Jun-1991', 'Sibling', 0.25, 88770);

****

**Insert data into the Payment table**

INSERT INTO Payment VALUES (1, 88761, '15-Jan-2023', 5000.00, 'Credit Card');

INSERT INTO Payment VALUES (2, 88762, '20-Feb-2023', 7500.00, 'Bank Transfer');

INSERT INTO Payment VALUES (3, 88763, '10-March-2023', 3000.00, 'Cheque');

INSERT INTO Payment VALUES (4, 88764, '05-Apr-2023', 10000.00, 'Cash');

INSERT INTO Payment VALUES (5, 88765, '15-May-2023', 4000.00, 'Credit Card');

INSERT INTO Payment VALUES (6, 88766, '25-Jun-2023', 6000.00, 'Bank Transfer');

INSERT INTO Payment VALUES (7, 88767, '18-Jul-2023', 5500.00, 'Cheque');

INSERT INTO Payment VALUES (8, 88768, '22-Aug-2023', 7000.00, 'Cash');

INSERT INTO Payment VALUES (9, 88769, '10-Sep-2023', 3200.00, 'Credit Card');

INSERT INTO Payment VALUES (10, 88770, '01-Oct-2023', 6000.00, 'Bank Transfer');



**Relationship Tables: -**

**1. Policyholder owns Policy**

CREATE TABLE Owns (

PolicyholderID INT,

PolicyNumber INT,

FOREIGN KEY (PolicyholderID) REFERENCES Policyholder(PolicyholderID),

FOREIGN KEY (PolicyNumber) REFERENCES Policy(PolicyNumber)

);

INSERT INTO owns VALUES (181, 88761);

INSERT INTO owns VALUES (182, 88762);

INSERT INTO owns VALUES (183, 88763);

INSERT INTO owns VALUES (184, 88764);

INSERT INTO owns VALUES (185, 88765);

INSERT INTO owns VALUES (186, 88766);

INSERT INTO owns VALUES (187, 88767);

INSERT INTO owns VALUES (188, 88768);

INSERT INTO owns VALUES (189, 88769);

INSERT INTO owns VALUES (190, 88770);

**2. Agent sells Policy**

CREATE TABLE sells (

AgentID INT,

PolicyNumber INT,

FOREIGN KEY (AgentID) REFERENCES Agent(AgentID),

FOREIGN KEY (PolicyNumber) REFERENCES Policy(PolicyNumber)

);

INSERT INTO sells VALUES(1451, 88761);

INSERT INTO sells VALUES(1452, 88762);

INSERT INTO sells VALUES(1453, 88763);

INSERT INTO sells VALUES(1454, 88764);

INSERT INTO sells VALUES(1455, 88765);

INSERT INTO sells VALUES(1456, 88766);

INSERT INTO sells VALUES(1457, 88767);

INSERT INTO sells VALUES(1458, 88768);

INSERT INTO sells VALUES(1459, 88769);

INSERT INTO sells VALUES(1460, 88770);

**3. Policy has Beneficiaries**

CREATE TABLE has (

BeneficiaryID INT,

PolicyNumber INT,

FOREIGN KEY (BeneficiaryID) REFERENCES Beneficiary(BeneficiaryID),

FOREIGN KEY (PolicyNumber) REFERENCES Policy(PolicyNumber)

);

INSERT INTO has VALUES (8841, 88761);

INSERT INTO has VALUES (8842, 88762);

INSERT INTO has VALUES (8843, 88763);

INSERT INTO has VALUES (8844, 88764);

INSERT INTO has VALUES (8845, 88765);

INSERT INTO has VALUES (8846, 88766);

INSERT INTO has VALUES (8847, 88767);

INSERT INTO has VALUES (8848, 88768);

INSERT INTO has VALUES (8848, 88769);

INSERT INTO has VALUES (8850, 88770);

**4. Policyholder makes Payment:**

CREATE TABLE Makes (

PolicyholderID INT,

PaymentID INT,

FOREIGN KEY (PolicyholderID) REFERENCES Policyholder(PolicyholderID),

FOREIGN KEY (PaymentID) REFERENCES Payment(PaymentID)

);

INSERT INTO makes VALUES (181, 1);

INSERT INTO makes VALUES (182, 2);

INSERT INTO makes VALUES (183, 3);

INSERT INTO makes VALUES (184, 4);

INSERT INTO makes VALUES (185, 5);

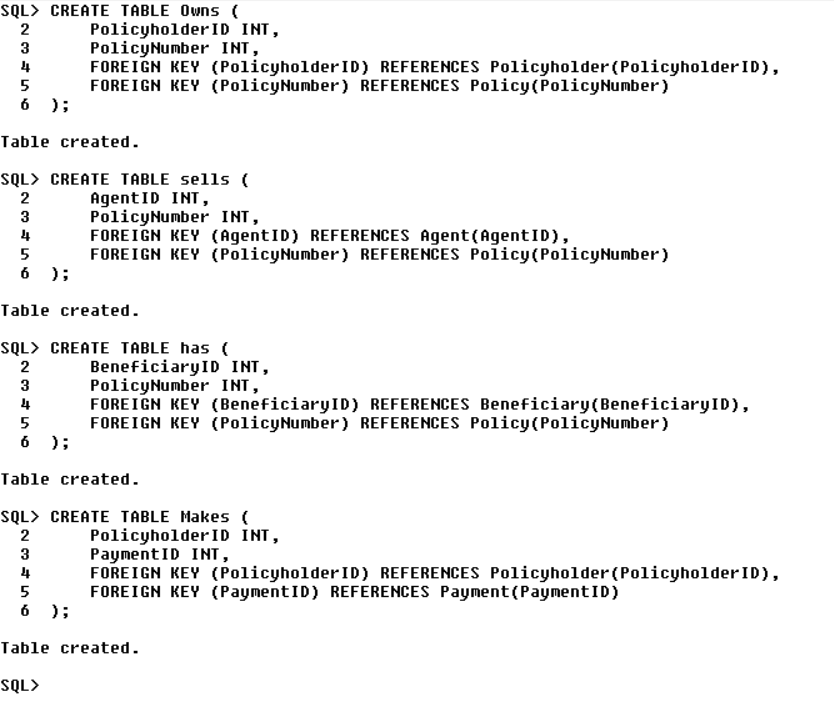
INSERT INTO makes VALUES (186, 6);

