IFT 530: Advanced DataBase Management Systems (2023 Spring)

Project Tile: Insurance Policy Database Management System

530 Final Project Written Submission

Submitted to Dr. Robert Rucker

By

Devi Subadra Venkatesan (ASU ID: 1228251806)

Balasubramanyam Prema Doddanari (ASU ID: 1227720899)

Praveen Kumar Siddela (ASU ID:1227371004)

On 28th April 2023

Table of Contents

Introduction for Insurance Policy Database Management System (IPDMS)

An insurance management system is a application designed to help insurance companies manage their policies, claims, and other related activities more efficiently. The system provides a platform for insurers to manage their business operations, from underwriting policies to handling claims, customer service, and compliance with regulatory requirements.

The system is designed to streamline the insurance process, reduce operational costs, and improve customer service. It enables insurers to manage policies more effectively, automate the claims process, and track and analyze data to identify patterns and trends that could lead to potential losses.

Insurance management systems come with a range of features and capabilities, including policy management, claims processing, reporting and analytics, compliance management, and customer relationship management. The system can be customized to meet the specific needs of an insurance company, allowing insurers to tailor the software to fit their unique business requirements.

Importance of the IPDMS

An insurance management system is an essential tool for efficiently managing the complex operations of the insurance industry. Its importance cannot be overstated, as it can streamline processes, ensure compliance, and manage risks effectively. By automating tasks such as data entry and claims processing, insurers can enhance operational efficiency, reduce errors, and optimize resource utilization, leading to cost savings. The system's ability to improve customer service is another significant advantage. Automated claims processing with real-time updates can

result in better customer satisfaction and retention rates, contributing to the long-term success of the company. Furthermore, an insurance management system provides a centralized database for all policies, claims, and relevant information, allowing insurers to manage and monitor data accurately. This, in turn, can help insurers make informed decisions based on precise data analysis, leading to better risk management and profitability. The system's role in ensuring regulatory compliance is crucial as insurance companies are obligated to adhere to strict regulatory requirements, and non-compliance can lead to costly penalties and legal issues. The system can provide alerts and reminders, helping insurers meet deadlines and complete necessary documentation. Lastly, an insurance management system can help insurers manage risks associated with underwriting policies by providing real-time data analysis and identifying potential losses. In summary, an insurance management system is a necessary tool for insurers to stay competitive in today's fast-paced business environment, as it can improve operational efficiency, customer service, compliance, risk management, and contribute to long-term success.

Description of Entities

The entities are “Customer”, ”Insurer”, ”Policy”, ”Policy Payment”, ”PolicyTypes”,”Address”. The First entity is Insurer , in which the primary key is insurereID, Foreign key is customerid, there are other attributes which are username, firstname, and lastname they were all self-referenced and marked with Mandatory and added Unique Constraints.The second entity is Customer, in which the primary key is customerid and the foreign key are PhoneNum and EmailId which are connected with Inclusive Or Constraint and addressid which will be connected directly to Address Entity which will store the address info of the Customers with a foreign key as countryCode and stateCode.

The main Entity is Policy as it is connected to the Customer with a Many-to-Many relation, PolicyPayment, and PolicyTypes entities with Policyid as a primary key along with InsuranceStatus which is depicted with Exclusive Or Constraint to represent any one status {Sanctioned,Cancelled, Renewed}is allowed.It has a Ternary relation of PolicyHasPolicyPaymentWhereThePaymentDate through policypaymentid from Policypayment with two subtypes (PaymentStatus and PaymentType) which are differentiated with Exclusive Or Constraint.

The Policy table is connected to the second major Entity which is Types, they are subcategorized into three groups -AutoPolicy(Automobiles), HousingPolicy and LifeInsurance.

AutoPolicy- Checks whether the vehicle has any accidents if so then it would dedudct the amount with autoPolicyName and foreign key -accidentid

HousingPolicy- Insurer checks the House conditions with Yearbuilt and any prev policies or claims on the house.

LifeInsurance-Term-MedicalPolicy it covers the medical insurances for customers for certain period of time and UniversalPolicy ensure the permanent insurance covered even for the death of the customer by “checking the basic eligible criteria (i.e) Age limit shouldn’t be greater than 60 years and health condition should be healthy.

List of the queries that are answered as part of the project:

Stored Procedures:

1. Get all the Policy details as a particular procedure.

   Example: If we want to get all the details of Policy, instead of executing it every time again and again, we can use a procedure called 'getCustomerPolicyDetails' which stores the SQL query.

2. Is there any Stored Procedure to retrieve Customer Payment policy details? Creates a stored procedure that selects from a particular state from the "Airport" table.

   Example: If we want the payment details for a particular customer then this Stored procedure 'getCustomerPaymentPolicyDetails' returns the Customer, Policy and payment details.

Triggers:

1. Triggers to perform Insert operation and that notify Customer if a Policy Payment Fails?

   Example: 'CustomerPaymentFailureNotify_Trigger' this trigger will insert the payment failure records in the EventLogger table. This trigger will be triggered whenever a record got inserted in PolicyPayment table.

2. Tiggers to perform Insert operation and that notifies the Customer policies which ends in a week?

   Example: 'CustomerPolicyDeadlineNotify_Trigger' this trigger will insert the Customer policy records which have policy Expire deadline of 7 days. This trigger will be triggered whenever a record get inserted in Policy table

Questions:

1. How many policies were issued in a particular time period?

2.How many Active Policies are there in the system?

3. Which Customer has the Maximum coverage amount?

4. Find out the Customer details who has paid their policy payment through Netbanking also display the Policy coverage amount?

5. Is there any Stored procedure to retrieve Policy details for a Customer?

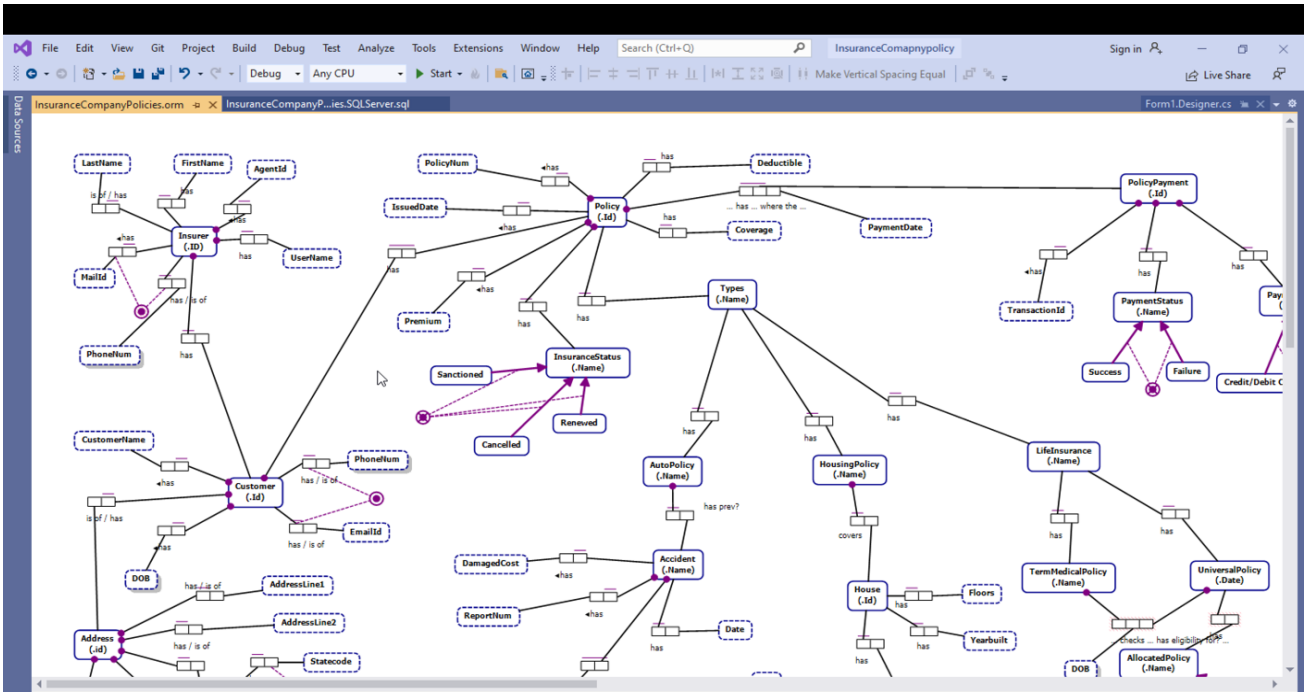
6. Is there any Stored procedure to retrieve Customer Payment Policy details?

7. Is there any trigger that notify Customer if a Policy Payment fails?

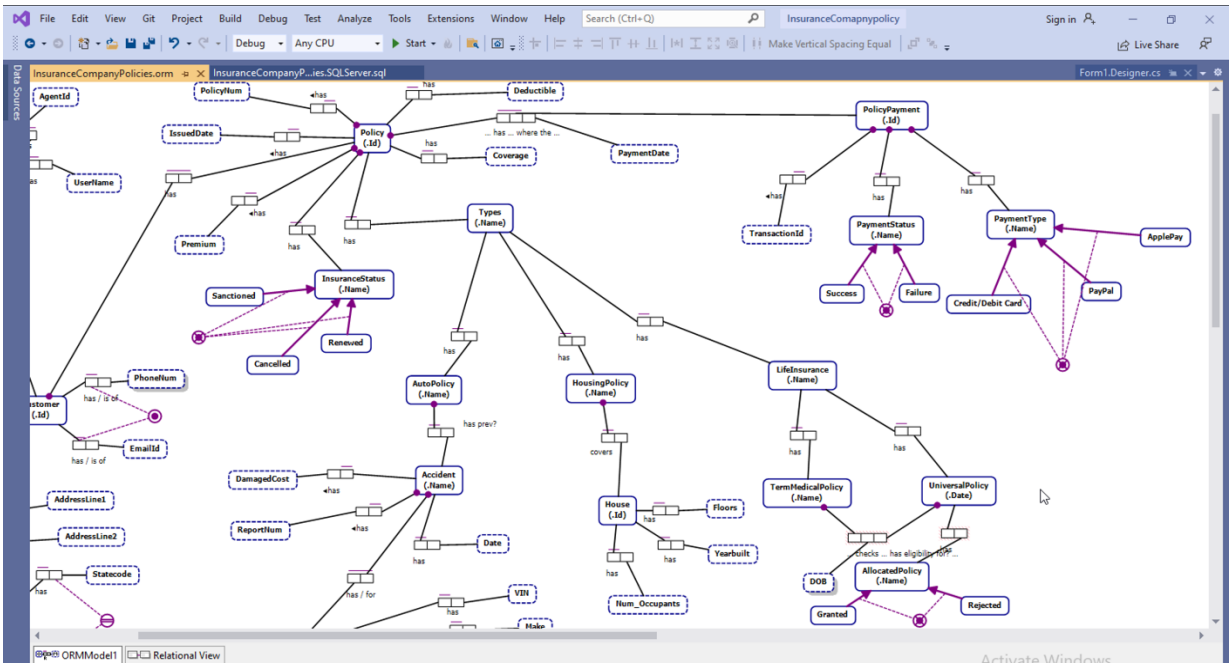
8. Is there any trigger that notifies the Customer policies which ends in a week?

ORM Diagram

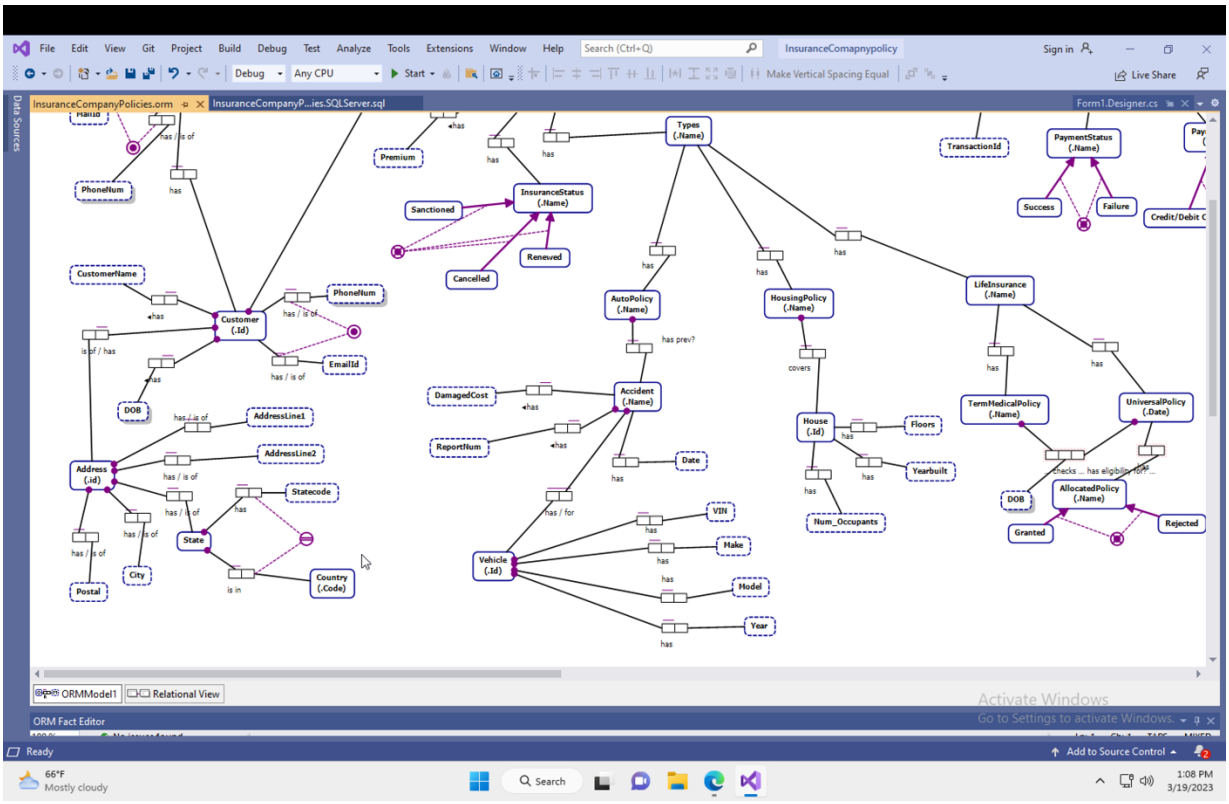
a)



b)


File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
InsuranceCompanyPolicies.orm
InsuranceCompanyP...ies.SQLServer.sql
Form1.Designer.cs
AgentId
PolicyNum
Deductible
IssuedDate
Policy (.Id)
Coverage
PaymentDate
PolicyPayment (.Id)
UserName
Premium
Types (.Name)
TransactionId
PaymentStatus (.Name)
PaymentType (.Name)
ApplePay
Success
Failure
Credit/Debit Card
PayPal
InsuranceStatus (.Name)
Sanctioned
Cancelled
Renewed
PhoneNum
EmailId
AutoPolicy (.Name)
HousingPolicy (.Name)
LifeInsurance (.Name)
has prev?
covers
DamagedCost
Accident (.Name)
ReportNum
Date
AddressLine1
AddressLine2
Statecode
House (.Id)
Floors
Yearbuilt
Num_Occupants
VIN
TermMedicalPolicy (.Name)
UniversalPolicy (.Date)
DOB
AllocatedPolicy (.Name)
Granted
Rejected
ORMModel1
Relational View

c)



Entities of the database with their attributes:

1. **Insurer(.ID)**
    - FirstName
    - LastName
    - AgentId
    - UserName
    - PhoneNum
    - MailId
2. **Customer(.ID)**
    - CustomerName

- DOB
- PhoneNum
- EmailId

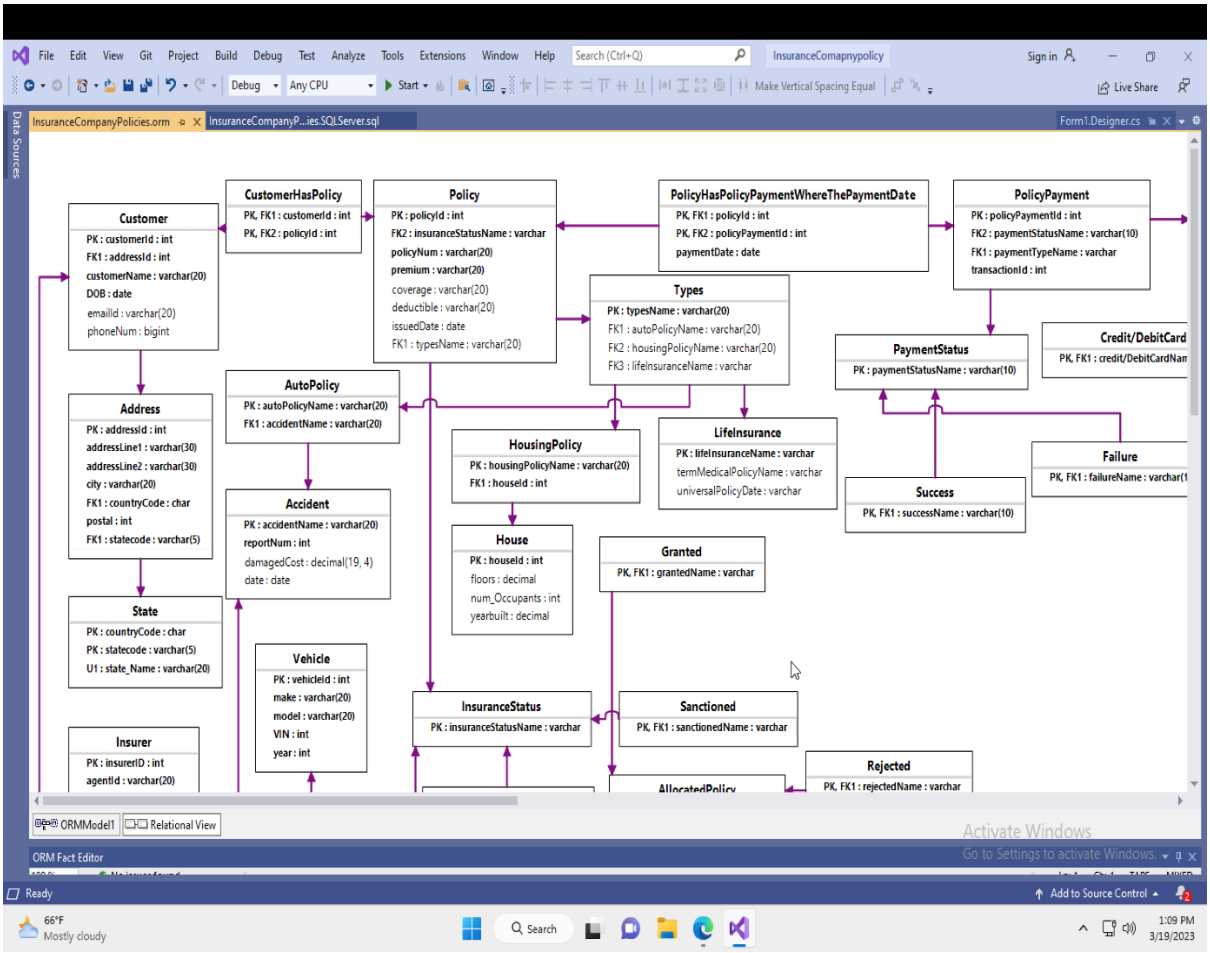
3. **Policy(.ID)**
    - IssuedDate
    - PolicyNum
    - coverage
    - Deductible
4. **PolicyPayment(.ID)**
    - PaymentType
    - PaymentStatus
    - TransactionId
5. **Types(.Name)**
    - AutoPolicy
    - HousingPolicy
    - LifeInsurance
6. **Accident(.Name)**
    - DamagedCost
    - ReportNum
    - Date
7. **House(.Name)**
    - Floors
    - YearBuilt

