Northeastern University
College of Engineering

IE 6750 Data Warehousing and Integration

Project Title

Policy Lapsation in Life Insurance

Milestone 3

Group 4

Aayush Amrute
Saurabh Chavan
Paritosh Vyawahare

amrute.a@northeastern.edu

chavan.sau@northeastern.edu

vyawahare.p@northeastern.edu

Submission Date: 02-19-2025

Table of Contents

Business Need for Data Warehouse

We are building a centralized data warehouse to streamline life insurance policy management and improve decision-making. This will enable better tracking of policy lapsation, customer retention, payment behaviours, and agent/branch performance. By consolidating data from multiple sources like web sales, agents, and banks, we ensure accurate insights and proactive strategies. The data warehouse will help analyse customer segments, optimize payment channels, and evaluate agent performance, ultimately reducing lapsation rates and enhancing business efficiency.

Data Warehouse Design Overview

Our data warehouse follows a constellation schema with three major fact tables:

Customer Retention Fact Table : Tracks policy lapsation trends, reinstatements, and premium payments to identify high-risk customers.

Agent & Branch Performance Fact Table : Evaluates agent performance, branch-level policy issuance, commissions earned.

Payment Analysis Fact Table : Analyzes payment behavior, success/failure rates, and the impact of autopay vs. non-autopay on retention.

These fact tables connect to multiple dimension tables to provide a holistic view of policy performance, customer behavior, and operational efficiency.

Dimensions & Hierarchies

The data warehouse is designed with well-structured dimensions that allow for efficient analysis across various hierarchies.

- Customer Dimension (dim_customers)

Attributes: Customer Key, Customer ID, Occupation, Marital Status, Age Group, Zip Code, City Key

Hierarchy: City → State → Country

Customer → Occupation → Income Group

- Policy Dimension (dim_policies)

Attributes: Policy Key, Policy Number, Status Code, Premium Frequency, Policy Type Key

Hierarchy: Policy Type → Lapsed/Non-Lapsed → Reinstatement Status

- Payment Method Dimension (dim_payment_methods)

Attributes: Payment Method Key, Auto Pay Key, Non-Auto Pay Key

Hierarchy: Payment Type → Autopay/Non-Autopay

- Agent & Branch Dimension

Agent Dimension (dim_agents): Agent Key, Agent ID, Agent Name, Status

Hierarchy: Agent → Branch → Region

Branch Dimension (dim_branches): Branch Key, Branch Name, Zip Code, City Key

Hierarchy: City → State → Country

- Time Dimension (dim_time)

Attributes: Time Key, Date, Day, Month, Quarter, Year

Hierarchy: Year → Quarter → Month → Day

Fact Tables & Measures

Our fact tables are designed to store quantitative metrics that allow for deep analytical insights.

- Customer Retention Fact Table (fact_customer_retention)

Foreign Keys: Customer Key, Policy Key, Payment Method Key, Issue Date Key, Lapsation Date Key, Reinstatement Date Key

Measures: Annual Income, Premium Amount, Sum Assured, Lapsation Flag (Yes/No), Reinstatement Flag (Yes/No), Autopay Enrolled (Yes/No)

- Agent & Branch Performance Fact Table (fact_agent_branch_performance)

Foreign Keys: Agent Key, Branch Key, Policy Key

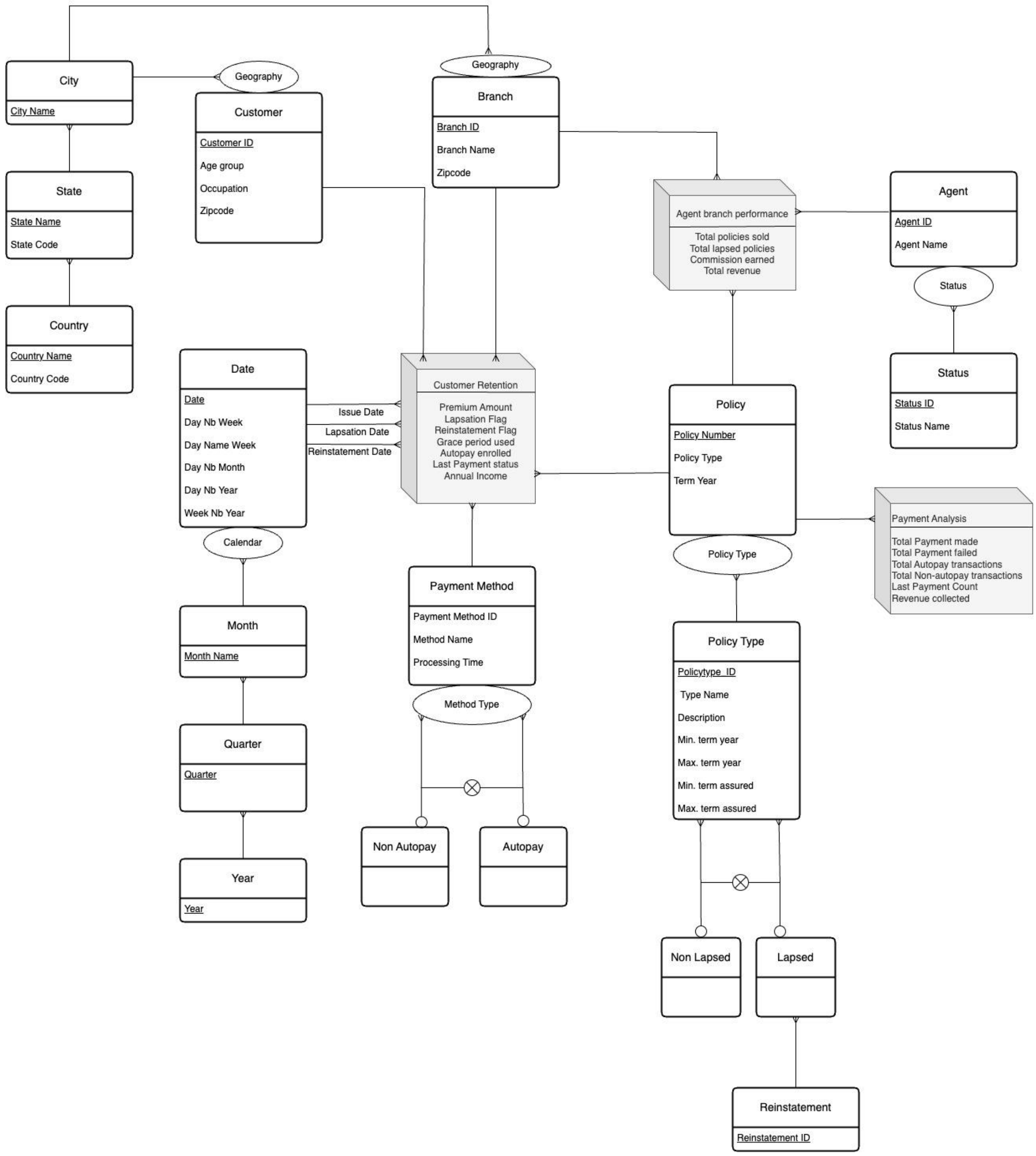
Measures: Total Policies Sold, Total Lapsed Policies, Commission Earned, Premium Revenue (Total Premium - Agent Commission)

