Київський національний університет імені Тараса Шевченка
Факультет комп'ютерних наук та кібернетики
Кафедра дослідження операцій

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА
за спеціальністю 113 «Прикладна математика»
на тему:

РОЗРОБКА І ЗАХИСТ БАЗИ ДАНИХ У MS SQL SERVER

Виконавець:
Бакалавр четвертого курсу

Руденко Данило Миколайович

Науковий керівник:
доктор фізико-математичних наук,
доцент, професор кафедри дослідження операцій

Самойленко Ігор Валерійович

Роботу заслухано на засіданні кафедри дослідження операцій та рекомендовано до захисту, протокол №9 від **23 травня 2023 р**.

Завідувач кафедри ДО  проф. Іксанов О.М.

Київ – 2023

ЗМІСТ

ВСТУПНА ЧАСТИНА

У сучасному світі велика кількість даних генерується щодня в різних галузях індустрії, технологій, медицини та бізнесу. Обробка і зберігання цих даних є дуже важливими завданнями, які вирішуються за допомогою баз даних.

Системи управління базами даних (СУБД) дозволяють створювати, зберігати та обробляти великі об'єми даних. Серед СУБД, які використовуються в сучасному світі, Microsoft SQL Server є однією з найпоширеніших та найпопулярніших. Ця СУБД має велику кількість інструментів для забезпечення ефективної розробки та управління базами даних.

Результатом дослідження буде створення БД у MS SQL Server, яка буде відповідати вимогам сучасних підходів до розробки БД. Крім того, будуть проаналізовані різні методи захисту даних у БД, що дозволить розробити ефективну систему захисту даних в БД.

Тому, метою даної дипломної роботи є розробка та захист БД у MS SQL Server з використанням сучасних технологій та методів забезпечення безпеки даних. Для досягнення мети необхідно вирішити такі завдання: вивчити теоретичні засади розробки та захисту БД, розробити модель даних для обраної предметної галузі, розробити та реалізувати БД у MS SQL Server, виконати тестування та оптимізацію БД, а також розробити стратегію забезпечення безпеки даних.

Однією з основних причин актуальності цієї теми є необхідність в захисті конфіденційної інформації, яка зберігається в БД. З міркувань безпеки, важливо розуміти загрози, які можуть бути пов'язані з незахищеними базами даних, тому що це може призвести до витоку даних, фінансових втрат і порушення репутації компанії.

У зв'язку з постійним зростанням кількості даних, які збираються та зберігаються в базах даних, виникає необхідність у підвищенні рівня захисту цих даних. Крім того, із збільшенням кількості даних зростає й кількість можливих загроз їхній безпеці, тому необхідно розробляти нові методи захисту та підвищувати ефективність наявних.

Розробка та захист баз даних в MS SQL Server є надзвичайно важливою проблемою, оскільки дані є ключовими складовими в сучасному світі та використовуються в різних галузях, включаючи медицину, банківський сектор, промисловість та інформаційні технології.

Розробка баз даних є складним процесом, який вимагає знання технологій та методик проектування, включаючи моделювання даних, нормалізацію та оптимізацію запитів. Крім того, важливим аспектом розробки є врахування вимог до захисту даних, щоб забезпечити конфіденційність, цілісність та доступність інформації.

Захист баз даних включає в себе багато аспектів, таких як фізична та логічна безпека, автентифікація та авторизація користувачів, шифрування даних, моніторинг активності користувачів та виявлення зловмисної діяльності. MS SQL Server надає багато засобів для захисту баз даних, включаючи захист на рівні сервера, бази даних та даних в ряді їхніх форматів.

Однією з актуальних проблем в сфері захисту баз даних є кібератаки та витоки даних. За даними Verizon's 2020 Data Breach Investigations Report, 45% усіх кібератак були спрямовані на бази даних. Тому, забезпечення надійного захисту баз даних від кібератак є критично важливим.

Об'єктом дослідження є база даних у системі управління базами даних MS SQL Server, яка є однією з найбільш поширених СУБД на сучасному ринку.

Дослідження зосередиться на вивченні можливостей системи управління базами даних MS SQL Server для розробки та захисту баз даних, а також на визначенні методів захисту даних в цій системі. В ході роботи будуть проаналізовані різні підходи до розробки баз даних і будуть використані методи захисту даних від несанкціонованого доступу, випадкових помилок, вразливостей, збоїв жорсткого диска тощо. Завданнями є вивчення можливостей СУБД для розробки та захисту баз даних, аналіз вразливостей та загроз, вивчення методів захисту даних та вивчення режимів доступу до баз даних.

Розділ 1. Теоретична частина

1.1 Огляд методів захисту БД

Методи захисту БД є дуже важливим елементом забезпечення безпеки даних. Нижче наведено деякі з методів захисту БД:

– Аутентифікація та авторизація: Це метод, який дозволяє перевірити, чи є користувач, який намагається отримати доступ до бази даних, дійсно тим, за кого він себе видає. Після того, як користувача було аутентифіковано, йому може бути дозволено або заборонено доступ до певних ресурсів бази даних за допомогою механізму авторизації.

Аутентифікація та авторизація є ключовими методами захисту БД, які забезпечують контроль доступу до бази даних і захищають її від несанкціонованого доступу.

Аутентифікація - це процес перевірки ідентичності користувача, який намагається отримати доступ до БД. Він зазвичай здійснюється за допомогою ідентифікатора користувача (логіну) та пароля. При цьому користувач повинен ввести вірний логін та пароль для того, щоб отримати доступ до БД.

Авторизація - це процес визначення повноважень користувача, який вже пройшов аутентифікацію. Вона визначає, на які ресурси має доступ користувач, і які дії він може виконувати з цими ресурсами. Наприклад, адміністратор бази даних може мати повний доступ до всіх таблиць БД, тоді як звичайні користувачі можуть бути обмежені у своїх діях, які вони можуть виконувати з базою даних.

Враховуючи значення аутентифікації та авторизації, користувачі повинні мати різні рівні доступу до БД в залежності від їхніх повноважень та потреб. Важливо також забезпечити виконання ряду правил та політик доступу до БД, так щоб доступ користувачів до даних був обмежений та контрольований.

– Шифрування даних: Цей метод полягає в захисті даних від несанкціонованого доступу, шляхом перетворення їх у незрозумілу форму за допомогою шифрування. Шифрування може застосовуватися на різних рівнях: на рівні диску, файлу або рядка даних.

Наприклад, можна зашифрувати всю базу даних або окремі поля таблиць, які містять конфіденційну інформацію. Також можна використовувати різні алгоритми шифрування, такі як AES (Advanced Encryption Standard), DES (Data Encryption Standard), RSA (Rivest-Shamir-Adleman), і багато інших.

Шифрування даних зазвичай включає дві основні функції - шифрування та розшифрування. Для зашифрування даних потрібно використовувати ключ, який є секретним. Цей ключ використовується для перетворення відкритого тексту в шифротекст. Для розшифрування даних необхідно мати доступ до того ж ключа, що використовувався для шифрування.

– Аудит: Цей метод полягає в реєстрації всіх подій, що відбуваються в базі даних. Інформація про події збирається та зберігається в окремому журналі. Аудит може допомогти виявити помилки, вторгнення або несанкціоновані зміни даних.

Під час аудиту можуть фіксуватися дії, такі як входи користувачів в систему, зміни даних в таблицях, створення нових об'єктів БД і багато іншого. Ці записи дій забезпечують зручний засіб для відстеження, хто і коли виконував ті чи інші дії в БД, що дає можливість виявити можливі проблеми з безпекою даних.

Аудит може бути проведений відповідно до різних стандартів та вимог, наприклад, стандарту PCI DSS, який регулює захист платіжних карток. Це дозволяє підприємствам виконувати різні правила, регулятиви та вимоги, які стосуються зберігання та обробки даних.

У випадку виявлення підозрілих дій або порушень безпеки, аудит дає можливість швидко виявити проблему і вжити заходів для її вирішення, що дозволяє зменшити ризики порушення безпеки даних.

– Фізичний захист: Цей метод полягає в забезпеченні безпеки самої фізичної інфраструктури бази даних. Наприклад, фізичний захист може включати

контроль доступу до серверного приміщення, забезпечення стабільної роботи електропостачання та систем охолодження.

Один з найважливіших аспектів фізичного захисту полягає у забезпеченні безпеки фізичних носіїв даних, таких як жорсткі диски, флеш-накопичувачі, CD або DVD диски. Ці носії можуть містити конфіденційну інформацію, тому їх потрібно зберігати в безпечних місцях, забезпечувати їхній захист від фізичних пошкоджень і втрати.

Для забезпечення фізичного захисту можна використовувати різні техніки, наприклад, системи контролю доступу, відеоспостереження, вогнегасні системи та інші. Крім того, важливо забезпечити резервне копіювання даних, щоб у разі фізичного пошкодження або втрати можна було відновити доступ до них.

– Захист від вторгнень: Цей метод полягає в застосуванні систем захисту від вторгнень. Такі системи можуть виявляти спроби вторгнення до бази даних та вживати заходи для їхнього запобігання.

Для захисту від вторгнень використовують різні методи, які можуть бути розділені на дві категорії: захист від зовнішніх атак та захист від внутрішніх загроз.

Для захисту від зовнішніх атак використовують фірмвари (firewalls), які блокують небезпечний трафік з Інтернету, забезпечують контроль доступу до мережі та встановлюють правила безпеки. Фірмвари можуть бути апаратними або програмними.

Для захисту від внутрішніх загроз використовують системи виявлення вторгнень (Intrusion Detection Systems, IDS). Ці системи моніторять активність користувачів та виявляють небезпечні дії, такі як спроби незаконного доступу, перевищення повноважень або зміна конфігурації системи. IDS можуть бути розташовані як на зовнішньому мережевому периметрі, так і внутрішніх сегментах мережі.

Також для захисту від вторгнень використовуються системи обмеження швидкості, які можуть знижувати швидкість виконання запитів користувачів в разі перевищення певних порогових значень. Це дозволяє запобігти спробам атак на ресурсоємкі операції, такі як перебір паролів або SQL-ін'єкції.

1.2 Перевірка та аудит БД

Перевірка та аудит БД - це процес перевірки та оцінки безпеки бази даних, щоб забезпечити, що система захисту даних працює належним чином та не виникає загроз безпеці. Цей процес включає в себе перевірку наявності вразливостей у базі даних, збір інформації про потенційні загрози та виконання аналізу журналів безпеки для виявлення підозрілих дій.

Під час аудиту БД, використовуються різні інструменти, такі як сканери вразливостей, аналізатори журналів, моніторинг мережі, інструменти збору інформації та інші. Оцінка безпеки бази даних дозволяє виявити можливі загрози та вразливості, що можуть бути використані для атак на систему.

Після проведення аудиту БД, розробляються рекомендації з приводу підвищення рівня безпеки, які можуть включати в себе запровадження нових методів захисту, оновлення програмного забезпечення, налаштування належного рівня доступу до даних та інші заходи. Такі рекомендації можуть допомогти забезпечити безпеку даних в базі даних та зменшити ризик викрадення, втрати або пошкодження даних.

Перевірка та аудит БД є важливим етапом в забезпеченні безпеки даних в БД. Цей процес допомагає виявити потенційні загрози безпеці даних та забезпечити їхню відповідність встановленим стандартам та політикам безпеки.

Один з прикладів реального використання аудиту БД - це використання аудиторських журналів для відстеження змін у БД та виявлення потенційних загроз безпеці даних. Наприклад, компанія може вести журнал аудиту, щоб відстежувати дії користувачів, такі як зміна даних, додавання нових записів, видалення даних тощо. Це допомагає виявити будь-які небажані дії та вчасно реагувати на них.

Інший приклад - це використання інструментів перевірки БД, таких як програми сканування вразливостей та тестування на проникнення, для оцінки рівня безпеки БД та виявлення потенційних вразливостей. Наприклад, програма Acunetix може

виявляти різноманітні вразливості БД, такі як SQL-ін'єкції, вразливість до крадіжки сесій та інші.

У цілому, перевірка та аудит БД дозволяє підтримувати високий рівень безпеки даних в БД та виявляти потенційні загрози безпеці даних, що допомагає вчасно реагувати на них та забезпечити безпеку даних у майбутньому.

1.3 Підходи до розробки БД

Підходи до розробки баз даних (БД) визначаються способом організації даних і взаємодії з ними. Розробка БД може бути проведена за допомогою різних підходів, в залежності від потреб користувачів і мети БД. Нижче наведено деякі з підходів до розробки БД:

- Реляційний підхід: це найбільш поширений підхід до розробки БД. За цим підходом, дані організовані у вигляді таблиць з реляціями між ними. У цьому підході використовується мова запитів SQL для створення, модифікації та отримання даних.
- Об'єктно-орієнтований підхід: цей підхід базується на використанні об'єктів як основного засобу для зберігання даних. Об'єктно-орієнтована модель даних описує структуру та поведінку об'єктів.
- Ієрархічний підхід: в цьому підході дані організовані у вигляді ієрархії, де кожен вузол має підвузли із відповідними даними. Цей підхід був популярний в минулому, але зараз його використання обмежене.
- Мережевий підхід: в цьому підході дані організовані у вигляді мережі, де кожен запис може мати декілька батьків. Цей підхід також був популярний в минулому, але зараз використовується дуже рідко.