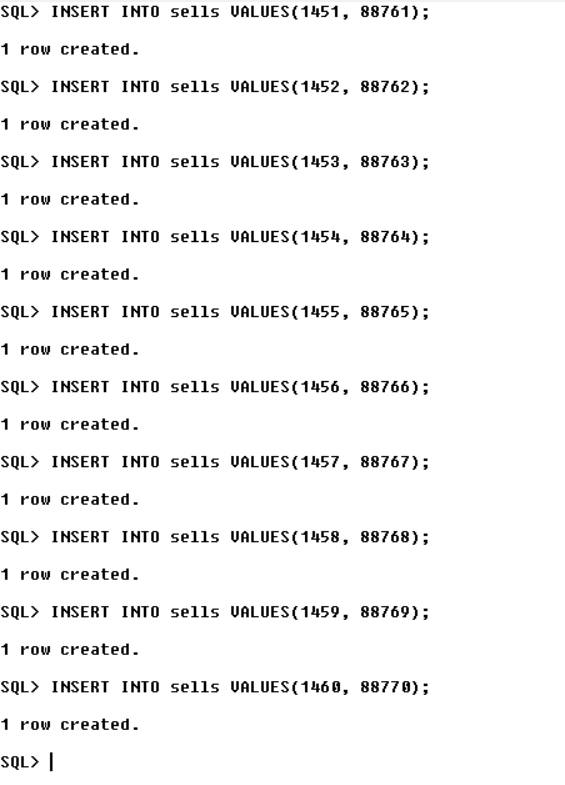
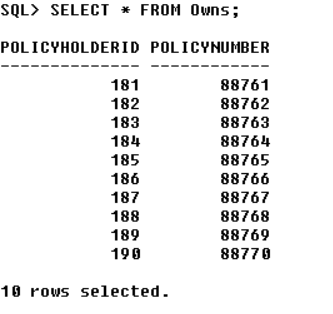
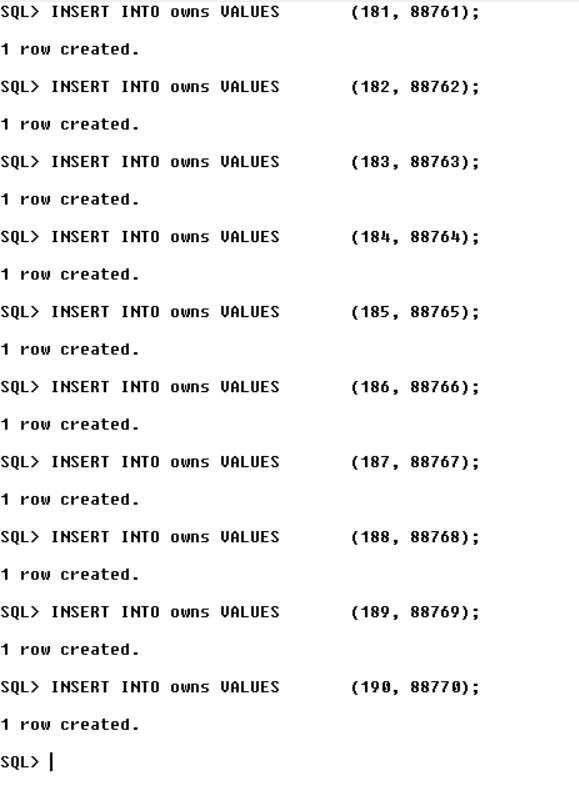
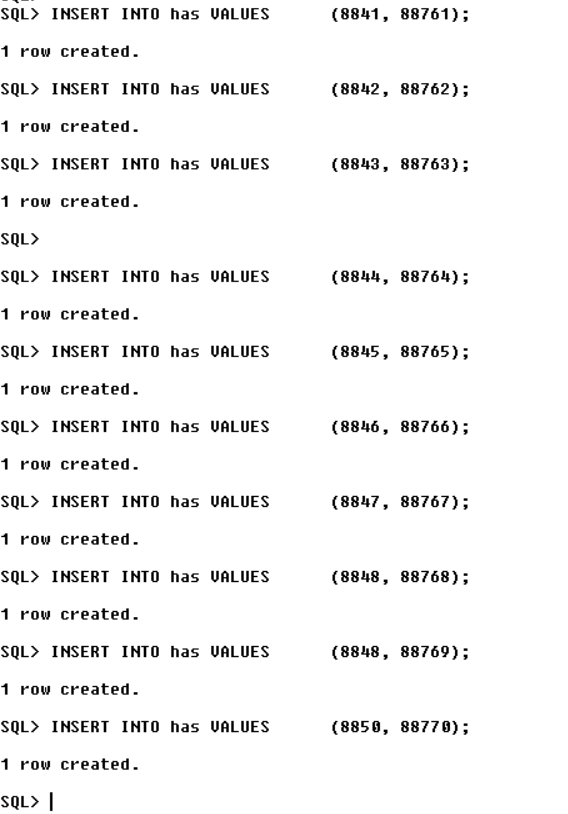
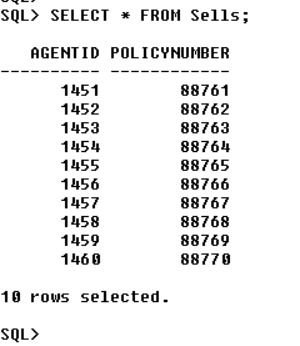
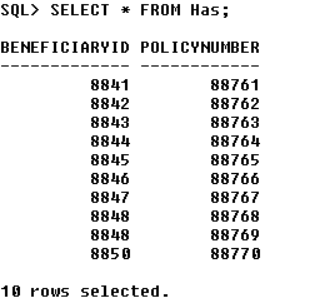
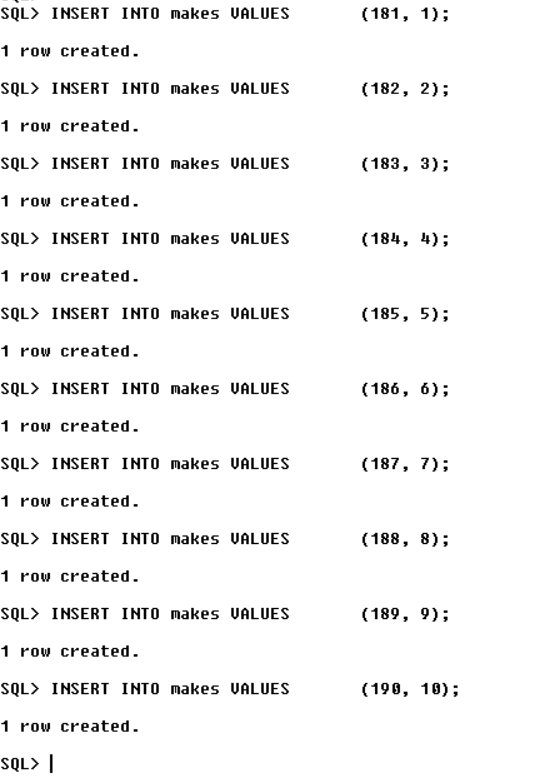
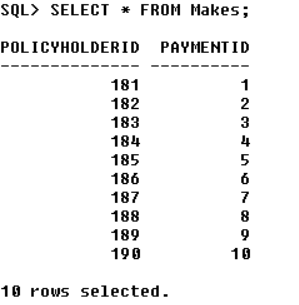
INSERT INTO makes VALUES (187, 7);