- Payment Analysis Fact Table (fact_payment_analysis)

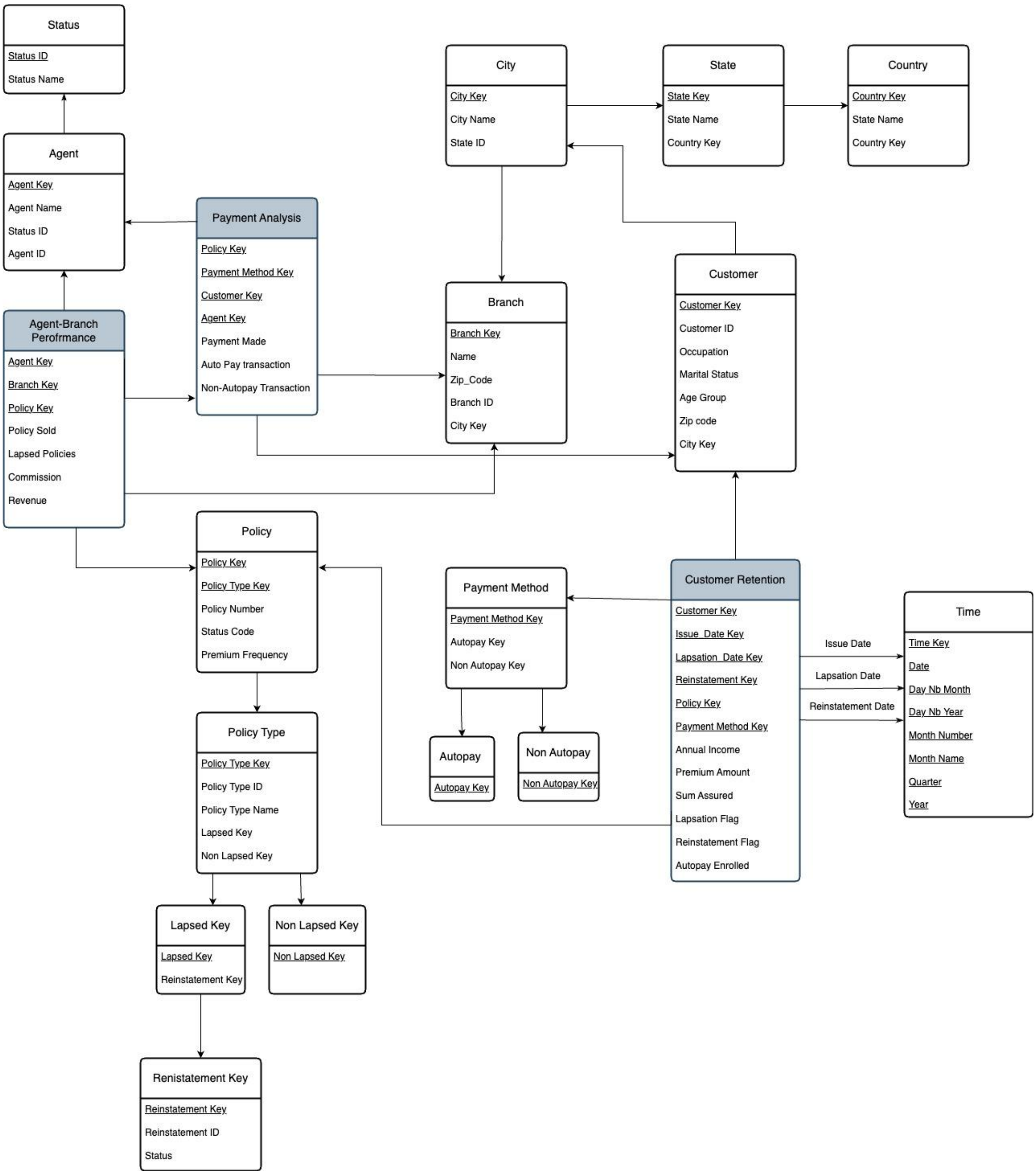
Foreign Keys: Policy Key, Customer Key, Payment Key, Agent Key

Measures: Total Payments Made, Total Auto Pay Transactions, Total Non-Auto Pay Transactions