1.4 Мова запитів SQL

Мова SQL (Structured Query Language) є стандартом для роботи з реляційними базами даних. Вона використовується для створення, модифікації та отримання даних з БД. SQL включає в себе такі елементи, як команди SELECT, INSERT, UPDATE та DELETE, що використовуються для взаємодії з даними.

Команда SELECT є найбільш часто використовуваною командою SQL. Вона дозволяє вибирати дані з таблиці за певними критеріями і виводити їх у вигляді таблиці. Команда INSERT дозволяє додавати дані до таблиці, а команда UPDATE - змінювати існуючі дані. Команда DELETE використовується для видалення даних з таблиці..

SQL має багато варіацій, включаючи MySQL, PostgreSQL, Oracle та Microsoft SQL Server. Кожна з цих варіацій має свої особливості, проте базуються на загальному стандарті SQL. Використання SQL в розробці БД дозволяє ефективно працювати з даними і забезпечувати їхню безпеку.

Приклади використання SQL:

– Вибірка всіх даних з таблиці "users":

```
SELECT * FROM users;
```

– Вибірка даних з таблиці "orders", в яких ціна більше 1000:

```
SELECT * FROM orders WHERE price > 1000;
```

– Вставка нового рядка в таблицю "users":

```
INSERT INTO users (name, email) VALUES ('John Smith', 'john@example.com');
```

– Видалення користувача з id=2:

```
DELETE FROM users WHERE id=2;
```

– Приклад утворення таблиці

```
CREATE TABLE users (
id INT NOT NULL AUTO_INCREMENT,
name VARCHAR(50) NOT NULL,
email VARCHAR(100) NOT NULL,
```

```
password VARCHAR(255) NOT NULL,
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
PRIMARY KEY (id),
UNIQUE KEY email (email)
);
```

У цьому прикладі створюється таблиця з назвою "users". Вона містить стовпці "id", "name", "email", "password" та "created_at". Стовпець "id" визначений як PRIMARY KEY, тому його значення будуть унікальними для кожного запису. Стовпець "email" визначений як UNIQUE KEY, тому два записи не можуть мати однакових значень у цьому стовпці.

Розділ 2. Практична частина

2.1 Аналіз вимог до БД

Аналіз вимог до бази даних є важливим етапом при розробці будь-якої бази даних. Цей процес допомагає зрозуміти потреби користувачів і встановити вимоги до даних, які повинні бути збережені в базі даних. Вимоги до БД включають такі елементи, як структура даних, типи даних, обмеження на дані та інші фактори.

Під час аналізу вимог до бази даних, звернемо увагу на наступні питання:

– Які дані потрібно зберігати в базі даних?

Нехай ми будемо займатися фінансовими технологіями. Тоді наша база даних повинна містити наступну інформацію:

Інформація про клієнтів: повне ім'я, дата народження, адреса, електронна адреса, номер телефону, номер паспорту, ідентифікаційний код.

Інформація про кредити: сума кредиту, термін погашення, відсоткова ставка, дата видачі, статус кредиту (активний, закритий, прострочений).

Історія платежів: інформація про всі платежі, які здійснювалися клієнтом, включаючи дату платежу, суму, тип платежу (проценти, основний борг, пеня), статус платежу (сплачений, прострочений, несплачений).

– Які залежності і зв'язки між даними повинні бути враховані?

Клієнти: клієнти, які взяли кредит, повинні бути пов'язані з кредитними договорами, які містять інформацію про тип кредиту, розмір кредиту, термін кредиту та процентну ставку.

Платежі: інформація про платежі повинна бути пов'язана з кредитними договорами та клієнтами, які роблять платежі.

Заборгованість: інформація про заборгованість клієнта повинна бути пов'язана з кредитними договорами та платежами.

Процентні ставки: інформація про процентні ставки повинна бути пов'язана з кредитними договорами.

– Які обмеження повинні бути на дані в базі даних?

Обмеження на тип даних: тип даних для кожного поля повинен бути вибраний так, щоб відповідати типу даних, які він міститиме. Наприклад, поле з датою повинно мати тип "дата" або "час", а поле з числом повинно мати тип "ціле" або "дробове".

Обмеження на довжину даних: для поля, що містить текст, повинно бути встановлено максимальну довжину, яка не дозволяє вводити більше символів, ніж ця довжина. Це допомагає уникнути введення неправильних даних, які можуть призвести до втрати цілісності даних.

Обмеження на доступ до даних: важливо забезпечити захист від несанкціонованого доступу до бази даних.

Ще одним прикладом обмеження є встановлення заборони на видалення деяких даних з бази даних для забезпечення їхнього збереження в разі потреби. Однак, якщо виникає необхідність видалення даних, можна встановити обмеження на рівні доступу до бази даних, щоб забезпечити, що тільки авторизовані користувачі можуть видаляти дані з бази даних.

– Які типи запитів будуть використовуватися для вибірки даних з бази?

SELECT - це запит для вибірки даних з однієї або більше таблиць. Він може бути використаний з різними функціями, такими як WHERE, GROUP BY, ORDER BY та іншими.

INSERT - це запит для додавання нових даних у таблицю.

UPDATE - це запит для оновлення існуючих даних у таблиці.

DELETE - це запит для видалення даних з таблиці.

JOIN - це запит для комбінування даних з двох або більше таблиць на основі спільних полів.

UNION - це запит для об'єднання даних з двох або більше таблиць з однаковою структурою.

– Які типи звітів та аналітики будуть використовуватися для аналізу даних?

Виділимо типи звітів та аналітики, які можуть бути корисними для нашої фінансової компанії:

Звіти про портфелі кредитів: ці звіти містять інформацію про загальний обсяг кредитів, їхній стан та ризики, що пов'язані з кожним кредитом.

Звіти по конверсії: цей тип аналітики допомагає будувати графіки конверсій з погашених кредитів у повторні.

Звіти про витрати та прибуток: ці звіти допомагають відслідковувати фінансові результати компанії та оцінювати її ефективність.

Аналіз платіжної дисципліни: цей тип аналітики дозволяє відслідковувати, наскільки клієнти вчасно сплачують свої кредитні зобов'язання.

Звіти про клієнтів: ці звіти містять інформацію про клієнтів, яка може бути корисною для подальшого розвитку бізнесу.

Існують різні програмні засоби для створення звітів та аналітики, такі як Excel, Tableau, Power BI та інші, які допоможуть створити звіти та аналітику зі зібраних даних в базі даних. У нашому випадку будемо користуватися останнім.

Тепер можна перейти до проектування структури бази даних. Цей етап включає створення схеми бази даних, таблиць та інші деталі реалізації. Ці кроки допомагають забезпечити ефективну роботу бази даних у майбутньому.

2.2 Розробка схеми БД

Для розробки схеми БД для Нашої fintech компанії, потрібно визначити всі дані, які будуть зберігатися в базі даних. Основними сутностями будуть клієнти, кредитні заявки, платежі, промокоди, смс для клієнтів, портфоліо.

Для кожної сутності потрібно визначити атрибути та їхні типи даних.

Для сутності "CLIENTS" можуть будуть визначені наступні атрибути: REGISTRATION_DATE, CLIENT_ID, SURNAME, NAME, MIDDLE_NAME, PERSONAL_ID, PHONE_NUMBER, EMAIL.

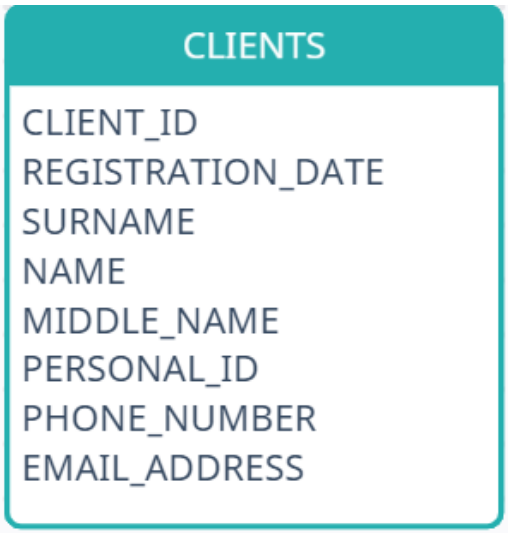


Рисунок 2.1 – Таблиця CLIENTS

- REGISTRATION_DATE - дата реєстрації клієнта в системі;
- CLIENT_ID - унікальний ідентифікатор клієнта в системі;
- SURNAME - прізвище клієнта;
- NAME - ім'я клієнта;
- MIDDLE_NAME - по-батькові клієнта;
- PERSONAL_ID - ідентифікаційний номер клієнта;
- PHONE_NUMBER - номер телефону клієнта;
- EMAIL - адреса електронної пошти клієнта.

Для сутності "APPLICATION" будуть визначені атрибути, такі як REQUEST_DATE, CLIENT_ID, ANKETA_ID, REQUESTED_AMOUNT, REQUESTED_TERM, DECISION.

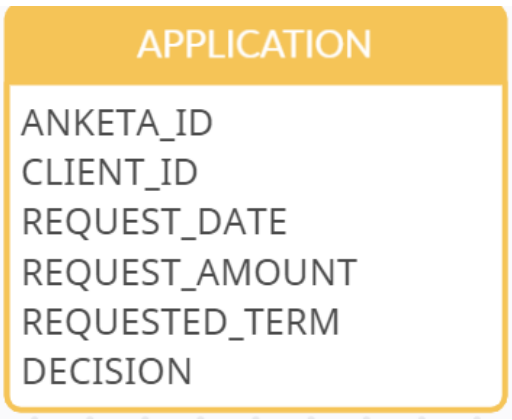


Рисунок 2.2 – Таблиця APPLICATIONS

- REQUEST_DATE - дата подання заявки;
- CLIENT_ID - ідентифікатор клієнта, який подав заявку;
- ANKETA_ID - ідентифікатор анкети, пов'язаної з заявкою;
- REQUESTED_AMOUNT - запропонована сума кредиту;
- REQUESTED_TERM - запропонований термін кредиту;
- DECISION - рішення по заявці (прийнята, відхилена).

Для сутності "PORTFOLIO" будуть визначені атрибути, такі як REPORT_DATE, CLIENT_ID, LOAN_ID, BEGIN_DATE, LOAN_AMOUNT, LOAN_TERM, INTEREST_RATE, ALL_AMOUNT, DPD, CLOSE_DATE.

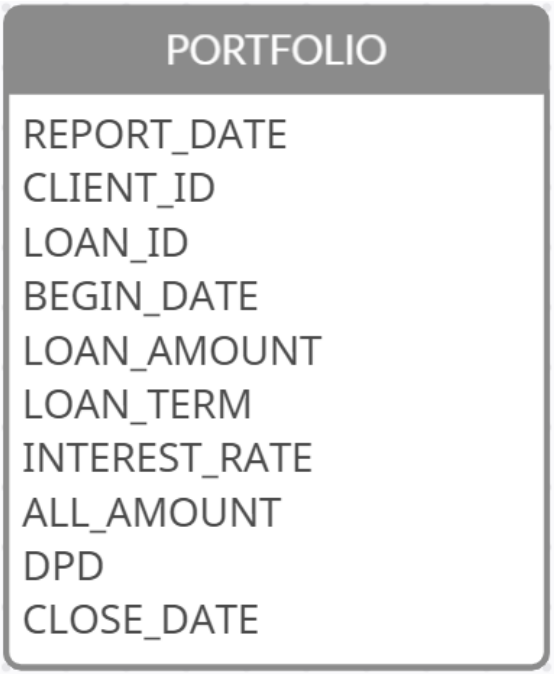


Рисунок 2.3 – Таблиця PORTFOLIO

- REPORT_DATE - дата звіту портфеля;
- CLIENT_ID - ідентифікатор клієнта, що отримав кредит;
- LOAN_ID - ідентифікатор кредиту;
- BEGIN_DATE - дата отримання кредиту;
- LOAN_AMOUNT - сума кредиту;
- LOAN_TERM - термін кредиту;
- INTEREST_RATE - відсоткова ставка за кредитом;

– ALL_AMOUNT - сума повернення кредиту з урахуванням відсотків та штрафних санкцій;

– DPD - кількість днів прострочки по кредиту;

– CLOSE_DATE - дата повного погашення кредиту.

Для сутності "PAYMENTS" будуть визначені атрибути, такі як PAYMENT_DATE, CLIENT_ID, LOAN_ID, PAID_BODY, PAID_INTEREST ,PAID_PENALTIES .

Рисунок 2.4 – Таблиця PAYMENTS

– PAYMENT_DATE - дата здійснення платежу за кредитом;

– CLIENT_ID - ідентифікатор клієнта, який здійснив платіж;

– LOAN_ID - ідентифікатор кредиту, за яким було здійснено платіж;

– PAID_BODY - сума, яка була сплачена як тіло кредиту;

– PAID_INTEREST - сума, яка була сплачена як відсотки по кредиту;

– PAID_PENALTIES - сума, яка була сплачена як штрафні санкції за порушення умов кредитного договору.

