Міністерство освіти і науки України
Національний університет «Львівська політехніка»

Інститут прикладної математики та фундаментальних наук (ІМФН)
Кафедра прикладної математики

Курсова робота

з навчальної дисципліни
“Надвеликі бази даних”
Варіант №4: «Екскурсії»

Виконала:
студентка групи ПМ-42
Лопата Злата Олександрівна
Прийняв:
к. т. н. доцент
Любінський Богдан Богданович

Львів-2025

Анотація

Курсова робота з дисципліни “Надвеликі бази даних” присвячена побудові OLAP (Online Analytical Processing) кубу на основі принципів та технологій Microsoft SQL Server, зокрема SSAS (SQL Server Analysis Services), SSIS (SQL Server Integration Services) і SSRS (SQL Server Reporting Services). У роботі розглядається повний цикл створення OLAP кубу: від підготовки джерел даних та ETL-процесу з використанням SSIS, до моделювання багатовимірних даних у SSAS та створення звітів у SSRS. Висвітлено ключові етапи, такі як проектування багатовимірної моделі, визначення вимірів та ієрархій, реалізація агрегацій, а також інтеграція кубу зі звітними системами.

Abstract

This course project for the discipline “Very Large Databases” is dedicated to the development of an OLAP (Online Analytical Processing) cube based on the principles and technologies of Microsoft SQL Server, in particular SSAS (SQL Server Analysis Services), SSIS (SQL Server Integration Services), and SSRS (SQL Server Reporting Services). The paper covers the complete lifecycle of OLAP cube creation, starting from data source preparation and the ETL process using SSIS, through multidimensional data modeling in SSAS, and concluding with report generation in SSRS. Key stages are highlighted, including the design of the multidimensional model, definition of dimensions and hierarchies, implementation of aggregations, and integration of the OLAP cube with reporting systems.

Зміст

Вступ 4
Розділ 1. Аналіз предметної області 6
Опис предметної області 6
Основні бізнес-процеси 6
Функціональні вимоги до системи 7
Бізнес-правила та обмеження 8
Побудова ER-діаграми 8
Розділ 2. Проектування бази даних 12
Логічне проектування 12
Структура бази даних 12
Створення SQL-server та бази даних 13
Заповнення таблиць та генерація даних 17
Розділ 3. Реалізація ETL-процесів 25
Побудова Data Warehouse 25
Створення SSIS для реалізації ETL-процесу 27
Розділ 4. Побудова OLAP-куба та аналітичні звіти 35
Побудова OLAP-куба: 36
Побудова звітів 40
Висновки 46
Список використаних джерел 48
Додаток 49
SQL-скрипт для бази даних Excursions 49
SQL-скрипт для бази даних Excursions_DWH: 51

Вступ

Тема: Розробка та аналіз надвеликої бази даних з використанням технологій Microsoft SQL Server.

Мета курсової роботи:

Розробити повнофункціональну систему керування надвеликою базою даних для обраної предметної області з реалізацією ETL-процесів, побудовою багатовимірного куба та створенням аналітичних звітів з використанням технологій Microsoft SQL Server (SSIS, SSAS, SSRS

Варіант 4: "Екскурсії"

Основні сутності: Екскурсії, Автобуси, Шофери, Замовники, Рахунки, Платежі.

Специфічні вимоги:

- Мінімум 10 000 екскурсій
- Мінімум 500 автобусів
- Історія за 5 років

Обов'язкові звіти:

1. Список екскурсій за період з вартістю
2. Графік використання автобусів
3. Фінансовий звіт за місяць
4. Довідка замовника
5. Аналіз популярності маршрутів

Актуальність теми:

На сучасному етапі розвитку туристичної галузі спостерігається зростання попиту на організовані екскурсійні послуги. Туристичні компанії, що

займаються проведенням екскурсій, за свою довгу діяльність обслуговують велику кількість замовників, використовують значну кількість транспортних засобів, мають багато персоналу, а також здійснюють численні фінансові операції. У таких умовах ефективне управління даними стає дуже важким без використання сучасних інформаційних систем і баз даних.

Тому розробка баз даних необхідна для екскурсійних організацій, щоб автоматизувати процеси планування екскурсій та роботу шоферів, ведення фінансового обліку та документації, адже за допомогою OLAP-куба можна створити для аналітики звіти. Важливим для якісного управління компаніями є можливість аналізу популярності маршрутів та перегляд фінансових показників за вибрані періоди.

Об’єкт дослідження: діяльність екскурсійної компанії.

Предмет дослідження: інформаційні процеси та база даних екскурсійної діяльності.

Розділ 1. Аналіз предметної області

Опис предметної області

Предметною областю даної курсової роботи є діяльність екскурсійної компанії, що спеціалізується на організації та проведенні автобусних екскурсій для фізичних та юридичних осіб. Компанія надає послуги з планування маршрутів, підбору транспортних засобів і водіїв, оформлення замовлень, виставлення рахунків та прийому платежів.

Екскурсії проводяться за заздалегідь визначеними маршрутами, мають встановлену дату, тривалість та вартість. Для проведення кожної екскурсії може використовуватися один автобус і призначається відповідальний шофер. Замовниками можуть виступати як приватні особи, так і організації (школи, туристичні агентства, підприємства).

Система повинна забезпечувати збереження історичних даних про проведені екскурсії, використання автобусів і роботу шоферів за період не менше п’яти років, а також підтримувати обробку великого обсягу інформації (понад 10 000 екскурсій).

Основні бізнес-процеси

1. **Планування екскурсій:**
   - створення нових екскурсій;
   - визначення маршруту, дати, тривалості та вартості;
   - автобус та шофер;
2. **Управління транспортними засобами:**
   - облік автобусів;
   - контроль завантаженості та використання тз;
   - аналіз графіку використання автобусів;
3. **Управління персоналом (шоферами):**

- збереження інформації про водіїв;
- призначення шоферів на екскурсії;

4. **Робота із замовниками:**
   - оформлення замовлень на екскурсії;
   - реєстрація замовників;
   - формування довідки замовника;
5. **Фінансові операції:**
   - формування рахунків на оплату;
   - реєстрація платежів;
   - контроль заборгованостей;
6. **Аналітика та звітність:**
   - формування звітів за певний період;
   - фінансовий звіт за місяць;
   - аналіз популярності маршрутів.

Функціональні вимоги до системи

- Інформаційна система екскурсій повинна забезпечувати:
- зберігання даних про екскурсії, автобуси, шоферів, замовників, рахунки та платежі;
- обробку не менше 10 000 записів про екскурсії та не менше 500 автобусів;
- ведення історії за п’ять років;
- додавання, редагування та видалення даних;
- пошук екскурсій за датою, маршрутом та замовником;
- графік використання автобусів;

- формування звітів;
- отримання довідки по конкретному замовнику;
- аналіз завантаженості автобусів та популярності маршрутів.