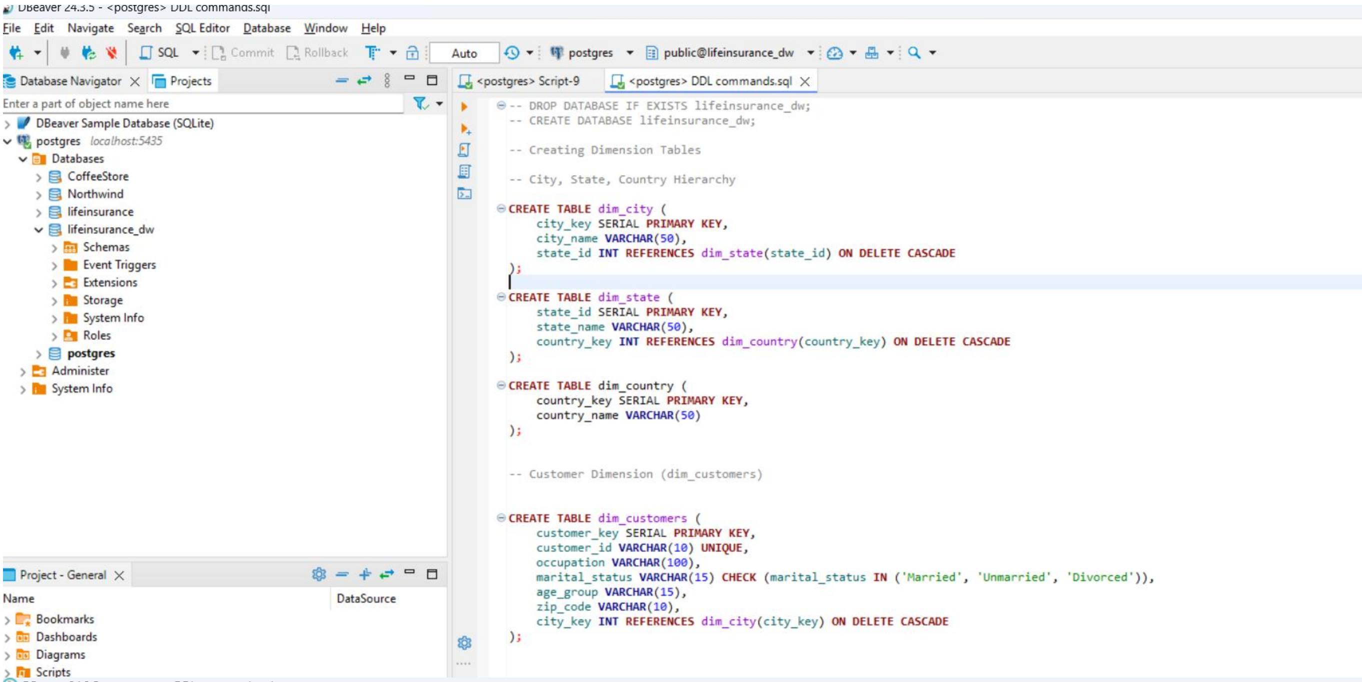
Conceptual DW Model

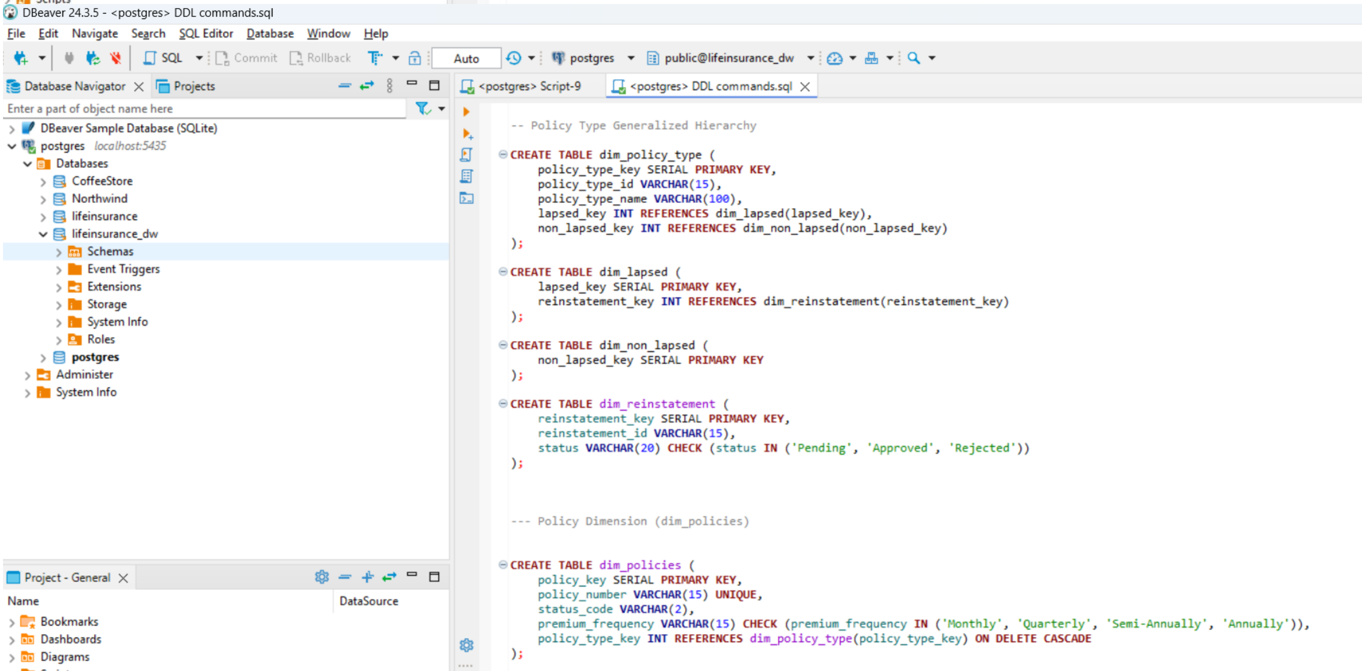


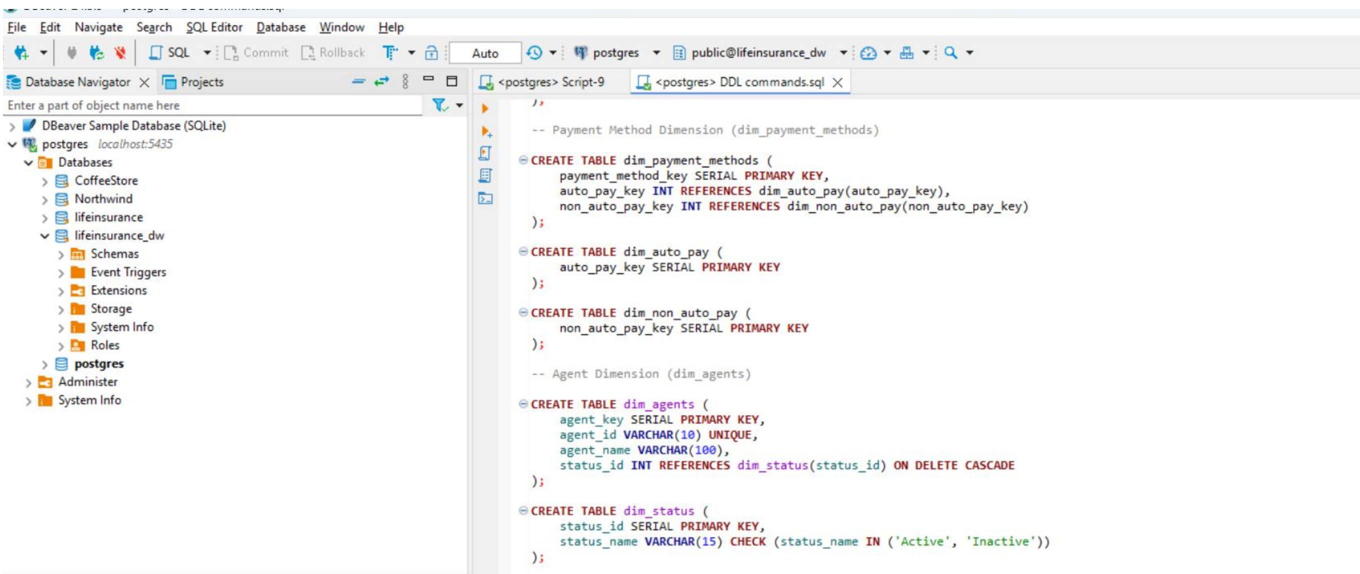
Logical DW Model

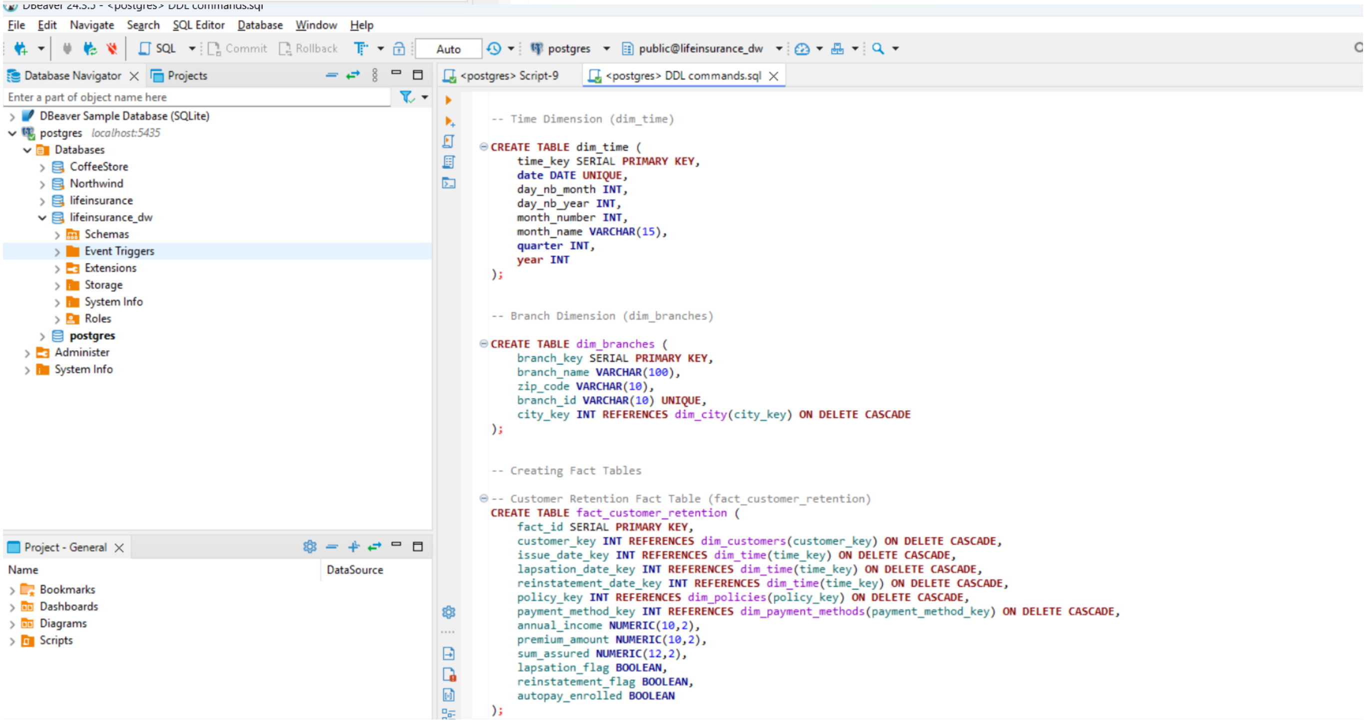


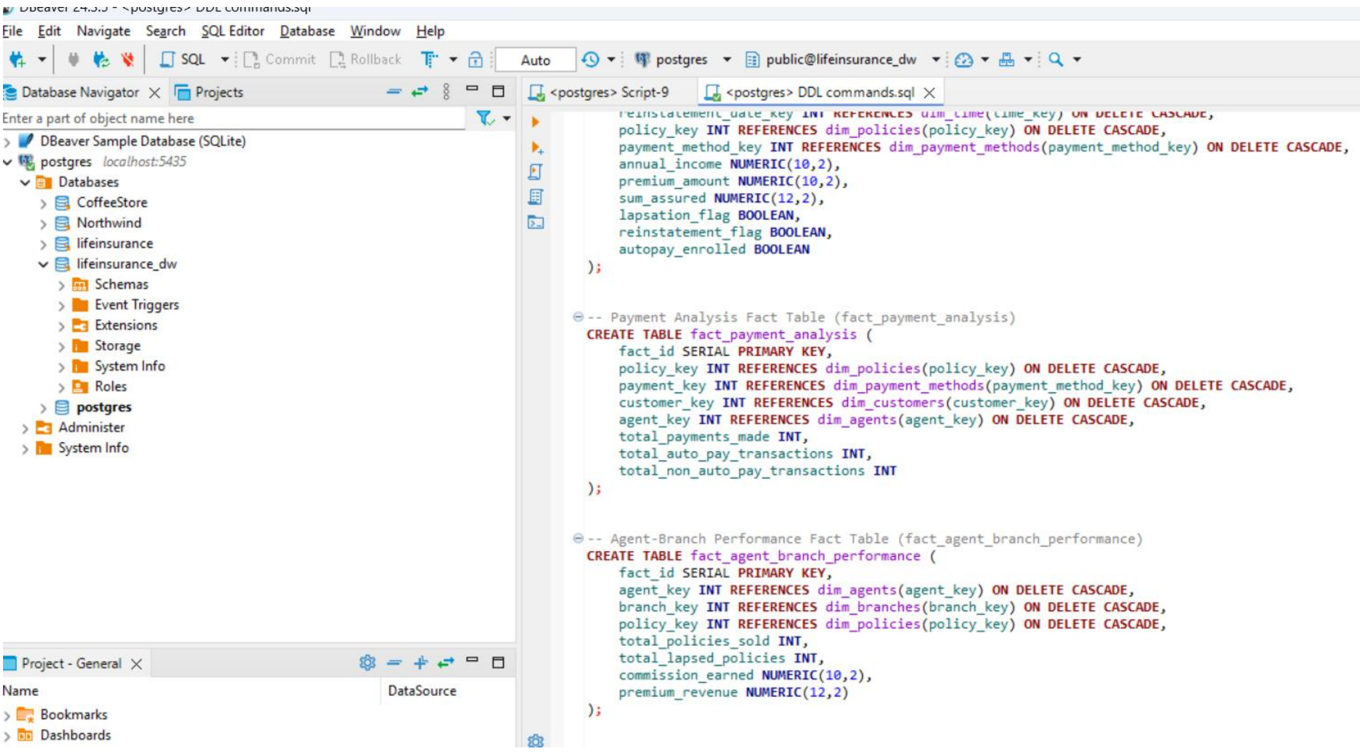
DB Implementation

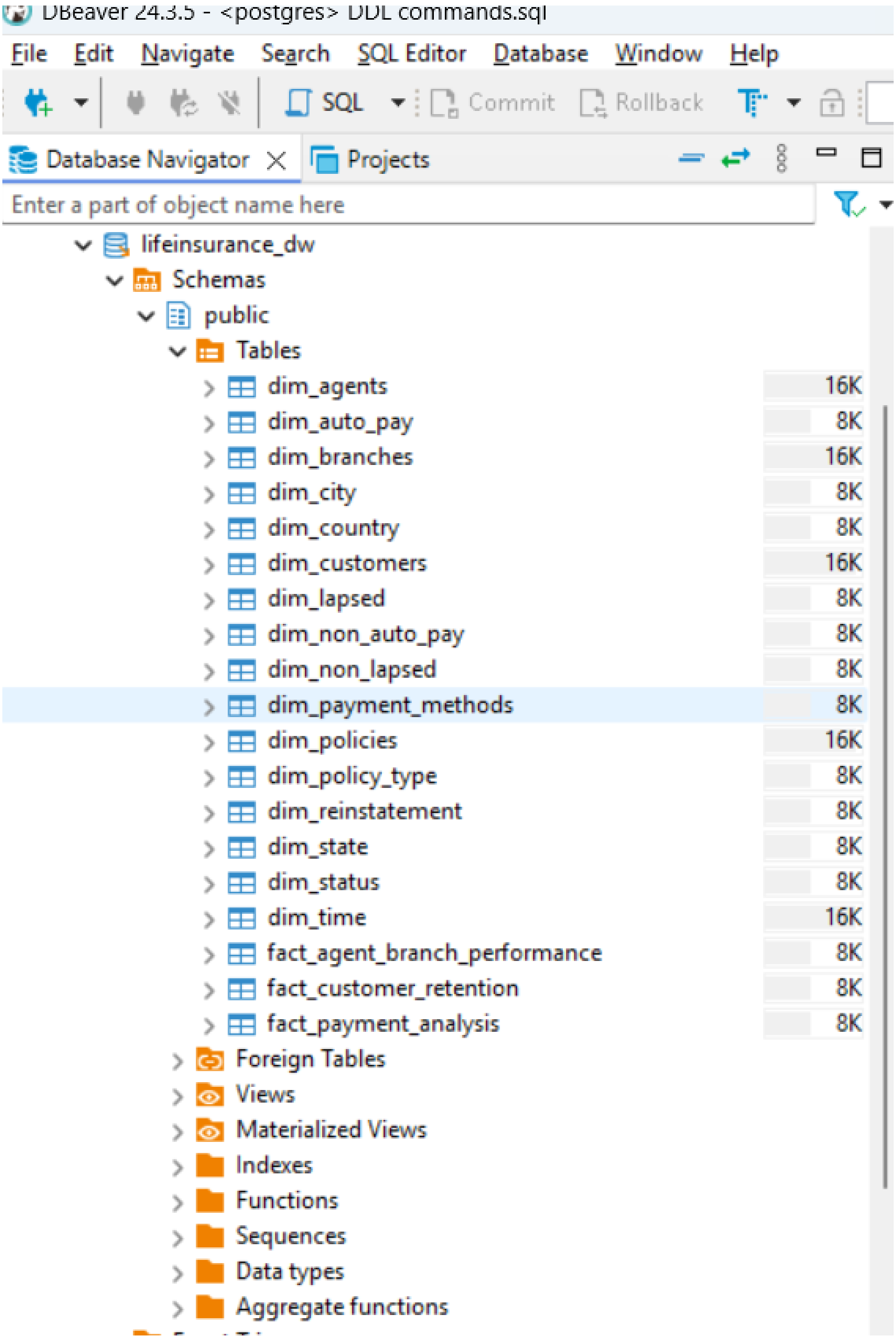




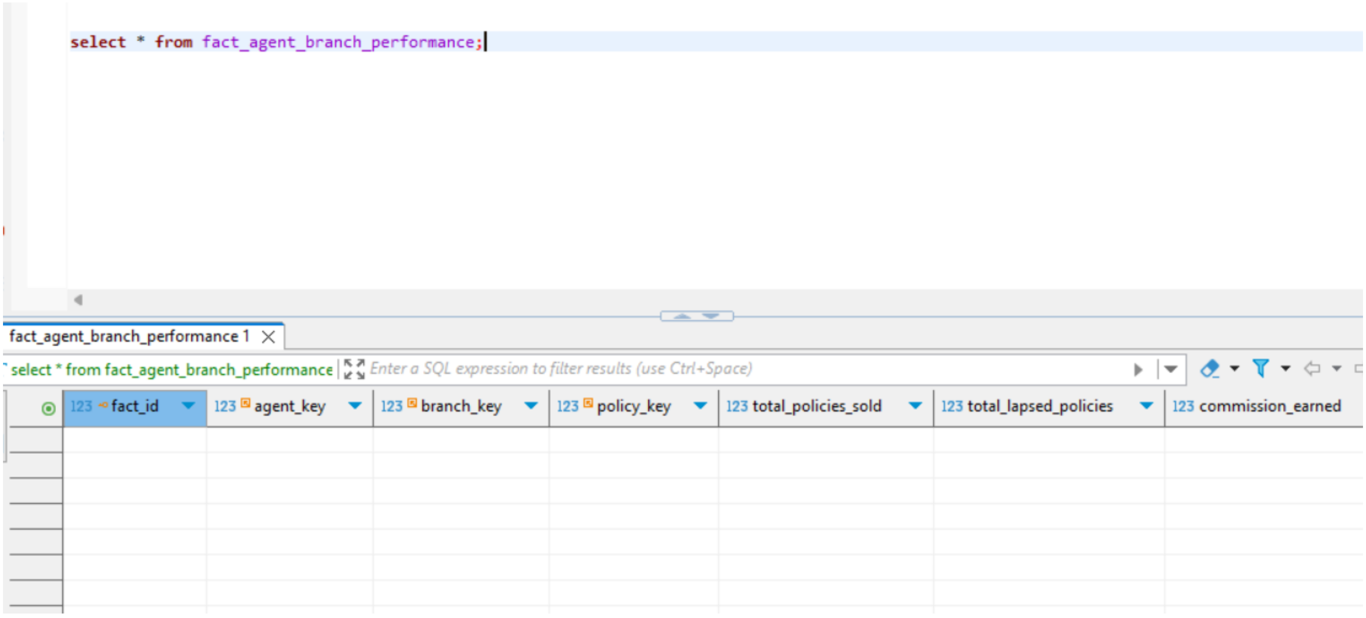

<postgres> Script-9
<postgres> DDL commands.sql
-- Payment Method Dimension (dim_payment_methods)
CREATE TABLE dim_payment_methods (
payment_method_key SERIAL PRIMARY KEY,
auto_pay_key INT REFERENCES dim_auto_pay(auto_pay_key),
non_auto_pay_key INT REFERENCES dim_non_auto_pay(non_auto_pay_key)
);
CREATE TABLE dim_auto_pay (
auto_pay_key SERIAL PRIMARY KEY
);
CREATE TABLE dim_non_auto_pay (
non_auto_pay_key SERIAL PRIMARY KEY
);
-- Agent Dimension (dim_agents)
CREATE TABLE dim_agents (
agent_key SERIAL PRIMARY KEY,
agent_id VARCHAR(10) UNIQUE,
agent_name VARCHAR(100),
status_id INT REFERENCES dim_status(status_id) ON DELETE CASCADE
);
CREATE TABLE dim_status (
status_id SERIAL PRIMARY KEY,
status_name VARCHAR(15) CHECK (status_name IN ('Active', 'Inactive'))
);