Для сутності "PROMOCODE_USING" будуть визначені атрибути, такі як REPORT_DATE, LOAN_ID, BONUS_NAME, DISCOUNT_PERCENT.

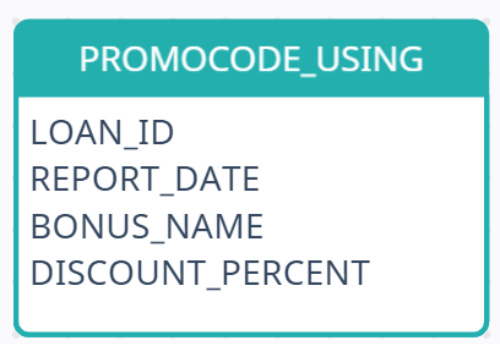


Рисунок 2.5 – Таблиця PROMOCODE_USING

– REPORT_DATE - атрибут, що зберігає дату, коли промокод було використано;

– LOAN_ID - атрибут, що зберігає ідентифікатор кредиту, до якого було застосовано промокод;

– BONUS_NAME - атрибут, що зберігає назву бонусу, який було отримано при використанні промокоду;

– DISCOUNT_PERCENT - атрибут, що зберігає відсоток знижки, який було отримано при використанні промокоду.

Для сутності "SMS" будуть визначені атрибути, такі як REPORT_DATE, CLIENT_ID, PHONE_NUMBER, NAME_SMS

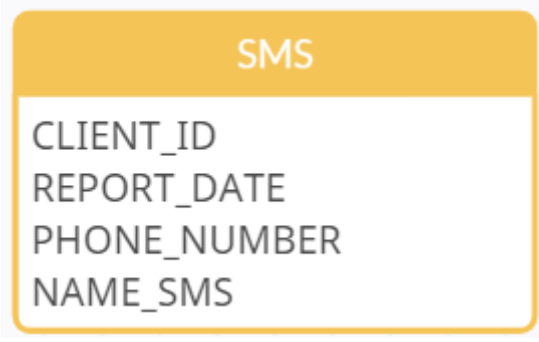


Рисунок 2.6 – Таблиця SMS

– CLIENT_ID - це числовий ідентифікатор клієнта, який отримав повідомлення SMS;

– REPORT_DATE - це дата, коли було відправлено повідомлення SMS;

– PHONE_NUMBER - це рядок, що містить номер телефону клієнта;

– NAME_SMS - це рядок, що містить текст повідомлення SMS.

2.3 Розробка SQL-запитів для БД

Звідси починається одна з найважливіших частин розробки бази даних - розробка SQL-запитів для доступу до даних в БД.

Напишемо у MS SQL скрипти для створення описаних раніше таблиць

CLIENTS:

```
CREATE TABLE CLIENTS (
    ID INT PRIMARY KEY NOT NULL,
    CLIENT_ID INT NOT NULL,
    REGISTRATION_DATE DATE NOT NULL,
    SURNAME NVARCHAR(255) NOT NULL,
    NAME NVARCHAR(255) NOT NULL,
    MIDDLE_NAME NVARCHAR(255) NULL,
    PERSONAL_ID NVARCHAR(30) NOT NULL,
    PHONE_NUMBER nvarchar(15) NOT NULL,
    EMAIL_ADDRESS nvarchar(100) NULL
);
```

Рисунок 2.7 – Створення таблиці CLIENTS

APPLICATION:

```
CREATE TABLE APPLICATION (
    ID INT PRIMARY KEY NOT NULL,
    ANKETA_ID INT NOT NULL,
    CLIENT_ID INT NOT NULL,
    REQUEST_DATE DATETIME2(0) NOT NULL,
    REQUEST_AMOUNT FLOAT NOT NULL,
    REQUESTED_TERM INT NOT NULL,
    DECISION NVARCHAR(30) NOT NULL
);
```

Рисунок 2.8 – Створення таблиці APPLICATION

PORTFOLIO:

```
CREATE TABLE PORTFOLIO (
    ID INT PRIMARY KEY NOT NULL,
    REPORT_DATE DATE NOT NULL,
    CLIENT_ID INT NOT NULL,
    LOAN_ID INT NOT NULL,
    BEGIN_DATE DATE NOT NULL,
    LOAN_AMOUNT FLOAT NOT NULL,
    LOAN_TERM INT NOT NULL,
    INTEREST_RATE FLOAT NOT NULL,
    ALL_AMOUNT FLOAT NOT NULL,
    DPD INT NOT NULL,
    CLOSE_DATE DATE NULL
);
```

Рисунок 2.9 – Створення таблиці PORTFOLIO

PAYMENTS:

```
CREATE TABLE PAYMENTS (
    ID INT PRIMARY KEY NOT NULL,
    PAYMENT_DATE DATETIME2(0) NOT NULL,
    CLIENT_ID INT NOT NULL,
    LOAN_ID INT NOT NULL,
    PAID_BODY FLOAT NOT NULL,
    PAID_INTEREST FLOAT NOT NULL,
    PAID_PENALTIES FLOAT NOT NULL
);
```

Рисунок 2.10 – Створення таблиці PAYMENTS

PROMOCODE_USING:

```
CREATE TABLE PROMOCODE_USING (
    ID INT PRIMARY KEY NOT NULL,
    LOAN_ID INT NOT NULL,
    REPORT_DATE DATE NOT NULL,
    BONUS_NAME NVARCHAR(255) NOT NULL,
    DISCOUNT_PERCENT INT NOT NULL
);
```

Рисунок 2.11 – Створення таблиці PROMOCODE_USING

SMS:

```
CREATE TABLE SMS(
    ID INT PRIMARY KEY NOT NULL,
    CLIENT_ID INT NOT NULL,
    REPORT_DATE DATE NOT NULL,
    PHONE_NUMBER NVARCHAR(15) NOT NULL,
    NAME_SMS NVARCHAR(100) NOT NULL
);
```

Рисунок 2.12 – Створення таблиці SMS

Атрибути в цих таблицях мають наступні типи даних:

ID, ANKETA_ID, CLIENT_ID, LOAN_ID, DPD, DISCOUNT_PERCENT, REQUESTED_TERM – INT, тобто це ціле число. Ідентифікатори використовуються як унікальні числові значення, що ідентифікують записи в наших таблицях бази даних. Цей тип даних має розмір 4 байти і забезпечує великий діапазон значень, що дозволяє зберігати велику кількість ідентифікаторів і забезпечує ефективність операцій

порівняння та пошуку за цими значеннями. Тобто використання цілочисельного типу даних для ідентифікаторів дозволяє зменшити обсяг зайнятого місця в базі даних і полегшити її роботу. Запрошений і встановлений термін кредиту та кількість днів також мають цей тип, оскільки вони означають кількість повних днів. Відсоток знижки також нехай буде цілим числом.

REPORT_DATE, BEGIN_DATE, CLOSE_DATE, REGISTRATION_DATE – DATE. Для атрибутів, які відображають дату цей тип даних є підходящим вибором, бо зберігає її у форматі 'рік-місяць-день' , тому можна здійснювати різні операції.

PAYMENT_DATE, REQUEST_DATE – DATETIME2(0). Оскільки у нас база даних, яка пов’язана з фінансовою сферою, нам дуже важлива точність та надійність даних, цей тип зберігає дату та час з точністю до мілісекунд. У нашому випадку, число 0 у дужках указує на те, що значення дати та часу зберігається з точністю до секунди.

NAME_SMS, PHONE_NUMBER, BONUS_NAME, DECISION, NAME, SURNAME, MIDDLE_NAME, EMAIL_ADDRESS, PERSONAL_ID – NVARCHAR(n). Всі ці атрибути мають текстовий формат даних, тому вони будуть мати цей тип даних.

PAID_BODY, PAID_INTEREST, PAID_PENALTIES, ALL_AMOUNT, INTEREST_RATE, LOAN_AMOUNT, REQUEST_AMOUNT – FLOAT. Суми платежів можуть мати дробову частину, наприклад, якщо частину платежу складає процентна ставка або штрафні санкції. Загальна суму, яка також може мати дробову частину, наприклад, якщо сума кредиту є нецілим числом. Відсоткова ставка також може бути не цілим числом.

Окремо зауважимо, що усі атрибути мають обмеження “not null”, щоб значення у цих стовпчиках не були порожніми, окрім атрибута CLOSE_DATE, оскільки у нашу табличку PORTFOLIO будуть записуватися дані кожного дня по кредиту клієнта, кредит може бути не закритий, тому значення цього стовпчика буде нульовим. Оскільки атрибут DPD має тип цілого числа, у дні, коли клієнт ще не має прострочки, ми можемо писати “0”.

Також можна побачити, що в кожній таблиці є атрибут ID, котрий є первинним ключем. Такий варіант був використаний з метою зв'язування таблиць між собою за допомогою атрибутів. Це рішення було прийнято, бо такі атрибути як CLIENT_ID,

ANKETA_ID, LOAN_ID можуть мати дублікати (наприклад у табличці PORTFOLIO). Атрибут ID ми не будемо використовувати як ключ між нашими табличками, він нам буде корисним, якщо ми будемо наповнювати нашу базу даних з іншої бази. Наприклад у нас є база даних, яка використовується в реальному часі для зберігання даних і їхньої обробки. Така база даних містить актуальну інформацію про бізнес-процеси, продукти, послуги або будь-яку іншу інформацію, яка потрібна для виконання операцій організації. Так звана “база даних на проді” розміщується на окремому сервері або групі серверів і має високу доступність і надійність для забезпечення продуктивності та безпеки.

Також атрибут ID можна буде використовувати як елемент сортування, оскільки чим більше числове значення ID, тим свіжіші дані. Ще одне застосування це - зв’язування таблички з самою собою.

Зараз для прикладу заповнимо базу даних даними

CLIENTS:

```
INSERT INTO CLIENTS (ID, CLIENT_ID, REGISTRATION_DATE, SURNAME, NAME, MIDDLE_NAME, PERSONAL_ID, PHONE_NUMBER, EMAIL_ADDRESS)
VALUES
(1, 1896772, '2021-12-04', 'Першенко', 'Перш', 'Першович', '1234567890', '380501234567', 'abcdefg@example.com'),
(2, 1726625, '2022-02-03', 'Другенко', 'ДругГ', 'Другович', '2136549875', '380671234567', 'hijklmn@example.com'),
(3, 723506, '2021-08-25', 'Третєнко', 'Третій', 'Третьович', '9876543212', '380931234567', 'opqrstu@example.com'),
(4, 1865787, '2022-02-09', 'Четверенко', 'Четверт', 'Четвертович', '1234567899', '380981234567', 'vwxyz@example.com'),
(5, 665956, '2021-08-05', 'П`ятенко', 'П`ят', 'П`ятович,', '5555666677', '380681234567', 'qazwsx@example.com'),
(6, 1594172, '2022-02-09', 'Шістенко', 'Шост', 'Шостович', '3322116655', '380501234567', 'edcrfv@example.com'),
(7, 1841235, '2021-07-21', 'Семенко', 'Сьом', 'Сьомович', '9988774455', '380441234567', 'tgbyhn@example.com'),
(8, 1922480, '2021-12-04', 'Восьменко', 'Восьм', 'Восьмович', '6655663325', '380981234567', 'ujmikl@example.com'),
(9, 1610742, '2021-08-05', 'Дев`ятенко', 'Дев`ят', 'Дев`ятович', '9182736450', '380502345678', 'polikm@example.com');
```

Рисунок 2.13 – Заповнення даними таблиці CLIENTS

	ID	CLIENT_ID	REGISTRATION_DATE	SURNAME	NAME	MIDDLE_NAME	PERSONAL_ID	PHONE_NUMBER	EMAIL_ADDRESS
1	1	1896772	2021-12-04	Першенко	Перш	Першович	1234567890	380501234567	abcdefg@example.com
2	2	1726625	2022-02-03	Другенко	ДругГ	Другович	2136549875	380671234567	hijklmn@example.com
3	3	723506	2021-08-25	Третєнко	Третій	Третьович	9876543212	380931234567	opqrstu@example.com
4	4	1865787	2022-02-09	Четверенко	Четверт	Четвертович	1234567899	380981234567	vwxyz@example.com
5	5	665956	2021-08-05	П`ятенко	П`ят	П`ятович,	5555666677	380681234567	qazwsx@example.com
6	6	1594172	2022-02-09	Шістенко	Шост	Шостович	3322116655	380501234567	edcrfv@example.com
7	7	1841235	2021-07-21	Семенко	Сьом	Сьомович	9988774455	380441234567	tgbyhn@example.com
8	8	1922480	2021-12-04	Восьменко	Восьм	Восьмович	6655663325	380981234567	ujmikl@example.com
9	9	1610742	2021-08-05	Дев`ятенко	Дев`ят	Дев`ятович	9182736450	380502345678	polikm@example.com