- occupents

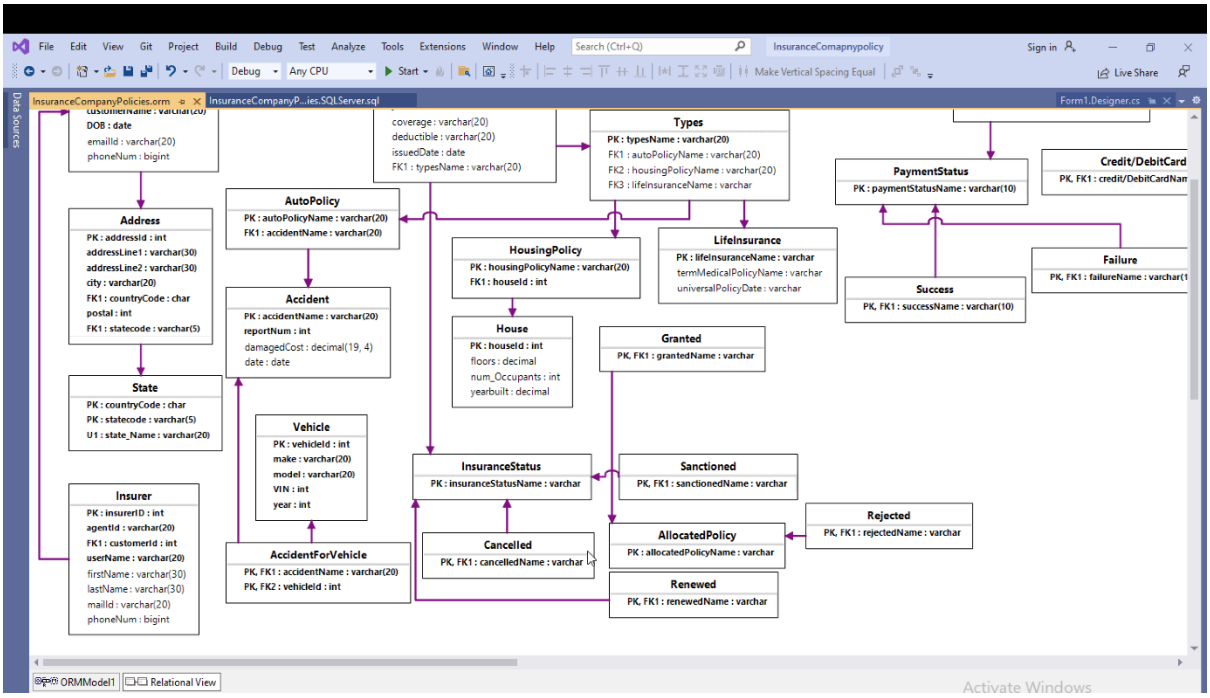
8. **Vehicle(.ID)**

- VIN
- Make
- Model
- Year

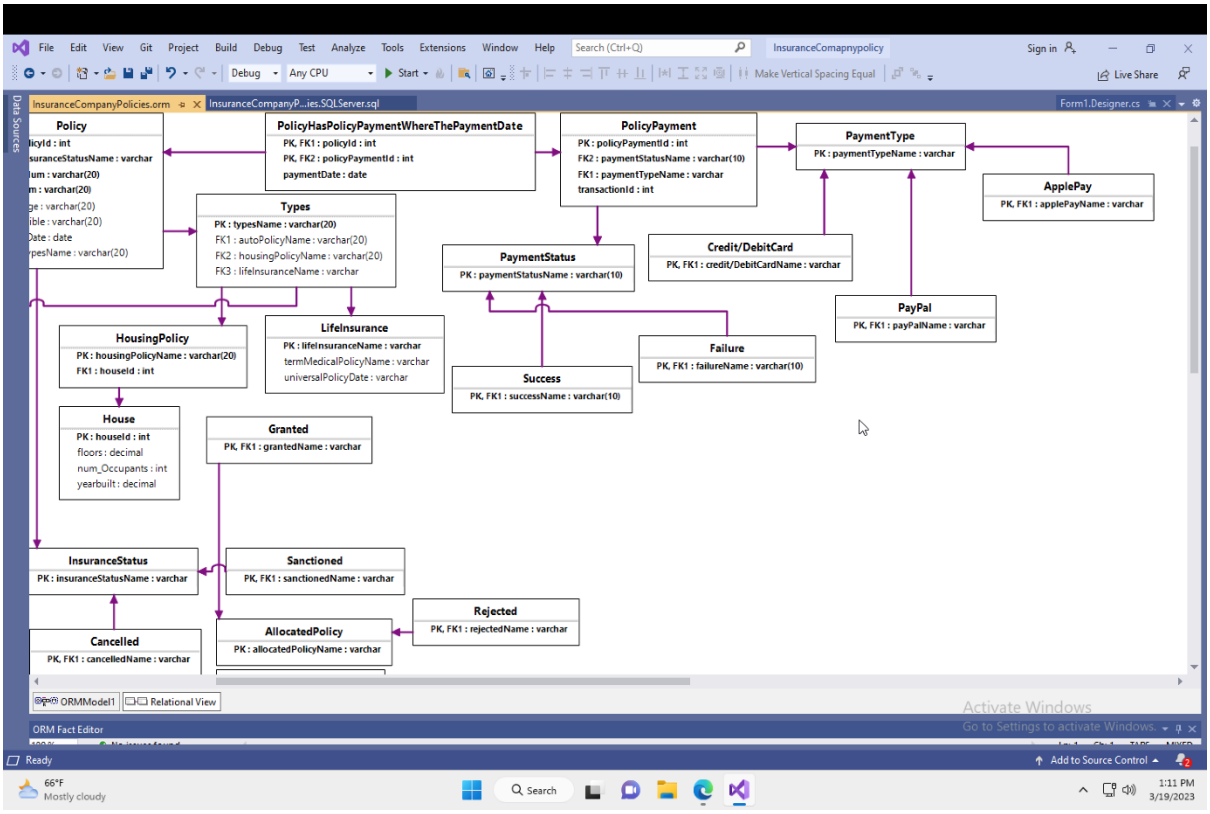
Relational Schema



b)


File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
InsuranceCompanyPolicies.orm
InsuranceCompanyP...ies.SQLServer.sql
Form1.Designer.cs
Address
PK : addressId : int
addressLine1 : varchar(30)
addressLine2 : varchar(30)
city : varchar(20)
FK1 : countryCode : char
postal : int
FK1 : statecode : varchar(5)
State
PK : countryCode : char
PK : statecode : varchar(5)
U1 : state_Name : varchar(20)
Insurer
PK : insurerID : int
agentId : varchar(20)
FK1 : customerId : int
userName : varchar(20)
firstName : varchar(30)
lastName : varchar(30)
mailId : varchar(20)
phoneNum : bigint
AutoPolicy
PK : autoPolicyName : varchar(20)
FK1 : accidentName : varchar(20)
Accident
PK : accidentName : varchar(20)
reportNum : int
damagedCost : decimal(19, 4)
date : date
Vehicle
PK : vehicleId : int
make : varchar(20)
model : varchar(20)
VIN : int
year : int
AccidentForVehicle
PK, FK1 : accidentName : varchar(20)
PK, FK2 : vehicleId : int
Types
PK : typesName : varchar(20)
FK1 : autoPolicyName : varchar(20)
FK2 : housingPolicyName : varchar(20)
FK3 : lifeInsuranceName : varchar
HousingPolicy
PK : housingPolicyName : varchar(20)
FK1 : houseId : int
House
PK : houseId : int
floors : decimal
num_Occupants : int
yearbuilt : decimal
LifeInsurance
PK : lifeInsuranceName : varchar
termMedicalPolicyName : varchar
universalPolicyDate : varchar
Granted
PK, FK1 : grantedName : varchar
InsuranceStatus
PK : insuranceStatusName : varchar
Sanctioned
PK, FK1 : sanctionedName : varchar
Cancelled
PK, FK1 : cancelledName : varchar
AllocatedPolicy
PK : allocatedPolicyName : varchar
Rejected
PK, FK1 : rejectedName : varchar
Renewed
PK, FK1 : renewedName : varchar
PaymentStatus
PK : paymentStatusName : varchar(10)
Credit/DebitCard
Failure
Success
PK, FK1 : successName : varchar(10)
ORMModel1
Relational View
Activate Windows

c)


File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
InsuranceCompanyPolicies.orm
InsuranceCompanyP...ies.SQLServer.sql
Form1.Designer.cs
Policy
PolicyHasPolicyPaymentWhereThePaymentDate
PK, FK1 : policyId : int
PK, FK2 : policyPaymentId : int
paymentDate : date
PolicyPayment
PK : policyPaymentId : int
FK2 : paymentStatusName : varchar(10)
FK1 : paymentTypeName : varchar
transactionId : int
PaymentType
PK : paymentTypeName : varchar
ApplePay
PK, FK1 : applePayName : varchar
Types
PK : typesName : varchar(20)
FK1 : autoPolicyName : varchar(20)
FK2 : housingPolicyName : varchar(20)
FK3 : lifeInsuranceName : varchar
PaymentStatus
PK : paymentStatusName : varchar(10)
Credit/DebitCard
PK, FK1 : credit/DebitCardName : varchar
PayPal
PK, FK1 : payPalName : varchar
HousingPolicy
PK : housingPolicyName : varchar(20)
FK1 : houseId : int
LifeInsurance
PK : lifeInsuranceName : varchar
termMedicalPolicyName : varchar
universalPolicyDate : varchar
Failure
PK, FK1 : failureName : varchar(10)
Success
PK, FK1 : successName : varchar(10)
House
PK : houseId : int
floors : decimal
num_Occupants : int
yearbuilt : decimal
Granted
PK, FK1 : grantedName : varchar
InsuranceStatus
PK : insuranceStatusName : varchar
Sanctioned
PK, FK1 : sanctionedName : varchar
Rejected
PK, FK1 : rejectedName : varchar
AllocatedPolicy
PK : allocatedPolicyName : varchar
Cancelled
PK, FK1 : cancelledName : varchar
ORMModel1
Relational View
ORM Fact Editor
Activate Windows
Go to Settings to activate Windows.
Ready
Add to Source Control
66°F
Mostly cloudy
Search
1:11 PM
3/19/2023

6. Job Flow:

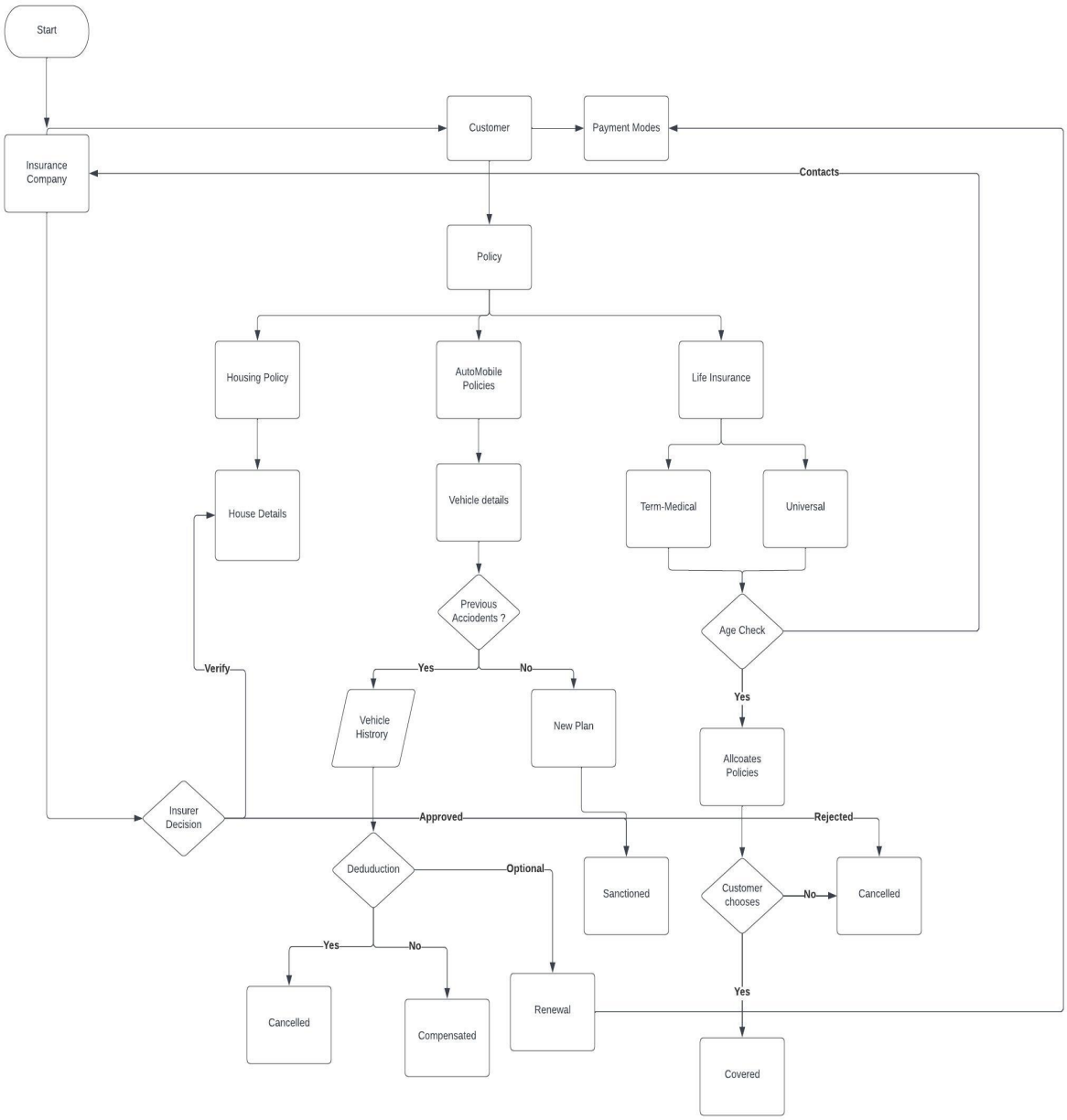

Start
Customer
Payment Modes
Insurance Company
Contacts
Policy
Housing Policy
AutoMobile Policies
Life Insurance
House Details
Vehicle details
Term-Medical
Universal
Previous Acciodents ?
Age Check
Verify
Yes
No
Yes
Vehicle Histrory
New Plan
Allcoates Policies
Insurer Decision
Approved
Rejected
Deduduction
Optional
Sanctioned
Customer chooses
No
Cancelled
Yes
No
Yes
Cancelled
Compensated
Renewal
Covered

Table Implementation, Datatypes and Column Constraints

In this project, we have used the following column constraints:

1. NOT NULL: The values in this column should not be null.
2. UNIQUE: The UNIQUE constraint ensures that all values in a column are different.
3. PRIMARY KEY: It is a unique key constraint which accepts only unique values in the data records and rejects any duplicate values.
4. FOREIGN KEY: It references the data from another table using a common column attribute to fetch the data.

1.Customer(.ID) – Primary Key

- CustomerName: Varchar(20)
- DOB: date
- PhoneNum: bigint
- EmailId :Varchar(20)
- addressId: Int- Foreign Key

2. Policy(.ID) – Primary Key

- IssuedDate: date
- PolicyNum: Varchar(20)
- coverage:varchar(20)
- typeName:Varchar(20)- Foreign Key

3. PolicyPayment(.ID) – Primary Key

- Paymentmethod: varchar(20)
- PaymentStatus:varchar(20)
- TransactionId:int

4. Types(.Name)-Varchar(20)

- AutoPolicy
- HousingPolicy
- LifeInsurance

DML & DDL

DDL and DML statements are used to define the data in the data bases and perform data manipulation

Data Definition Language (DDL): DDL statements are used to create, modify, or delete database objects such as tables, indexes, and constraints. We have used some DDL commands like, CREATE command for creating the database for the project, ALTER statements to manipulate the structure.

Example: Create a database to store all tables and create tables and then create the tables.

Data Manipulation Language (DML): It is used to manipulate the existing data. We have used statements like INSERT to enter data into the student- university admissions databases and UPDATE to perform any updating in the existing data.

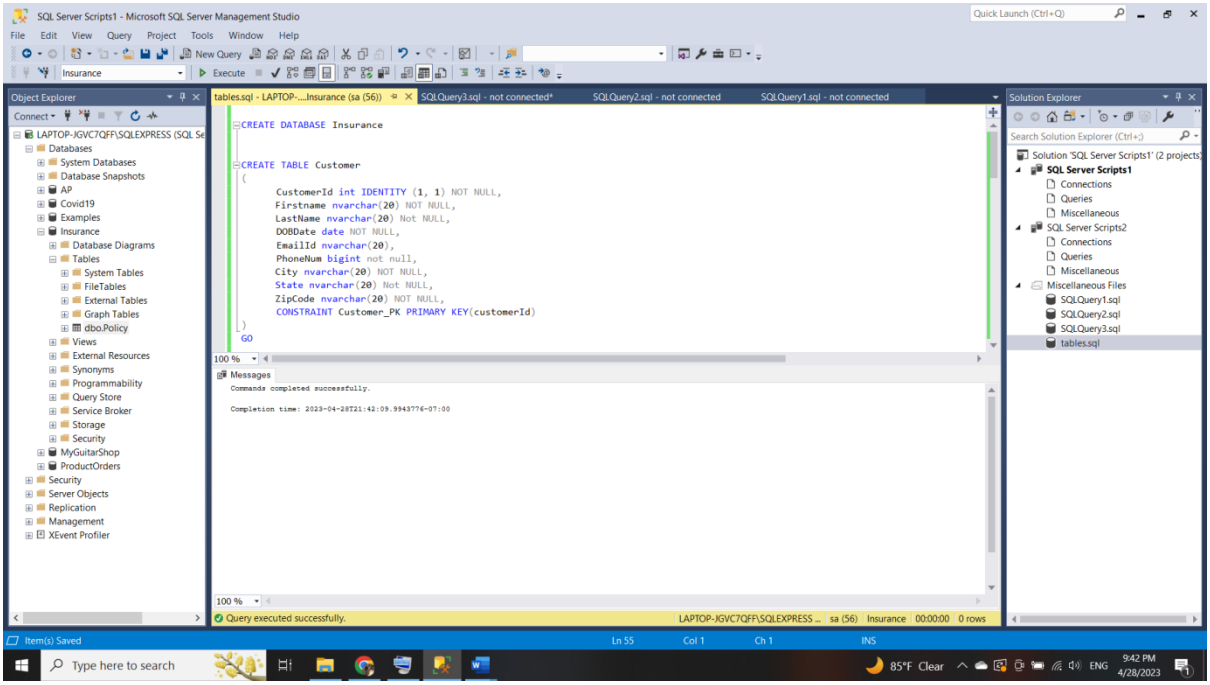
Example: INSERT statement to store Customer details

Database Tables

```
CREATE DATABASE Insurance


CREATE TABLE Customer
(
    CustomerId int IDENTITY (1, 1) NOT NULL,
    Firstname nvarchar(20) NOT NULL,
    LastName nvarchar(20) Not NULL,
    DOBDate date NOT NULL,
    EmailId nvarchar(20),
    PhoneNum bigint not null,
    City nvarchar(20) NOT NULL,
    State nvarchar(20) Not NULL,
    ZipCode nvarchar(20) NOT NULL,
    CONSTRAINT Customer_PK PRIMARY KEY(customerId)
)
GO


CREATE TABLE Policy
(
    PolicyId int IDENTITY (1, 1) NOT NULL,
    PolicyTypeID INT  NOT NULL,
    CustomerID INT NOT NULL,
    Coverage  bigint NOT NULL,
    IssuedDate  DateTime NOT NULL,
    Expiry DateTime NOT NULL,
    CONSTRAINT Policy_PK PRIMARY KEY(PolicyId)
)
GO


CREATE TABLE PolicyTypes
```