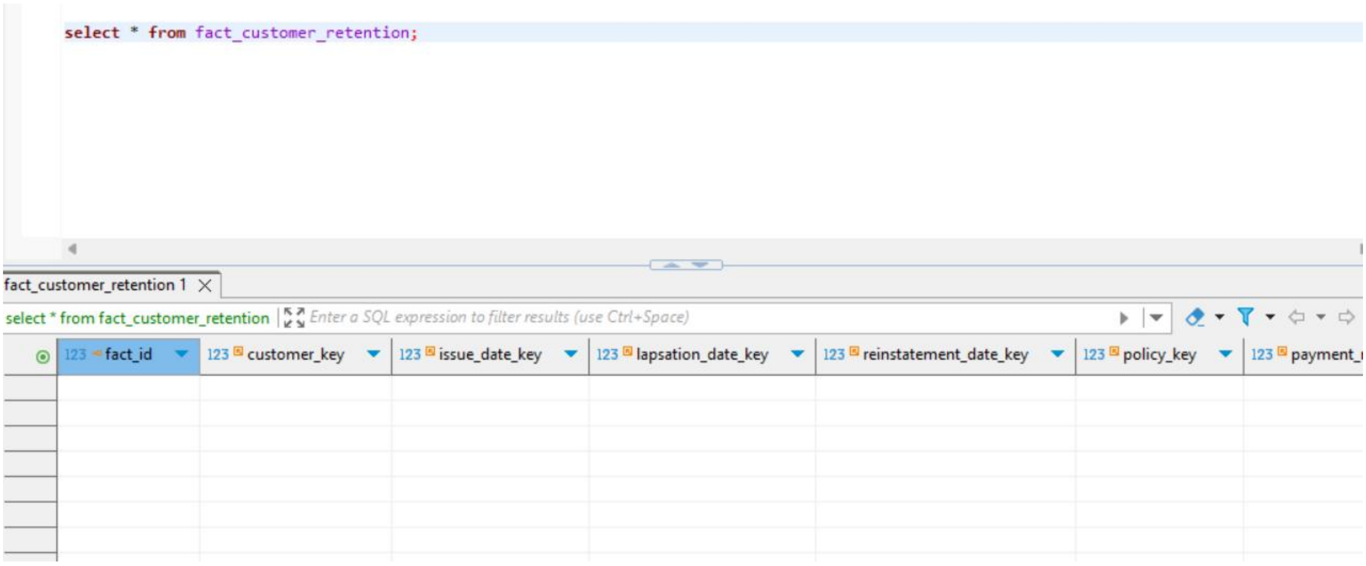

<postgres> Script-9
<postgres> DDL commands.sql
-- Time Dimension (dim_time)
CREATE TABLE dim_time (
time_key SERIAL PRIMARY KEY,
date DATE UNIQUE,
day_nb_month INT,
day_nb_year INT,
month_number INT,
month_name VARCHAR(15),
quarter INT,
year INT
);
-- Branch Dimension (dim_branches)
CREATE TABLE dim_branches (
branch_key SERIAL PRIMARY KEY,
branch_name VARCHAR(100),
zip_code VARCHAR(10),
branch_id VARCHAR(10) UNIQUE,
city_key INT REFERENCES dim_city(city_key) ON DELETE CASCADE
);
-- Creating Fact Tables
-- Customer Retention Fact Table (fact_customer_retention)
CREATE TABLE fact_customer_retention (
fact_id SERIAL PRIMARY KEY,
customer_key INT REFERENCES dim_customers(customer_key) ON DELETE CASCADE,
issue_date_key INT REFERENCES dim_time(time_key) ON DELETE CASCADE,
lapsation_date_key INT REFERENCES dim_time(time_key) ON DELETE CASCADE,
reinstatement_date_key INT REFERENCES dim_time(time_key) ON DELETE CASCADE,
policy_key INT REFERENCES dim_policies(policy_key) ON DELETE CASCADE,
payment_method_key INT REFERENCES dim_payment_methods(payment_method_key) ON DELETE CASCADE,
annual_income NUMERIC(10,2),
premium_amount NUMERIC(10,2),
sum_assured NUMERIC(12,2),
lapsation_flag BOOLEAN,
reinstatement_flag BOOLEAN,
autopay_enrolled BOOLEAN
);


File Edit Navigate Search SQL Editor Database Window Help
SQL
Commit
Rollback
Auto
postgres
public@lifeinsurance_dw
Database Navigator
Projects
<postgres> Script-9
<postgres> DDL commands.sql
Enter a part of object name here
DBeaver Sample Database (SQLite)
postgres localhost:5435
Databases
CoffeeStore
Northwind
lifeinsurance
lifeinsurance_dw
Schemas
Event Triggers
Extensions
Storage
System Info
Roles
postgres
Administer
System Info
Project - General
Name
DataSource
Bookmarks
Dashboards
policy_key INT REFERENCES dim_policies(policy_key) ON DELETE CASCADE,
payment_method_key INT REFERENCES dim_payment_methods(payment_method_key) ON DELETE CASCADE,
annual_income NUMERIC(10,2),
premium_amount NUMERIC(10,2),
sum_assured NUMERIC(12,2),
lapsation_flag BOOLEAN,
reinstatement_flag BOOLEAN,
autopay_enrolled BOOLEAN
);
-- Payment Analysis Fact Table (fact_payment_analysis)
CREATE TABLE fact_payment_analysis (
fact_id SERIAL PRIMARY KEY,
policy_key INT REFERENCES dim_policies(policy_key) ON DELETE CASCADE,
payment_key INT REFERENCES dim_payment_methods(payment_method_key) ON DELETE CASCADE,
customer_key INT REFERENCES dim_customers(customer_key) ON DELETE CASCADE,
agent_key INT REFERENCES dim_agents(agent_key) ON DELETE CASCADE,
total_payments_made INT,
total_auto_pay_transactions INT,
total_non_auto_pay_transactions INT
);
-- Agent-Branch Performance Fact Table (fact_agent_branch_performance)
CREATE TABLE fact_agent_branch_performance (
fact_id SERIAL PRIMARY KEY,
agent_key INT REFERENCES dim_agents(agent_key) ON DELETE CASCADE,
branch_key INT REFERENCES dim_branches(branch_key) ON DELETE CASCADE,
policy_key INT REFERENCES dim_policies(policy_key) ON DELETE CASCADE,
total_policies_sold INT,
total_lapsed_policies INT,
commission_earned NUMERIC(10,2),
premium_revenue NUMERIC(12,2)
);

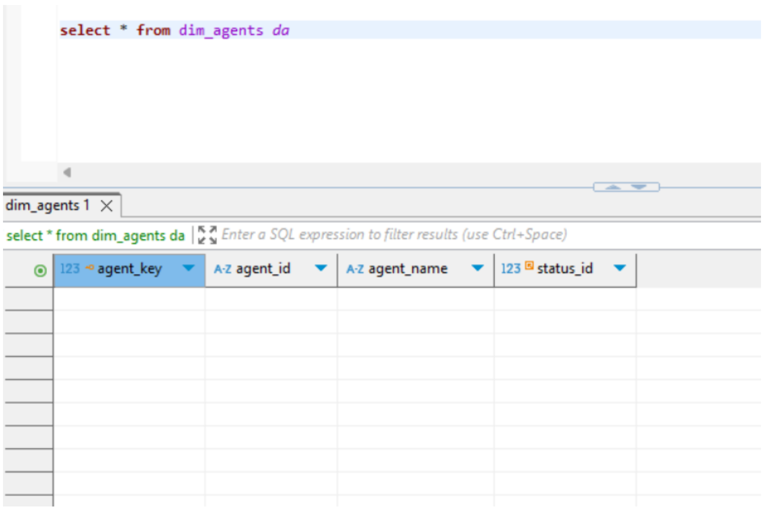

DBeaver 24.3.5 - <postgres> DDL commands.sql
File Edit Navigate Search SQL Editor Database Window Help
SQL
Commit
Rollback
Database Navigator
Projects
Enter a part of object name here
lifeinsurance_dw
Schemas
public
Tables
dim_agents 16K
dim_auto_pay 8K
dim_branches 16K
dim_city 8K
dim_country 8K
dim_customers 16K
dim_lapsed 8K
dim_non_auto_pay 8K
dim_non_lapsed 8K
dim_payment_methods 8K
dim_policies 16K
dim_policy_type 8K
dim_reinstatement 8K
dim_state 8K
dim_status 8K
dim_time 16K
fact_agent_branch_performance 8K
fact_customer_retention 8K
fact_payment_analysis 8K
Foreign Tables
Views
Materialized Views
Indexes
Functions
Sequences
Data types
Aggregate functions