Рисунок 2.14 – Дані в таблиці CLIENTS

APPLICATION:

```
INSERT INTO APPLICATION (ID, ANKETA_ID, CLIENT_ID, REQUEST_DATE, REQUEST_AMOUNT, REQUEST_NUMBER, REQUESTED_TERM, DECISION)
VALUES
    (1, 200037, 1896772, '2023-03-11 00:10:30', 1500, 3, 10, 'approved'),
    (2, 183057, 1726625, '2023-03-13 00:04:57', 2000, 1, 15, 'approved'),
    (3, 204282, 723506, '2023-03-13 01:09:08', 10000, 2, 10, 'approved'),
    (4, 261056, 1865787, '2023-03-14 00:11:55', 8000, 4, 30, 'approved'),
    (5, 250977, 665956, '2023-03-16 00:05:35', 5000, 2, 7, 'approved'),
    (6, 255222, 1594172, '2023-03-16 01:50:18', 2000, 1, 15, 'approved'),
    (7, 574380, 1841235, '2023-04-03 07:19:26', 1750, 1, 10, 'approved'),
    (8, 2568, 1922480, '2023-04-14 18:40:15', 3000, 4, 15, 'rejected'),
    (9, 2752, 1610742, '2023-04-06 07:29:33', 500, 3, 30, 'rejected');
```

Рисунок 2.15 – Заповнення даними таблиці APPLICATION

	ID	ANKETA_ID	CLIENT_ID	REQUEST_DATE	REQUEST_AMOUNT	REQUEST_NUMBER	REQUESTDET_TERM	DECISION
1	1	200037	1896772	2023-03-11 00:10:30	1500	3	10	approved
2	2	183057	1726625	2023-03-13 00:04:57	2000	1	15	approved
3	3	204282	723506	2023-03-13 01:09:08	10000	2	10	approved
4	4	261056	1865787	2023-03-14 00:11:55	8000	4	30	approved
5	5	250977	665956	2023-03-16 00:05:35	5000	2	7	approved
6	6	255222	1594172	2023-03-16 01:50:18	2000	1	15	approved
7	7	574380	1841235	2023-04-03 07:19:26	1750	1	10	approved
8	8	2568	1922480	2023-04-14 18:40:15	3000	4	15	rejected
9	9	2752	1610742	2023-04-06 07:29:33	500	3	30	rejected

Рисунок 2.16 – Дані в таблиці APPLICATION

PORTFOLIO (на зображенні будуть показані внесені дані лише по одному клієнту, бо в цій табличці показуються дані кожного дня по кредиту, поки клієнт його не закриє. Якщо у табличці APPLICATION у клієнта в DECISION буде стояти “rejected”, у табличку PORTFOLIO ця анкета не потрапить):

```
INSERT INTO PORTFOLIO (ID, REPORT_DATE, CLIENT_ID, LOAN_ID, BEGIN_DATE, LOAN_AMOUNT, LOAN_TERM, INTEREST_RATE, ALL_AMOUNT, DPD, CLOSE_DATE)
VALUES
    (1, '2023-03-17', 665956, 250977, '2023-03-17', 5000, 7, 0.5, 5025, 0, NULL),
    (2, '2023-03-18', 665956, 250977, '2023-03-17', 5000, 7, 0.5, 5050, 0, NULL),
    (3, '2023-03-19', 665956, 250977, '2023-03-17', 5000, 7, 0.5, 5025, 0, NULL),
    (4, '2023-03-20', 665956, 250977, '2023-03-17', 5000, 7, 0.5, 5075, 0, NULL),
    (5, '2023-03-21', 665956, 250977, '2023-03-17', 5000, 7, 0.5, 0, 0, '2023-03-21');
    ...
```

Рисунок 2.17 – Заповнення даними таблиці PORTFOLIO

	ID	REPORT_DATE	CLIENT_ID	LOAN_ID	BEGIN_DATE	LOAN_AMOUNT	LOAN_TERM	INTEREST_RATE	ALL_AMOUNT	DPD	CLOSE_DATE
1	1	2023-03-17	665956	250977	2023-03-17	5000	7	0.5	5025	0	NULL
2	2	2023-03-18	665956	250977	2023-03-17	5000	7	0.5	5050	0	NULL
3	3	2023-03-19	665956	250977	2023-03-17	5000	7	0.5	5025	0	NULL
4	4	2023-03-20	665956	250977	2023-03-17	5000	7	0.5	5075	0	NULL
5	5	2023-03-21	665956	250977	2023-03-17	5000	7	0.5	0	0	2023-03-21

Рисунок 2.18 – Дані в таблиці PORTFOLIO

PAYMENTS:

```
INSERT INTO PAYMENTS(ID, PAYMENT_DATE, CLIENT_ID, LOAN_ID, PAID_BODY, PAID_INTEREST, PAID_PENALTIES) VALUES
  (1, '2023-03-17 18:29:51', 1896772, 200037, 1500, 45, 0),
  (2, '2023-03-28 12:30:31', 1726625, 183057, 2000, 150, 0);
  ...
```

Рисунок 2.19 – Заповнення даними таблиці PAYMENTS

	ID	PAYMENT_DATE	CLIENT_ID	LOAN_ID	PAID_BODY	PAID_INTEREST	PAID_PENALTIES
1	1	2023-03-17 18:29:51	1896772	200037	1500	45	0
2	2	2023-03-28 12:30:31	1726625	183057	2000	150	0

Рисунок 2.20 –Дані в таблиці PAYMENTS

PROMOCODE_USING:

```
INSERT INTO PROMOCODE_USING (LOAN_ID, REPORT_DATE, BONUS_NAME, DISCOUNT_PERCENT)
VALUES
    (574380, '2023-04-03', 'Красивий', 100),
    (250977, '2023-03-16', '5-19 днів без кредиту', 40),
    (255222, '2023-03-17', 'День народження', 50);
```

Рисунок 2.21 – Заповнення даними таблиці PROMOCODE_USING

	LOAN_ID	REPORT_DATE	BONUS_NAME	DISCOUNT_PERCENT
1	574380	2023-04-03	Красивий	100
2	250977	2023-03-16	5-19 днів без кредиту	40
3	255222	2023-03-17	День народження	50

Рисунок 2.22 – Дані в таблиці PROMOCODE_USING

SMS:

```
INSERT INTO SMS (ID, CLIENT_ID, REPORT_DATE, PHONE_NUMBER, NAME_SMS)
VALUES
    (1, 2752, '2023-04-06', '380502345678', 'Відзилення'),
    (2, 2568, '2023-04-14', '380981234567', 'Відхилення'),
    (3, 200037, '2023-03-17', '380501234567', 'Оплата пройшла успішно');
```

Рисунок 2.23 – Заповнення даними таблиці SMS

	ID	CLIENT_ID	REPORT_DATE	PHONE_NUMBER	NAME_SMS
1	1	2752	2023-04-06	380502345678	Відзилення
2	2	2568	2023-04-14	380981234567	Відхилення
3	3	200037	2023-03-17	380501234567	Оплата пройшла успішно

Рисунок 2.24 – Дані в таблиці SMS

2.4 Налаштування доступу до БД

Оскільки ми створюємо базу даних для фінансової компанії, і в нас зберігається персональні дані наших клієнтів, ми повинні якомога краще захистити нашу БД. Розпочнемо з найпростіших стандартних шляхів та спробуємо надалі вдосконалити.

Для налаштування доступу до бази даних створимо користувача і надамо йому відповідні права доступу. Створення користувача і надання йому доступу до БД виконується адміністратором бази даних або особою з відповідними правами доступу.

Знизу наведений рисунок, на якому зображене вікно створення нового користувача в нашій базі даних.

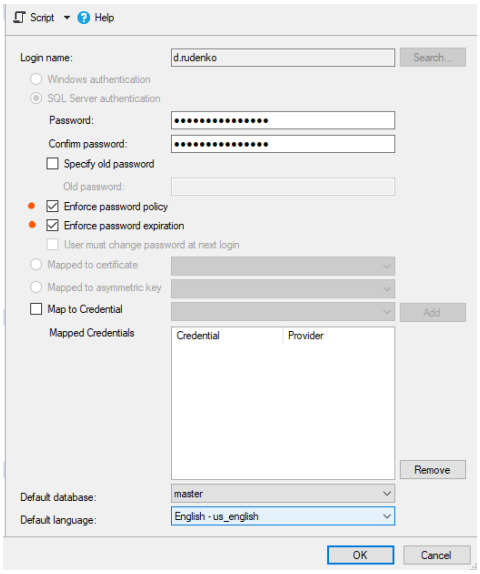


Рисунок 2.25 – Вікно створення користувача у MS SQL Server

Пропоную навпроти “Enforce password policy” та “Enforce password expiration” виставити прапорець. Ці два пункти змушують нашого користувача придумати складний пароль, який буде відповідати усім вимогам, та міняти його через певний час. Це створено для того, щоб навіть, якщо зловмисник вгадає складний пароль, щоб він не мав необмежений у часі доступ до наших даних. Так виглядає вікно, у якому нашого користувача просять поміняти пароль, бо пройшов певний час. Це вікно

з’являється лише в тому випадку, якщо при вході в базу користувач увів правильний логін та пароль.

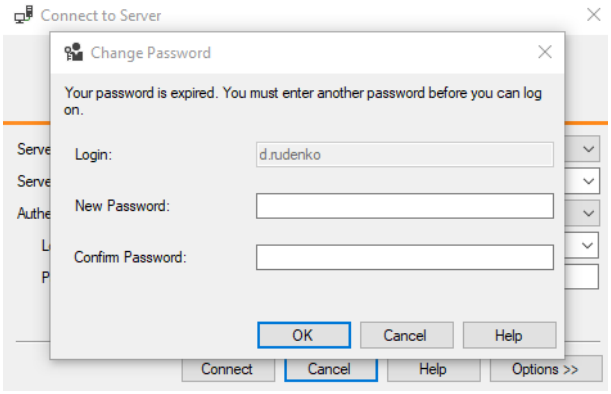


Рисунок 2.26 – Вікно оновлення паролю

За замовчуванням кожен новий користувач, який створюється в базі даних, автоматично отримує роль public, яка дає йому доступ до всіх об'єктів, до яких не встановлено обмежень доступу. Роль sysadmin - це найвища роль на сервері, яка дає повний доступ до всіх можливостей MS SQL Server, включаючи створення, зміну та видалення баз даних, налаштування сервера, налаштування безпеки, тому її варто надавати лише перевіреному колу осіб.

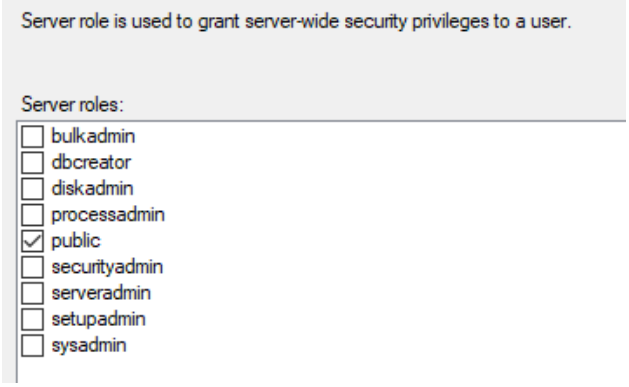


Рисунок 2.27 – Ролі користувача

Також можна створювати власні ролі, та додавати туди користувачів. Роль може надавати можливість вибірки, додавання, оновлення та видалення даних з таблиць.

Якщо ми хочемо обмежити доступ до певних таблиць, ми можемо скористатися такими способами:

– Створюємо ролі, даємо їм певні дозволи та додаємо користувачів. Найголовнішим кроком у цьому методі є те, що нам треба буде видалити доступ

до цих таблиць у ролі “public” , бо це дозволить заборонити доступ до таблиці всім користувачам, які не мають спеціальних дозволів.

1) Створити нову роль "collection" та надати їй дозвіл на доступ до таблиці "PAYMENTS":

```
USE fin_comp;
GO
CREATE ROLE collection;
GO
GRANT SELECT, INSERT, UPDATE, DELETE ON PAYMENTS TO collection;
GO
```

Рисунок 2.28 – Створення ролі

2) Додати користувача "d.rudenko" до ролі "collection":

```
USE fin_comp;
GO
EXEC sp_addrolemember 'collection', 'd.rudenko';
GO
```

Рисунок 2.29 – Додавання користувача до ролі

3) Видалити дозвіл на доступ до таблиці "PAYMENTS" у ролі public:

```
USE fin_comp;
GO
REVOKE SELECT, INSERT, UPDATE, DELETE ON PAYMENTS TO public;
GO
```

Рисунок 2.30 – Видалення доступу

– Створюємо перегляд (view), та додаємо туди лише потрібні дані. Наприклад: у нас є табличка CLIENTS, у якій містяться PERSONAL_ID та PHONE_NUMBER, то щоб ці поля не були доступними до перегляду усім користувачам ми створимо представлення. Це можна зробити у самому MS SQL Server Management Studio за допомогою функціоналу програми, або прописавши скрипт:

```
1  CREATE VIEW CLIENTS_VIEW AS
2  SELECT REGISTRATION_DATE, CLIENT_ID, SURNAME, NAME, MIDDLE_NAME, EMAIL
3  FROM CLIENTS;
```

Рисунок 2.31 – Створення представлення

У цьому дослідженні користувачу буде надано повний доступ до бази даних без будь-яких обмежень. Це доцільно, оскільки в даному випадку єдиним користувачем є автор цього дослідження.