Бізнес-правила та обмеження

1. Одна екскурсія проводиться з використанням одного автобуса та одного шофера**.**
2. Один автобус не може бути призначений на дві екскурсії в один і той самий час.
3. Шофер не може одночасно брати участь у двох екскурсіях.
4. Кожна екскурсія має фіксовану вартість**.**
5. Для кожної екскурсії формується один рахунок.
6. Рахунок може бути оплачений одним або кількома платежами.
7. Сума платежів не перевищує суму рахунку.
8. Видалення екскурсії можливе лише за відсутності прив’язаних платежів.
9. Усі фінансові операції повинні зберігатися в системі для подальшого аналізу.

Побудова ER-діаграми

ER-діаграма має такі основні сутності: Екскурсії, Автобуси, Шофери, Замовники, Рахунки, Платежі. Розглянемо кожну сутність окремо.

Атрибути сутності «Екскурсії»

- excursion_id — унікальний ідентифікатор екскурсії;
- route_name — назва маршруту;
- start_date — дата проведення екскурсії;
- duration_hours — тривалість екскурсії в годинах;
- price — вартість екскурсії;

- bus_id — ідентифікатор автобуса;
- driver_id — ідентифікатор шофера;
- customer_id — ідентифікатор замовника.

Атрибути сутності «Автобуси»

- bus_id — унікальний ідентифікатор автобуса;
- model — модель автобуса;
- plate_number — номер автобуса;
- capacity — кількість місць;
- year — рік випуску.

Атрибути сутності «Шофери»

- driver_id — унікальний ідентифікатор шофера;
- full_name — ПІБ шофера;
- experience_years — стаж роботи;
- phone — контактний номер телефону.

Атрибути сутності «Замовники»

- customer_id — унікальний ідентифікатор замовника;
- name — ПІБ або назва організації;
- type — тип замовника (фізична або юридична особа);
- phone — контактний номер;
- email — електронна пошта.

Атрибути сутності «Рахунки»

- invoice_id — унікальний ідентифікатор рахунку;
- excursion_id — ідентифікатор екскурсії;
- invoice_date — дата виставлення рахунку;

- total_amount — загальна сума;
- status — статус оплати.

Атрибути сутності «Платежі»

- payment_id — унікальний ідентифікатор платежу;
- invoice_id — ідентифікатор рахунку;
- payment_date — дата платежу;
- amount — сума платежу.

Зв’язки між сутностями

Сутність 1	Сутність 2	Кардина-льність	Опис
Автобуси	Екскурсії	1 : N	Один автобус може використовуватися для багатьох екскурсій, але кожна екскурсія використовує лише один автобус
Шофери	Екскурсії	1 : N	Кожна екскурсія має одного шофера, але один шофер може вести багато екскурсій.
Замовники	Екскурсії	1 : N	Кожна екскурсія оформлюється одним замовником, але замовник може замовити багато екскурсій.
Екскурсії	Рахунки	1 : 1	Для кожної екскурсії формується один рахунок.

Рахунки	Платежі	1 : N	Один рахунок може бути оплачений кількома платежами.

ER-діаграма

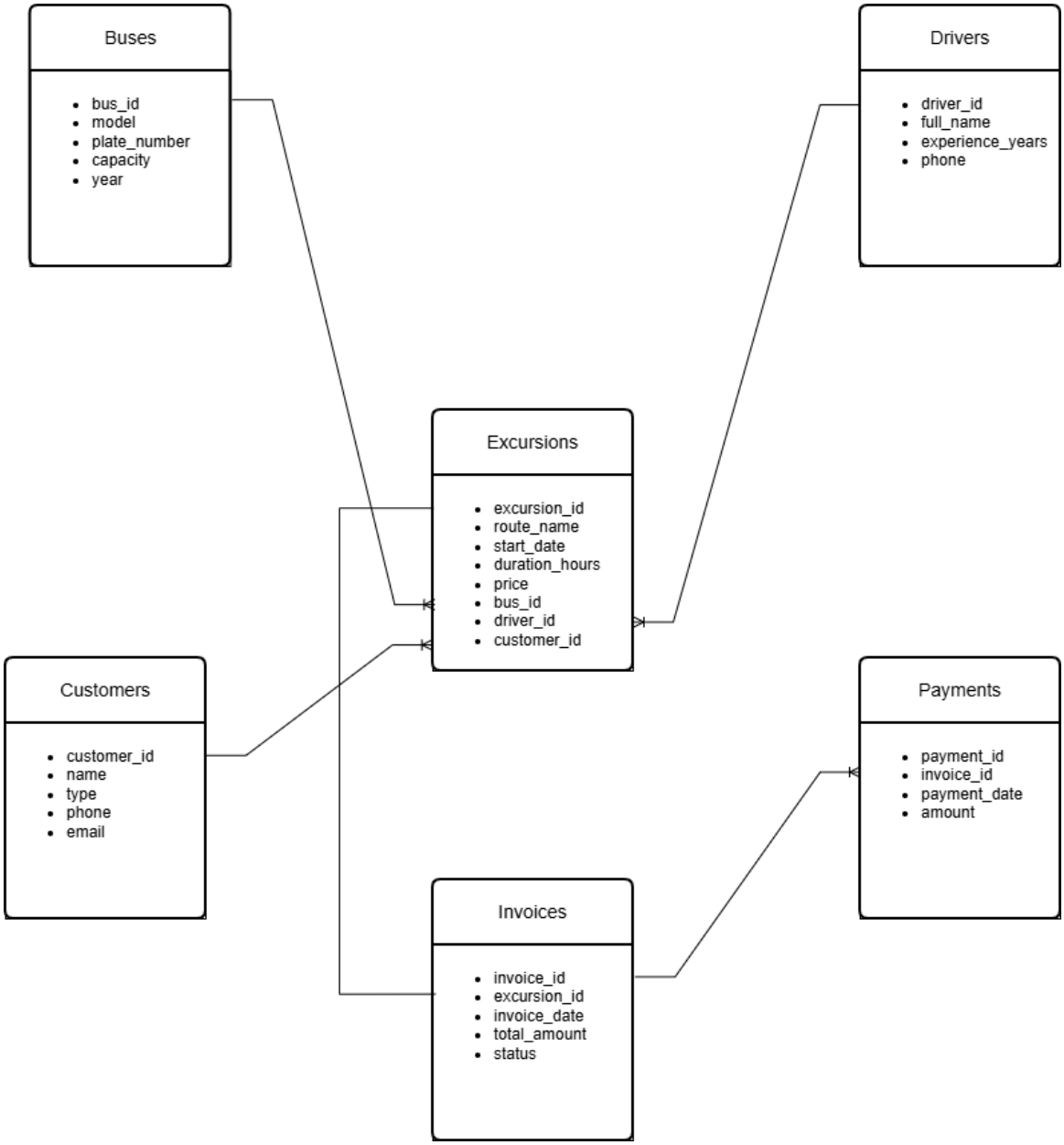


рис.1 ER-діаграма

Розділ 2. Проектування бази даних

Логічне проектування

Логічне проєктування бази даних (рис.1) виконано з використанням принципів нормалізації з метою усунення надлишковості даних та забезпечення цілісності інформації. База даних приведена щонайменше до третьої нормальної форми (3НФ).