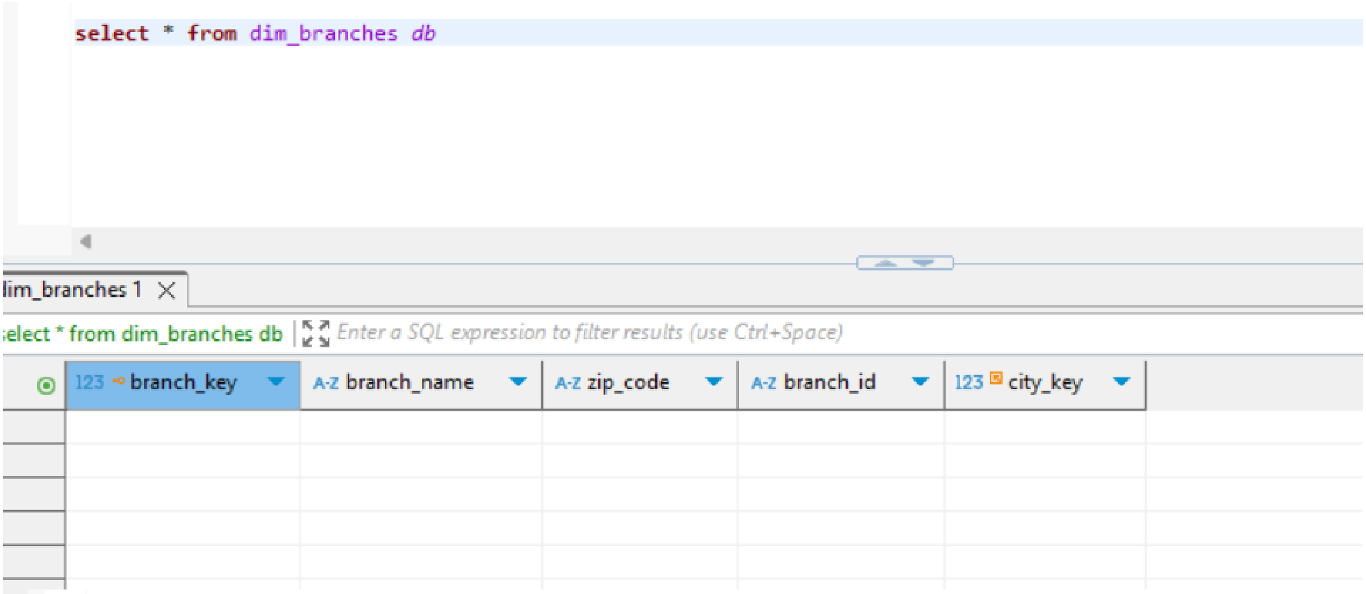
Fact Tables (few samples)