2.5 Розробка засобів захисту БД

Якщо ми хочемо створити велику фінансову компанію, то нам потрібен буде сервер, де будуть зберігатися наші дані. Наприклад, у цьому дослідженні є таблиця CLIENTS, у якій по одному клієнту є лише один рядок, тобто, кожний клієнт є унікальним. А наприклад у PORTFOLIO міститься інформація по кредиту, у той час коли клієнт може мати дуже багато кредитів, також у цій таблиці зберігаються дані кожного дня. Якщо наш клієнт візьме кредит терміном 365 днів та оплатить у останній день, ми матимемо 365 записів по одному клієнту по його одному кредиту. Уже можна уявити, якщо наша компанія буде зростати, то нам треба буде десь усі дані зберігати. Звісно, дані можна чистити, але історичні дані можуть бути корисними, щоб будувати стратегії, проводити аналітику та використовувати нові або старі методики, які принесуть якнайбільшу вигоду. Кожного дня може з’являтися по 1000 нових клієнтів, 2000 заяв та 1500 платежів. Треба зауважити, що ці числа не є рекордними у цій сфері, до того ж у період війни.

В залежності від кількості даних, треба буде займати якесь приміщення для серверів, які обов’язково треба охороняти. Доступ до них повинен бути обмежений, та їхнє місцезнаходження не розкрите. Не можна відкидати й проблеми з електрикою, з якою зіткнулися жителі України, тому компанії для того, щоб бізнес продовжував функціонувати, переносили свої сервера у більш безпечні місця, або використовували

безперебойні джерела електроенергії, і це все лише для серверів. Окрім цього за серверами повинен бути постійний контроль у разі помилки з боку користувачів. Існує можливість, що ті працівники, що виконують аналітичні завдання та одночасно вигружають значні об’єми даних, можуть спричинити перевантаження бази даних та викликати її аварійне відключення.

Якщо наша компанія у майбутньому буде мати більше ніж один проект, то доцільно буде створювати окремо БД для окремого проекта, але якщо ідея буде у розповсюдженні по всьому світу та по інших країнах, треба буде мати більше одного сервера і бажано у різних місцях.

У попередньому пункті було сказано про ролі та надання/видалення доступу до таблиць, таким чином ми зможемо вберегти важливі дані, які не бажано розповсюджувати серед усієї компанії. Тобто з підвищенням кваліфікації працівнику будуть відткриватися доступи до інших таблиць з даними.

Не можна забувати й про аудит БД. Ми повинні контролювати усі дії у нашій базі даних та ким вони були зроблені, щоб у разі чого мати повну детальну інформації та уникнути небажаних входів, видалення/зміни даних, перегрузки БД. Наприклад якщо користувач запустить якийсь складний скрипт у якому він використовував таблиці з великим обсягом даних. У нашому випадку ми маємо табличку PORTFOLIO, у якій, як було сказано раніше, зберігається велика кількість даних. Уявімо, що наша компанія існує вже 10 років і хтось виконає запит, в якому буде викликати декілька разів цю табличку через конструкцію “with”, яка є дуже вимогливою до оперативної пам’яті, бо дозволяє створювати тимчасові таблиці-підзапити. Якщо українська компанія в умовах війни може мати за березень 2023 року 1.7 мільйонів записів лише у одній таблиці, можна уявити скільки буде даних через 10 років у нашій БД. Для уникнення небажаних перегрузок треба обмежувати кількість вигружаємих даних функцією TOP (n), де n – кількість строк, які користувач хоче вивести. Також треба зауважити, що наприклад у аналогічному інструменті для управління базами даних DBeaver кількість відображаємих строк автоматично обмежується, також можна виставити своє значення.

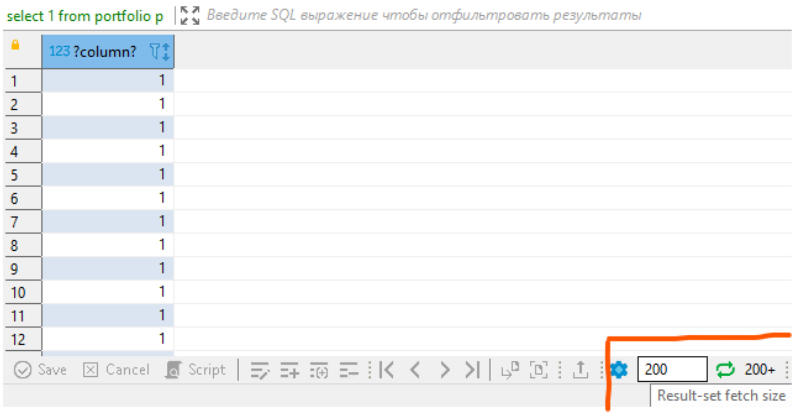


Рисунок 2.32 – Скріншот з DBeaver

Ситуація, коли треба вигружати повністю таблицю, може виникати або дуже рідко, або коли база даних відносно молода. Для обмеження даних, можна використовувати умову ($where\ date \geq$ 'дата, з якої треба показати дані').

Буває ситуація, коли ми хочемо записувати дані в базу лише для того, щоб зрозуміти чи заповнив клієнт відповідне поле. Наприклад, якщо ми в майбутньому захочемо створити власний застосунок та відслідковувати кроки реєстрації. Клієнту обов'язково треба буде заносити основний номер телефону та, по можливості, додатковий, але вони повинні бути різними. Ці два поля будуть зберігатися у нашій базі даних, але замість 4 цифр посередені будуть стояти зірочки. Можливі ситуації:

– Клієнт заповнив обидва поля (38067****676; 38076***767);
– Клієнт заповним лише обов'язкове поле (38067****676; NULL);
– Клієнт не заповнив обидва поля (NULL, NULL).

Таким чином, ми вирішили проблему, коли треба вигрузити інформацію про проходження кроку "контактна інформація", не розкриваючи конфіденційну інформацію.

Ще одним методом є шифрування.

Щоб зашифрувати дані, треба:

– Створити сертифікат за допомогою команди CREATE CERTIFICATE;
– Створити симетричний ключ за допомогою команди CREATE SYMMETRIC KEY і прив'язати його до сертифіката;

– Використати функцію ENCRYPTBYKEY для зашифрування даних та збереження їх у базі даних.

Приклад того, як може виглядати зашифрований PERSONAL_ID одного з клієнтів:

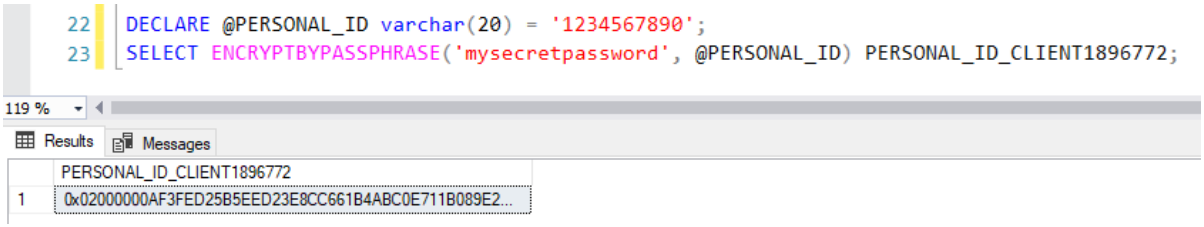


Рисунок 2.33 – Шифрування

Щоб дешифрувати дані треба:

– Створити сертифікат та симетричний ключ, які були використані для шифрування даних;

– Використати функцію DECRYPTBYKEY для дешифрування даних.

Приклад того, як може виглядати дешифрований PERSONAL_ID одного з клієнтів:

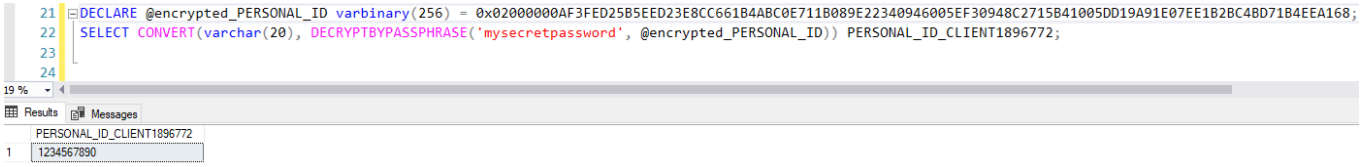


Рисунок 2.34 – Дешифрування

Варто зазначити, що шифрування даних може бути корисним для збереження конфіденційної інформації у базі даних. Проте, використання шифрування також може затримувати процеси пошуку та фільтрації даних, тому важливо уважно розглянути переваги та недоліки використання шифрування для конкретного випадку.

PERSONAL_ID та PHONE_NUMBER можна не шифрувати в деяких випадках, коли необхідна доступність цих даних у відкритому вигляді для ефективної роботи системи або в разі, коли вони не містять особливо конфіденційної інформації. Наприклад, номер телефону може бути доступним відкритим текстом для забезпечення можливості зв'язку з клієнтом або для зручності внесення даних, але при

цьому його необхідно захистити від несанкціонованого доступу, зламів інформаційної системи тощо. Також, PERSONAL_ID може бути не шифрований, якщо це не є обов'язковою вимогою законодавства або політики безпеки даних організації.

Наша компанія має багато критичних сервісів. Щоб підвищити рівень безпеки, доступ до цих сервісів має бути забезпечений через VPN-підключення, яке дозволяє працювати з віддаленими серверами та інфраструктурою компанії так, ніби ви знаходитесь в офісі. Якщо не використувувати даний метод, шанс зламати нашу БД підвищується. Зловмисники можуть отримати доступ до конфеденційних даних нашої компанії та клієнтів для шантажу, Якщо БД не захищена VPN, то зламувач може здійснити "прослуховування" мережевого трафіку і отримати доступ до даних,

Сервер VPN у дослідженні буде працювати за допомогою протоколу IPSec (IKEv2), але також можна було використовувати програмне забезпечення Cisco AnyConnect та openconnect. Для входу необхідний активний обліковий запис (ім'я користувача та пароль) в домені компанії

При спробі входу на сервер у MS SQL без підключення VPN, користувач побачить помилку входу

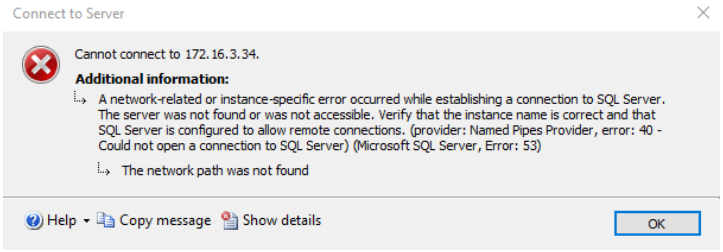


Рисунок 2.35 – Помилка при під’єднанні

У цьому повідомленні й сказано про помилку, яка пов’язана з мережою. Отже, без підключення VPN зловмисник не зможе ні видалити БД, ні зробити зміни, ні вигрузити дані, ні подивитися на її структуру. Таким чином ми обмежили коло осіб, які можуть користуватися нашим сервером та БД. Навіть, якщо виникне потреба звільнити працівника, який має усі доступи, вистачить видалити його доменний обліковий запис.

Отже, маємо декілька варінтів захисту БД від несанкціоного входу, але не можна забувати про те, що навіть досвідчений працівник може випадково видалити/змінити дані, тому треба завжди мати резервну копію базу даних, яка буде регулярно оновлючатися та зберігатися у безпечному місці.

Одним з найвідоміших прикладів порушення безпеки даних є атака на компанію Equifax у 2017 році, під час якої злочинці отримали доступ до особистих даних понад 143 мільйонів клієнтів. Компанія могла б вжити наведені у цьому дослідженні заходів для захисту своєї бази даних таких як : встановлення міцних паролів та двофакторної автентифікації для всіх користувачів з доступом до БД, встановлення мережевих файрволів та обмеження доступу до бази даних зовнішніх користувачів, дозволяючи доступ лише з віддалених робочих станцій з правильними ключами безпеки VPN.

Це був приклад того, як розумне використання засобів захисту баз даних може допомогти компанії уникнути проблем з безпекою. Навіть якщо у компанії є деякі заходи захисту БД, можна їх оптимізувати, вдосконалити та доповнити новими засобами захисту.

2.6 Тестування та валідація розробленої БД

Тестування та валідація розробленої бази даних є важливим етапом в процесі розробки. Його мета полягає у виявленні помилок, пропущених або некоректно виконаних дій, а також перевірці правильності роботи БД.

Так як ми налаштували доступ до серверу через VPN, треба спочатку його підключити

Рисунок 2.36 – Підключення VPN

Далі підключаємося до бази даних, указуючи сервер та використовуючи логін і пароль користувача, якого ми створили.