INSERT INTO makes VALUES (188, 8);

INSERT INTO makes VALUES (189, 9);

INSERT INTO makes VALUES (190, 10);

****

**** **** **** 

**Queries: -**

**1**. **Query**: List Beneficiaries and Their Relationship for a Specific Policyholder:

**Question**: List the beneficiaries and their relationships for a specific policyholder, given the PolicyholderID.

SET LINESIZE 200

SET PAGESIZE 100

COLUMN Policyholder\_FirstName FORMAT A20

COLUMN Policyholder\_LastName FORMAT A20

COLUMN Beneficiaryy\_FirstName FORMAT A20

COLUMN Beneficiaryy\_LastName FORMAT A20

COLUMN RelationshipToPolicyholder FORMAT A20

SELECT

ph.FirstName AS Policyholder\_FirstName,

ph.LastName AS Policyholder\_LastName,

b.FirstName AS Beneficiaryy\_FirstName,

b.LastName AS Beneficiaryy\_LastName,

b.RelationshipToPolicyholder

FROM

Policyholder ph

INNER JOIN

Beneficiary b ON ph.PolicyholderID = b.PolicyholderID

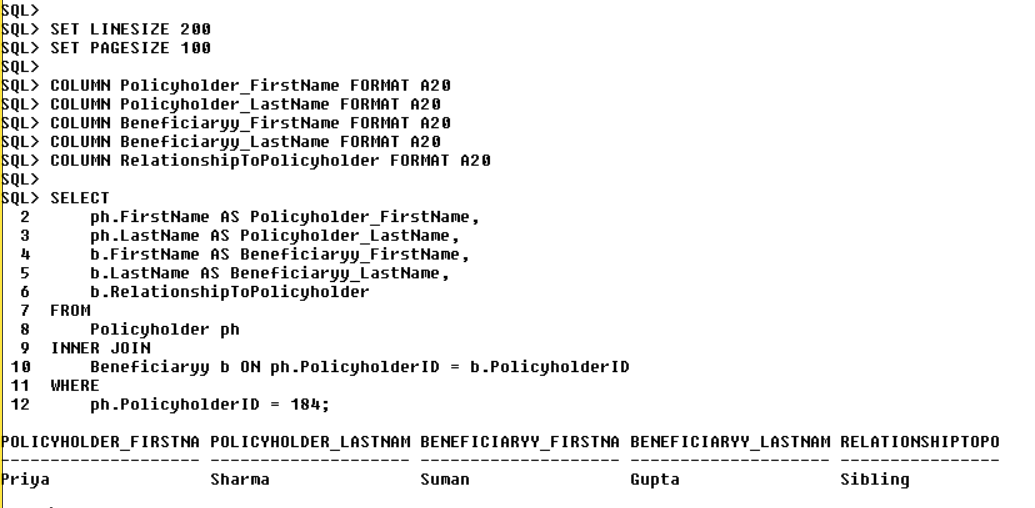
WHERE

ph.PolicyholderID = 184;

**Explanation:**

This query retrieves the beneficiaries and their relationships for a specific policyholder.

This query helps identify who will receive the benefits in the event of a claim for a specific policyholder.



**2**. **Query**: Policy with Highest Premium

**Question**: Find the policy with the highest premium amount and display its details.

SELECT \*

FROM (

SELECT

PolicyNumber,

PolicyType,

PremiumAmount

FROM

Policyy

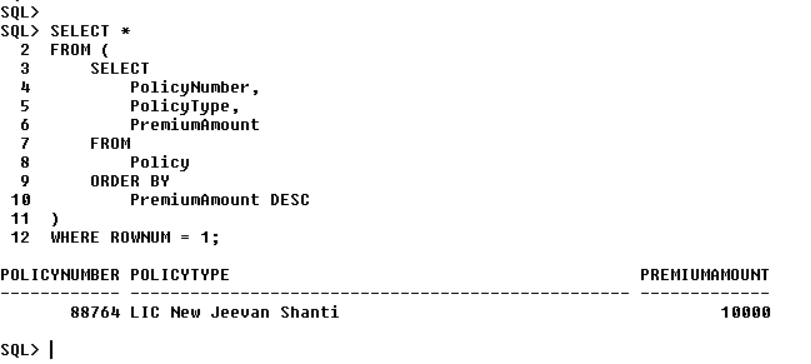
ORDER BY

PremiumAmount DESC

)

WHERE ROWNUM = 1;

**Explanation**: This query identifies the most expensive policy in terms of premium which can be useful for analysis or management decisions.