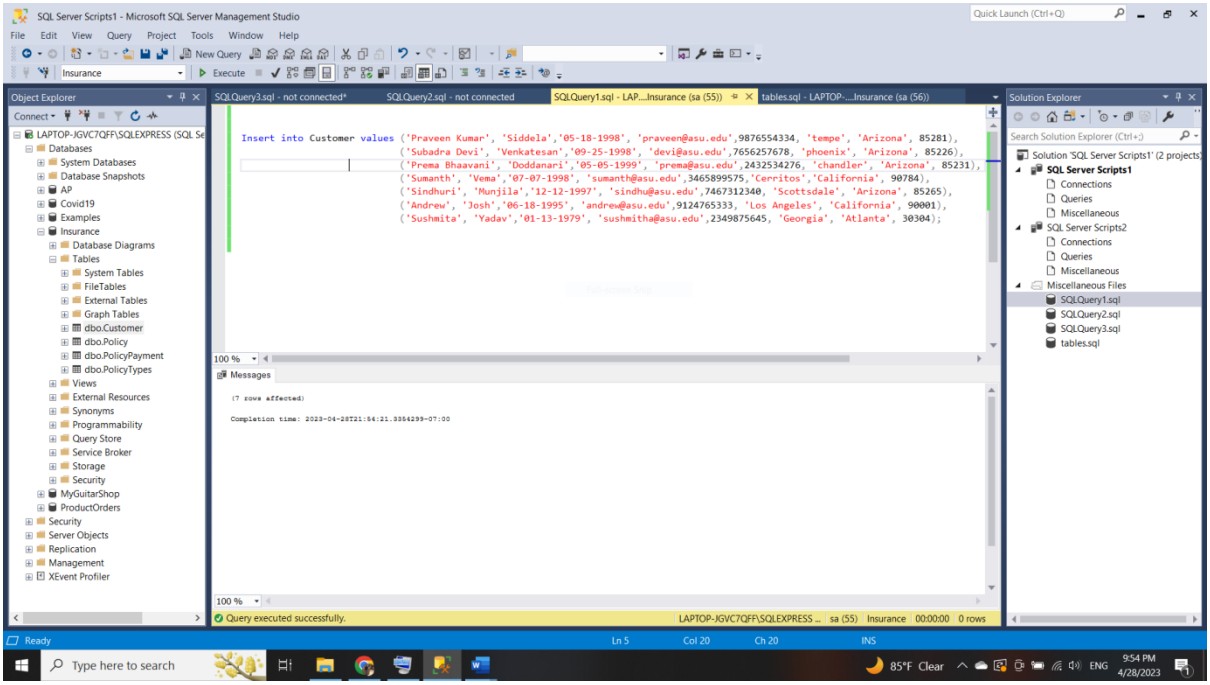
```
(
    PolicyTypeID INT IDENTITY (1, 1) NOT NULL,
    PolicyName nvarchar(20) NOT NULL,
    CONSTRAINT Types_PK PRIMARY KEY(PolicyTypeID)
)
GO



CREATE TABLE PolicyPayment
(
    PaymentId int IDENTITY (1, 1) NOT NULL,
    PolicyID int NOT NULL,
    PaymentStatus  nvarchar(30) NOT NULL,
    PaymentType nvarchar(30) NOT NULL,
    TransactionId bigint NOT NULL,
    CONSTRAINT PolicyPayment_PK PRIMARY KEY(PaymentId)
)
GO



ALTER TABLE Policy ADD CONSTRAINT Customer_FK FOREIGN KEY (CustomerID) REFERENCES
Customer(CustomerID) ON DELETE NO ACTION ON UPDATE NO ACTION
GO



ALTER TABLE Policy ADD CONSTRAINT PolicyType_FK FOREIGN KEY (PolicyTypeID) REFERENCES
PolicyTypes(PolicyTypeID) ON DELETE NO ACTION ON UPDATE NO ACTION
GO



ALTER TABLE PolicyPayment ADD CONSTRAINT PolicyPayment_FK FOREIGN KEY (PolicyID)
REFERENCES Policy(PolicyID) ON DELETE NO ACTION ON UPDATE NO ACTION
GO
```

```
CREATE TABLE EventLogger
(CustomerId INT ,
 PolicyID INT,
 Logtime DateTime,
 [Message] NVARCHAR(100))
GO

CREATE TABLE CustomerPolicyDeadline
(CustomerId INT ,
 Firstname nvarchar(20) NOT NULL,
 LastName nvarchar(20) Not NULL,
 PolicyID INT,
 Coverage  bigint NOT NULL,
 IssuedDate DateTime NOT NULL,
 Expiry DateTime NOT NULL,
 Logtime DateTime,
 [Message] NVARCHAR(100))
GO
```

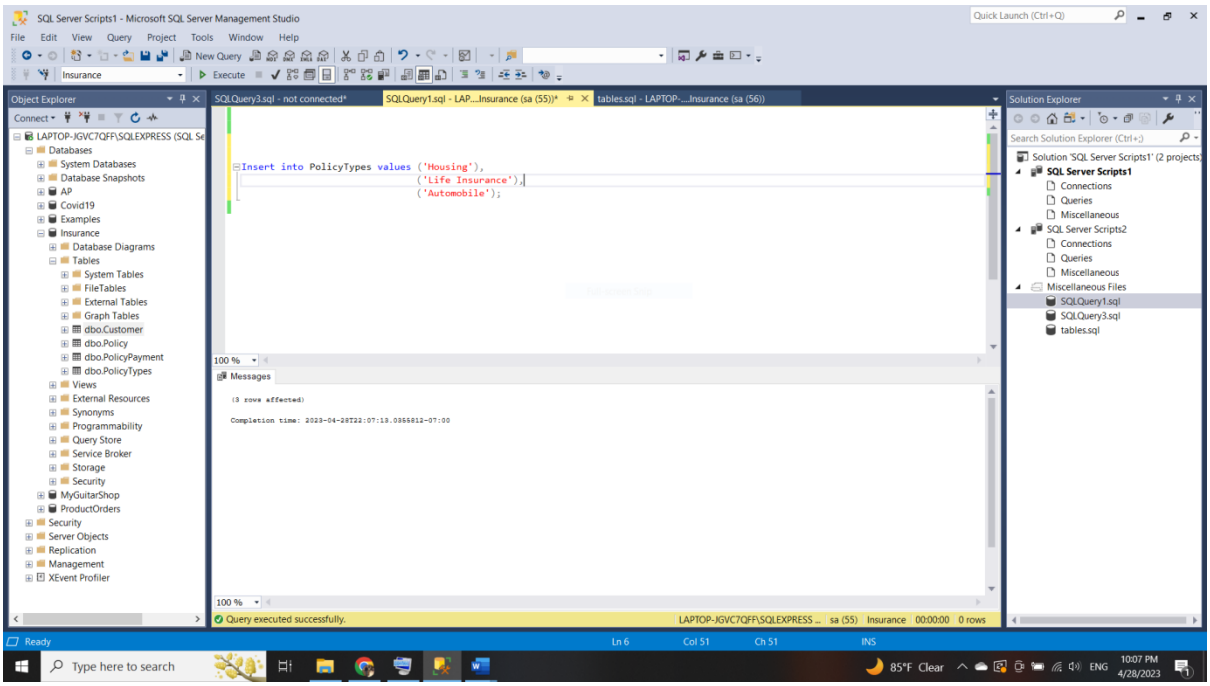


Inserting Data into Tables:

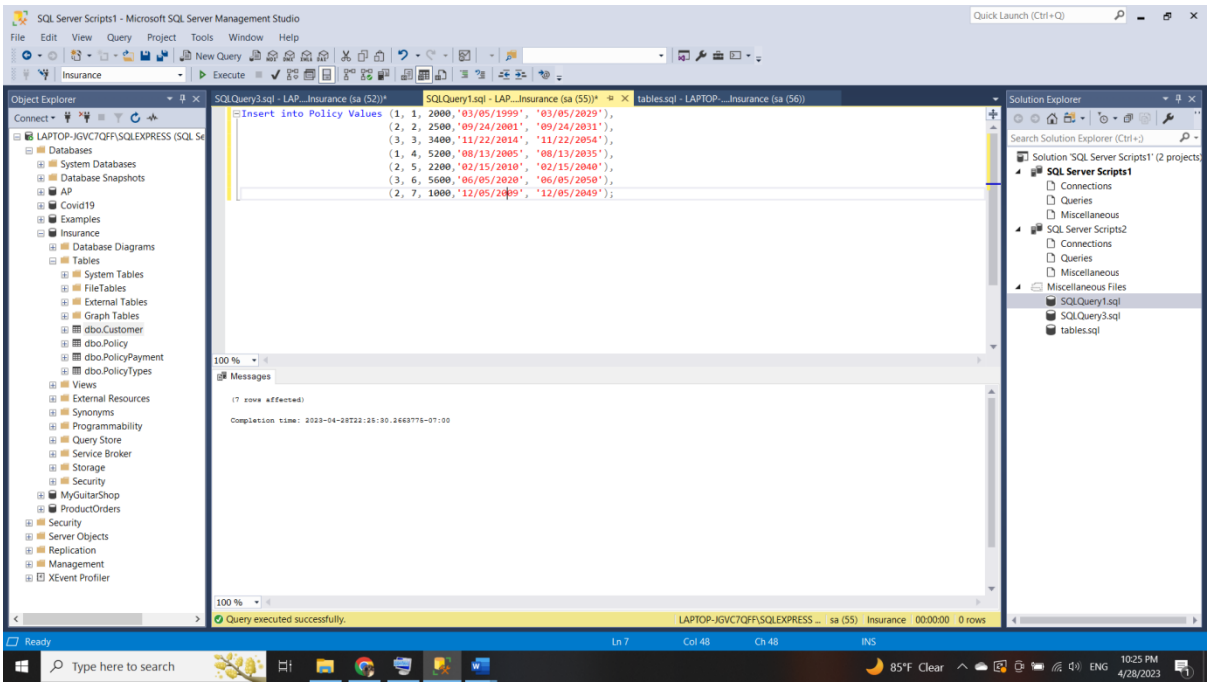
```
Insert into Customer values
('Praveen Kumar', 'Siddela','05-18-1998', 'praveen@asu.edu',9876554334, 'tempe', 'Arizona', 85281),
('Subadra Devi', 'Venkatesan','09-25-1998', 'devi@asu.edu',7656257678, 'phoenix', 'Arizona', 85226),
('Prema Bhaavani', 'Doddanari','05-05-1999', 'prema@asu.edu',2432534276, 'chandler', 'Arizona', 85231),
('Sumanth', 'Vema','07-07-1998', 'sumanth@asu.edu',3465899575,'Cerritos','California', 90784),
('Sindhuri', 'Munjila','12-12-1997', 'sindhu@asu.edu',7467312340, 'Scottsdale', 'Arizona', 85265),
('Andrew', 'Josh','06-18-1995', 'andrew@asu.edu',9124765333, 'Los Angeles', 'California', 90001),
('Sushmita', 'Yadav','01-13-1979', 'sushmitha@asu.edu',2349875645, 'Georgia', 'Atlanta', 30304);
```




```
INSERT INTO PolicyTypes VALUES ('Housing'),
                               ('Life Insurance'),
                               ('Automobile');
```




```
INSERT INTO Policy VALUES (1, 1, 2000,'03/05/1999', '03/05/2029'),
                          (2, 2, 2500,'09/24/2001', '09/24/2031'),
                          (3, 3, 3400,'11/22/2014', '11/22/2054'),
                          (1, 4, 5200,'08/13/2005', '08/13/2035'),
                          (2, 5, 2200,'02/15/2010', '02/15/2040'),
                          (3, 6, 5600,'06/05/2020', '06/05/2050'),
                          (2, 7, 1000,'12/05/2009', '12/05/2049');
```




```
Insert into PolicyPayment values(1,'Success','Cash',9127),
                                (2,'Success','Credit/Debit',7389),
                                (3,'Success','Netbanking',2739),
                                (4,'Success','Cash',1935),
                                (5,'Fail','Credit/Debit',4792),
                                (6,'Success','Credit/Debit',6500),
                                (7,'Fail','Netbanking',2794);
```

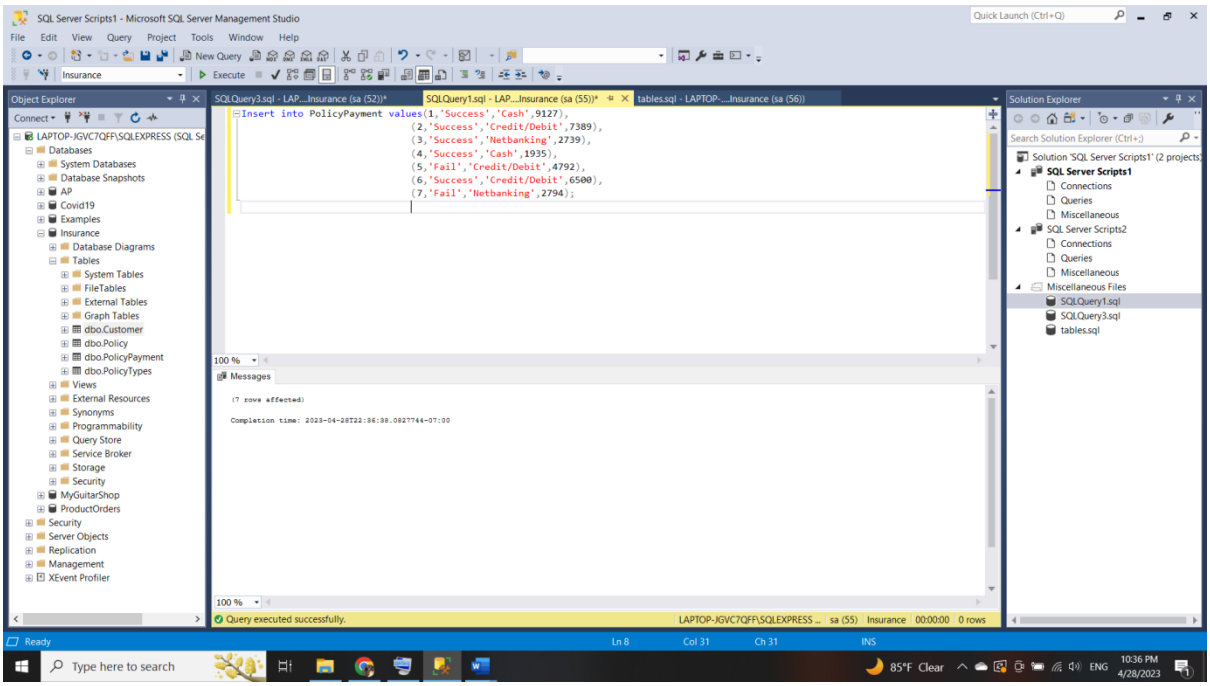
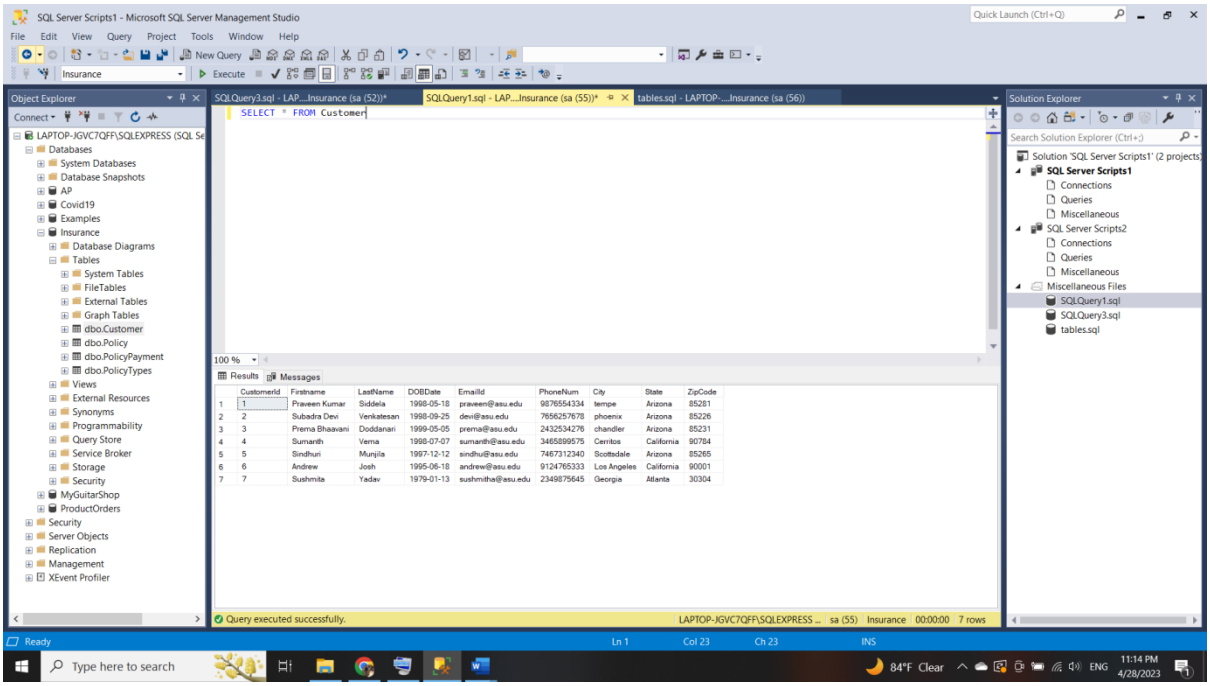
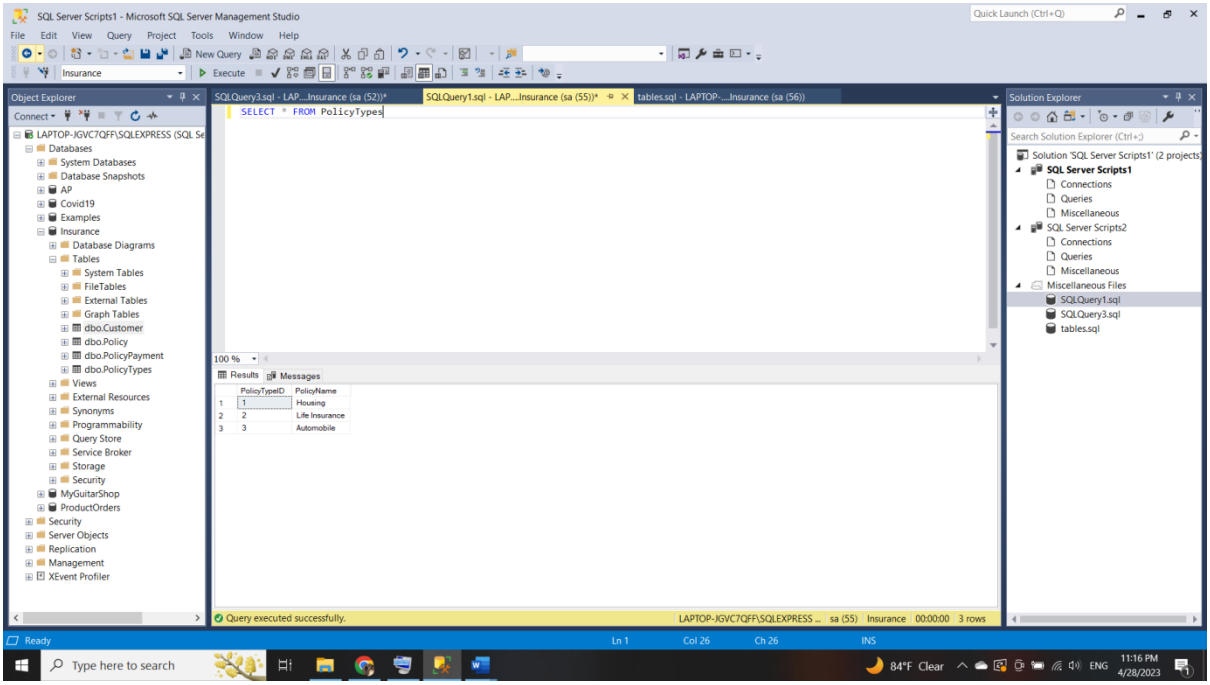


Table Contents:

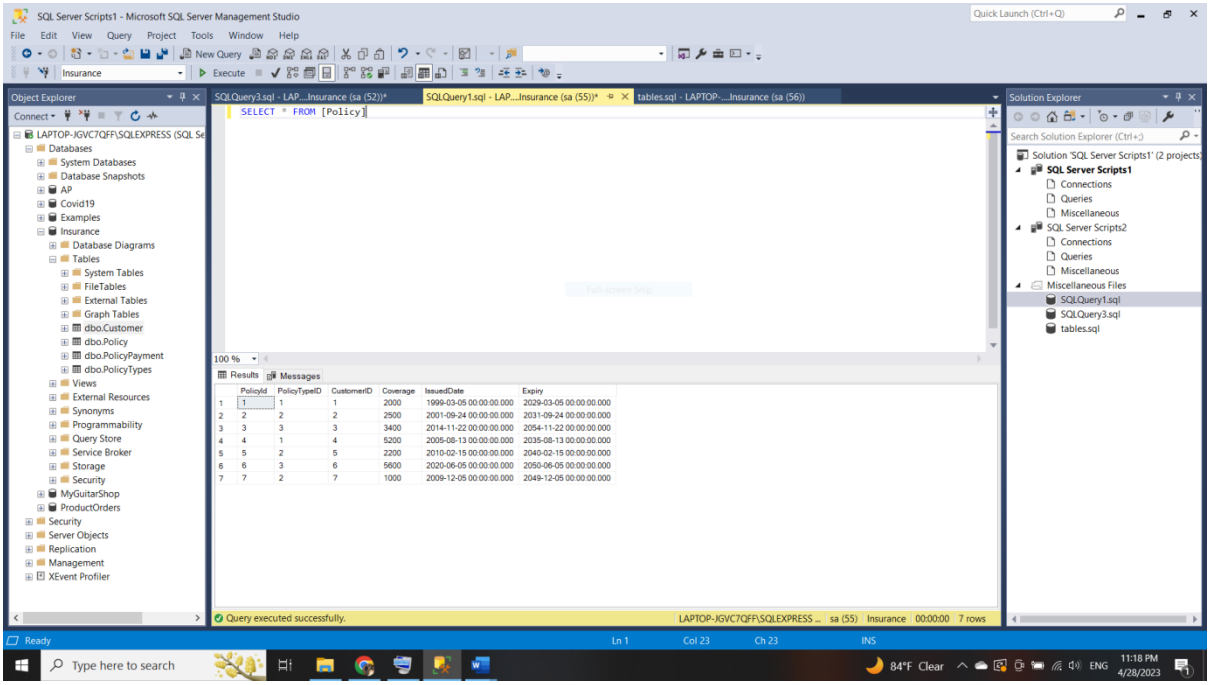
SELECT * FROM Customer



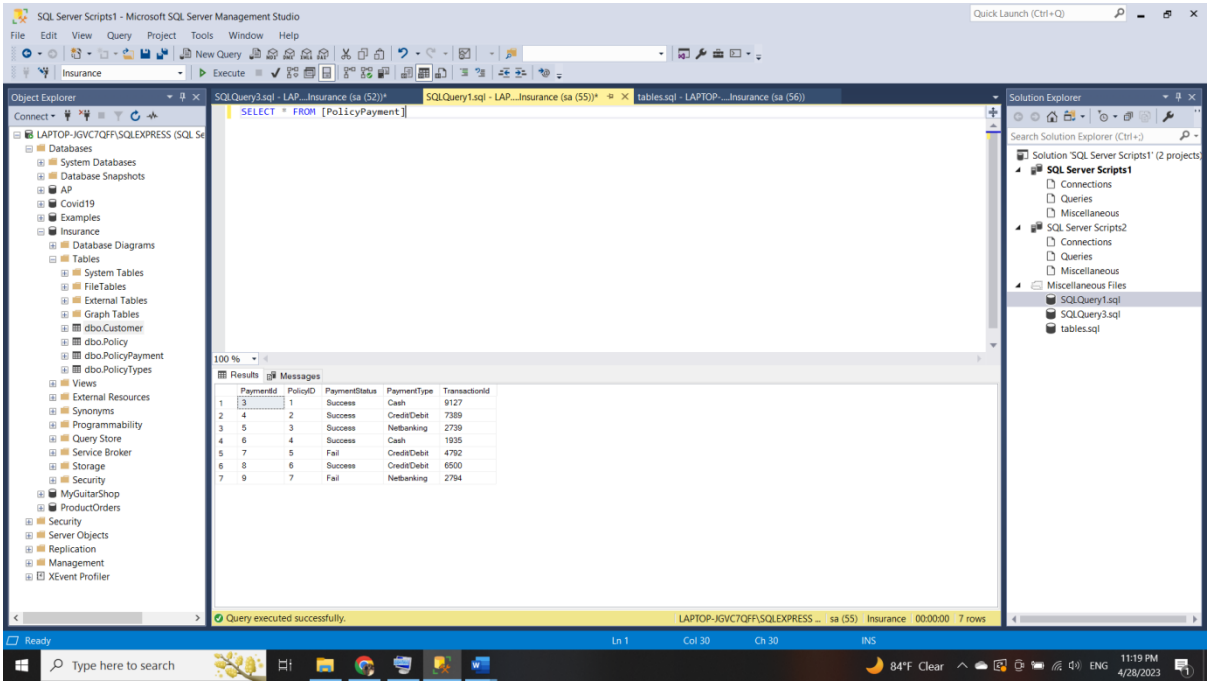
SELECT * FROM PolicyTypes



SELECT * FROM [Policy]