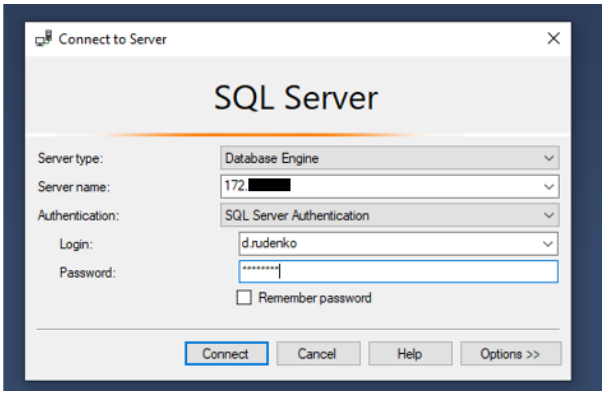


Рисунок 2.37 – Вікно авторизації

Якщо б VPN не був підключений, при спробі під’єднання ми б отримали помилку, яка була висвітлена у попередньому пункті. При введенні неправильного логіну або паролю, у систему не пустить.

При спробі утворення view, ми отримуємо помилку:

“ Msg 262, Level 14, State 18, Procedure test_view, Line 1 [Batch Start Line 0] CREATE VIEW permission denied in database…”.

При спробі створення таблиці, ми отримуємо помилку :

“Msg 262, Level 14, State 1, Line 1 CREATE TABLE permission denied in database…”.

При спробі додавання даних у існуючу таблицю, ми отримуємо помилку:

“Msg 229, Level 14, State 5, Line 1 The INSERT permission was denied on the object 'APPLICATION'…”

При спробі прочитати дані, наш скрипт успішно виконується, адже користувачу була надана роль “public”, яка не дозволяє створювати таблиці, представлення або додавати дані у таблицю.

Дозаповнимо нашу базу новими даними та спробуємо зробити вибірку, у якій виведемо клієнтський ідентифікатор, номер телефону та найбільшу запрошену суму цим клієнтом для того, щоб показати, що номер телефону неможливо прочитати, навіть якщо нам він треба.

	CLIENT_ID	PHONE_NUMBER	max_REQUEST_AMOUNT
1	2960197	38073****4741	4100
2	2814626	38063****9833	5000
3	2443295	38096****9153	1000
4	1678461	38068****8251	3000
5	2928923	38068****1830	4000
6	1951003	38096****4223	5000
7	1684404	38098****9017	20000
8	2578679	38097****0954	5000
9	2091045	38097****4980	10000
10	2384634	38098****8352	10000
11	2192255	38063****2256	8500
12	1652957	38050****7109	24000
13	1793824	38066****4872	6000
14	334190	38099****5952	1000
15	2752484	38097****4474	5000
16	2281303	38098****3447	4000
17	3080292	38050****5690	1100
18	2203208	38063****0161	3000
19	1924974	38067****6693	3000

Рисунок 2.38 – Вибірка даних

Це означає, що ми можемо впенитися, що клієнт, який запрошував таку суму, має номер телефону. Далі, наприклад, для проведення акції надсилання смс, ми можемо надіслати у департамент, який займається цими питаннями, список клієнтів та суму, попередньо прибравши з вибірки клієнтів, у котрих у стовпчику PHONE_NUMBER не вказаний номер (але у нашому випадку номер повинен бути завжди, бо база була побудована таким чином, що значення стовпчику PHONE_NUMBER ненульове). Цей приклад слугував для демонстрації того, як можна скрити конфеденційні дані.

Методи забезпечення безпеки бази даних спрацювали, а це означає, що користувачі бази даних можуть працювати з нею, не наражаючи компанію на таку небезпеку як злив даних.

Оскільки БД налаштована правильно, тоді можна приступити до тестування саме її наповнення, до вибрки даних з таблиць за певними умовами.

2.7 Аналіз та звітність

База створена, вона у безпеці, дані заповнені. Але у такому вигляді ці дані не є дуже корисними. Для того, щоб отримувати потрібну нам вигрузку даних, ми будемо використовувати мову SQL. Для аналізу ланих можна утворювати різного рівня

складності запити. У даному дослідженні наведемо приклади, які будуть корисними саме для нашого випадку, та є авторськими завданнями та скриптами.

З метою проведення маркетингових заходів можна сформувати бази даних "Позитивних" клієнтів за наведеними нижче параметрами:

– Кількість прострочених днів не перевищує 3 дні;
– Клієнт без кредиту протягом 5-120 днів.

```
-----"Позитивні" клієнти:
 --клієнти без прострочки
select z.*, a.REPORT_DATE  from
(select CLIENT_ID, max(DPD) DPD
from PORTFOLIO
group by CLIENT_ID) z
left join
 --клієнти без кредиту
(select CLIENT_ID , max(REPORT_DATE) REPORT_DATE from PORTFOLIO
group by CLIENT_ID ) a
on a.CLIENT_ID=z.CLIENT_ID
where z.DPD <=3 and a.REPORT_DATE >=dateadd(day,-121,GETDATE() ) and a.REPORT_DATE <dateadd(day,-5,GETDATE() )
```

Рисунок 2.39 – SQL-скрипт

Клієнти без прострочки: Виконується підзапит, який групує дані з таблиці PORTFOLIO за CLIENT_ID і обирає максимальне значення прострочки (DPD) для кожного клієнта. Потім зовнішнє поєднання (LEFT JOIN) з додатковим підзапитом додає до результатів поле REPORT_DATE (остання дата звіту) для кожного клієнта. В кінці WHERE умовою обмежуємо результати, щоб DPD було не більше 3 і REPORT_DATE було протягом останніх 5-120 днів.

Отже, цей скрипт вибирає клієнтів, які не мають прострочок (DPD <= 3) і які не мали кредитів протягом останніх 5-120 днів (a.REPORT_DATE >= dateadd(day,-121,GETDATE()) і a.REPORT_DATE < dateadd(day,-5,GETDATE())). Результати включають поля CLIENT_ID, EMAIL_ADDRESS і REPORT_DATE з таблиці PORTFOLIO для цих клієнтів. Результат:

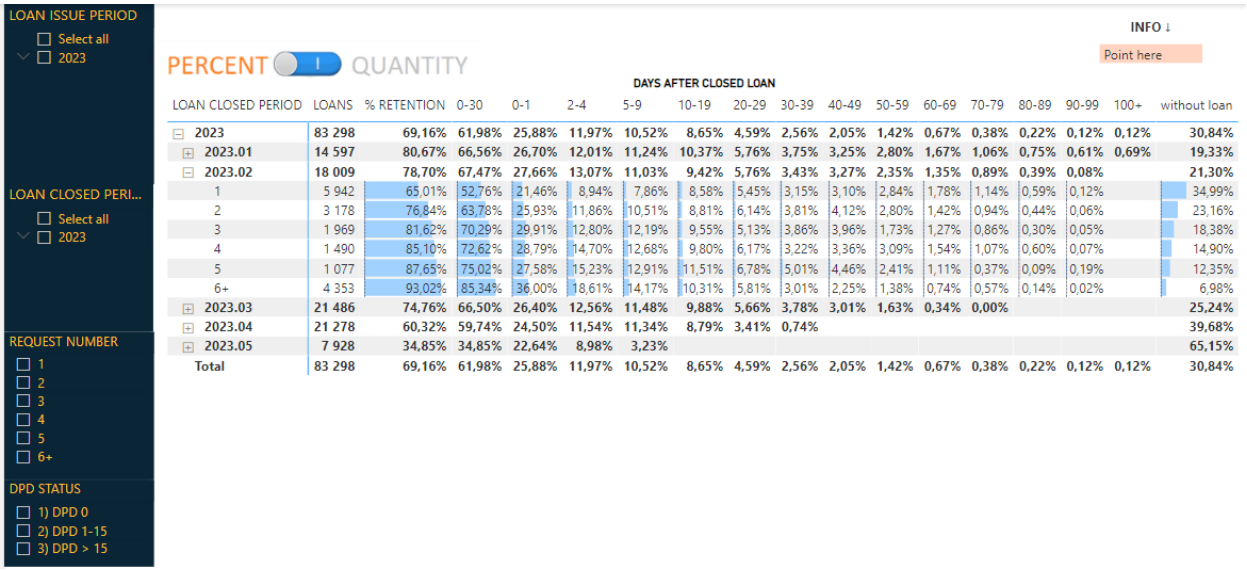
	CLIENT_ID	EMAI_ADDRESS	DPD	REPORT_DATE
1	278	danylo@rudenko	0	2023-04-20
2	452	danylo@rudenko	0	2023-04-20
3	742	danylo@rudenko	0	2023-04-19
4	960	danylo@rudenko	0	2023-04-15
5	1170	danylo@rudenko	0	2023-04-12
6	1298	danylo@rudenko	1	2023-02-06
7	1301	danylo@rudenko	0	2023-05-03
8	1484	danylo@rudenko	0	2023-05-05
9	1598	danylo@rudenko	0	2023-04-07
10	2049	danylo@rudenko	2	2023-04-28
11	2485	danylo@rudenko	0	2023-03-15
12	2982	danylo@rudenko	0	2023-01-25
13	3228	danylo@rudenko	0	2023-04-26
14	3324	danylo@rudenko	0	2023-03-19

Рисунок 2.40 – Фрагмент результату виконання SQL-скрипту (2.39)

Це був приклад простої вибірки клієнтів, але окрім цього за допомогою скриптів та допоміжних програм можна отримувати інформативні та зручні звіти. Ми будемо використовувати PowerBI, аналітичний інструмент, розроблений тією ж самою компанією, що й розглядаємий MS SQL, тобто компанією Microsoft. Він є дуже зручним для введення бізнесу, бо має два компоненти, в одному з яких користувач створює, налаштовує звіти та візуалізацію, а в другому, який є хмарною платформою, публікує, інтерактивно взаємодіє з даними та спільно працює з колегами.

За допомогою створеної БД можна побудувати звіти з retention, які будуть показувати лояльність клієнтів. Це поняття є дуже важливим у бізнес-аналітиці. З даними, які містяться у нашій базі даних, можна побудувати декілька таких звітів, які схожі між собою логікою, але є дуже корисними для аналітики.

Retention detailed:



LOAN CLOSED PERIOD	LOANS	% RETENTION	0-30	0-1	2-4	5-9	10-19	20-29	30-39	40-49	50-59	60-69	70-79	80-89	90-99	100+	without loan
2023	83 298	69,16%	61,98%	25,88%	11,97%	10,52%	8,65%	4,59%	2,56%	2,05%	1,42%	0,67%	0,38%	0,22%	0,12%	0,12%	30,84%
2023.01	14 597	80,67%	66,56%	26,70%	12,01%	11,24%	10,37%	5,76%	3,75%	3,25%	2,80%	1,67%	1,06%	0,75%	0,61%	0,69%	19,33%
2023.02	18 009	78,70%	67,47%	27,66%	13,07%	11,03%	9,42%	5,76%	3,43%	3,27%	2,35%	1,35%	0,89%	0,39%	0,08%		21,30%
1	5 942	65,01%	52,76%	21,46%	8,94%	7,86%	8,58%	5,45%	3,15%	3,10%	2,84%	1,78%	1,14%	0,59%	0,12%		34,99%
2	3 178	76,84%	63,78%	25,93%	11,86%	10,51%	8,81%	6,14%	3,81%	4,12%	2,80%	1,42%	0,94%	0,44%	0,06%		23,16%
3	1 969	81,62%	70,29%	29,91%	12,80%	12,19%	9,55%	5,13%	3,86%	3,96%	1,73%	1,27%	0,86%	0,30%	0,05%		18,38%
4	1 490	85,10%	72,62%	28,79%	14,70%	12,68%	9,80%	6,17%	3,22%	3,36%	3,09%	1,54%	1,07%	0,60%	0,07%		14,90%
5	1 077	87,65%	75,02%	27,58%	15,23%	12,91%	11,51%	6,78%	5,01%	4,46%	2,41%	1,11%	0,37%	0,09%	0,19%		12,35%
6+	4 353	93,02%	85,34%	36,00%	18,61%	14,17%	10,31%	5,81%	3,01%	2,25%	1,38%	0,74%	0,57%	0,14%	0,02%		6,98%
2023.03	21 486	74,76%	66,50%	26,40%	12,56%	11,48%	9,88%	5,66%	3,78%	3,01%	1,63%	0,34%	0,00%				25,24%
2023.04	21 278	60,32%	59,74%	24,50%	11,54%	11,34%	8,79%	3,41%	0,74%								39,68%
2023.05	7 928	34,85%	34,85%	22,64%	8,98%	3,23%											65,15%
Total	83 298	69,16%	61,98%	25,88%	11,97%	10,52%	8,65%	4,59%	2,56%	2,05%	1,42%	0,67%	0,38%	0,22%	0,12%	0,12%	30,84%

Рисунок 2.41 – Звіт у PowerBI Retention detailed

У даному звіті враховуються всі погашені кредити (за обраний період/порядковий номер) та потім будується графік їх конверсії в повторні (з деталізацією за кількістю днів без кредиту, що пройшли до видачі наступного кредиту).