Перша нормальна форма (1НФ): Усі відношення мають атомарні атрибути. Повторювані групи відсутні, кожне поле містить лише одне значення. Для кожної таблиці визначено первинний ключ.

Друга нормальна форма (2НФ): Усі неключові атрибути повністю функціонально залежать від первинного ключа. Оскільки у таблицях використовуються прості (не складені) первинні ключі, часткові залежності відсутні.

Третя нормальна форма (3НФ): Усі неключові атрибути залежать лише від первинного ключа і не мають транзитивних залежностей. Зокрема, інформація про замовника зберігається окремо від рахунків і екскурсій, що виключає дублювання даних та порушення цілісності.

Структура бази даних

База даних для предметної області «Екскурсії» буде складатися з 6 таблиць:

Таблиця Excursions містить основні дані про екскурсії, зокрема маршрут, дату проведення, тривалість в годинах та вартість. Також у таблиці зберігаються зовнішні ключі, що забезпечують зв’язок з таблицями Buses, Drivers та Customers. Дана таблиця є центральною в базі даних, оскільки саме з нею пов’язані фінансові операції та аналітичні звіти.

Таблиця Buses містить інформацію про кожний автобус. Вона містить дані про модель автобуса, номер, місткість та рік випуску.

Таблиця Drivers містить інформацію про шоферів, зокрема ПІБ, стаж роботи та номер телефону. Зв’язок між таблицями Drivers та Excursions дозволяє контролювати призначення водіїв на екскурсії та запобігати одночасній участі шофера у декількох екскурсіях.

Таблиця Customers містить дані замовників екскурсій, як фізичних так і юридичних. Містить їх контактні дані (номер та електрона пошта) та використовується для формування довідок і фінансових звітів за конкретними клієнтами.

Таблиця Invoices призначена для обліку рахунків, які формуються для кожної екскурсії. У таблиці зберігається дата виставлення рахунку, загальна сума та статус оплати. Між таблицями Excursions і Invoices реалізовано зв’язок типу «один до одного», що відповідає бізнес-правилу формування одного рахунку на одну екскурсію.

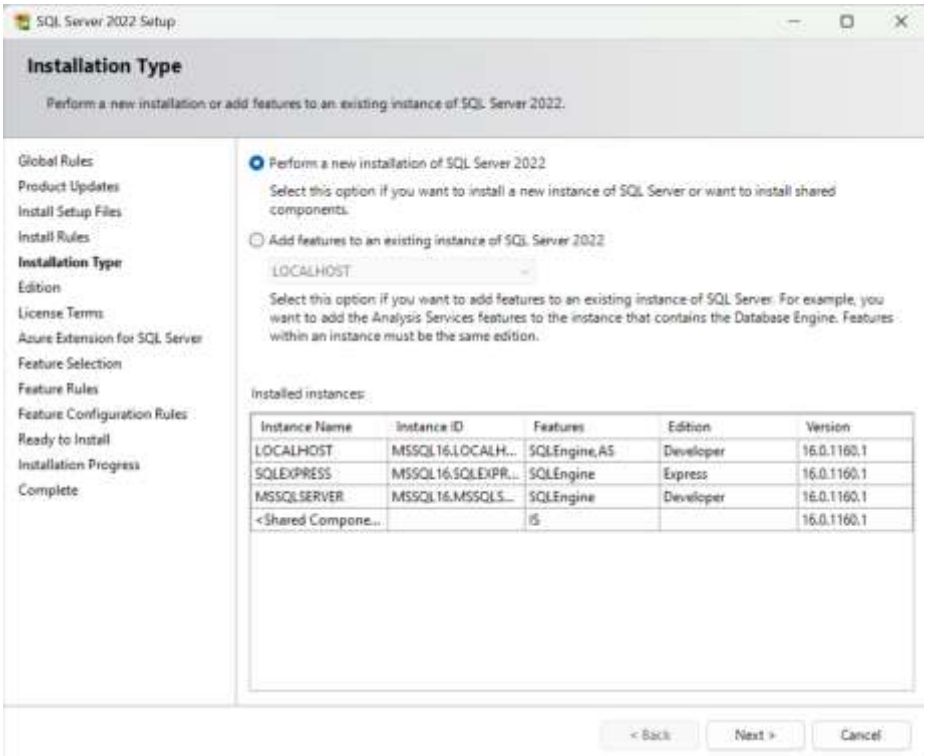
Таблиця Payments містить інформацію про здійснені платежі за рахунками. Один рахунок може бути оплачений кількома платежами, що реалізує зв’язок типу «один до багатьох» між таблицями Invoices та Payments. Дані з цієї таблиці використовуються для контролю заборгованостей та формування фінансових звітів.

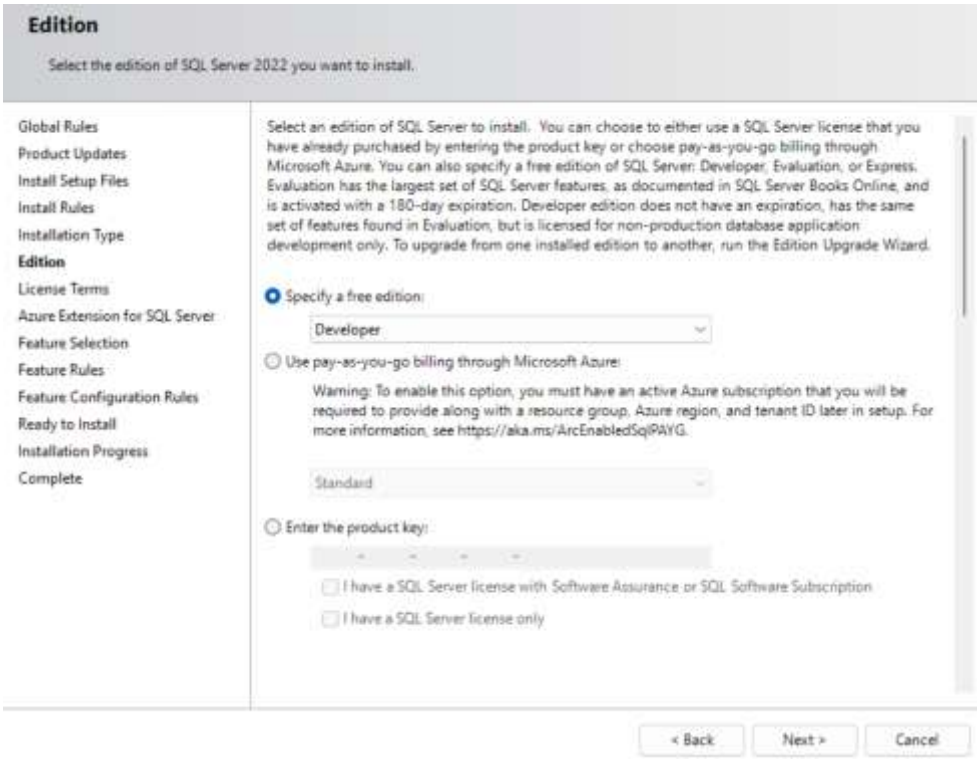
Створення SQL-server та бази даних

Завантажуємо програму SQL Server 2022 Setup, щоб створити новий сервер. Спочатку треба вибрати New SQL Server standalone installation:

New SQL Server standalone installation or add features to an existing installation

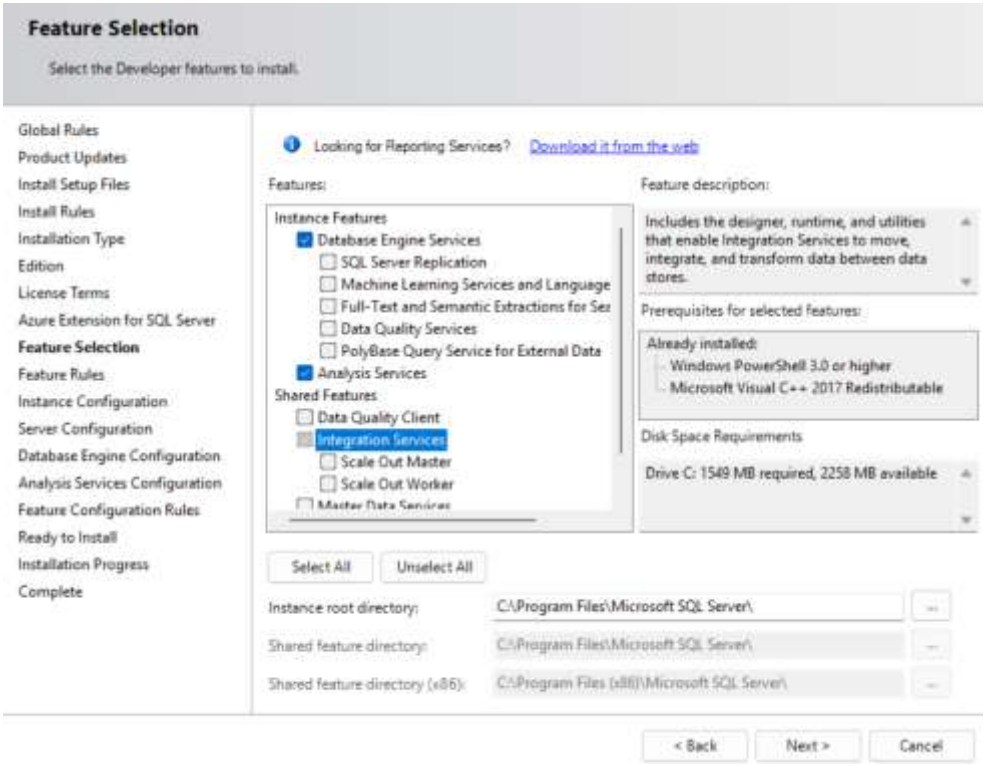
Launch a wizard to install SQL Server 2022 in a non-clustered environment or to add features to an existing SQL Server 2022 instance.



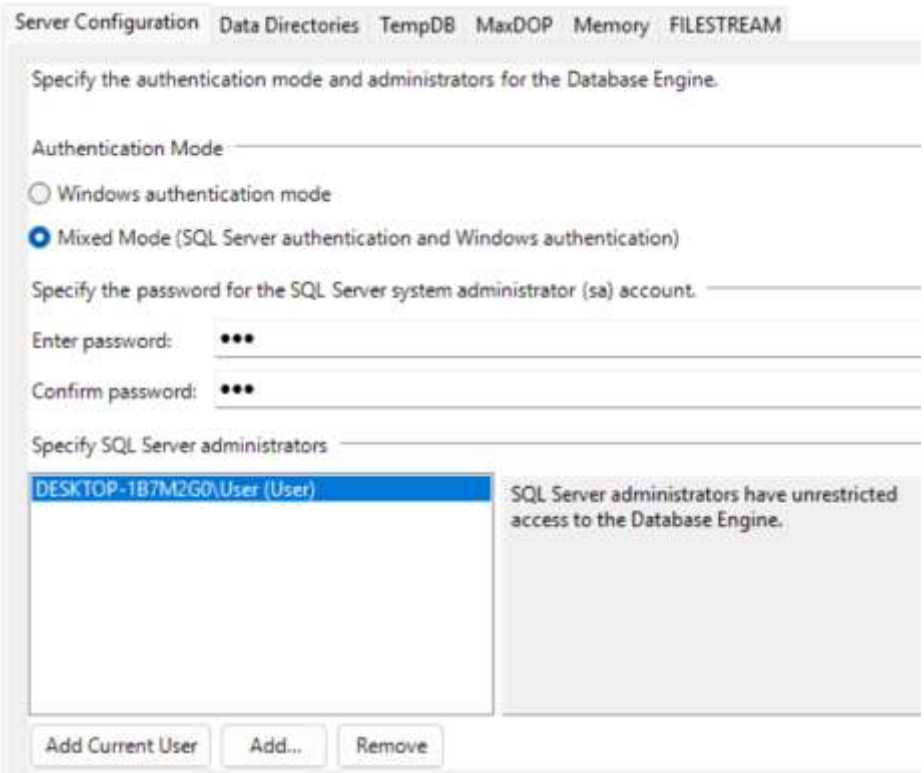


Вибираємо Developer. Далі потрібно обрати три Features:

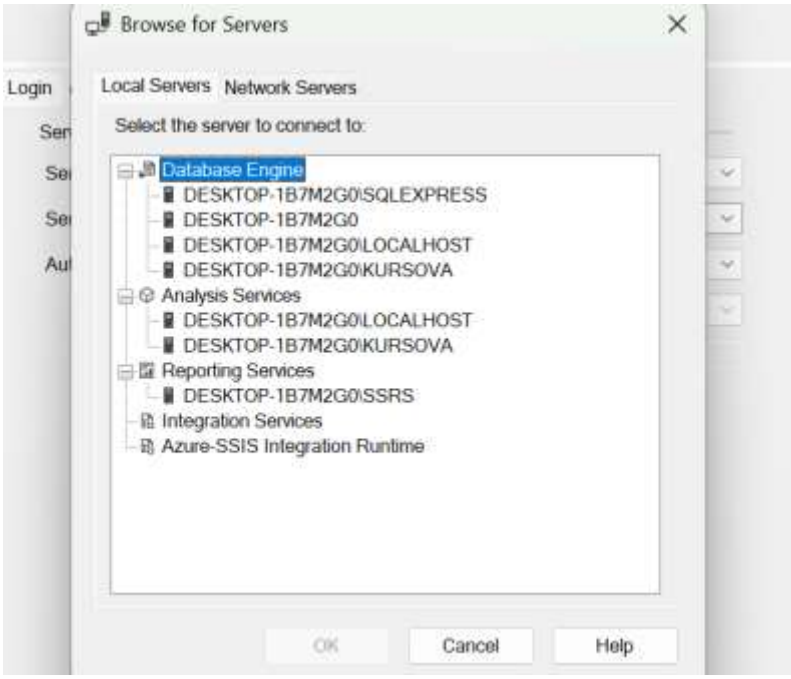
- Database Engine Services
- Analysis Services
- Integration Services



Даю назву серверу KURSOVA та додаю користувача sa з паролем 123. Та інсталюю.