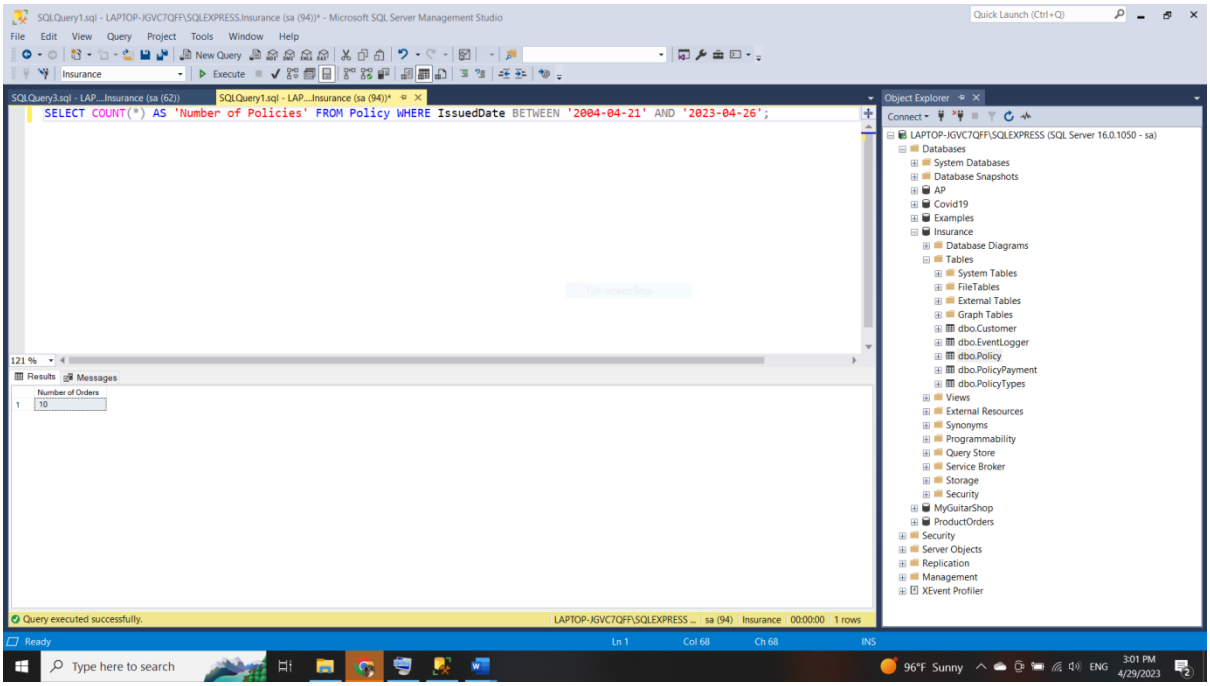
```
SELECT * FROM [PolicyPayment]
```



Answers for Questions

1. How many policies were issued in a particular time period?

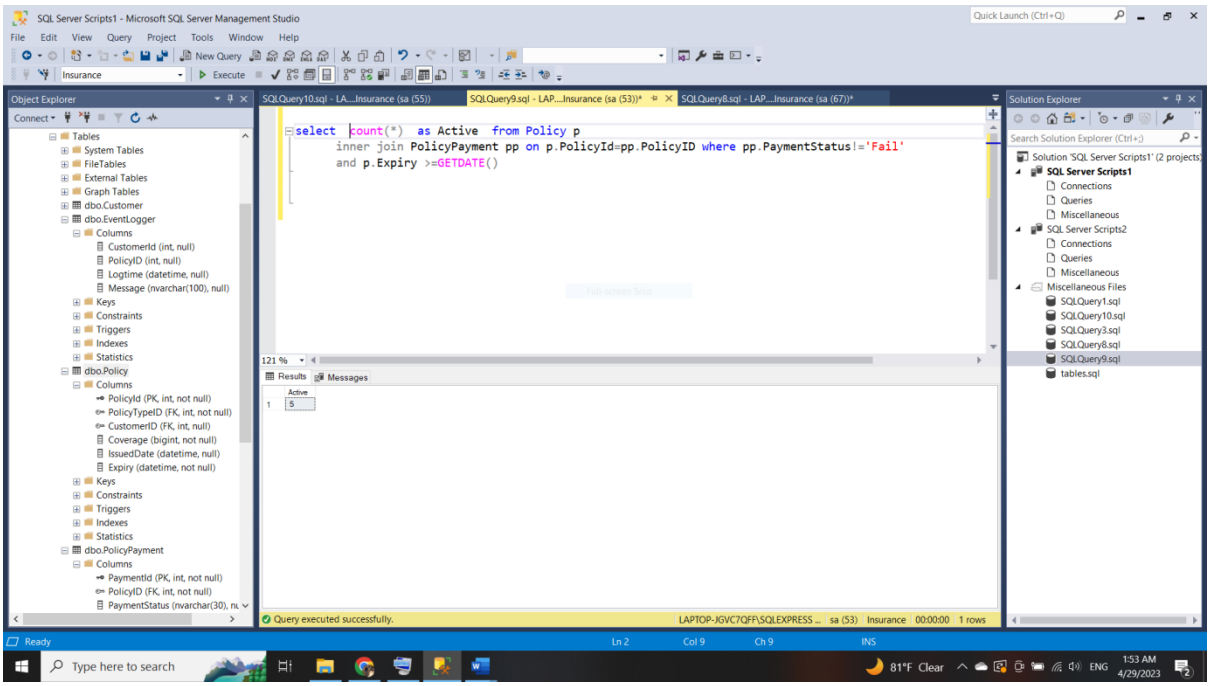
```
SELECT COUNT(*) AS 'Number of Policies' FROM Policy WHERE IssuedDate BETWEEN '2004-04-21' AND '2023-04-26';
```



2) How many Active Policies are there in the system?

A)

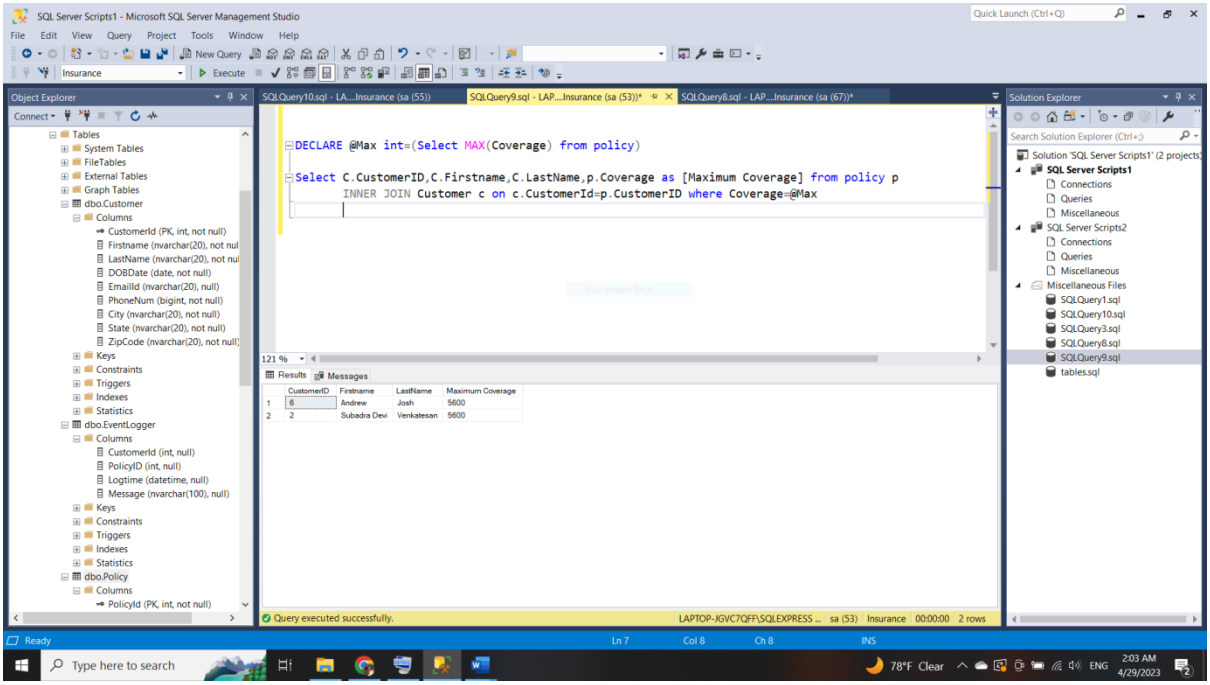
```
select count(*)  as Active  from Policy p
inner join PolicyPayment pp on p.PolicyId=pp.PolicyID where pp.PaymentStatus!='Fail'
and p.Expiry >=GETDATE()
```



3) Which Customer has the Maximum coverage amount?

A)

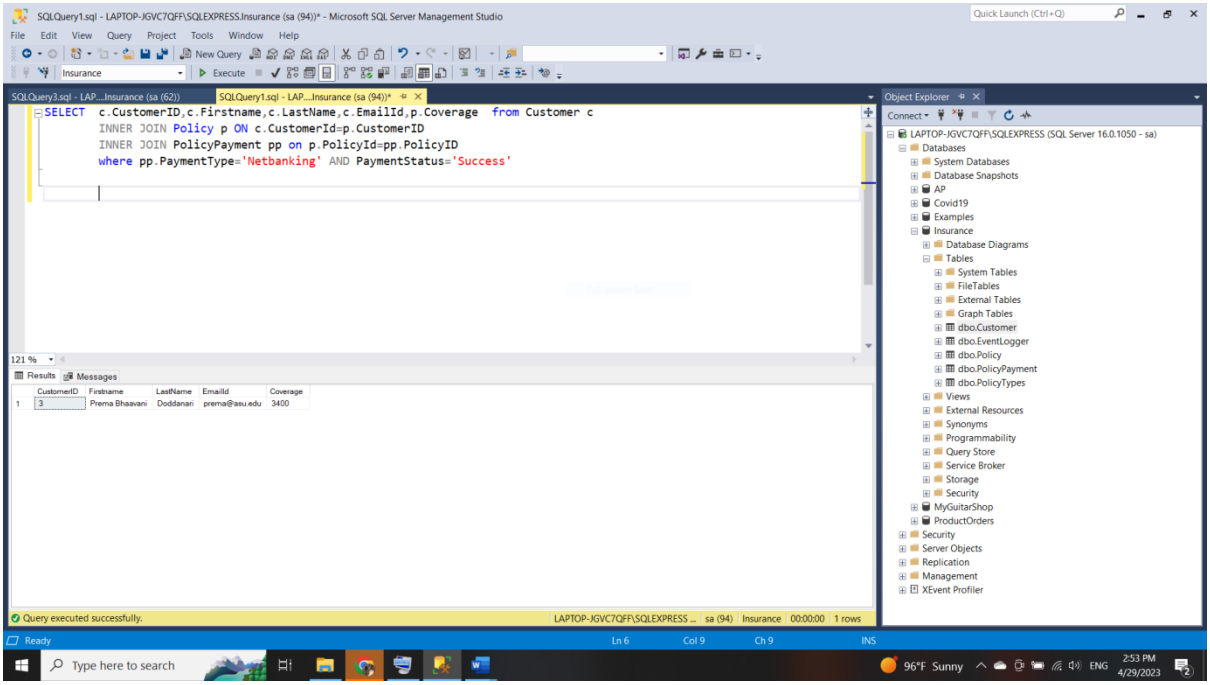
```
DECLARE @Max int=(Select MAX(Coverage) from policy)

Select C.CustomerID,C.Firstname,C.LastName,p.Coverage as [Maximum Coverage] from policy p INNER JOIN Customer c on c.CustomerId=p.CustomerID where Coverage=@Max
```



4) Find out the Customer details who has paid their policy payment through Netbanking also display the Policy coverage amount?

A)

```
SELECT  c.CustomerID,c.Firstname,c.LastName,c.EmailId,p.Coverage  from Customer c
      INNER JOIN Policy p ON c.CustomerId=p.CustomerID
      INNER JOIN PolicyPayment pp on p.PolicyId=pp.PolicyID
            where pp.PaymentType='Netbanking' AND PaymentStatus='Success'
```

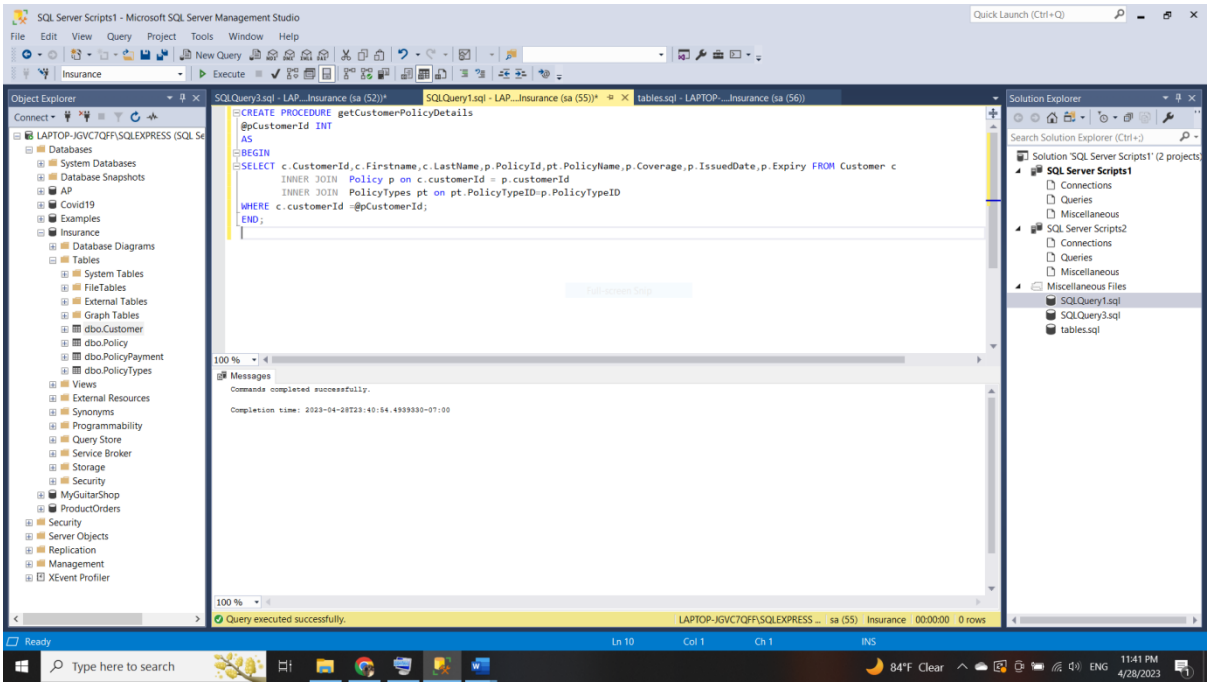


1) Is there any Stored procedure to retrieve Customer Policy details?

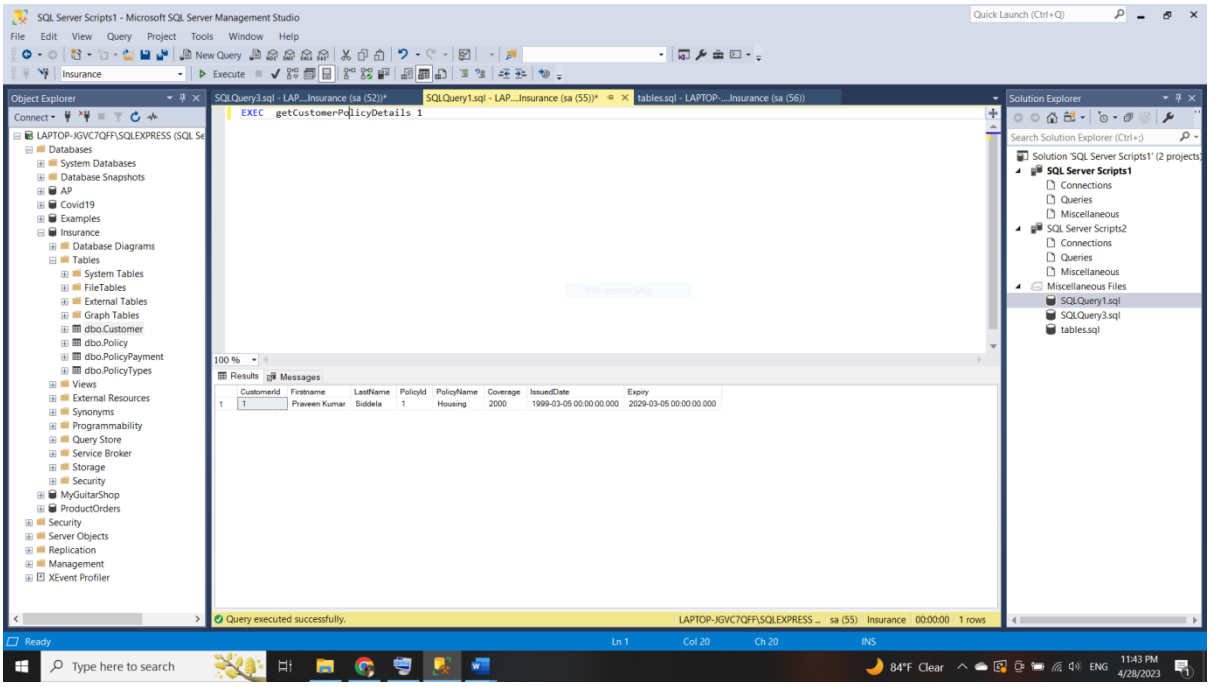
A)

```
CREATE PROCEDURE getCustomerPolicyDetails
    @pCustomerId INT
    AS
    BEGIN
    SELECT
    c.CustomerId,c.Firstname,c.LastName,p.PolicyId,pt.PolicyName,p.Coverage,p.IssuedDate,p.Expiry
    FROM Customer c
        INNER JOIN  Policy p on c.customerId = p.customerId
```

```
        INNER JOIN  PolicyTypes pt on pt.PolicyTypeID=p.PolicyTypeID
WHERE c.customerId =@pCustomerId;
END;
```