**3**. **Query**: Beneficiary Share Calculation

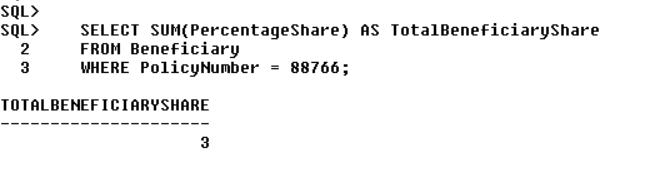
**Question**: Calculate the total percentage share assigned to beneficiaries for a specific policy.

SELECT SUM(PercentageShare) AS TotalBeneficiaryShare

FROM Beneficiary

WHERE PolicyNumber = 88766;

**Explanation**: This query helps determine the total beneficiary share for a particular policy.



**4**. **Query**: Payment History for a Policy.

**Question**: Retrieve the payment history (dates and amounts) for a specific policy using its PolicyNumber.

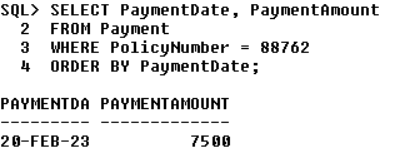
SELECT PaymentDate, PaymentAmount

FROM Payment

WHERE PolicyNumber = 88762

ORDER BY PaymentDate;

**Explanation**: This query provides a detailed payment history for a specific policy, which can be useful for tracking payments.



**5**. **Query**: Retrieve Policyholders and their Policy Details

**Question**: Provide a list of policyholders and their respective policies, including policy numbers, types, start dates, and end dates.

SELECT

P.FirstName,

P.LastName,

P.DateOfBirth,

P.Email,

Po.PolicyNumber,

Po.PolicyType,

Po.PolicyStartDate,

Po.PolicyEndDate

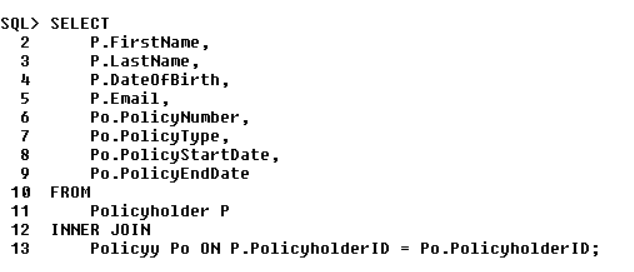
FROM

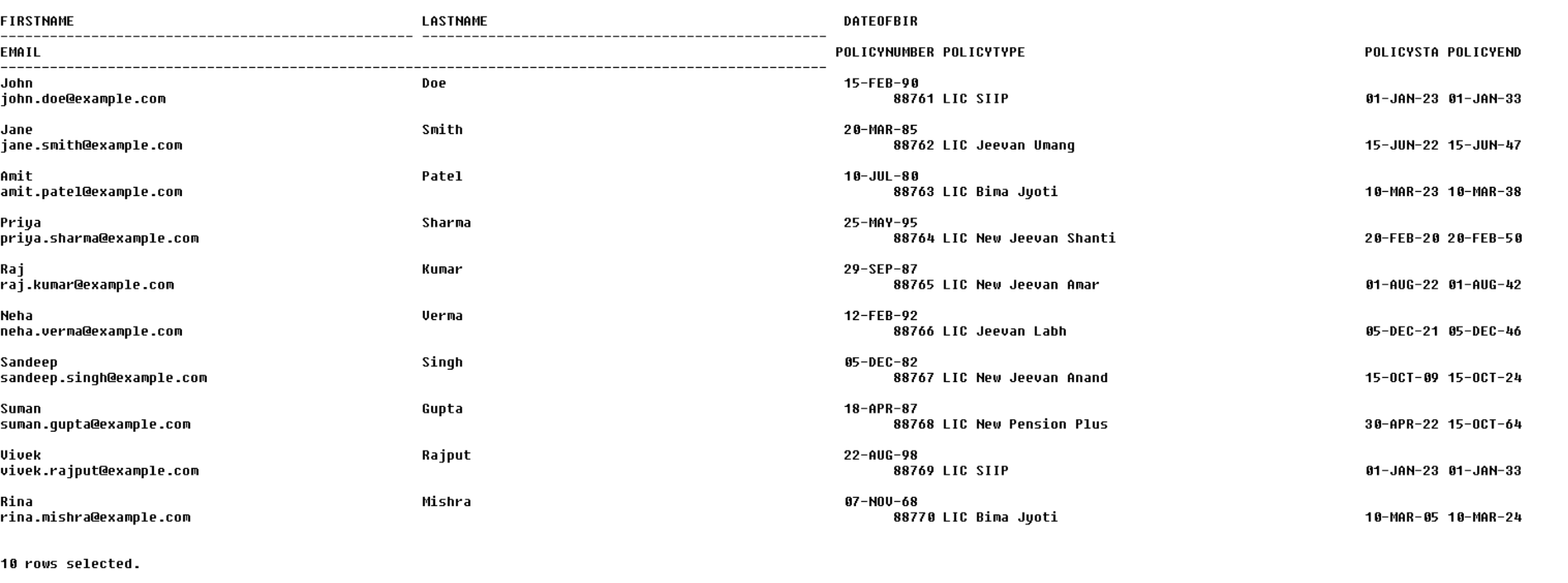
Policyholder P

INNER JOIN

Policyy Po ON P.PolicyholderID = Po.PolicyholderID;

**Explanation**: This query retrieves a list of policyholders along with their policy details, including policy number, type, start date, and end date.