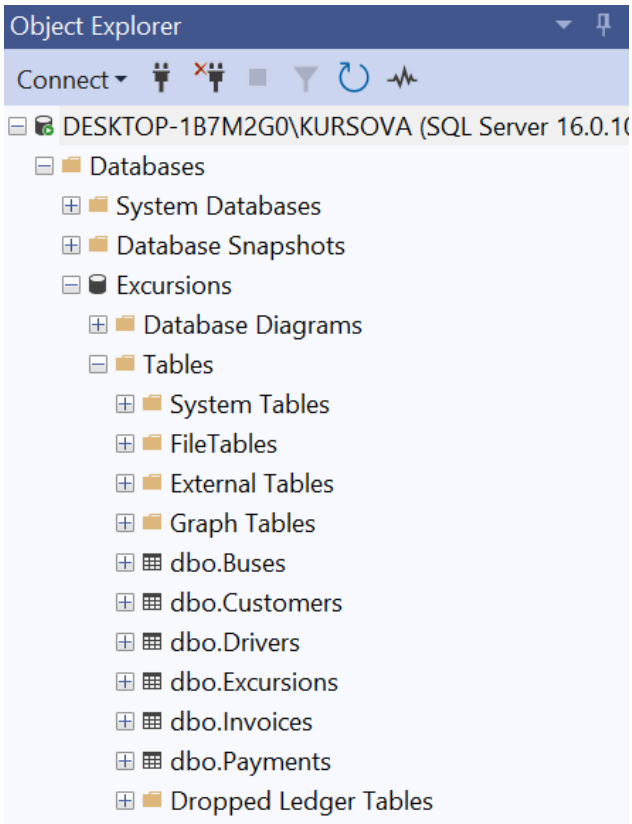


Дивимось список серверів в Microsoft SQL Server Management Studio:



Сервери KURSOVA з’явились.

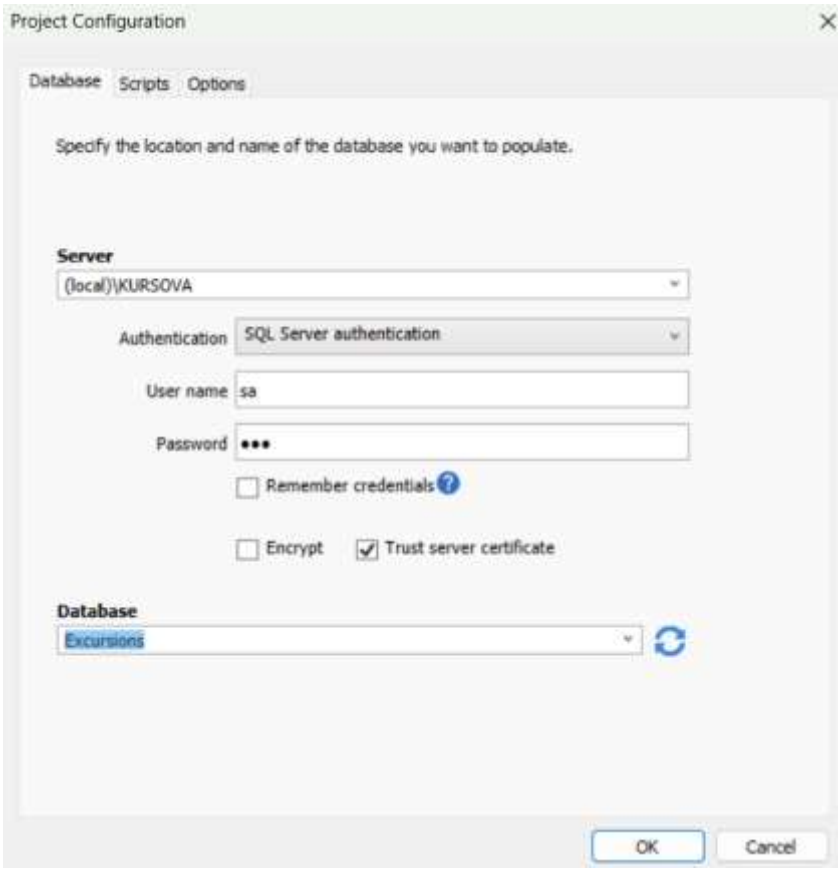
База даних «Excursions» створена в середовищі Microsoft SQL Server Management Studio за допомогою SQL-запиту CREATE DATABASE і створено відповідні таблиці за допомогою SQL-скриптів, які наведені в додатку.



Заповнення таблиць та генерація даних

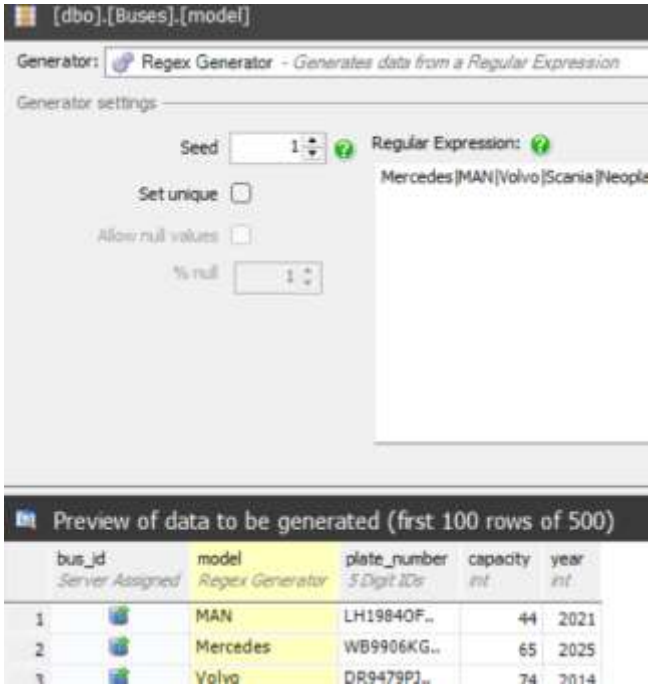
Для наповнення бази даних тестовими даними було використано програмний засіб Redgate SQL Data Generator, який дозволяє автоматично створювати великі обсяги даних з урахуванням структури бази даних, зовнішніх ключів та обмежень цілісності.