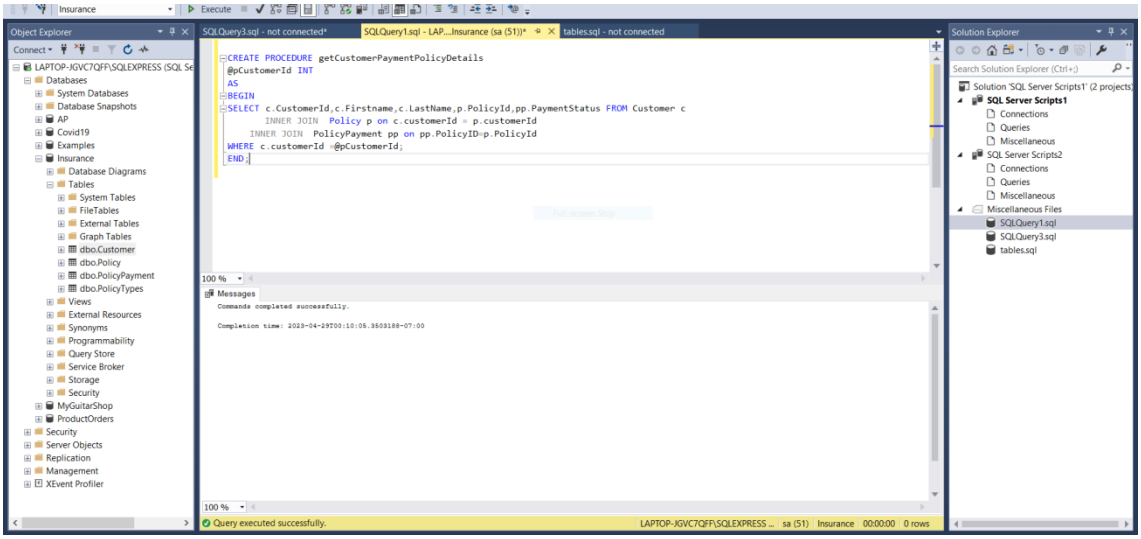

```
EXEC  getCustomerPolicyDetails 1
```



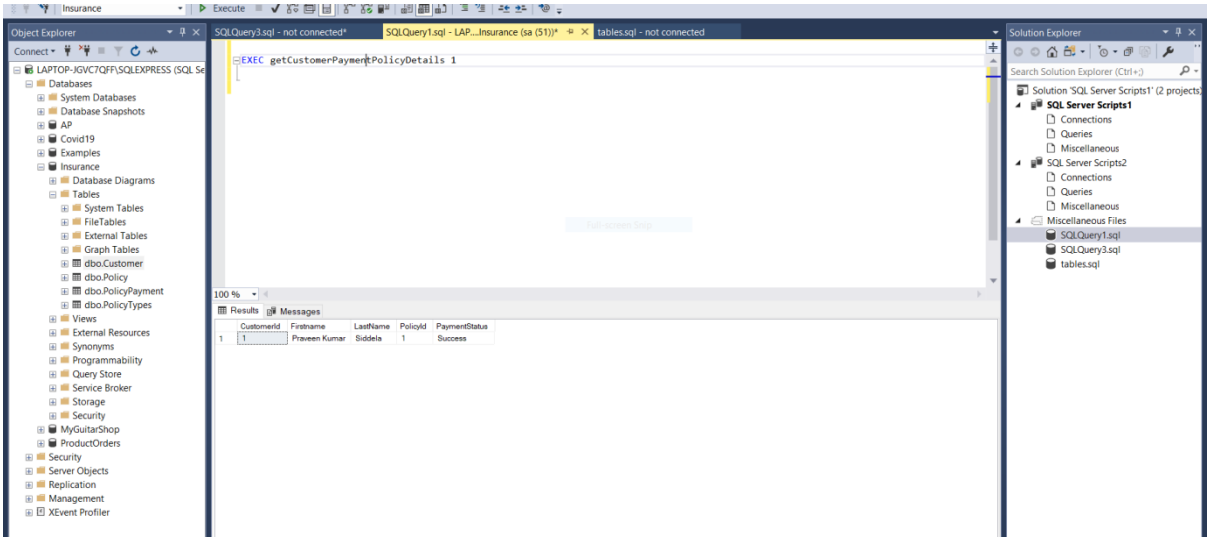
2) Is there any Stored procedure to retrieve Customer Payment Policy details?

A)

```
CREATE PROCEDURE getCustomerPaymentPolicyDetails

@pCustomerId INT
AS
BEGIN
SELECT c.CustomerId,c.Firstname,c.LastName,p.PolicyId,pp.PaymentStatus FROM Customer c
    INNER JOIN  Policy p on c.customerId = p.customerId
        INNER JOIN  PolicyPayment pp on pp.PolicyID=p.PolicyId
WHERE c.customerId =@pCustomerId;
END;
```




```
EXEC getCustomerPaymentPolicyDetails 1
```

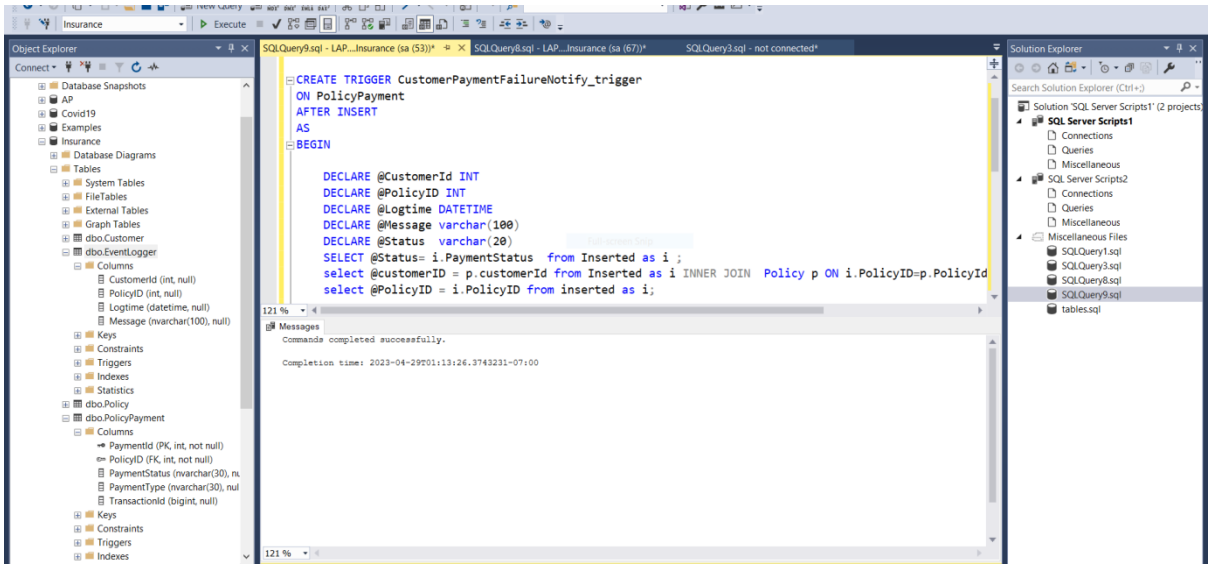


3) Is there any trigger that notify Customer if a Policy Payment fails?

A)

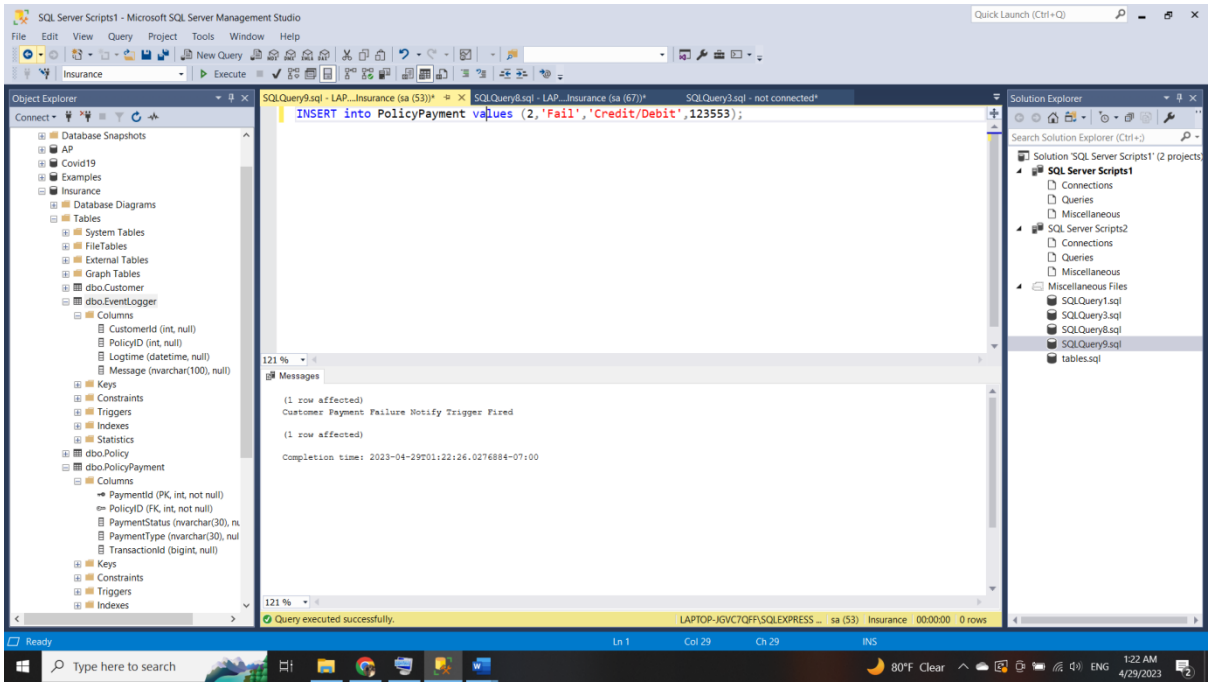
```
CREATE TRIGGER CustomerPaymentFailureNotify_trigger
ON PolicyPayment
AFTER INSERT
AS
BEGIN

    DECLARE @CustomerId INT
    DECLARE @PolicyID INT
    DECLARE @Logtime DATETIME
    DECLARE @Message varchar(100)
    DECLARE @Status  varchar(20)
    SELECT @Status= i.PaymentStatus  from Inserted as i ;
    select @customerID = p.customerId from Inserted as i INNER JOIN  Policy p ON i.PolicyID=p.PolicyId;
    select @PolicyID = i.PolicyID from inserted as i;

    IF @Status='Fail'
    BEGIN
  INSERT INTO EventLogger
  VALUES (@CustomerId, @PolicyID, GETDATE(),'Payment Failed');
    END
```

```
PRINT 'Customer Payment Failure Notify Trigger Fired'
END;
```



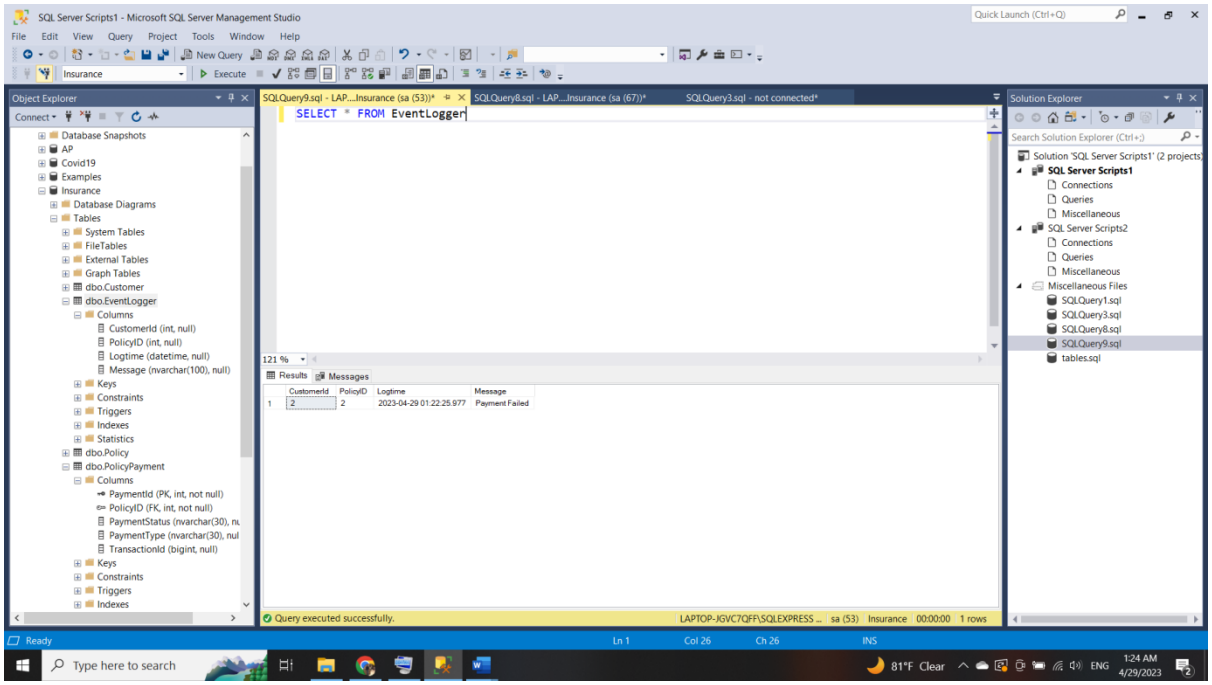
Customer Payment Failure Notify trigger will fires whenever there is an Failure insert happens into Policy Payment table, So inserting a record into Policy Payment table to check the trigger.

```
INSERT into PolicyPayment values (2,'Fail','Credit/Debit',123553);
```



Checking EventLogger Table

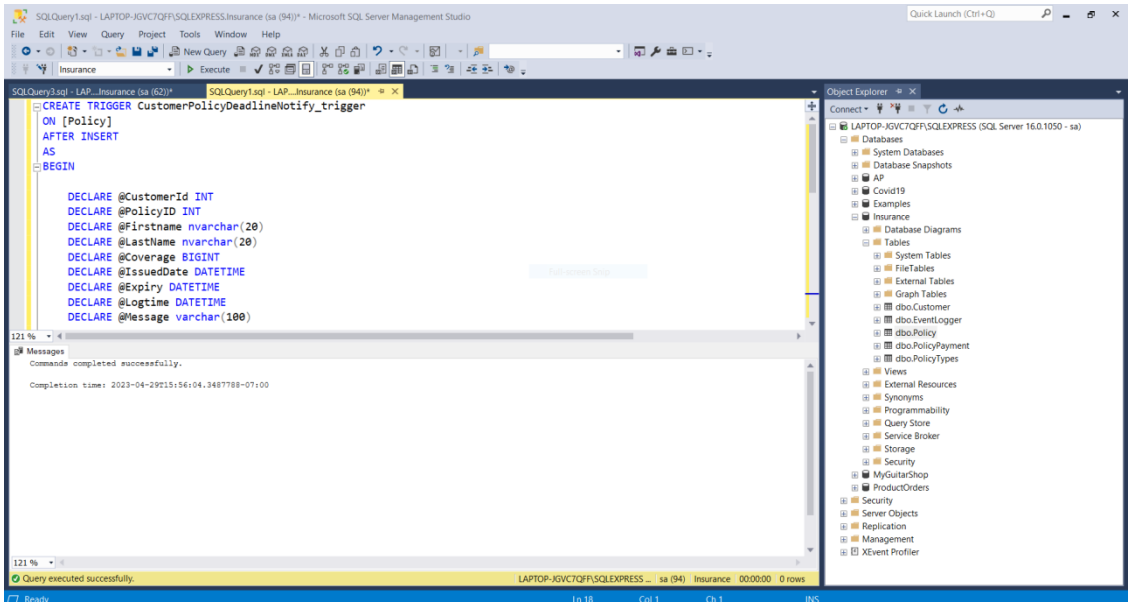
```
SELECT * FROM EventLogger
```



4) Is there any trigger that notifies the Customer policies which ends in a week?

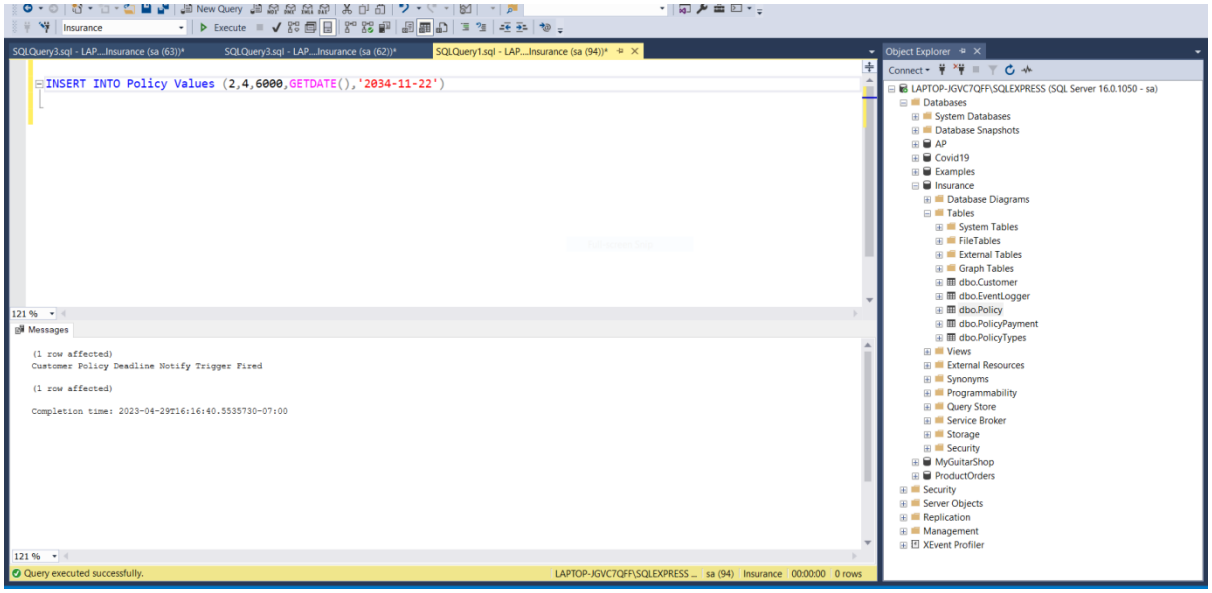
```
CREATE TRIGGER CustomerPolicyDeadlineNotify_trigger
ON [Policy]
AFTER INSERT
AS
BEGIN
        DECLARE @CustomerId INT
        DECLARE @PolicyID INT
        DECLARE @Firstname nvarchar(20)
        DECLARE @LastName nvarchar(20)
        DECLARE @Coverage BIGINT
        DECLARE @IssuedDate DATETIME
        DECLARE @Expiry DATETIME
        DECLARE @Logtime DATETIME
```

```
        DECLARE @Message varchar(100)

 INSERT INTO
CustomerPolicyDeadline(CustomerId,Firstname,LastName,PolicyID,Coverage,IssuedDate,Expiry,Logti
me,[Message])
SELECT
p.CustomerID,c.Firstname,c.LastName,p.PolicyId,p.Coverage,p.IssuedDate,p.Expiry,GETDATE(),'Poli
cy expiries in 7 days'
            from  Policy p INNER JOIN Customer c ON c.CustomerID=p.CustomerID
                WHERE p.Expiry=CONVERT(DATE,DATEADD(DAY,7,GETDATE()));


        PRINT 'Customer Policy Deadline Notify Trigger Fired'
END;
```



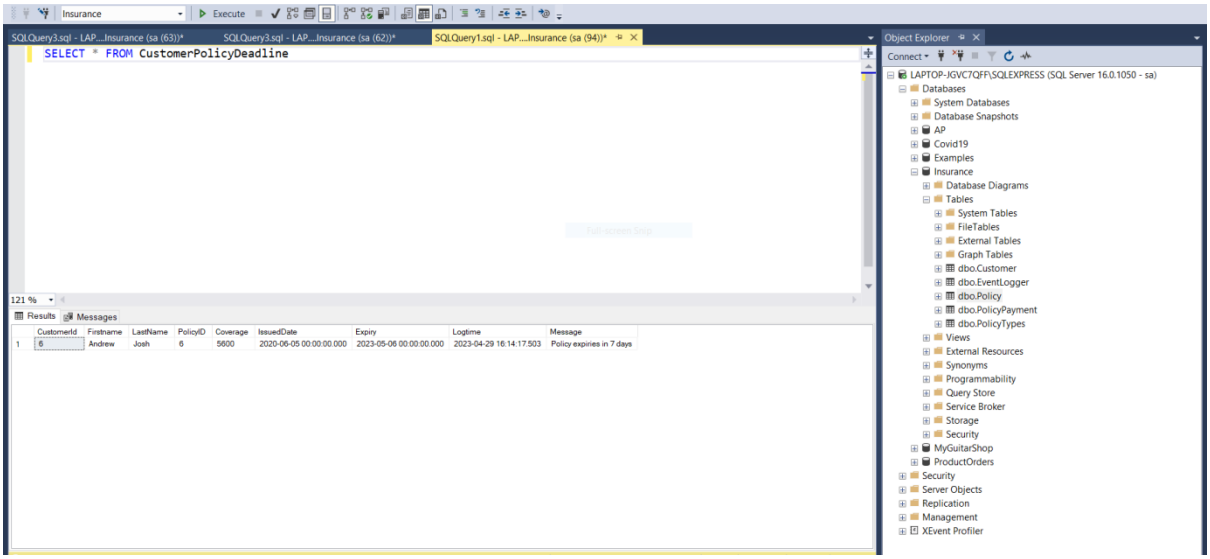
Customer policy Deadline Notify trigger will fires whenever there is an insert happens into Policy table it checks all the policies expiration dates and it get triggered if an Expiry date of policy is in 7 days. So, inserting a record into Policy table to check the trigger.

```
INSERT INTO Policy Values (2,4,6000,GETDATE(),'2034-11-22')
```