**6**. **Query**: List Policies with Premium Amount Greater Than a Threshold

**Question**: Give a list of policies with premium amounts exceeding 5000.00.

SELECT

PolicyNumber,

PolicyType,

PremiumAmount

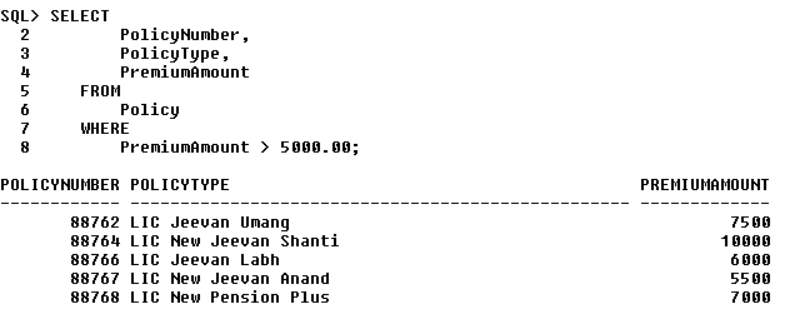
FROM

Policyy

WHERE

PremiumAmount > 5000.00;

**Explanation**: This query retrieves policies with a premium amount greater than 5000.00



**7**. **Query**: List Policyholders in a Specific City

**Question**: Provide a list of policyholders who reside in Mumbai, including their names, addresses, and the city they live in.

SELECT

FirstName,

LastName,

City

FROM

Policyholder P

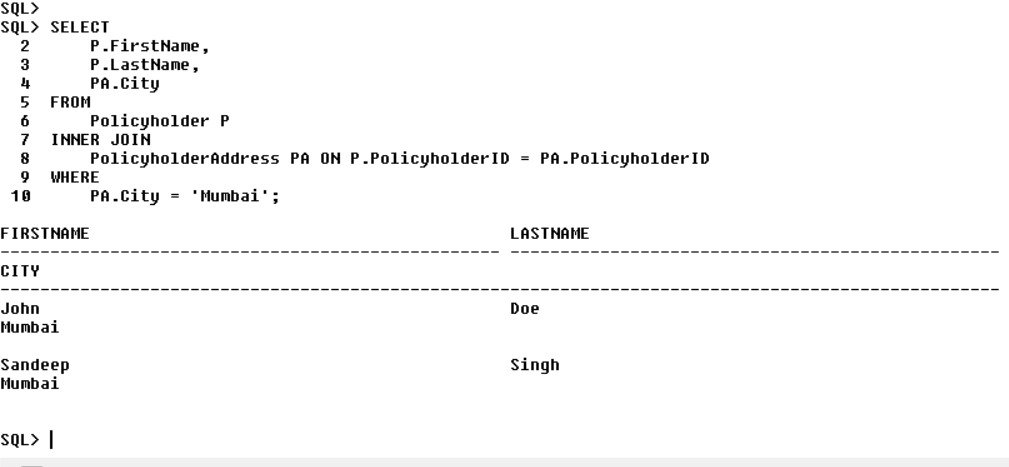
INNER JOIN

PolicyholderAddress PA ON P.PolicyholderID = PA.PolicyholderID

WHERE

City = 'Mumbai';

**Explanation**: This query retrieves policyholders who live in the city of Mumbai.



**8**. **Query**: Retrieve Policyholders with Age Greater Than 35.

**Question**: Give a list of policyholders whose age exceeds 35 years

SELECT

FirstName,

LastName,

Age

FROM

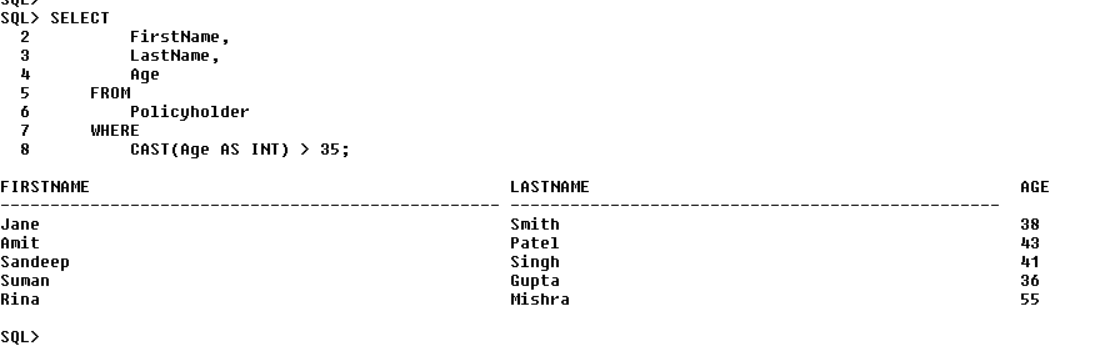
Policyholder

WHERE

CAST(Age AS INT) > 35;

**Explanation**: This query retrieves policyholders whose age is greater than 35 years (casting the Age column to an integer for comparison).

These queries can help an LIC company manage its policies, track payments, and gain insights into their policyholders and agents.



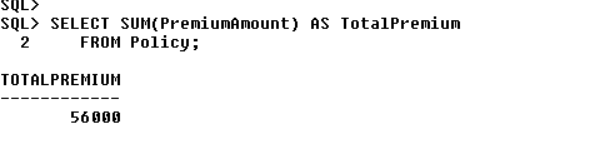
**9**. **Query**: Calculate Total Premium Collection

**Question**: What is the total premium amount that LIC has collected for all its policies?

SELECT SUM(PremiumAmount) AS TotalPremium

FROM Policyy;

**Explanation**: This query calculates the total premium amount collected by summing up the premium amounts from all policies.