Робота з перемикачем:

– PERCENT – відображаються дані у відсотках

– QUANTITY – відображаються дані в кількості

Робота з фільтрами:

– LOAN ISSUE PERIOD – період видачі кредиту

– LOAN CLOSED PERIOD - період погашення кредиту (включаючи майбутні дати)

– REQUEST NUMBER – порядковий номер зявки погашеного кредиту

– DPD STATUS – кількість днів прострочки на погашеному кредиті

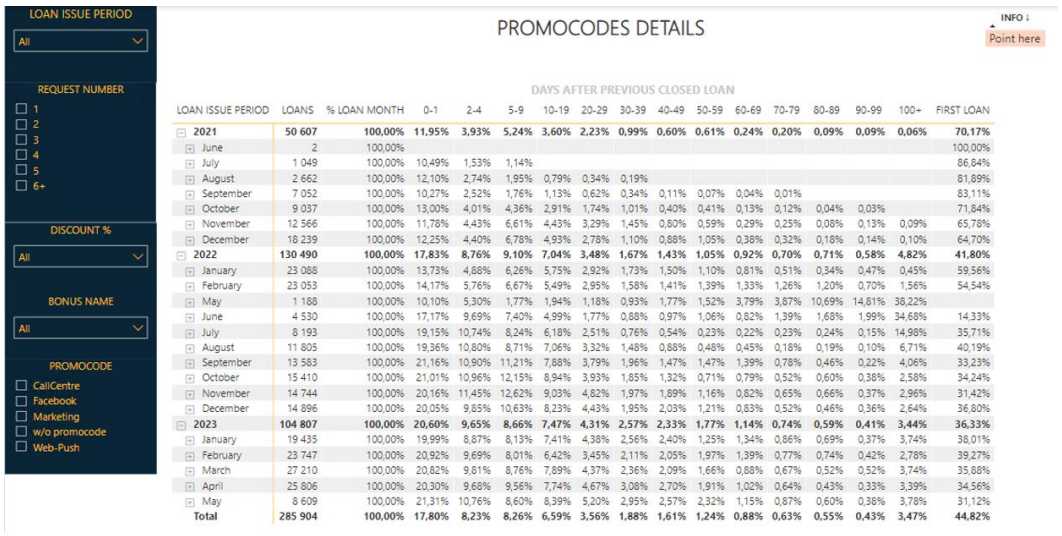
Close to application:

Виглядає так само, але будуть інші результати

У даному звіті враховуються всі погашені кредити (за обраний період/порядковий номер) і далі будується графік їх конверсії в подачу заявки на повторний кредит (з деталізацією за кількістю днів без кредиту, що пройшли до подачі заявки на наступний кредит).

Робота з перемикачем та фільтрами така сама.

Promocode details:



LOAN ISSUE PERIOD	LOANS	% LOAN MONTH	0-1	2-4	5-9	10-19	20-29	30-39	40-49	50-59	60-69	70-79	80-89	90-99	100+	FIRST LOAN
2021	**50 607**	**100,00%**	**11,95%**	**3,93%**	**5,24%**	**3,60%**	**2,23%**	**0,99%**	**0,60%**	**0,61%**	**0,24%**	**0,20%**	**0,09%**	**0,09%**	**0,06%**	**70,17%**
June	2	100,00%														100,00%
July	1 049	100,00%	10,49%	1,53%	1,14%											86,64%
August	2 662	100,00%	12,10%	2,74%	1,95%	0,79%	0,34%	0,19%								81,89%
September	7 052	100,00%	10,27%	2,52%	1,76%	1,13%	0,62%	0,34%	0,11%	0,07%	0,04%	0,01%				83,11%
October	9 037	100,00%	13,00%	4,01%	4,36%	2,91%	1,74%	1,01%	0,40%	0,41%	0,13%	0,12%	0,04%	0,03%		71,84%
November	12 566	100,00%	11,78%	4,43%	6,61%	4,43%	3,29%	1,45%	0,80%	0,59%	0,29%	0,25%	0,08%	0,13%	0,09%	65,78%
December	18 239	100,00%	12,25%	4,40%	6,78%	4,93%	2,78%	1,10%	0,88%	1,05%	0,38%	0,32%	0,18%	0,14%	0,10%	64,70%
2022	**130 490**	**100,00%**	**17,83%**	**8,76%**	**9,10%**	**7,04%**	**3,48%**	**1,67%**	**1,43%**	**1,05%**	**0,92%**	**0,70%**	**0,71%**	**0,58%**	**4,82%**	**41,80%**
January	23 088	100,00%	13,73%	4,88%	6,26%	5,75%	2,92%	1,73%	1,50%	1,10%	0,81%	0,51%	0,34%	0,47%	0,45%	59,56%
February	23 053	100,00%	14,17%	5,76%	6,67%	5,49%	2,95%	1,58%	1,41%	1,39%	1,33%	1,26%	1,20%	0,70%	1,56%	54,54%
May	1 188	100,00%	10,10%	5,30%	1,77%	1,94%	1,18%	0,93%	1,77%	1,52%	3,79%	3,87%	10,69%	14,81%	38,22%	
June	4 530	100,00%	17,17%	9,69%	7,40%	4,99%	1,77%	0,88%	0,97%	1,06%	0,82%	1,39%	1,68%	1,99%	34,68%	14,33%
July	8 193	100,00%	19,15%	10,74%	8,24%	6,18%	2,51%	0,76%	0,54%	0,23%	0,22%	0,23%	0,24%	0,15%	14,98%	35,71%
August	11 805	100,00%	19,36%	10,80%	8,71%	7,06%	3,32%	1,48%	0,88%	0,48%	0,45%	0,18%	0,19%	0,10%	6,71%	40,19%
September	13 583	100,00%	21,16%	10,90%	11,21%	7,88%	3,79%	1,96%	1,47%	1,47%	1,39%	0,78%	0,46%	0,22%	4,06%	33,23%
October	15 410	100,00%	21,01%	10,96%	12,15%	8,94%	3,93%	1,85%	1,32%	0,71%	0,79%	0,52%	0,60%	0,38%	2,58%	34,24%
November	14 744	100,00%	20,16%	11,45%	12,62%	9,03%	4,82%	1,97%	1,89%	1,16%	0,82%	0,65%	0,66%	0,37%	2,96%	31,42%
December	14 896	100,00%	20,05%	9,85%	10,63%	8,23%	4,43%	1,95%	2,03%	1,21%	0,83%	0,52%	0,46%	0,36%	2,64%	36,80%
2023	**104 807**	**100,00%**	**20,60%**	**9,65%**	**8,66%**	**7,47%**	**4,31%**	**2,57%**	**2,33%**	**1,77%**	**1,14%**	**0,74%**	**0,59%**	**0,41%**	**3,44%**	**36,33%**
January	19 435	100,00%	19,99%	8,87%	8,13%	7,41%	4,38%	2,56%	2,40%	1,25%	1,34%	0,86%	0,69%	0,37%	3,74%	38,01%
February	23 747	100,00%	20,92%	9,69%	8,01%	6,42%	3,45%	2,11%	2,05%	1,97%	1,39%	0,77%	0,74%	0,42%	2,78%	39,27%
March	27 210	100,00%	20,82%	9,61%	8,76%	7,89%	4,37%	2,36%	2,09%	1,66%	0,88%	0,67%	0,52%	0,52%	3,74%	35,88%
April	25 806	100,00%	20,30%	9,68%	9,56%	7,74%	4,67%	3,08%	2,70%	1,91%	1,02%	0,64%	0,43%	0,33%	3,39%	34,56%
May	8 609	100,00%	21,31%	10,76%	8,60%	8,39%	5,20%	2,95%	2,57%	2,32%	1,15%	0,87%	0,60%	0,38%	3,78%	31,12%
Total	**285 904**	**100,00%**	**17,80%**	**8,23%**	**8,26%**	**6,59%**	**3,56%**	**1,88%**	**1,61%**	**1,24%**	**0,88%**	**0,63%**	**0,55%**	**0,43%**	**3,47%**	**44,82%**

Рисунок 2.42 – Звіт у PowerBI Promocode details

У даному звіті відображається відсоток виданих кредитів в періоді, з деталізацією за виданим порядковим номером та застосованим промокодом.

Робота з фільтрами майже така сама окрім:

– PROMOCODE – призначення промокоду (CC, Маркетинг, Facebook, Web-Push). Призначення визначається в скрипті в залежності від назви
– DISCOUNT % – номінал промокоду
– CODE – промокод

ВИНСОВКИ

Результати дослідження показали, що створена база даних і використані методи захисту даних забезпечують високий рівень безпеки та конфіденційності інформації. База даних була успішно заповнена актуальними даними про клієнтів, включаючи інформацію про прострочені дні та статус кредиту. Це надає можливість для подальшого аналізу та використання даних для маркетингових цілей.

Застосування мови SQL дозволило побудувати різні звіти, спрямовані на аналіз retention та лояльності клієнтів. Звіти з retention надають організації уявлення про те, який відсоток клієнтів залишається з компанією протягом певного періоду. Це дозволяє оцінити ефективність стратегій залучення та утримання клієнтів. Крім того, аналіз залучення клієнтів за різними параметрами, наприклад, кількість прострочених днів та період без кредиту, дозволяє виокремити групи "Позитивних" клієнтів, що можуть бути цільовою аудиторією для маркетингових заходів.

Отримані результати свідчать про важливість ретеншну та лояльності клієнтів для організації. Шляхом аналізу цих даних, організація може отримати цінну інформацію про те, як зберігати клієнтів та покращити їх задоволеність. Результати дослідження підтверджують можливості бази даних і мови SQL в аналізі та використанні даних про клієнтів для вирішення бізнес-задач.

Далі можна дослідити більш глибокі аспекти лояльності клієнтів, такі як причини, які впливають на їхнє рішення залишатися або покидати організацію. Це може включати аналіз клієнтських опитувань, збору додаткової інформації про взаємодію клієнтів з продуктом або послугами компанії. Такий аналіз дозволить ідентифікувати ключові чинники, які впливають на лояльність клієнтів і встановити зв'язки між різними факторами та рівнем лояльності.

Дослідження може бути спрямоване на розробку персоналізованих стратегій залучення та утримання клієнтів. Це може включати вивчення індивідуальних потреб клієнтів, виявлення сегментів клієнтів зі схожими характеристиками та використання цих даних для розробки націленої комунікації та маркетингових пропозицій.

У цій області існує безліч перспектив для подальших досліджень і вдосконалення заходів безпеки баз даних. Дослідження можуть бути спрямовані на розробку нових алгоритмів шифрування, які були б ще більш ефективними, надійними та стійкими до атак, на розвиток більш точних та ефективних систем виявлення вторгнень, які здатні виявляти навіть складні та хитрі атаки або розробку автоматизованих засобів та інструментів для виявлення вразливостей, контролю доступу та аналізу безпеки баз даних.

Важливим напрямком досліджень є розробка та вдосконалення методів захисту від внутрішніх загроз, таких як несанкціонований доступ з боку співробітників або зловживання даними.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. SQL Server technical documentation - SQL Server | Microsoft Learn
2. Microsoft security documentation - Security documentation | Microsoft Learn
3. "Database System Concepts" - Автори: Abraham Silberschatz, Henry F. Korth, S. Sudarshan.
4. "SQL Cookbook" - Автор: Anthony Molinaro.
5. "Database Security and Auditing: Protecting Data Integrity and Accessibility" - Автор: Hassan A. Afyouni.
6. "Microsoft SQL Server 2019: A Beginner's Guide" автора Dusan Petkovic.
7. "Microsoft SQL Server 2019 Developer's Guide" автора Dejan Sarka, Milos Radivojevic, William Durkin, та ін.