Checking CustomerPolicyDeadline Table

SELECT * FROM CustomerPolicyDeadline

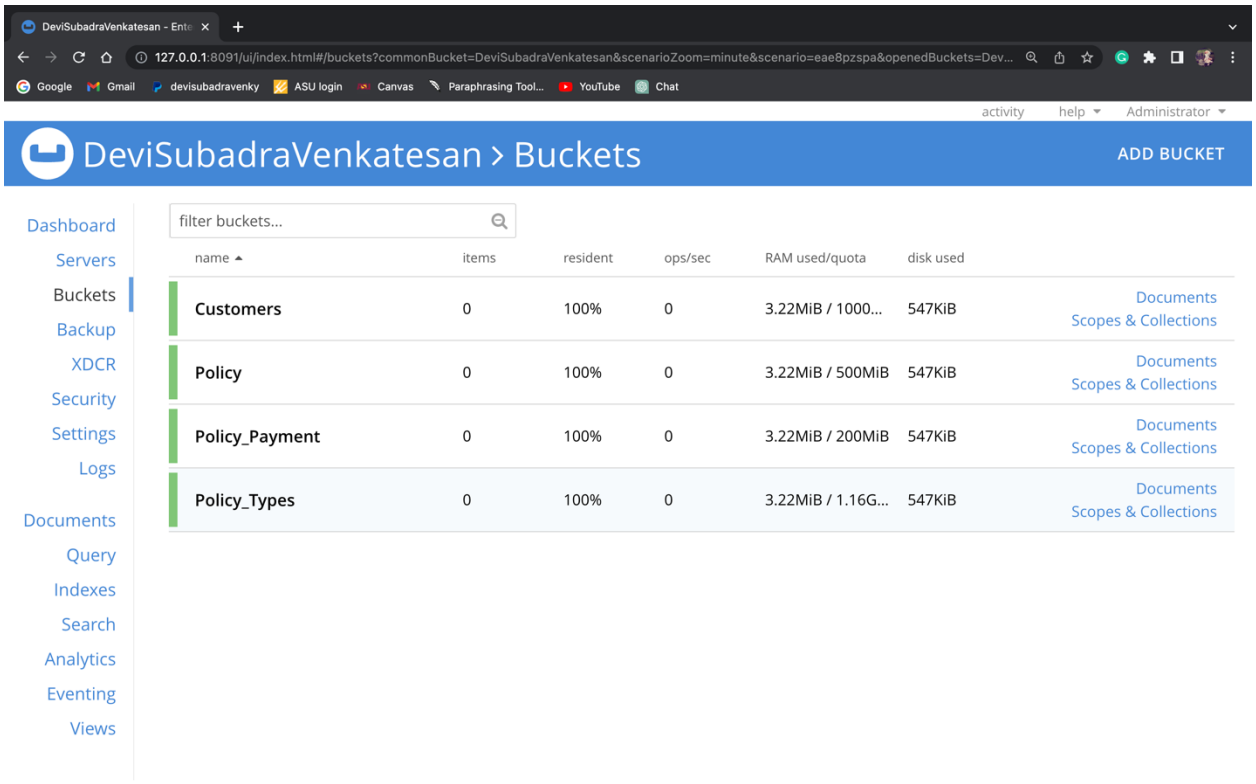


No SQL – Couchbase

We have created four buckets as mentioned below:

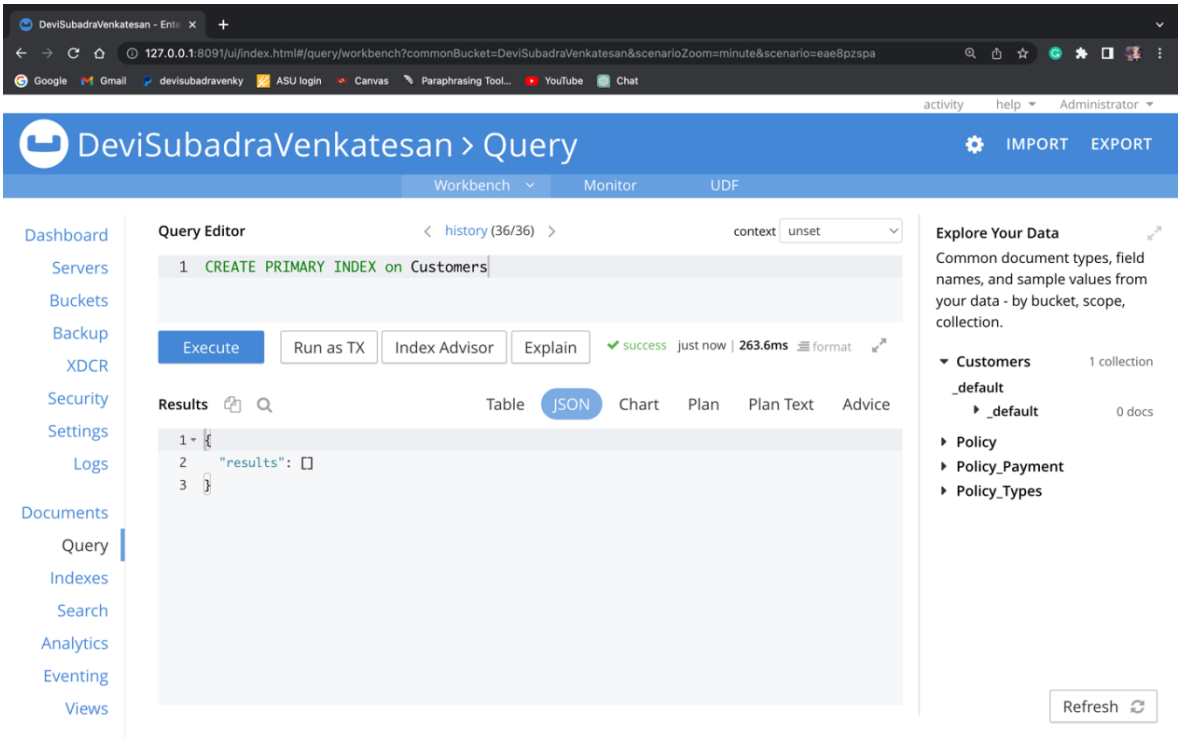
1. Customer

2. Policy
3. Policy Types
4. Policy Payment

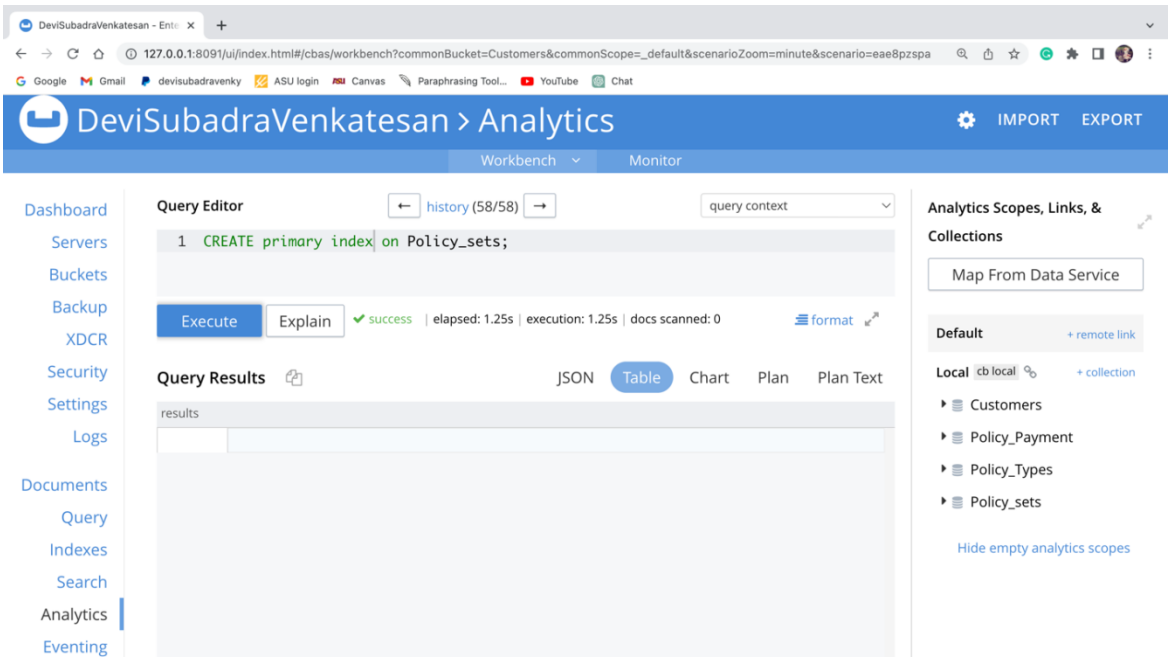


Creating Primary Indexes:

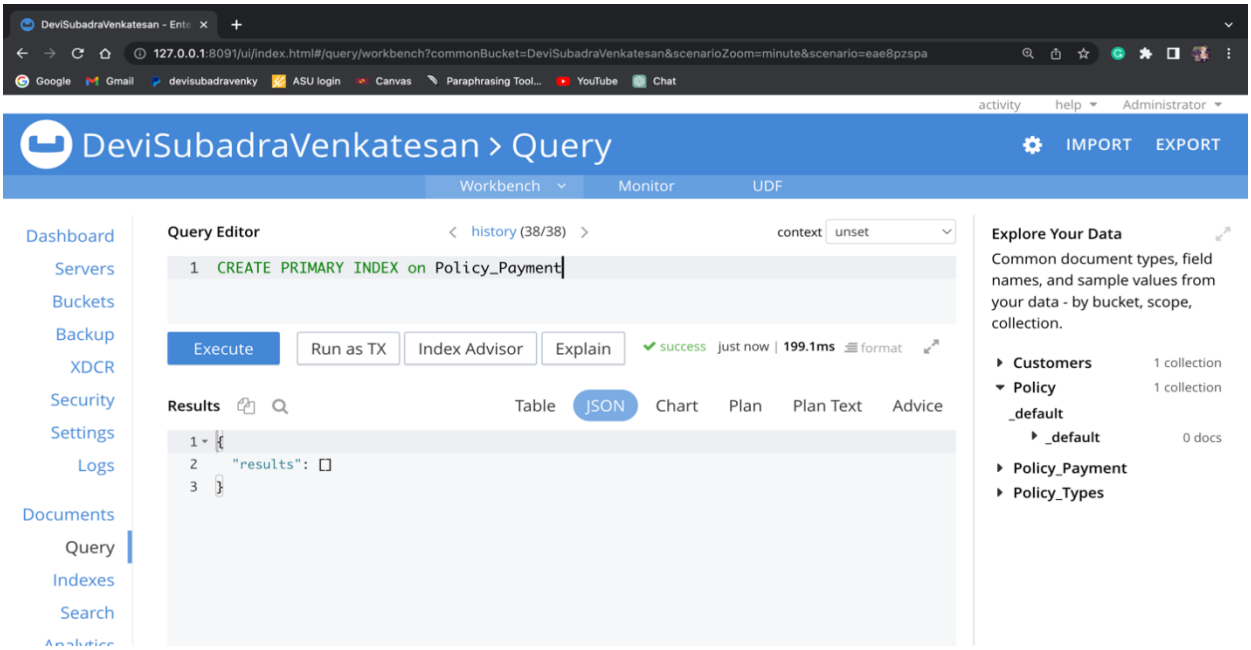
Query: CREATE PRIMARY INDEX on Customer



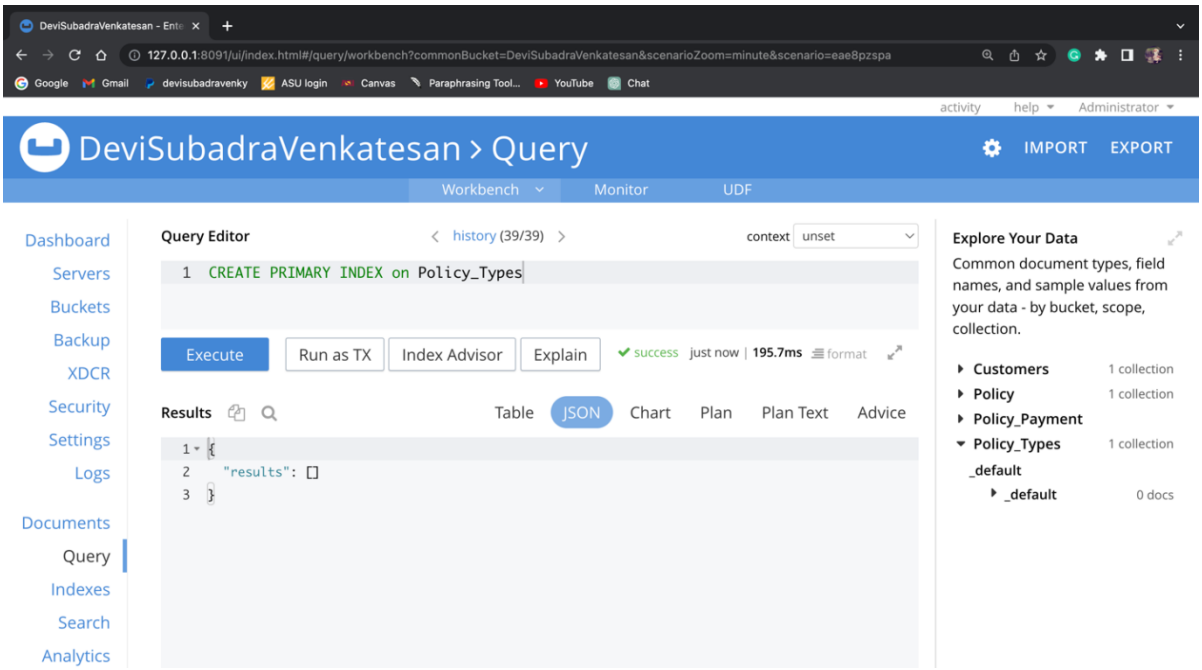
Query: CREATE PRIMARY INDEX on Policy_sets



Query: CREATE PRIMARY INDEX on Policy_Payment



Query: CREATE PRIMARY INDEX on Policy_Types



JSON Documents

Customer Bucket:

```
insert into customers(key,value)
values("1",      {
  "customerId": "001",
  "customerName": "Devi",
  "phoneNumber": "123-456-7890"
}),("2",{
"customerId": "002",
  "customerName": "Praveen",
  "phoneNumber": "555-555-1212"
}),("3",{
"customerId": "003",
  "customerName": "Prema",
  "phoneNumber": "888-123-4567"
}),("4",{
"customerId": "004",
  "customerName": "Sindhu",
  "phoneNumber": "555-123-4567"
}),("5",{
"customerId": "005",
  "customerName": "Sumanth",
  "phoneNumber": "777-555-1234"
}),("6",{
"customerId": "006",
  "customerName": "Andrew",
  "phoneNumber": "123-456-7890"
}),("7",{
```

"customerId": "007",
"customerName": "Sushmita",
"phoneNumber": "123-456-7890"
})

Policy_sets Bucket:

insert into Policy_sets (key,value)
values("1", {
"customerId":"001",
"Amount": "2000",
"Issued date":"03/05/1999",
"Expiry date": "03/05/2029"
}),("2",{
"customerId":"002",
"Amount": 2500,
"Issued date":"09/24/2001",
"Expiry date": "09/24/2031"
}),("3",{
"customerId":"003",
"Amount": "3400",
"Issued date":"11/22/2014",
"Expiry date": "11/22/2054"
}),("4",{
"customerId":"004",
"Amount": "5200",
"Issued date":"08/13/2005",
"Expiry date": "08/13/2035"
}),("5",{

```
  "customerId":"005",
 "Amount":"2200",
"Issued date":"02/15/2010",
"Expiry date":  "02/15/2040"
}),("6",{
  "customerId":"006",
 "Amount": "1000",
"Issued date":"03/05/2011",
"Expiry date": "07/22/2030"
}),("7",{
  "customerId": "007",
 "Amount": "1050",
"Issued date":"03/12/2003",
"Expiry date": "09/07/2030"
})
```

Policy_Payment Bucket:

```
insert into Policy_Payment (key,value)
values("1",      {
  "id": 1,
 "policy_id": 123,
 "amount": 500,
 "method": "Cash",
 "Transaction id": 9127,
 "payment_date": "2022-04-15",
 "status": "Success"
}),("2",{
  "id": 2,
```

```
  "policy_id": 456,
  "amount": 250,
  "method": "Credit/Debit",
  "Transaction id": 7389,
  "payment_date": "2022-05-01",
  "status": "Success"
},("3",{
 "id": 3,
  "policy_id": 789,
  "amount": 750,
  "method": "NetBanking",
  "Transaction id": 2739,
  "payment_date": "2022-06-15",
  "status": "Success"
},("4",{
"id": 4,
  "policy_id": 234,
  "amount": 1000,
  "method": "Cash",
  "Transaction id": 1935,
  "payment_date": "2022-07-01",
  "status": "Success"
},("5",{
 "id": 5,
  "policy_id": 500,
  "amount": 1055,
  "method": "Credit/Debit",
  "Transaction id": 4792,
```

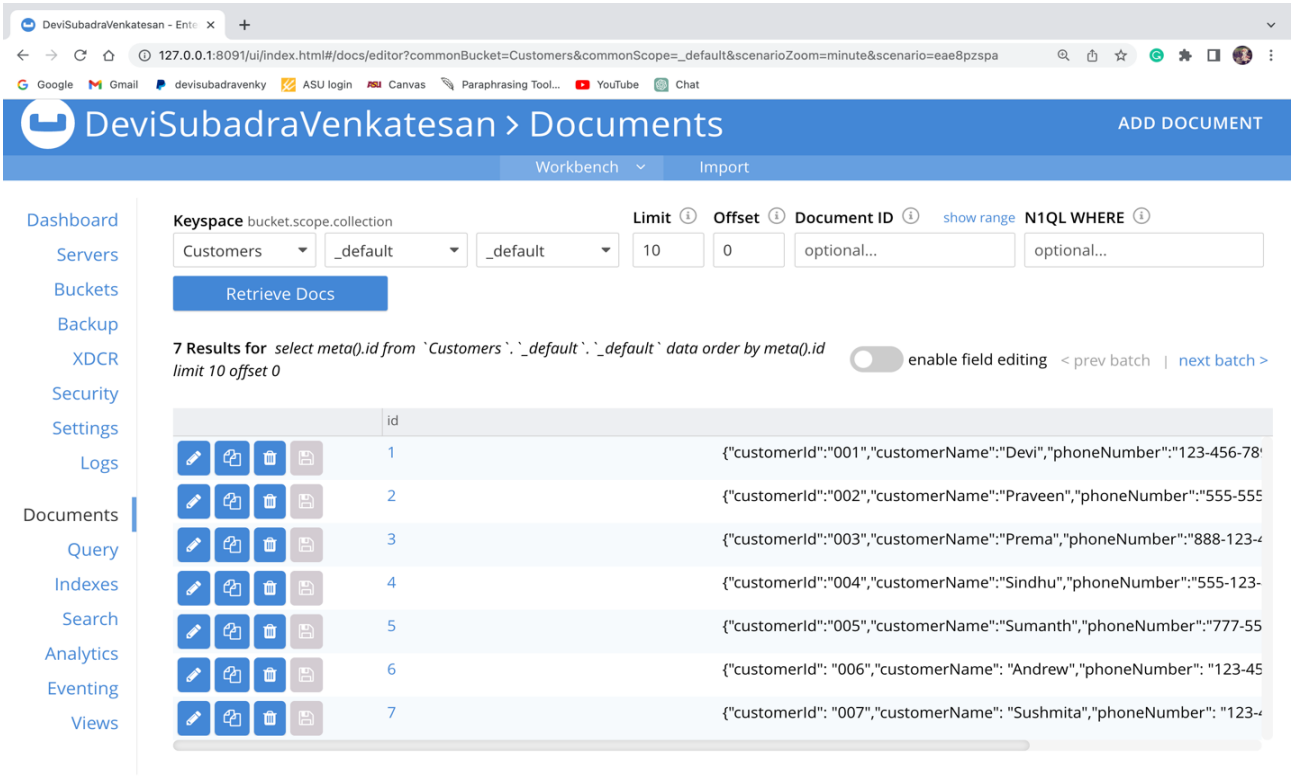
```
  "payment_date": "2022-08-11",
  "status": "Fail"
 }),("6",{
  "id": 6,
  "policy_id": 340,
  "amount": 1055,
  "method": "Credit/Debit",
  "Transaction id": 6500,
  "payment_date": "2022-08-11",
  "status": "Success"
 }),("7",{
 "id": 7,
  "policy_id": 110,
  "amount": 1055,
  "method": "NetBanking",
  "Transaction id": 2794,
  "payment_date": "2022-08-11",
  "status": "Fail"
 })
```