**10**. **Query**: Find Policies with Expiry Date Within a Year.

**Question:** Identify policies that are set to expire within the next year.

SELECT

PolicyNumber,

PolicyType,

PolicyEndDate

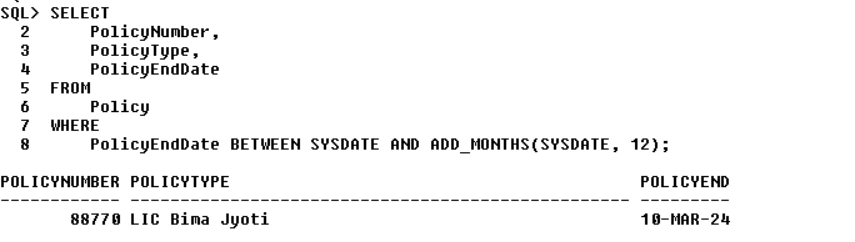
FROM

Policyy

WHERE

PolicyEndDate BETWEEN SYSDATE AND ADD\_MONTHS(SYSDATE, 12);

**Explanation**: This query identifies policies that are set to expire within the next year.



**Functions: -**

**Function**: Get Policyholder Information

**Question**: Provide the full name, date of birth, email, and phone number of the policyholder with PolicyholderID 188.

CREATE OR REPLACE FUNCTION GetPolicyholderInformation(

policyholderID IN NUMBER

)

RETURN SYS\_REFCURSOR

IS

cur SYS\_REFCURSOR;

BEGIN

OPEN cur FOR

SELECT FirstName || ' ' || LastName AS FullName,

DateOfBirth,

Email,

Phone

FROM Policyholder

WHERE PolicyholderID = policyholderID;

RETURN cur;

END GetPolicyholderInformation;

/

Procedure for the function: -

VAR result REFCURSOR;

BEGIN

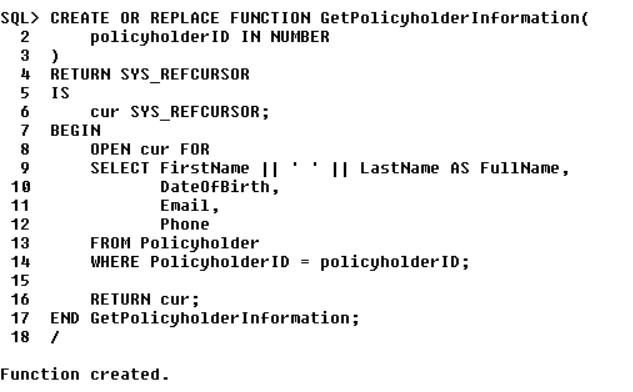
:result := GetPolicyholderInformation(188);

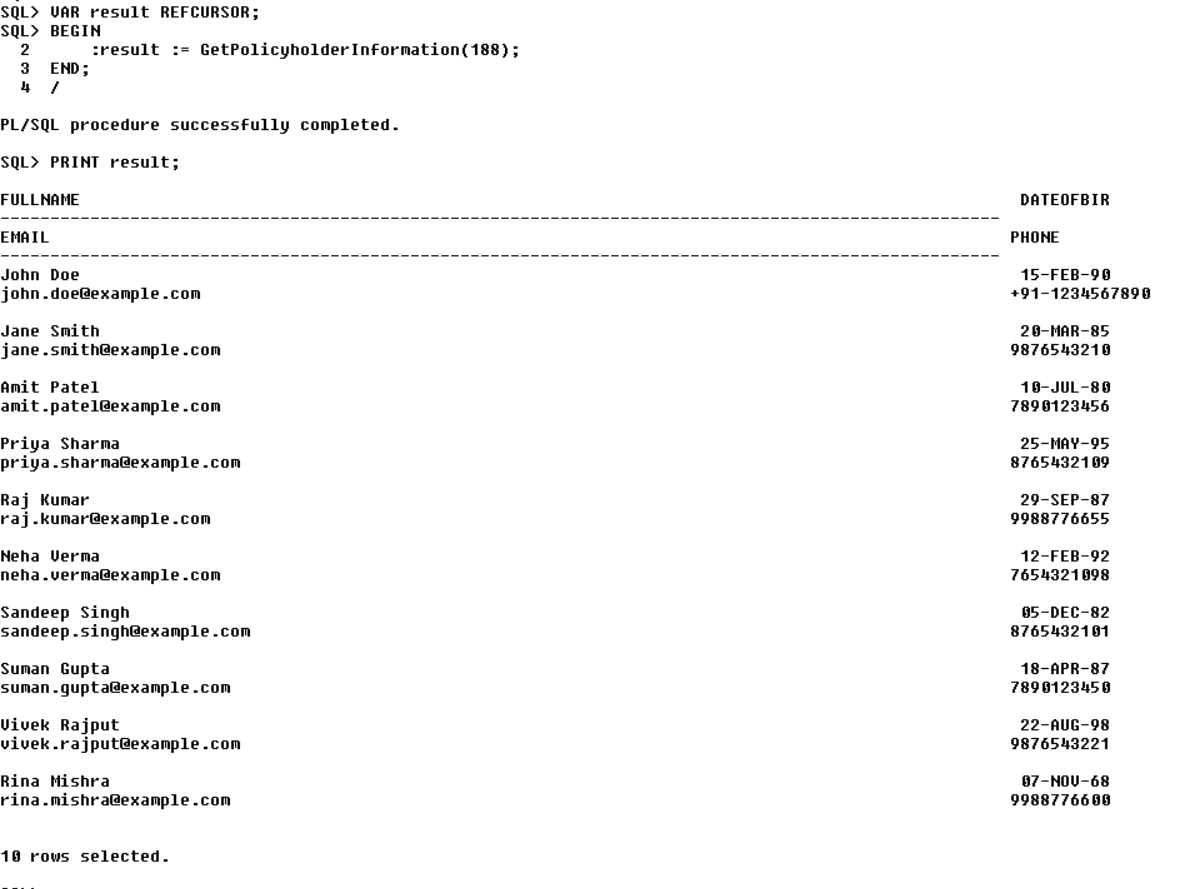
END;

PRINT result;

**Explanation**: This function retrieves information about a policyholder based on their PolicyholderID. It returns the policyholder's full name, date of birth, email, and phone number.

This function allows LIC to easily retrieve essential information about a policyholder by specifying their unique PolicyholderID, which can be helpful for customer support and policy management.



****

**Procedures: -**

**Question**: Create a procedure to calculate the total coverage amount and retrieve policyholder details for a specific policy type, based on the provided PolicyType.