Додаток А

Скрипт для “Retention detailed”, “Close to application” та “Promocode using”:

```
select  [CLIENT_ID ], [LOAN_ID ], [LOAN_NUMBER ], [BEGIN_DATE ], [CLOSE_DATE ], begin_date_next,
        FORMAT ([CLOSE_DATE ],  'yyyy-MM' ) close_period,
        FORMAT ([BEGIN_DATE ], 'yyyy-MM')  begin_period,
        DATEDIFF(day, [CLOSE_DATE ], begin_date_next) diffday,
        case    when    DATEDIFF(day, [CLOSE_DATE ], begin_date_next) <=  4  then '1. 0-4'
                when    DATEDIFF(day, [CLOSE_DATE ], begin_date_next) <=  9  then '2. 5-9'
                when    DATEDIFF(day, [CLOSE_DATE ], begin_date_next) <= 30  then '3. 10-30'
                when    DATEDIFF(day, [CLOSE_DATE ], begin_date_next) <= 60  then '4. 31-60'
                when    DATEDIFF(day, [CLOSE_DATE ], begin_date_next) <= 90  then '5. 61-90'
                when    DATEDIFF(day, [CLOSE_DATE ], begin_date_next) <= 120 then '6. 91-120'
                when    DATEDIFF(day, [CLOSE_DATE ], begin_date_next) <= 150 then '7. 121-150'
                when    DATEDIFF(day, [CLOSE_DATE ], begin_date_next) <= 180 then '8. 151-180'
                when    DATEDIFF(day, [CLOSE_DATE ], begin_date_next) >  180 then '9. 180+'
                else 'only one credit'
        end days_without_credits,
        case    when    DATEDIFF(day, [CLOSE_DATE ], begin_date_next) <=  1  then '000-001'
                when    DATEDIFF(day, [CLOSE_DATE ], begin_date_next) <=  4  then '002-004'
                when    DATEDIFF(day, [CLOSE_DATE ], begin_date_next) <=  9  then '005-009'
                when    DATEDIFF(day, [CLOSE_DATE ], begin_date_next) <= 19  then '010-019'
                when    DATEDIFF(day, [CLOSE_DATE ], begin_date_next) <= 29  then '020-029'
                when    DATEDIFF(day, [CLOSE_DATE ], begin_date_next) <= 39  then '030-039'
                when    DATEDIFF(day, [CLOSE_DATE ], begin_date_next) <= 49  then '040-049'
                when    DATEDIFF(day, [CLOSE_DATE ], begin_date_next) <= 59  then '050-059'
                when    DATEDIFF(day, [CLOSE_DATE ], begin_date_next) <= 69  then '060-069'
                when    DATEDIFF(day, [CLOSE_DATE ], begin_date_next) <= 79  then '070-079'
                when    DATEDIFF(day, [CLOSE_DATE ], begin_date_next) <= 89  then '080-089'
                when    DATEDIFF(day, [CLOSE_DATE ], begin_date_next) <= 99  then '090-099'
                when    DATEDIFF(day, [CLOSE_DATE ], begin_date_next) >  99  then '100+'
                else 'without loan'
        end retention_group,
```

Рисунок А.1 – Перша частина скрипту для побудови звітів для retention

```
DATEDIFF(day, [CLOSE_DATE ], lead_request_date ) as days_to_app,

case    when    DATEDIFF(day, [CLOSE_DATE ], lead_request_date) <   0   then 'other credit'
        when    DATEDIFF(day, [CLOSE_DATE ], lead_request_date) <   5   then '0-4'
        when    DATEDIFF(day, [CLOSE_DATE ], lead_request_date) <   10  then '5-9'
        when    DATEDIFF(day, [CLOSE_DATE ], lead_request_date) <   30  then '10-30'
        when    DATEDIFF(day, [CLOSE_DATE ], lead_request_date) >=  31  then '31+'
        else 'only one credit'
end group_days_to_app,
case    when    DATEDIFF(day, [CLOSE_DATE ], lead_request_date) <=  1  then '000-001'
        when    DATEDIFF(day, [CLOSE_DATE ], lead_request_date) <=  4  then '002-004'
        when    DATEDIFF(day, [CLOSE_DATE ], lead_request_date) <=  9  then '005-009'
        when    DATEDIFF(day, [CLOSE_DATE ], lead_request_date) <= 19  then '010-019'
        when    DATEDIFF(day, [CLOSE_DATE ], lead_request_date) <= 29  then '020-029'
        when    DATEDIFF(day, [CLOSE_DATE ], lead_request_date) <= 39  then '030-039'
        when    DATEDIFF(day, [CLOSE_DATE ], lead_request_date) <= 49  then '040-049'
        when    DATEDIFF(day, [CLOSE_DATE ], lead_request_date) <= 59  then '050-059'
        when    DATEDIFF(day, [CLOSE_DATE ], lead_request_date) <= 69  then '060-069'
        when    DATEDIFF(day, [CLOSE_DATE ], lead_request_date) <= 79  then '070-079'
        when    DATEDIFF(day, [CLOSE_DATE ], lead_request_date) <= 89  then '080-089'
        when    DATEDIFF(day, [CLOSE_DATE ], lead_request_date) <= 99  then '090-099'
        when    DATEDIFF(day, [CLOSE_DATE ], lead_request_date) >  99  then '100+'
        else 'without application'
end app_retention_group,
last_date_closed,
DATEDIFF(day, last_date_closed, begin_date) as dayspromodiff,
```

Рисунок А.2 – Друга частина скрипту для побудови звітів для retention

```
case    when    DATEDIFF(day, last_date_closed, begin_date) <=  1  then '000-001'
        when    DATEDIFF(day, last_date_closed, begin_date) <=  4  then '002-004'
        when    DATEDIFF(day, last_date_closed, begin_date) <=  9  then '005-009'
        when    DATEDIFF(day, last_date_closed, begin_date) <= 19  then '010-019'
        when    DATEDIFF(day, last_date_closed, begin_date) <= 29  then '020-029'
        when    DATEDIFF(day, last_date_closed, begin_date) <= 39  then '030-039'
        when    DATEDIFF(day, last_date_closed, begin_date) <= 49  then '040-049'
        when    DATEDIFF(day, last_date_closed, begin_date) <= 59  then '050-059'
        when    DATEDIFF(day, last_date_closed, begin_date) <= 69  then '060-069'
        when    DATEDIFF(day, last_date_closed, begin_date) <= 79  then '070-079'
        when    DATEDIFF(day, last_date_closed, begin_date) <= 89  then '080-089'
        when    DATEDIFF(day, last_date_closed, begin_date) <= 99  then '090-099'
        when    DATEDIFF(day, last_date_closed, begin_date) >  99  then '100+'
        when    LOAN_NUMBER = 1 then 'first_loan'
        else 'no info'
end promo_retention_group,
case    when [LOAN_NUMBER ] <  6    and [CLOSE_DATE ] is not null   then    cast([LOAN_NUMBER ] as varchar)
        when [LOAN_NUMBER ] >= 6    and [CLOSE_DATE ] is not null   then    '6+'
        else 'no closed loans'
        end close_loan_number,
case    when [LOAN_NUMBER ] <  6 then   cast([LOAN_NUMBER ] as varchar)
        when [LOAN_NUMBER ] >= 6 then   '6+'
        end as number_loan,
case    when [LOAN_NUMBER ] <  3 then   cast([LOAN_NUMBER ] as varchar)
        when [LOAN_NUMBER ] <  5 then   '3-4'
        when [LOAN_NUMBER ] >= 5 then   '5+'
        end as number_loan2,
```

Рисунок А.3 – Третя частина скрипту для побудови звітів для retention

```
case    when    [DPD ] <=   0   then    '1) DPD 0'
        when    [DPD ]  <=  15  then    '2) DPD 1-15'
        when    [DPD ] >    15  then    '3) DPD > 15'
        end dpd_status,
[DPD ],
datediff(day, [BEGIN_DATE ], [CLOSE_DATE ]) FACT_TERM, [LOAN_TERM ] as term_backet,
case    when [LOAN_TERM ] <= 5  then '1. 5'
        when [LOAN_TERM ] <= 9  then '2. 6-9'
        when [LOAN_TERM ] <= 15 then '3. 10-15'
        when [LOAN_TERM ] <= 20 then '4. 16-20'
        when [LOAN_TERM ] <= 25 then '5. 21-25'
        when [LOAN_TERM ] <= 30 then '6. 26-30'
        else 'other'
end issued_term,
case    when datediff(day, [BEGIN_DATE ], [CLOSE_DATE ]) <= 1   then '1. 0-1'
        when datediff(day, [BEGIN_DATE ], [CLOSE_DATE ]) <= 5   then '2. 2-5'
        when datediff(day, [BEGIN_DATE ], [CLOSE_DATE ]) <= 9   then '3. 6-9'
        when datediff(day, [BEGIN_DATE ], [CLOSE_DATE ]) <= 15  then '4. 10-15'
        when datediff(day, [BEGIN_DATE ], [CLOSE_DATE ]) <= 20  then '5. 16-20'
        when datediff(day, [BEGIN_DATE ], [CLOSE_DATE ]) <= 25  then '6. 21-25'
        when datediff(day, [BEGIN_DATE ], [CLOSE_DATE ]) <= 30  then '7. 26-30'
        when datediff(day, [BEGIN_DATE ], [CLOSE_DATE ]) >  30  then '8. >30'
        else 'active'
end group_fact_term,
```

Рисунок А.4 – Четверта частина скрипту для побудови звітів для retention

```
case    when datepart(day, [BEGIN_DATE ]) <=  5 then '1. 1-5'
        when datepart(day, [BEGIN_DATE ]) <= 10 then '2. 6-10'
        when datepart(day, [BEGIN_DATE ]) <= 15 then '3. 11-15'
        when datepart(day, [BEGIN_DATE ]) <= 20 then '4. 16-20'
        when datepart(day, [BEGIN_DATE ]) <= 25 then '5. 21-25'
        when datepart(day, [BEGIN_DATE ]) >  25 then '6. >25'
        else 'other'
end group_day,
case    when DISCOUNT_NAME is null then 'w/o promocode'
    else DISCOUNT_NAME
end code,

case    when DISCOUNT_PERCENT is null then 'w/o promocode'
    else cast(DISCOUNT_PERCENT as varchar)
end discount,

case    when BONUS_NAME LIKE N'%промо для КЦ%' or BONUS_NAME LIKE N'%сотрудник%' then 'CallCentre'
        when BONUS_NAME LIKE N'%Facebook%' then 'Facebook'
        when BONUS_NAME LIKE N'%пуш-рассылка%' then 'Web-Push'
        when DISCOUNT_PERCENT is not null then 'Marketing'
        when DISCOUNT_PERCENT is null then 'w/o promocode'
end PROMOCODE,

BONUS_NAME
```

Рисунок А.5 – П’ята частина скрипту для побудови звітів для retention

```
from    (   select  [CLIENT_ID ], [LOAN_ID ], [LOAN_NUMBER ], [LOAN_TERM ], [BEGIN_DATE ], [CLOSE_DATE ],
                    LEAD([BEGIN_DATE ] ) over(partition by [CLIENT_ID ] order by [LOAN_NUMBER ] ) as begin_date_next,
                    (LAG(close_date) over (partition by clients.CLIENT_ID order by LOAN_NUMBER)) as last_date_closed,
                    [DPD ], lead_application_id, lead_request_date
            from    (   select  opened.[CLIENT_ID ],
                                opened.[LOAN_ID ],
                                closed.[CLOSE_DATE ],
                                opened.[LOAN_NUMBER ],
                                opened.[LOAN_TERM ],
                                opened.[BEGIN_DATE ],
                                opened.[DPD ],

                                a.lead_application_id,
                                a.lead_request_date
                        from    (   select  [CLIENT_ID ], [LOAN_NUMBER ], [LOAN_TERM ], portfolio.[LOAN_ID ], [BEGIN_DATE ], max([DPD ]) as [DPD ]
                            from    (   select  [CLIENT_ID ], [LOAN_NUMBER ], [LOAN_ID ], [BEGIN_DATE ], [LOAN_TERM ]
                                        from PORTFOLIO p
                                        group by  [CLIENT_ID ], [LOAN_NUMBER ], [LOAN_ID ], [BEGIN_DATE ], [LOAN_TERM ], PRODUCT_NAME
                                    ) as portfolio
                                    left join
                                    (   select  p.[LOAN_ID ], p.[DPD ]
                                        from PORTFOLIO p
                                        union
                                        select  distinct  pm.[LOAN_ID ], pm.[DPD ]
                                        from PAYMENTS pm
                                    ) as dpd
                                    on dpd.[LOAN_ID ] = portfolio.[LOAN_ID ]
                            group by [CLIENT_ID ], [LOAN_NUMBER ], [LOAN_TERM ], portfolio.[LOAN_ID ], [BEGIN_DATE ]
                        ) as opened
```

Рисунок А.6 – Шоста частина скрипту для побудови звітів для retention

```
                        -- закрытые кредиты
                        left join
                        (   select  [LOAN_ID ], min([CLOSE_DATE ]) [CLOSE_DATE ]
                            from PORTFOLIO p2
                            where [CLOSE_DATE ] is not null
                            group by [LOAN_ID ]
                        ) as closed
                        on closed.[LOAN_ID ] = opened.[LOAN_ID ]
                                                    left join
                        ( -- вычесляем дату слудеющей заявки после закрытия кредита
                        select [CLIENT_ID ],
                        [ANKETA_ID ] ,
                        lead([ANKETA_ID ]) over(partition by [CLIENT_ID ] order by request_number) lead_application_id,
                        request_date,
                        lead(request_date) over(partition by [CLIENT_ID ] order by request_number) lead_request_date,
                        request_number ,
                        lead(request_number) over(partition by [CLIENT_ID ] order by request_number) lead_request_number

                    from APPLICATION) a
                    on opened.[LOAN_ID ] = a.[ANKETA_ID ]
                ) clients

    ) retention
```

Рисунок А.7 – Сьома частина скрипту для побудови звітів для retention

```
                left join
                    (select a.[ANKETA_ID ] ,
                            pc.DISCOUNT_NAME,
                            pc.BONUS_NAME,
                            pc.DISCOUNT_PERCENT
                    from APPLICATION a
                left join
            PROMOCODE_USING pc on a.[ANKETA_ID ] = pc.loan_id
            ) pc on retention.[LOAN_ID ] = pc.[ANKETA_ID ]
```

Рисунок А.8 – Восьма частина скрипту для побудови звітів для retention