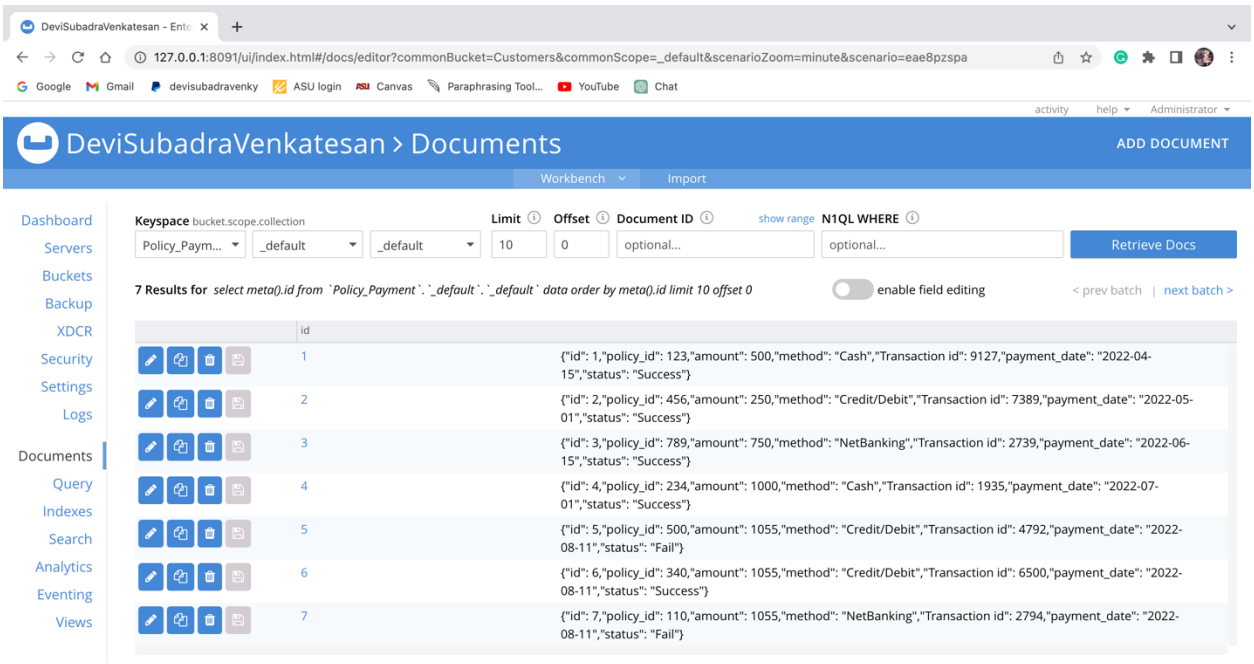
Policy_Types Bucket:

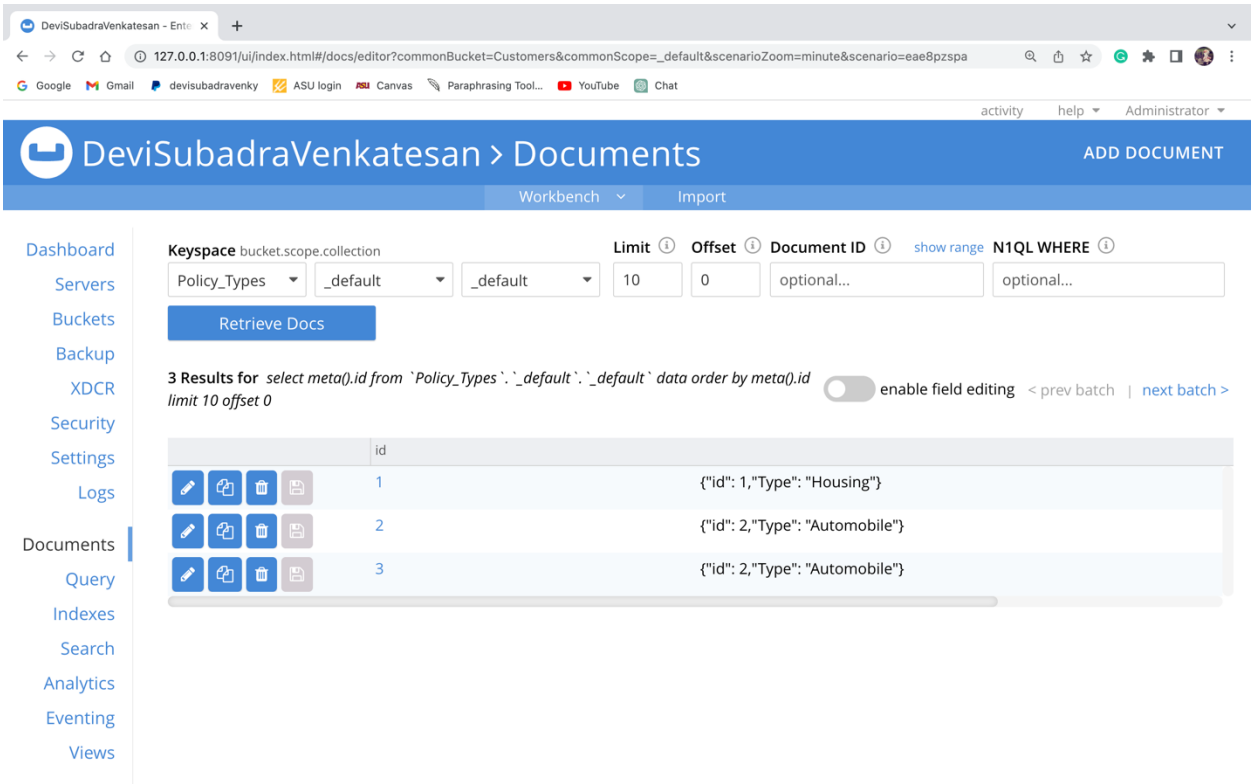
```
insert into Policy_Types (key,value)
values("O1",
        {
        "id": 1,
      "Type": "Housing"
}),("O2",{
  "id": 2,
```

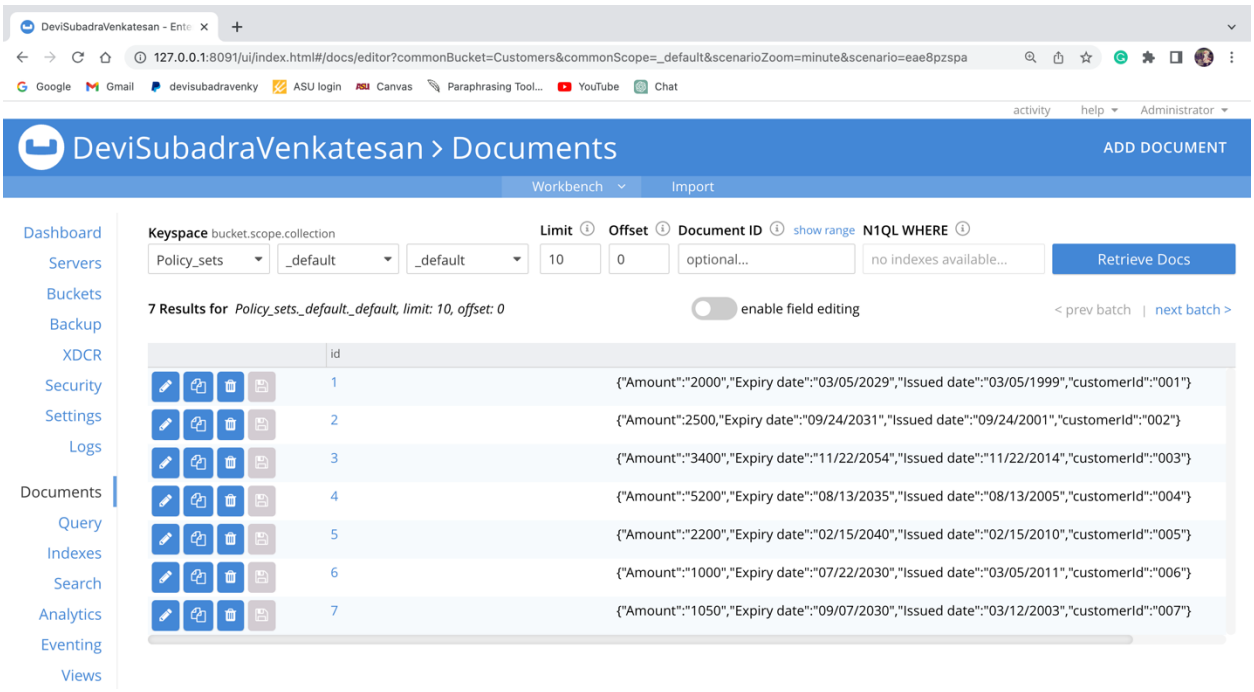
```
  "Type": "Automobile"
}
),("O3", {
  "id": 2,
  "Type": "Automobile"
}
```


Documents




DeviSubadraVenkatesan - Ente
127.0.0.1:8091/ui/index.html#/docs/editor?commonBucket=Customers&commonScope=_default&scenarioZoom=minute&scenario=eae8pzspa
Google
Gmail
devisubadravenky
ASU login
Canvas
Paraphrasing Tool...
YouTube
Chat
activity
help
Administrator
DeviSubadraVenkatesan > Documents
ADD DOCUMENT
Workbench
Import
Dashboard
Servers
Buckets
Backup
XDCR
Security
Settings
Logs
Documents
Query
Indexes
Search
Analytics
Eventing
Views
Keyspace bucket.scope.collection
Policy_Paym...
_default
_default
Limit
10
Offset
0
Document ID
optional...
show range
N1QL WHERE
optional...
Retrieve Docs
7 Results for select meta().id from `Policy_Payment`.`_default`.`_default` data order by meta().id limit 10 offset 0
enable field editing
< prev batch | next batch >
id
1 {"id": 1,"policy_id": 123,"amount": 500,"method": "Cash","Transaction id": 9127,"payment_date": "2022-04-15","status": "Success"}
2 {"id": 2,"policy_id": 456,"amount": 250,"method": "Credit/Debit","Transaction id": 7389,"payment_date": "2022-05-01","status": "Success"}
3 {"id": 3,"policy_id": 789,"amount": 750,"method": "NetBanking","Transaction id": 2739,"payment_date": "2022-06-15","status": "Success"}
4 {"id": 4,"policy_id": 234,"amount": 1000,"method": "Cash","Transaction id": 1935,"payment_date": "2022-07-01","status": "Success"}
5 {"id": 5,"policy_id": 500,"amount": 1055,"method": "Credit/Debit","Transaction id": 4792,"payment_date": "2022-08-11","status": "Fail"}
6 {"id": 6,"policy_id": 340,"amount": 1055,"method": "Credit/Debit","Transaction id": 6500,"payment_date": "2022-08-11","status": "Success"}
7 {"id": 7,"policy_id": 110,"amount": 1055,"method": "NetBanking","Transaction id": 2794,"payment_date": "2022-08-11","status": "Fail"}

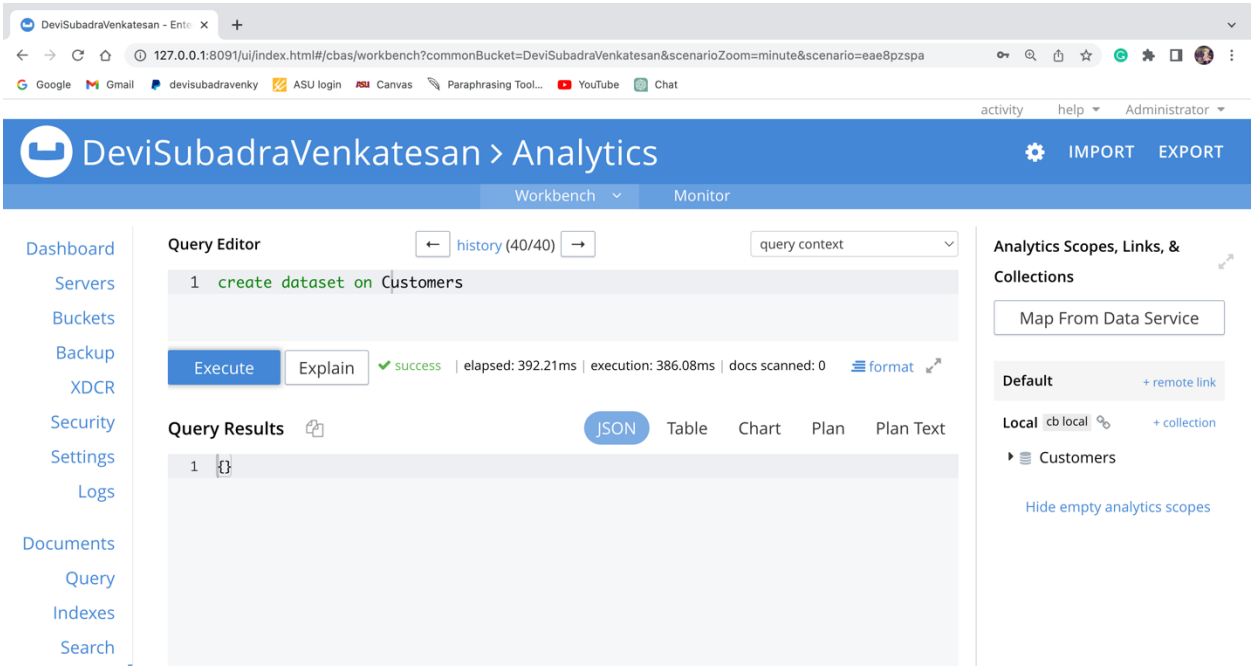

DeviSubadraVenkatesan - Ente
127.0.0.1:8091/ui/index.html#/docs/editor?commonBucket=Customers&commonScope=_default&scenarioZoom=minute&scenario=eae8pzspa
Google
Gmail
devisubadravenky
ASU login
Canvas
Paraphrasing Tool...
YouTube
Chat
activity
help
Administrator
DeviSubadraVenkatesan > Documents
ADD DOCUMENT
Workbench
Import
Dashboard
Servers
Buckets
Backup
XDCR
Security
Settings
Logs
Documents
Query
Indexes
Search
Analytics
Eventing
Views
Keyspace bucket.scope.collection
Policy_Types
_default
_default
Limit
10
Offset
0
Document ID
optional...
show range
N1QL WHERE
optional...
Retrieve Docs
3 Results for select meta().id from `Policy_Types`.`_default`.`_default` data order by meta().id limit 10 offset 0
enable field editing
< prev batch | next batch >
id
1 {"id": 1,"Type": "Housing"}
2 {"id": 2,"Type": "Automobile"}
3 {"id": 2,"Type": "Automobile"}



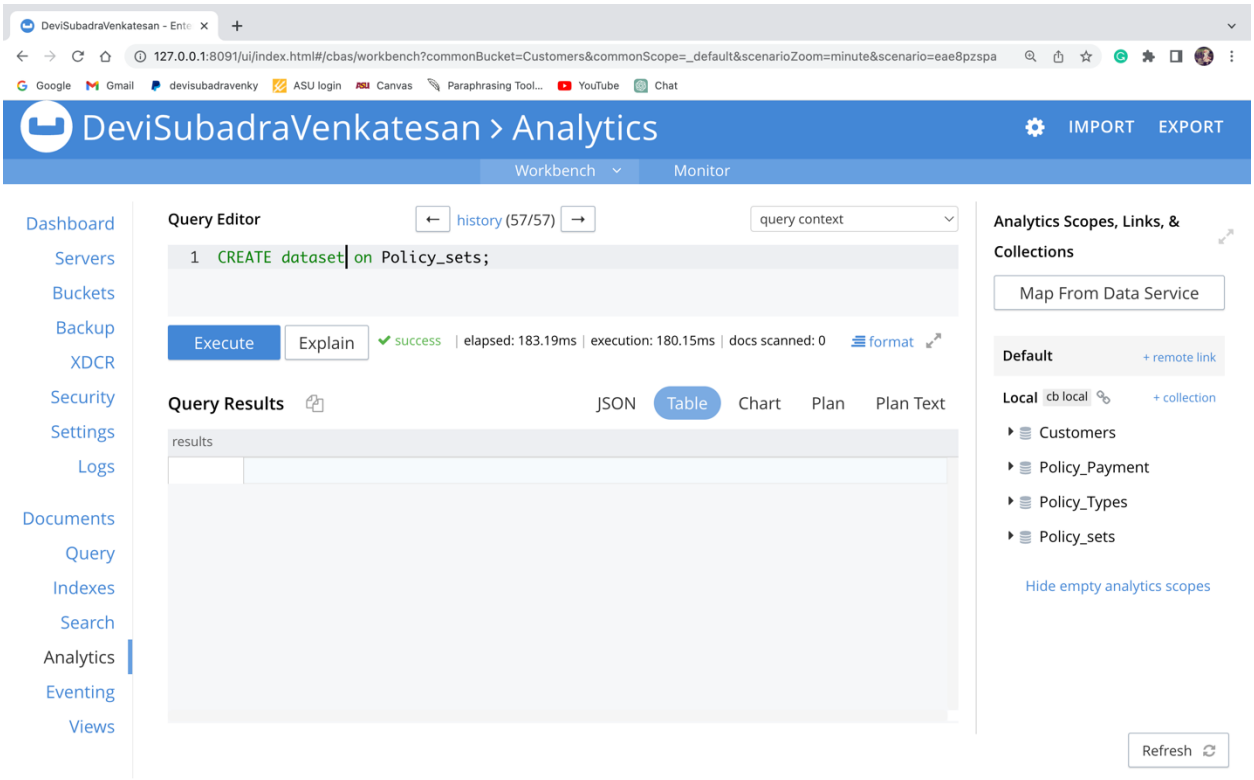
Query Services and Analytics Services

Creation of Datasets:

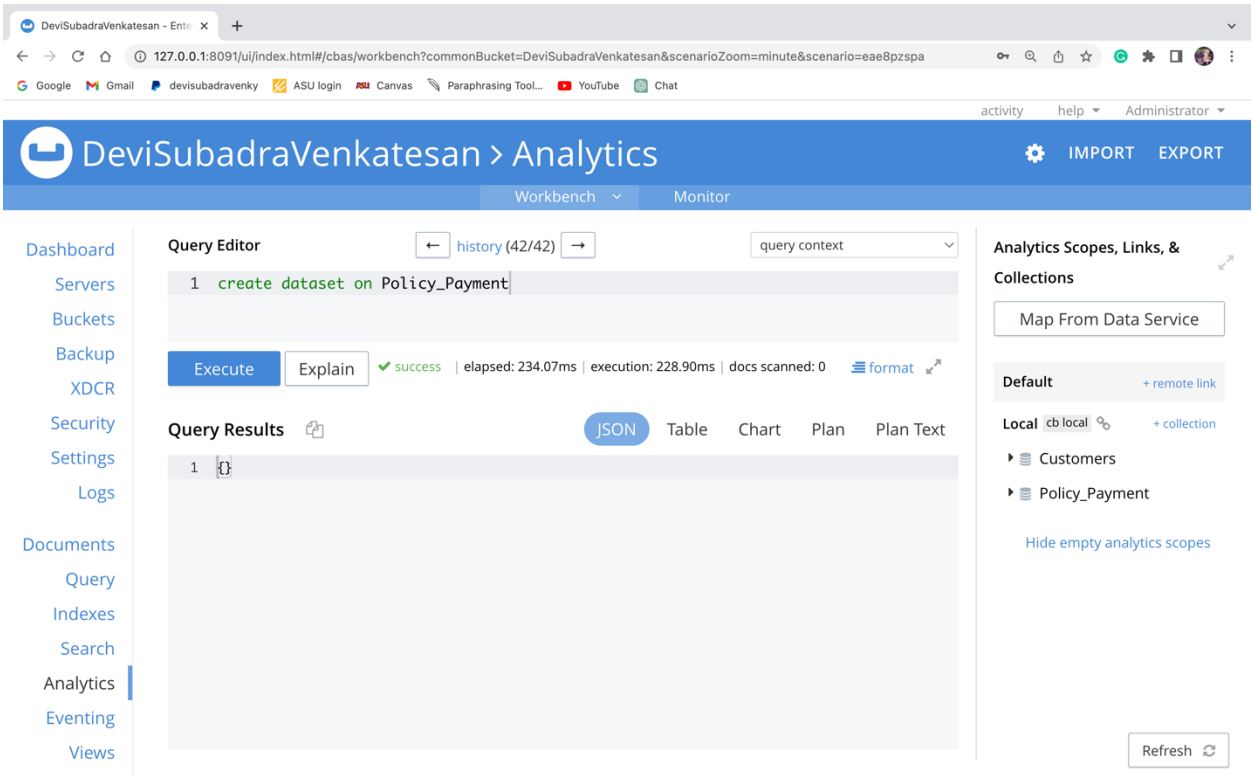
Query: CREATE DATASET on Customers



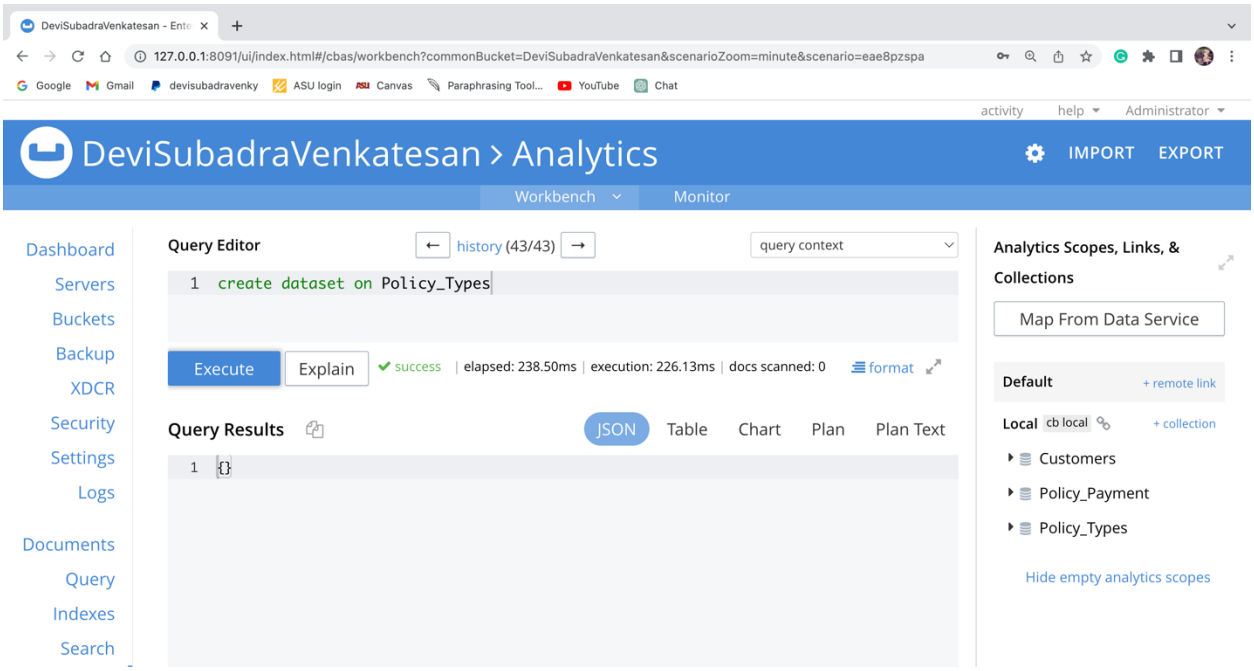
Query: CREATE DATASET on Policy_sets



Query: CREATE DATASET on Policy_Payment



Query: CREATE DATASET on Policy_Types

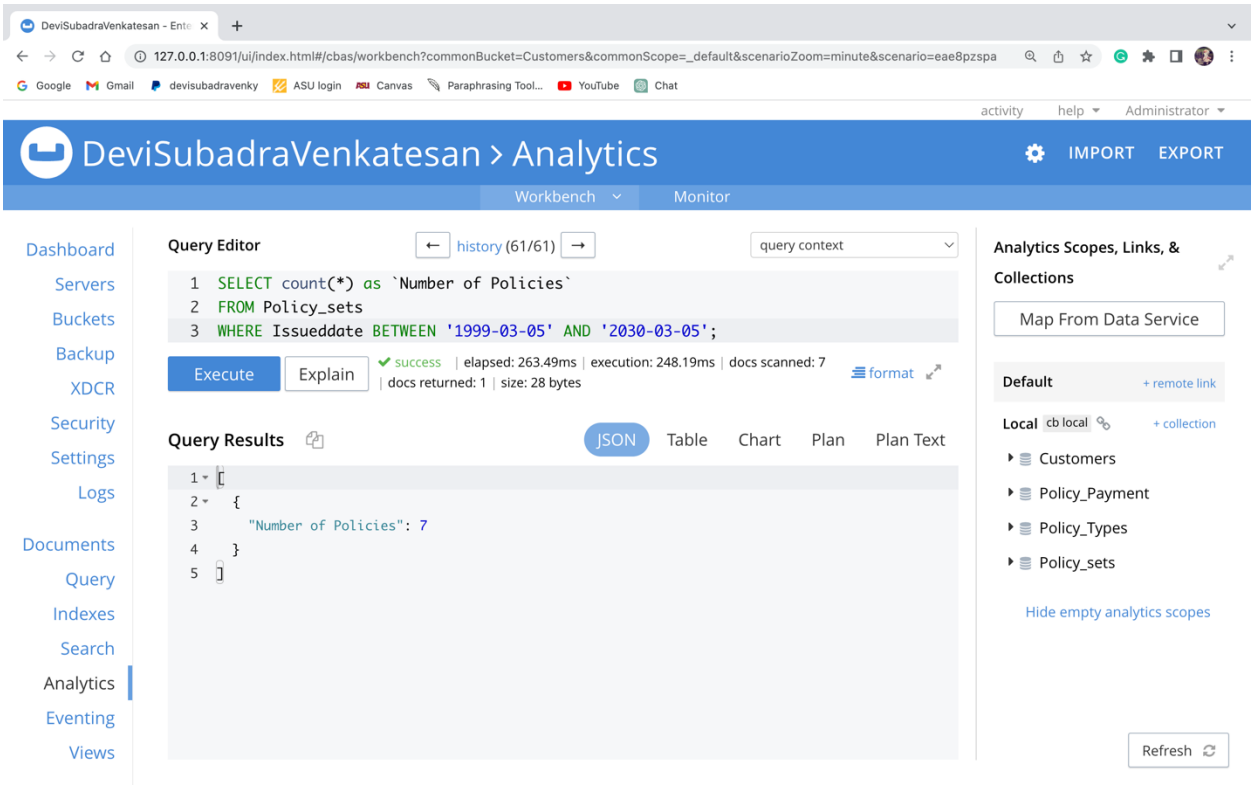


Queries in Couchbase

1. **How many policies were issued in a particular time period?**

```
SELECT count(*) as `Number of Policies`
FROM Policy_sets
WHERE Issueddate BETWEEN '1999-03-05' AND '2030-03-05';
```

Screenshot:

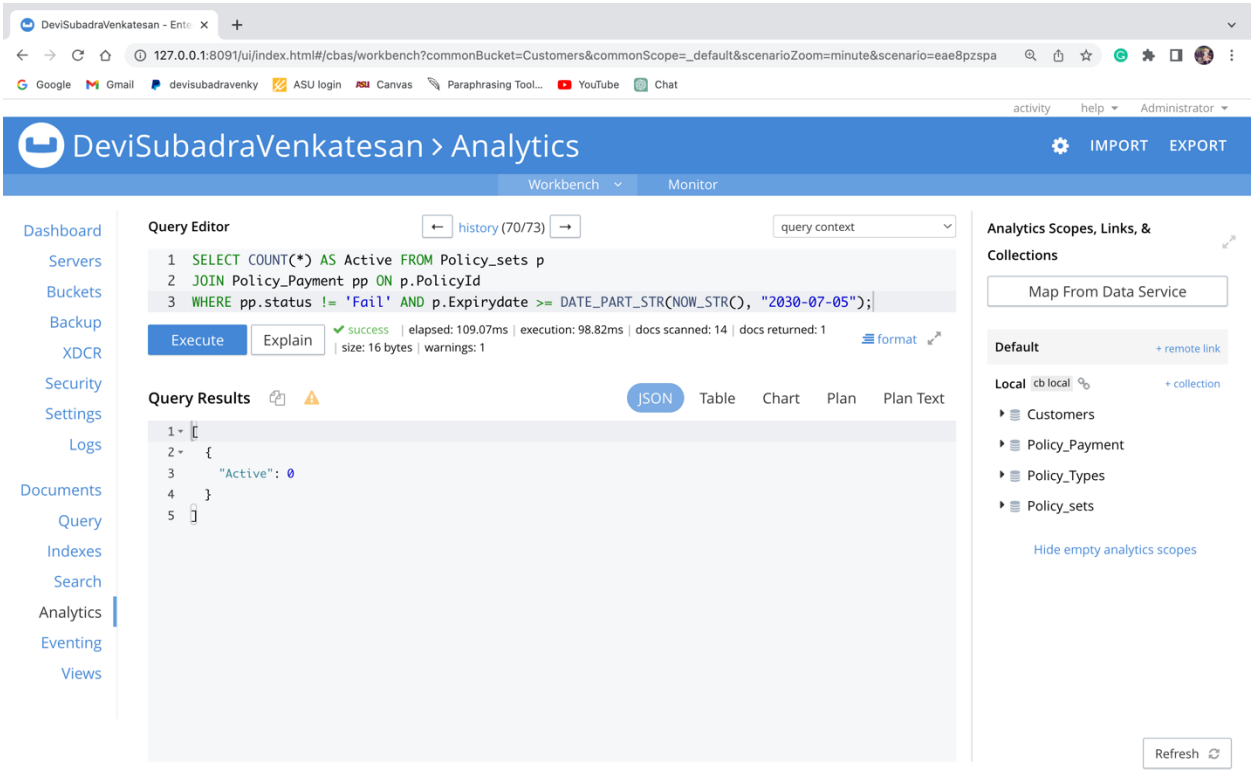


2) How many Active Policies are there in the system?

A)

SELECT COUNT(*) AS Active FROM Policy_sets p
JOIN Policy_Payment pp ON p.PolicyId
WHERE pp.status != 'Fail' AND p.Expirydate >= DATE_PART_STR(NOW_STR(), "2030-07-05");

Screenshot:



3)Find out the Customer details who has paid their policy payment through Netbanking also display the Policy coverage amount?

Query:

SELECT c.customerId, c.customerName, p.Amount, pp.method

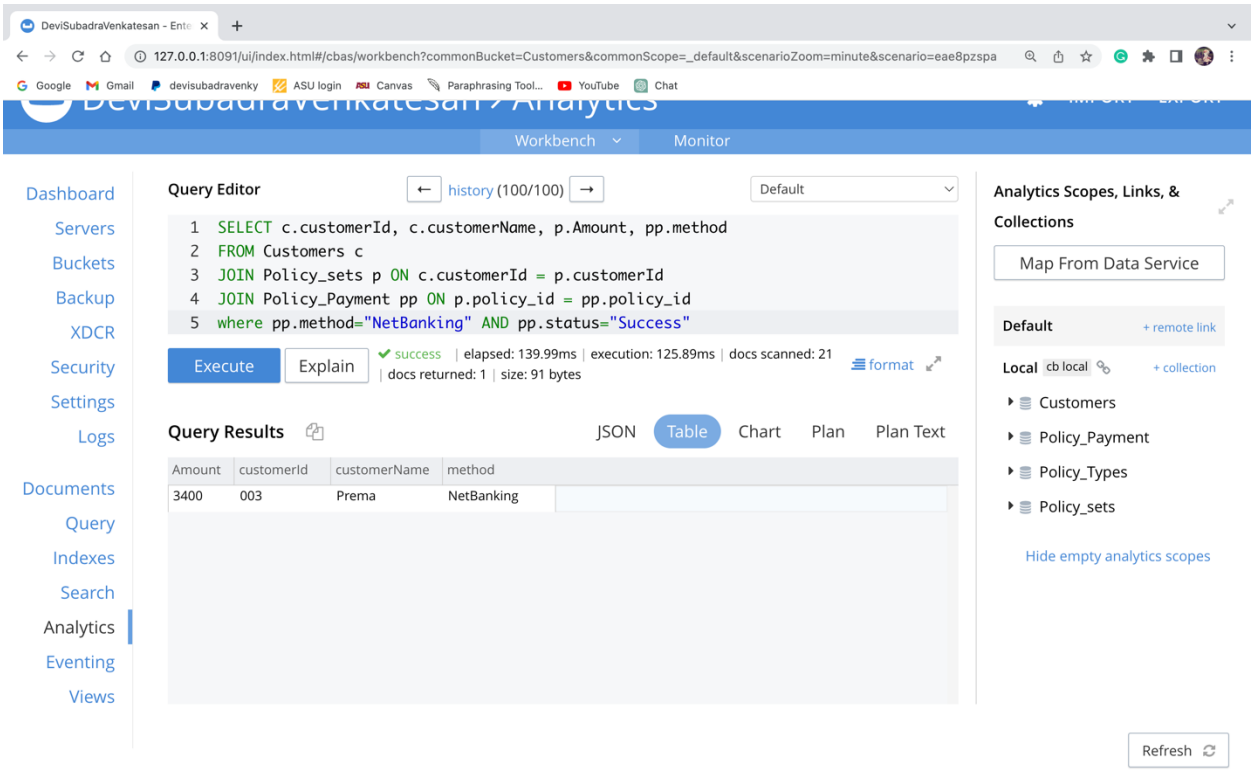
FROM Customers c

JOIN Policy_sets p ON c.customerId = p.customerId

JOIN Policy_Payment pp ON p.policy_id = pp.policy_id

where pp.method="NetBanking" AND pp.status="Success"

Screenshot:



4) Which Customer has the Maximum coverage amount?

Query:

Select c.customerId,c.customerName,MAX(p.Amount) as MaximumCoverage

from Policy_sets p

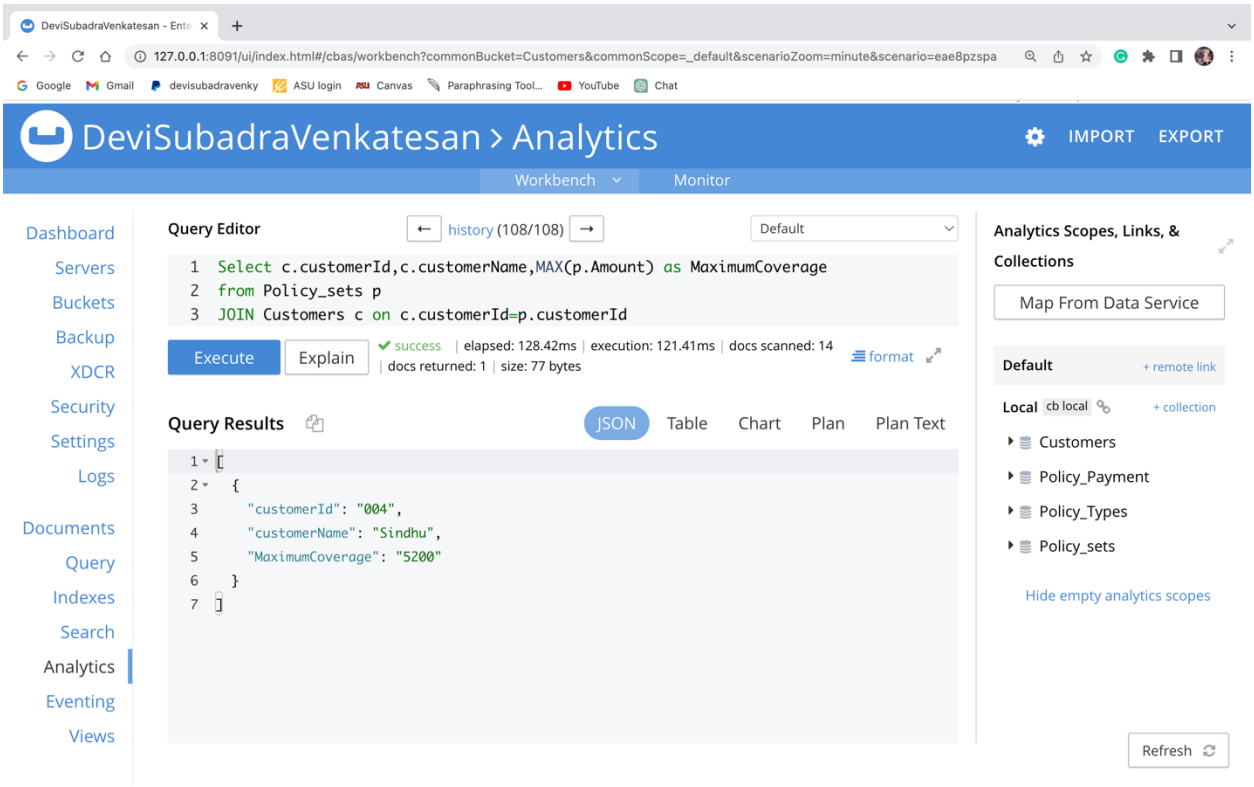
JOIN Customers c on c.customerId=p.customerId

GROUP BY c.customerId,c.customerName

ORDER BY MaximumCoverage DESC

LIMIT 1;

Screenshot:


DeviSubadraVenkatesan - Ente
127.0.0.1:8091/ui/index.html#/cbas/workbench?commonBucket=Customers&commonScope=_default&scenarioZoom=minute&scenario=eae8pzspa
Google
Gmail
devisubadravenky
ASU login
Canvas
Paraphrasing Tool...
YouTube
Chat
DeviSubadraVenkatesan > Analytics
IMPORT
EXPORT
Workbench
Monitor
Dashboard
Servers
Buckets
Backup
XDCR
Security
Settings
Logs
Documents
Query
Indexes
Search
Analytics
Eventing
Views
Query Editor
history (108/108)
Default
1 Select c.customerId,c.customerName,MAX(p.Amount) as MaximumCoverage
2 from Policy_sets p
3 JOIN Customers c on c.customerId=p.customerId
Execute
Explain
success | elapsed: 128.42ms | execution: 121.41ms | docs scanned: 14 | docs returned: 1 | size: 77 bytes
format
Query Results
JSON
Table
Chart
Plan
Plan Text
1 [
2 {
3 "customerId": "004",
4 "customerName": "Sindhu",
5 "MaximumCoverage": "5200"
6 }
7]
Analytics Scopes, Links, & Collections
Map From Data Service
Default
+ remote link
Local
cb local
+ collection
Customers
Policy_Payment
Policy_Types
Policy_sets
Hide empty analytics scopes
Refresh

Summary

The Insurance Policy Database Management System (IPDMS) is a software application that helps insurance companies manage their policies, claims, compliance, and other related activities more efficiently. The system is designed to streamline the insurance process, reduce operational costs, and improve customer service.

IPDMS provides a range of features and capabilities, including policy management, claims processing, reporting and analytics, compliance management, and customer relationship management. It can be customized to meet the specific needs of an insurance company, allowing insurers to tailor the software to fit their unique business requirements.

The importance of an IPDMS cannot be overstated, as it can streamline processes, ensure compliance, and manage risks effectively. By automating tasks such as data entry and claims processing, insurers can enhance operational efficiency, reduce errors, and optimize resource utilization, leading to cost savings.

Moreover, IPDMS's ability to improve customer service is another significant advantage. Automated claims processing with real-time updates can result in better customer satisfaction and retention rates, contributing to the long-term success of the company.

The main agenda of this Insurance Policy Database Management System (IPDMS)is to allocate sustained customized Policies by checking whether any prior accidents, health or natural calamities would have occurred for the vehicle, person and house respectively. The system ensures the entire system is recorded in a suitable manner.

Lastly, an insurance management system can help insurers manage risks associated with underwriting policies by providing real-time data analysis and identifying potential losses. In

summary, an insurance management system is a necessary tool for insurers to stay competitive in today's fast-paced business environment, as it can improve operational efficiency, customer service, compliance, risk management, and contribute to long-term success.

Conclusion

In conclusion, the efficient handling of passenger luggage is critical for the success of airlines and airports. Ultimately, the adoption of Automated Baggage Handling Systems can improve the way airports handle passenger luggage, leading to a better overall travel experience for everyone involved. Regarding further work, it would be beneficial to continue researching and developing innovative luggage handling solutions that can further streamline the process and reduce the likelihood of lost or delayed luggage. Additionally, there is a need to explore the potential environmental impact of luggage handling systems and find ways to make them more sustainable. In terms of the value of NoSQL and SQL in combination, both have their strengths and weaknesses. NoSQL databases are better suited for handling unstructured or semi-structured data and can handle large volumes of data with high velocity. In contrast, SQL databases are better suited for handling structured data and can perform complex queries efficiently. By combining the two, organizations can leverage the strengths of each database type and create a more comprehensive and flexible data management system. This can help organizations to better manage their data and gain valuable insights from it

Reference

1. Terry Haplin, (2015). Object role modeling fundamentals: A Practical Guide to Data Modeling with ORM. Technics Publications
2. Bryan Syverson, Joel Murach , Murach's SQL Server 2016 for developers.