CREATE OR REPLACE PROCEDURE GetPolicyTypeSummary (

p\_PolicyType VARCHAR2

)

AS

v\_TotalCoverageAmount NUMBER(12, 2) := 0;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Policy Type: ' || p\_PolicyType);

SELECT SUM(CoverageAmount) INTO v\_TotalCoverageAmount

FROM Policyy

WHERE PolicyType = p\_PolicyType;

DBMS\_OUTPUT.PUT\_LINE('Total Coverage Amount: ' || v\_TotalCoverageAmount);

END;

/

-- Get the policyholders and their details for the given policy type

SET SERVEROUTPUT ON;

DECLARE

p\_PolicyType VARCHAR2(50) := 'LIC New Pension Plus';

CURSOR policyholder\_cursor IS

SELECT ph.PolicyholderID, ph.FirstName, ph.LastName, ph.DateOfBirth, ph.Email, ph.Phone

FROM Policyholder ph

JOIN Policyy po ON ph.PolicyholderID = po.PolicyholderID

WHERE po.PolicyType = p\_PolicyType;

BEGIN

FOR policyholder\_row IN policyholder\_cursor

LOOP

DBMS\_OUTPUT.PUT\_LINE('Policyholder ID: ' || policyholder\_row.PolicyholderID);

DBMS\_OUTPUT.PUT\_LINE('Name: ' || policyholder\_row.FirstName || ' ' || policyholder\_row.LastName);

DBMS\_OUTPUT.PUT\_LINE('Date of Birth: ' || TO\_CHAR(policyholder\_row.DateOfBirth, 'YYYY-MM-DD'));

DBMS\_OUTPUT.PUT\_LINE('Email: ' || policyholder\_row.Email);

DBMS\_OUTPUT.PUT\_LINE('Phone: ' || policyholder\_row.Phone);

DBMS\_OUTPUT.PUT\_LINE('');

END LOOP;

GetPolicyTypeSummary('LIC New Pension Plus');

END;

/

**Explanation:**

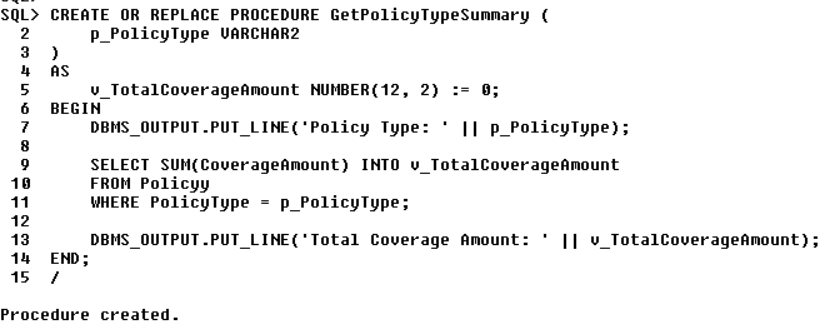
- This procedure takes a `PolicyType` as input and calculates the total coverage amount for policies of that type.

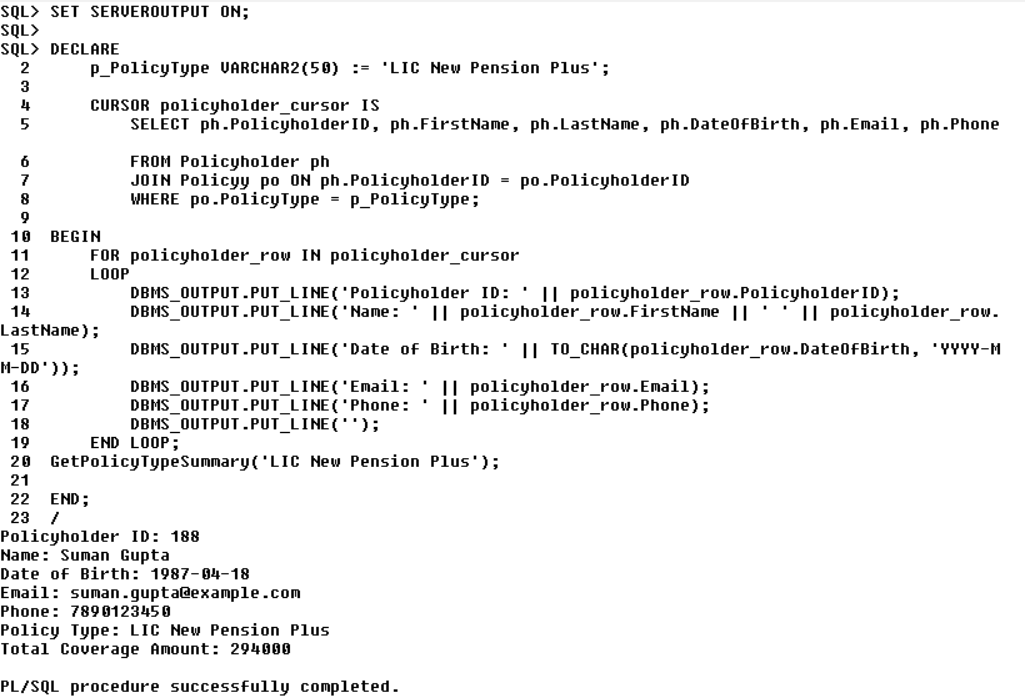
- It displays the policy type and the total coverage amount.

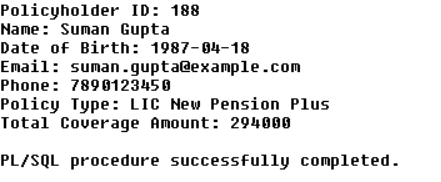
- It then retrieves the policyholders and their details for the given policy type.

- The procedure uses `DBMS\_OUTPUT.PUT\_LINE` to print the results.

This procedure calculates the total coverage amount for a specific policy type and provides a summary of the policies and policyholders for that type. This can help the company analyze the coverage amount and policyholder details for a particular insurance product.