Підключилась до сервера KURSOVA та бази даних Excursions:

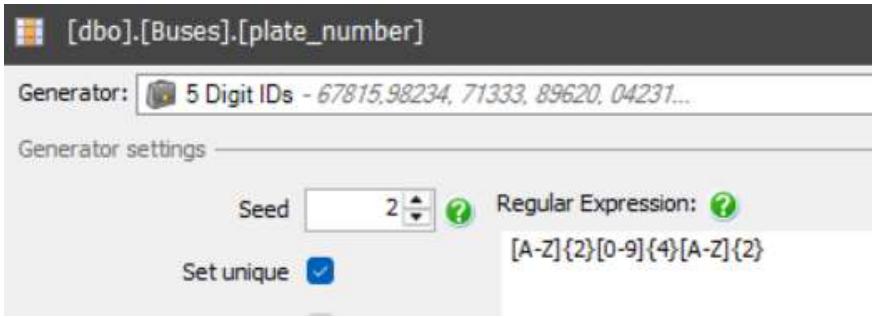


В таблиці Buses згенеруємо 500 рядочків:

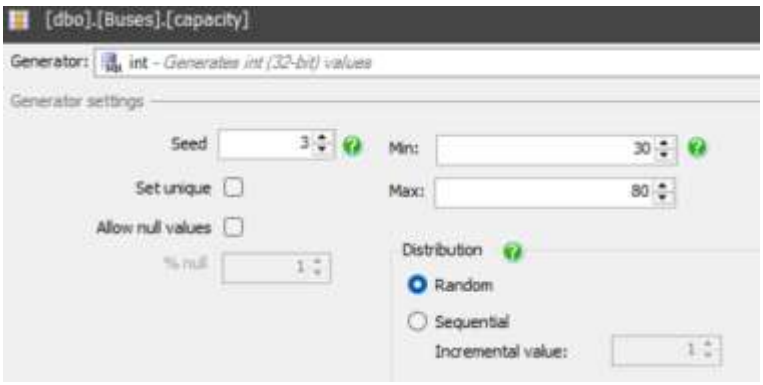
Для моделі автобуса вписала варіанти моделі і рандомно вони прописались, знизу бачимо згенеровані дані:



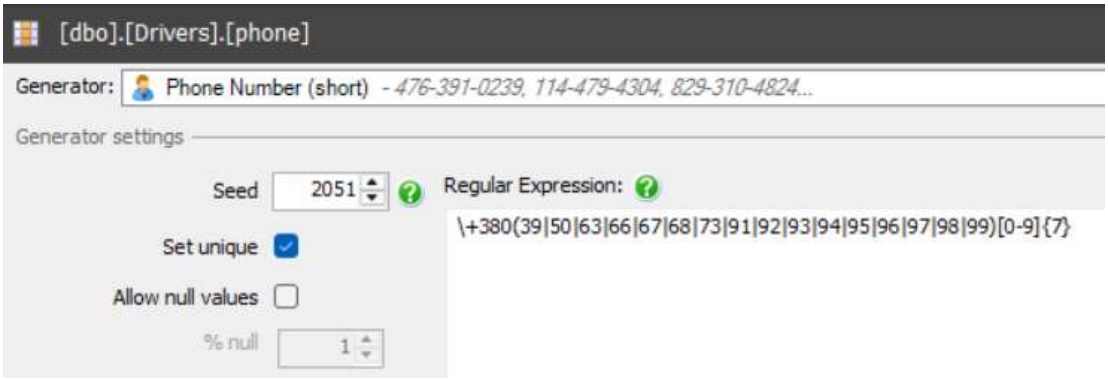
Для коректного виводу державного номера автобуса в Regular Expression напишемо:



Кількість місць в автобусі обмежемо від 30 до 80, а роки його випуску з 2000 по 2025:



Генеруємо номери телефонів, щоб розпочинались з актуальних номерів операторів (так само буде і в таблиці Customers):

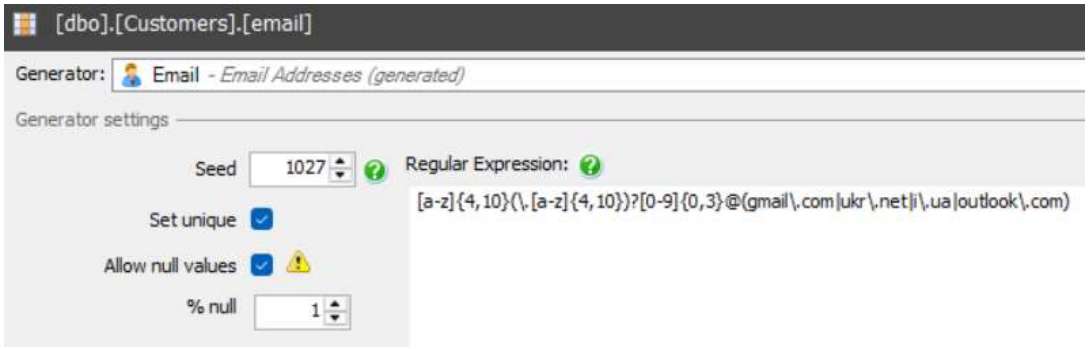


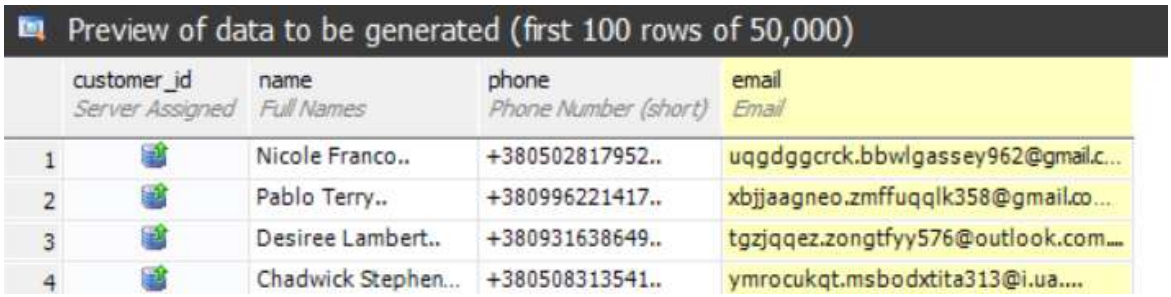
Отримали результати в таблиці Driver:

Preview of data to be generated (first 100 rows of 400)

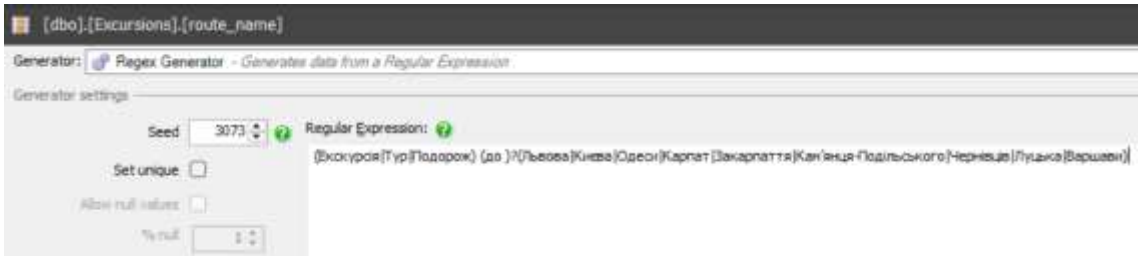
	driver_id *Server Assigned*	full_name *Full Names*	experience_years *int*	phone *Phone Number (short)*
1		Ginger Allen	18	+380926226538
2		Erika Waller	9	+380507649638
3		Tania Herrera	18	+380736340055
4		Frank Rojas	4	+380674226499
5		Norma Bean	20	+380979209868