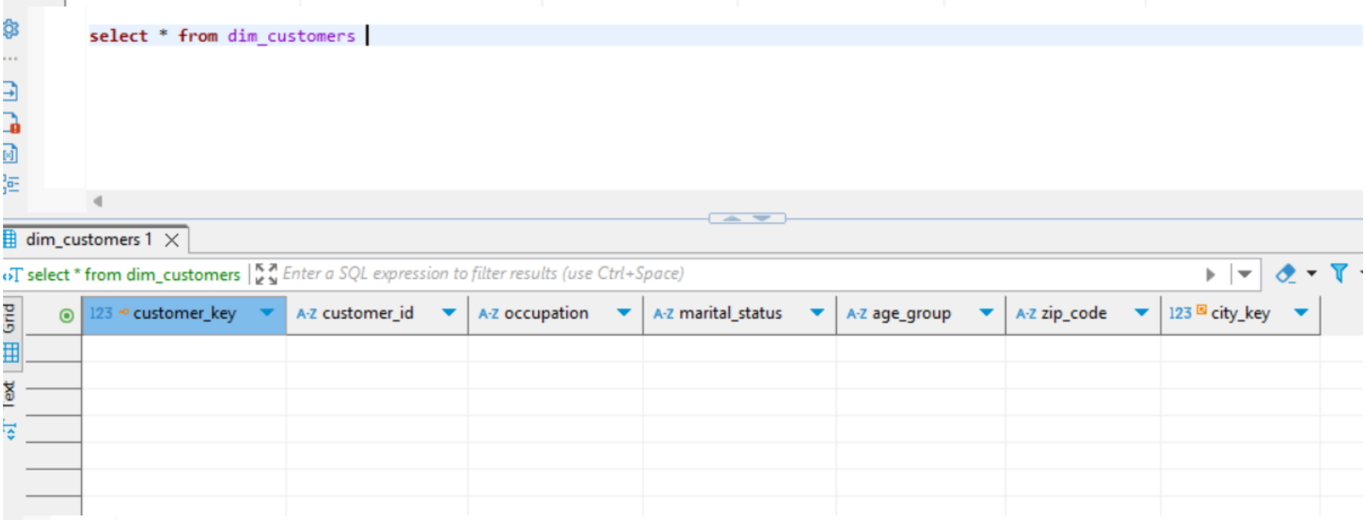


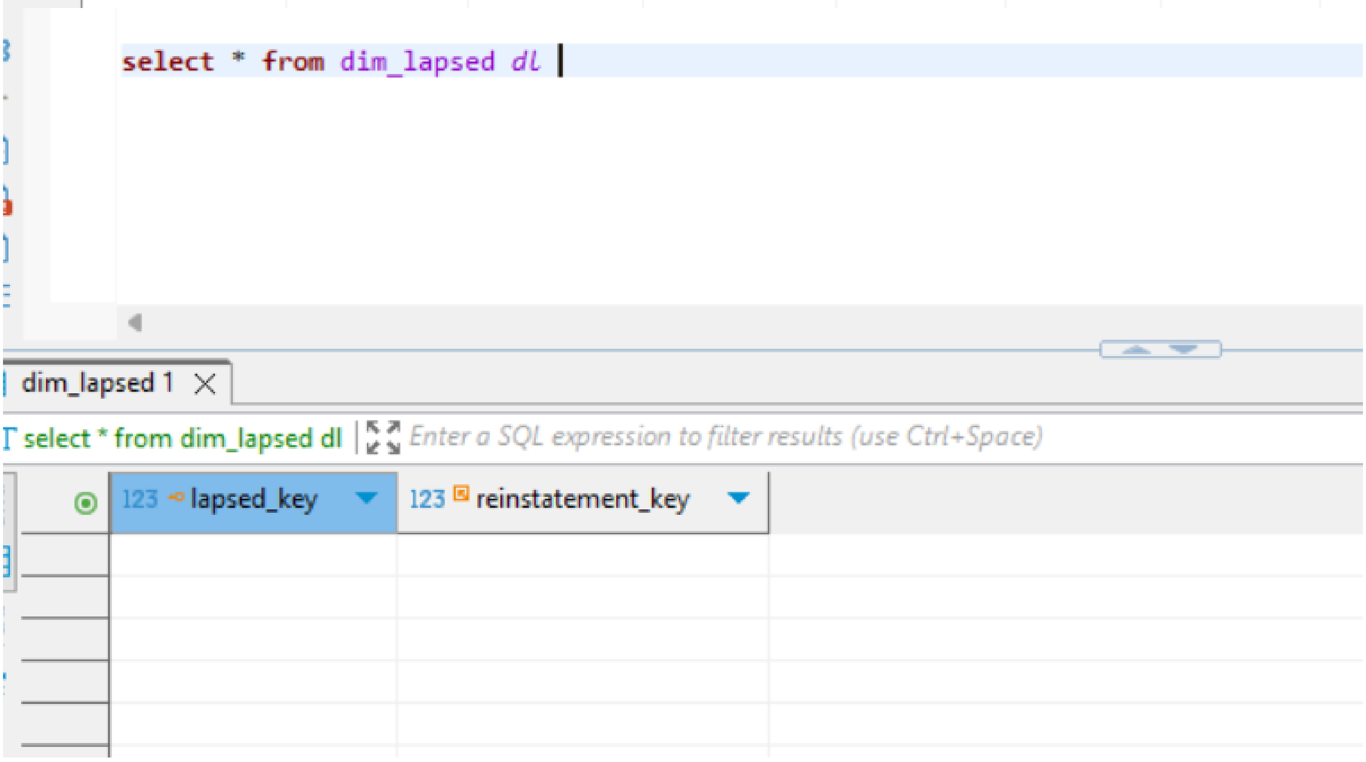


Dimension Tables:




select * from dim_branches db
lim_branches 1
select * from dim_branches db
Enter a SQL expression to filter results (use Ctrl+Space)
123 branch_key
A-Z branch_name
A-Z zip_code
A-Z branch_id
123 city_key


select * from dim_customers
dim_customers 1
select * from dim_customers
Enter a SQL expression to filter results (use Ctrl+Space)
Grid
Text
123 customer_key
A-Z customer_id
A-Z occupation
A-Z marital_status
A-Z age_group
A-Z zip_code
123 city_key


select * from dim_lapsed dl
dim_lapsed 1
select * from dim_lapsed dl
Enter a SQL expression to filter results (use Ctrl+Space)
123 lapsed_key
123 reinstatement_key

Primary Events

In our data warehouse, we focus on key events related to policy management, customer retention, agent performance, and payment processing. One of the most critical events is policy lapsation, which occurs when a customer fails to make timely premium payments, leading to policy termination. To track and analyze lapsation trends, we capture events like policy issuance, premium payments, lapsation, and reinstatement.

Another key event is payment transactions, where we monitor AutoPay vs. Non-AutoPay methods, successful vs. failed payments, and grace period utilization. Understanding payment behavior helps in optimizing payment channels and reducing missed payments.

Additionally, we track agent and branch performance by capturing events such as policies sold, commissions earned. This allows us to identify top-performing agents and branches while also highlighting areas for improvement.

By capturing these primary events in our data warehouse, we can generate meaningful insights that help improve customer retention, optimize payment processes, and enhance overall business performance.

Thinking ahead

Our data warehouse enables advanced OLAP operations to analyze policy lapsation, payment behavior, agent performance, and customer retention. These operations provide insights that drive business decisions, optimize payment processes, and improve customer engagement. Below are the key OLAP queries designed for analysis.

1. What is the quarterly premium revenue collected, and how does it vary by payment method?

Operation: ROLLUP

Res1 ← ROLLUP(fact_payment_analysis, dim_time → quarter, dim_payment_methods → payment_method, SUM(total_premium_paid) AS QuarterlyRevenue)

Insight: This analysis helps in understanding revenue trends and the contribution of different payment methods to total collections.

2. How does policy lapsation vary across different policy types and reinstatement statuses?

Operation: DRILLDOWN

Res1 ← DRILLDOWN(fact_customer_retention, dim_policies → policy_type, dim_lapsed → reinstatement_status, COUNT(lapsed_policies))

Insight: This allows us to drill down into lapsation trends based on policy types and reinstatement rates, helping to identify high-risk policies.

3. Which agent and branch have the highest policy sales, and how do they compare in terms of lapsation?

Operation: ROLLUP

Res1 ← ROLLUP(fact_agent_branch_performance, dim_agents → agent_name, dim_branches → branch_name, SUM(policies_sold) AS TotalSales, SUM(lapsed_policies) AS LapsedPolicies)

Insight: This helps in evaluating agent and branch efficiency by comparing sales and lapsation rates.

4. What is the success rate of policy reinstatement after lapsation?

Operation: DICE

Res1 ← DICE(fact_customer_retention, (lapsation_flag = '1' AND reinstatement_flag = '1'))

Res2 ← ROLLUP(Res1, dim_time → year, COUNT(reinstated_policies) / COUNT(lapsed_policies) * 100 AS ReinstatementRate)

Insight: Helps in understanding how often customers reinstate their policies after lapsation and identifying opportunities to improve reinstatement strategies.

5. What percentage of payments are AutoPay vs. Non-AutoPay, and how does it impact lapsation?

Operation: DRILLACROSS

Res1 ← DRILLACROSS(fact_payment_analysis, fact_customer_retention)

Res2 ← ROLLUP(Res1, dim_payment_methods → payment_method, COUNT(total_payments) AS TotalPayments, COUNT(lapsed_policies) AS LapsedPolicies)

Insight: Helps in understanding the relationship between payment methods and lapsation trends, guiding strategies to encourage AutoPay adoption.

6. How do lapsation rates vary across different customer income groups and age segments?

Operation: ROLLUP

Res1 ← ROLLUP(fact_customer_retention, dim_customers → age_group, dim_customers → annual_income, COUNT(lapsed_policies) AS LapsedPolicies)

Insight: Helps in segmenting high-risk customer groups based on demographics, supporting targeted retention efforts.

7. How does premium payment behavior vary over time, and which months show higher overdue payments?

Operation: DICE + ROLLUP

Res1 ← DICE(fact_payment_analysis, (payment_status = 'Overdue'))

Res2 ← ROLLUP(Res1, dim_time → month, COUNT(overdue_payments) AS MonthlyOverdues)

Insight: Helps in identifying seasonal payment trends and planning proactive reminders for customers.

8. Which agents contribute the most revenue, and what percentage of their policies are lapsed?

Operation: DRILLDOWN

Res1 ← DRILLDOWN(fact_agent_branch_performance, dim_agents → agent_name, SUM(premium_revenue) AS TotalRevenue, COUNT(lapsed_policies) AS LapsedPolicies)

Res2 ← ROLLUP(Res1, (lapsed_policies / policies_sold) * 100 AS LapsationRate)

Insight: Helps in evaluating the effectiveness of agents in selling and retaining policies.

9. How do different branches compare in terms of total policies issued and lapsation rates?

Operation: ROLLUP

Res1 ← ROLLUP(fact_agent_branch_performance, dim_branches → branch_name, COUNT(total_policies_issued) AS TotalPolicies, COUNT(lapsed_policies) AS LapsedPolicies)

Insight: Helps in assessing the performance of different branches and understanding which locations experience higher lapsation.