**Triggers: -**

**Question**: How can we ensure that each new policy inserted into the "Policy" table is automatically assigned a unique policy number, taking into account the existing policies?

Create a trigger to generate a new policy number

CREATE OR REPLACE TRIGGER GeneratePolicyNumber

BEFORE INSERT ON Policy

FOR EACH ROW

BEGIN

SELECT MAX(PolicyNumber) + 1 INTO :NEW.PolicyNumber FROM Policy;

END;

/

**Explanation**:

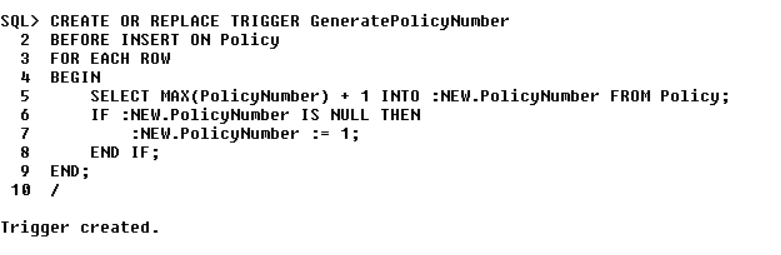
- This trigger is named "GeneratePolicyNumber."

- It is set to fire before an INSERT operation on the "Policy" table.

- For each new row being inserted, it calculates the next policy number by finding the maximum policy number in the table and adding 1 to it.

- If there are no existing policies, it sets the policy number to 1 as the initial policy number.

Trigger for an LIC (Life Insurance Corporation) could be to automatically generate a new policy number when a new policy is inserted into the "Policy" table. This trigger will ensure that each policy is assigned a unique policy number.



**Future Enhancement**

1. **Performance Optimization:**
   * Implement indexing and query optimization techniques to enhance the performance of database queries, especially for tables with a large number of records.
   * Use database performance monitoring tools to identify and address bottlenecks in real-time, ensuring efficient data retrieval and storage.
2. **Data Security and Encryption:**
   * Implement advanced encryption and data security measures to protect sensitive customer information.
   * Enhance user authentication and authorization mechanisms to control access to the database, ensuring that only authorized personnel can perform specific actions.
3. **Data Backup and Recovery:**
   * Develop automated backup and recovery processes to protect data against loss or corruption.
   * Implement a robust disaster recovery plan to minimize downtime and data loss in case of unforeseen events.
4. **Historical Data Retention:**
   * Incorporate a mechanism for archiving historical policy and payment data to maintain data integrity, meet compliance requirements, and enable historical analysis.
   * Define data retention policies to automatically archive or delete data based on business rules and regulatory requirements.
5. **Scalability:**
   * Design the database to be scalable, allowing for the addition of more policyholders, policies, and agents as the company grows.
   * Implement sharding or partitioning strategies to distribute data across multiple servers for improved performance and scalability.
6. **Data Warehouse Integration:**
   * Integrate a data warehousing solution to consolidate data from the PMD for business intelligence, analytics, and reporting purposes.
   * Enable the extraction, transformation, and loading (ETL) of data from the PMD into the data warehouse for advanced reporting and data analysis.
7. **Business Intelligence and Reporting:**
   * Implement a business intelligence (BI) layer to provide executives, managers, and analysts with dashboards and reports to make informed decisions.
   * Enable ad-hoc querying and data visualization tools for data analysis.
8. **Audit Trails and Logging:**
   * Enhance audit trails to track changes made to the database, including data modifications, user access, and system events.
   * Implement comprehensive logging to record database activities for security and compliance purposes.
9. **Concurrent Data Access:**
   * Enhance database concurrency control mechanisms to manage simultaneous data access by multiple users and applications, ensuring data consistency and integrity.
10. **Data Privacy Compliance:**
    * Implement features to comply with evolving data privacy regulations such as GDPR, CCPA, or any applicable regional laws.
    * Enable data anonymization or pseudonymization capabilities to protect policyholder privacy.
11. **Real-time Data Integration:**
    * Develop real-time data integration interfaces with external systems, such as banks for payment processing, to ensure seamless and up-to-date data synchronization.
    * Utilize message queues or data streaming platforms for real-time data exchange.
12. **Geospatial Data:**
    * If relevant, include geospatial data capabilities to track the location of policyholders, agents, and branch offices. This can aid in market analysis and targeted marketing.
13. **Database Version Control:**
    * Implement version control for database schema and configuration to track changes and roll back in case of issues during updates.
14. **Database as a Service (DBaaS):**
    * Consider transitioning to a DBaaS model to reduce the operational burden of managing the database infrastructure, allowing the DBMS provider to handle maintenance, scalability, and security.
15. **Machine Learning and Predictive Analytics:**
    * Incorporate machine learning and predictive analytics models to assess risk, predict policyholder behavior, and personalize insurance offerings.
16. **Chatbots and AI Integration:**
    * Integrate chatbots and AI-driven customer service solutions to enhance customer interactions and support.
17. **Blockchain for Trust and Transparency:**
    * Explore the use of blockchain technology to increase trust and transparency in transactions, particularly in the management of claims and payments.
18. **Compliance Monitoring:**
    * Implement automated compliance monitoring to ensure that the database adheres to industry regulations and internal policies, alerting administrators to potential violations.
19. **Data Quality Management:**
    * Deploy data quality tools to detect and correct data inconsistencies, errors, and duplications, improving the overall accuracy of the database.
20. **Advanced Data Analytics and Data Mining:**
    * Develop advanced data analytics and data mining capabilities to extract insights from the database for strategic decision-making, fraud detection, and market analysis.

**Conclusion**

In conclusion, the Policy Management Database, when viewed through the lens of a Database Management System, acts as the backbone of LIC's operations. It ensures data accuracy, enables efficient data retrieval, fosters strong customer relationships, ensures regulatory compliance, scales with business growth, enhances data security, and empowers data-driven decision-making. All of these factors combine to make the PMD an indispensable tool for LIC's success in the insurance industry, ultimately benefiting both the company and its valued policyholders.