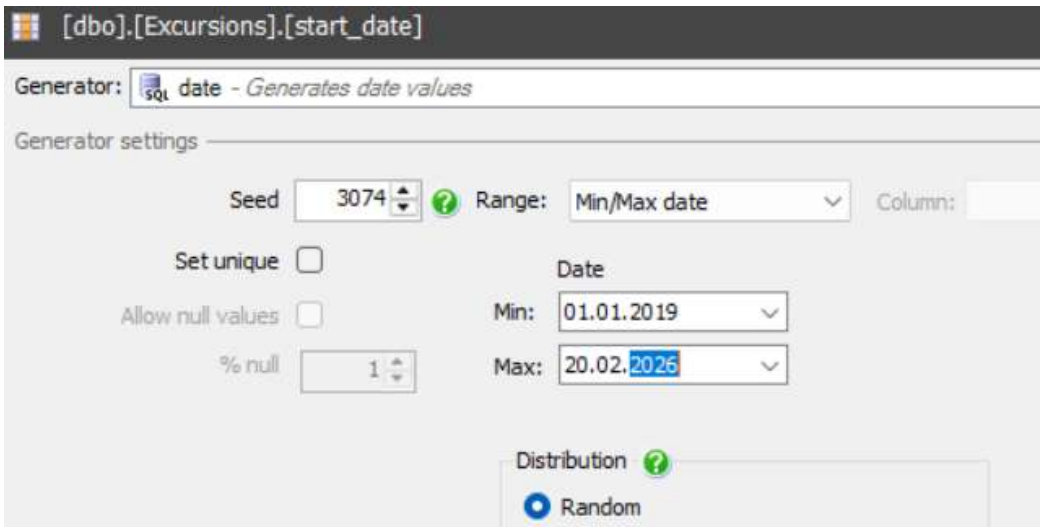
Прописуємо Regular Expression для email, щоб розширення були існуючі:



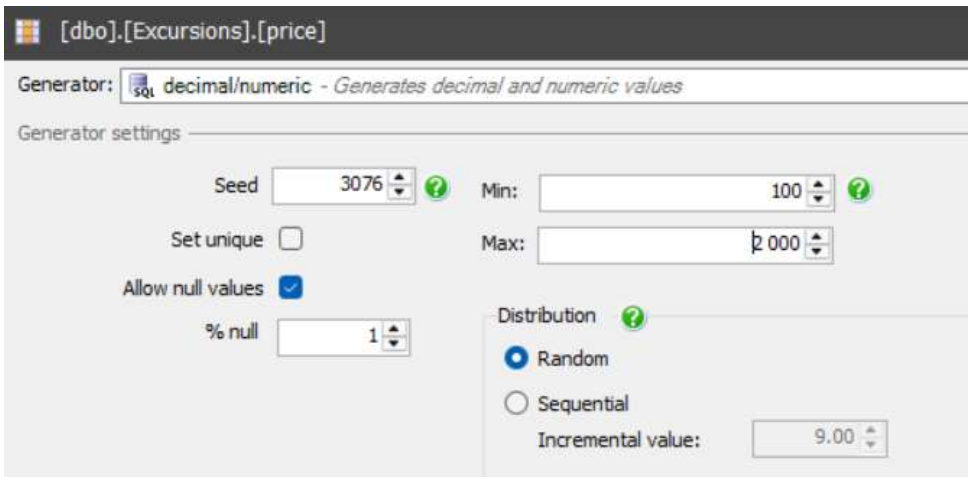


В таблиці Excursions прописую варіанти назв екскурсій так щоб вони могли мінятися і дати екскурсій будуть існувати з 01.01.2019 і заплановані вже до 20.02.2026:

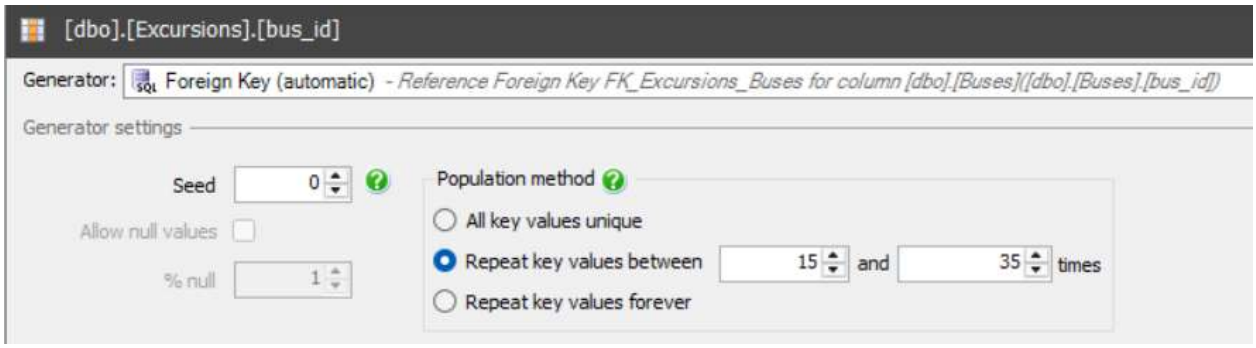


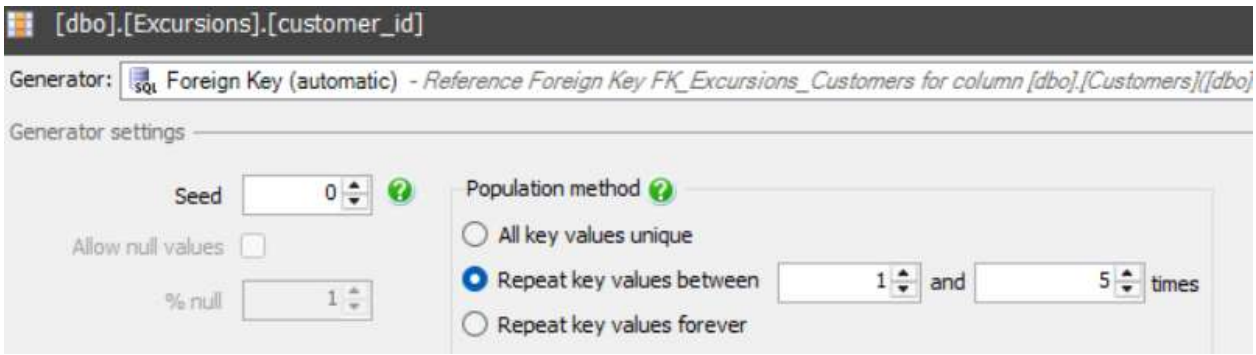


Ціни на екскурсії будуть від 100 до 2000 грн:

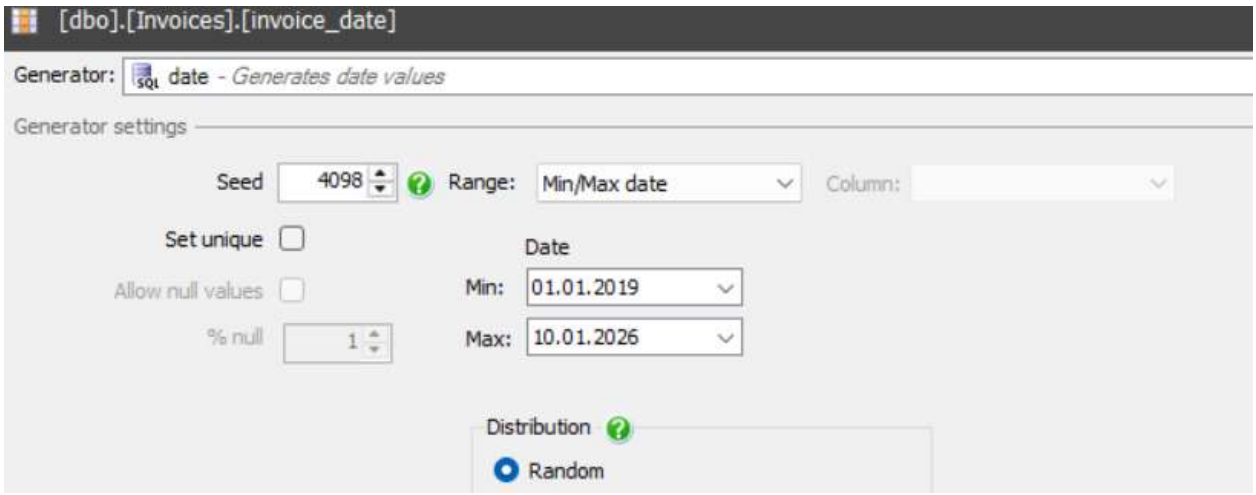


Кожний автобус може зустрічатися від 15 до 35 разів, і замовник від 1 разу до 5 ходив на екскурсії організовані екскурсійною організацією:





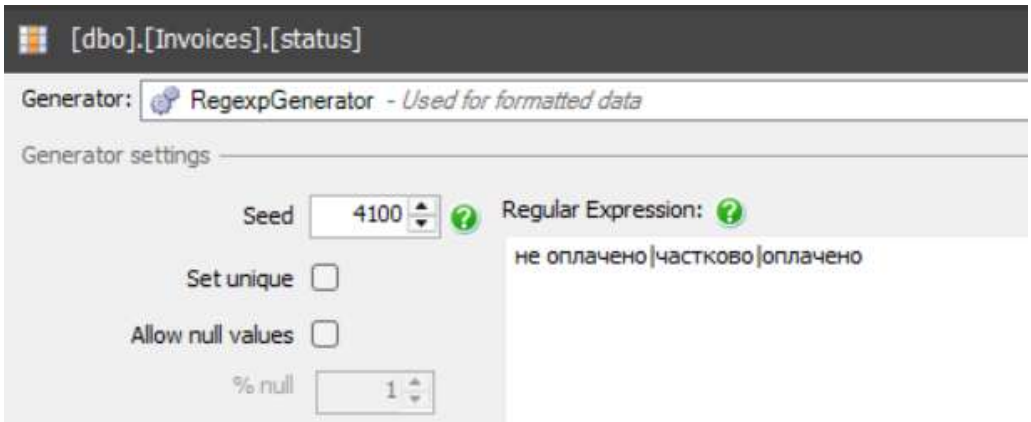
Для таблиці Invoices використано унікальні значення зовнішнього ключа excursion_id, що відповідає моделі «один рахунок на одну екскурсію» та запобігає дублюванню фінансових документів.



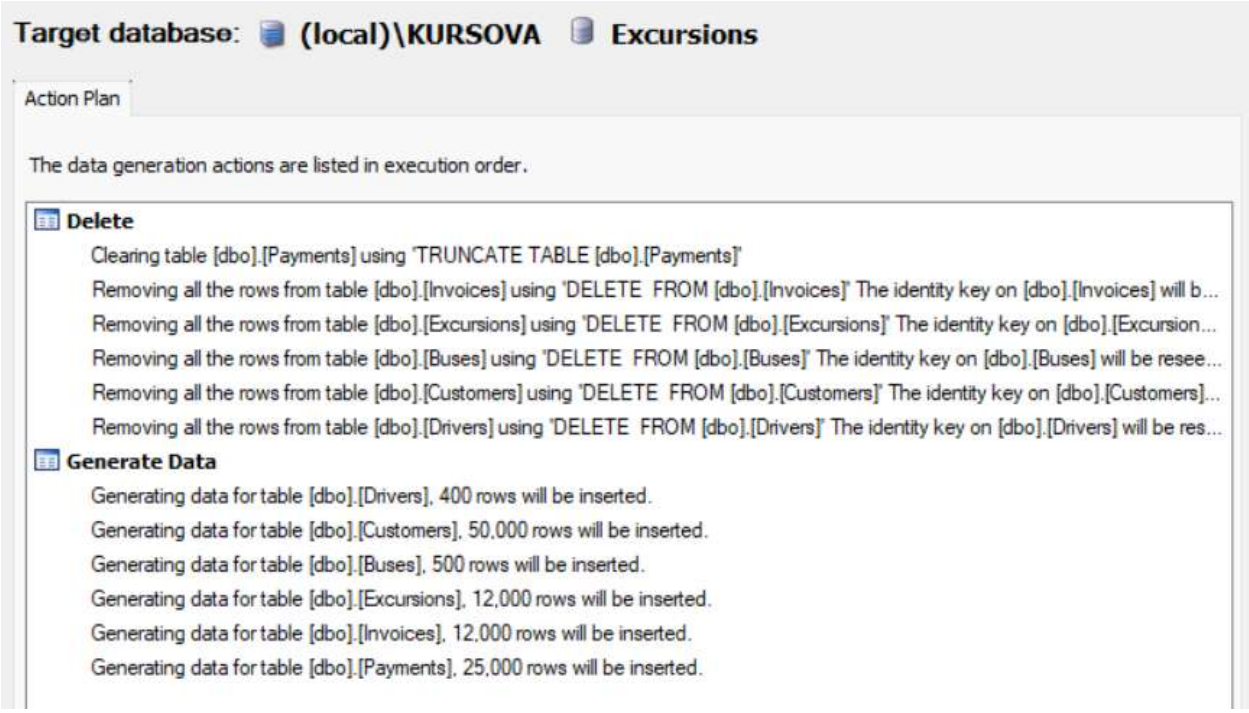
Поле total_amount формується на основі агрегованих даних таблиці Payments після генерації тестових даних, оскільки інструмент генерації не підтримує міжтабличні агрегації. Тому поки так, а потім в sql - запит зробимо:

```
UPDATE Invoices
SET total_amount = p.sum_amount
FROM Invoices i
JOIN (
   SELECT invoice_id, SUM(amount) AS sum_amount
   FROM Payments
   GROUP BY invoice_id
) p ON i.invoice_id = p.invoice_id;
```

Для таблиці Invoices генеруємо статуси (неоплачені, оплачено чи частково):



Отже, за допомогою програми Redgate SQL Data Generator в кожній таблиці нагенеровано логічну кількість даних для екскурсійної компанії:



Тепер перевіряємо записи:

```
1  SELECT COUNT(*)
2  FROM [Excursions].[dbo].[Buses]
```

89 % No issues found

Results Messages

	(No column name)
1	500

```
1  SELECT COUNT(*)
2  FROM [Excursions].[dbo].[Customers]
```

89 % No issues found

Results Messages

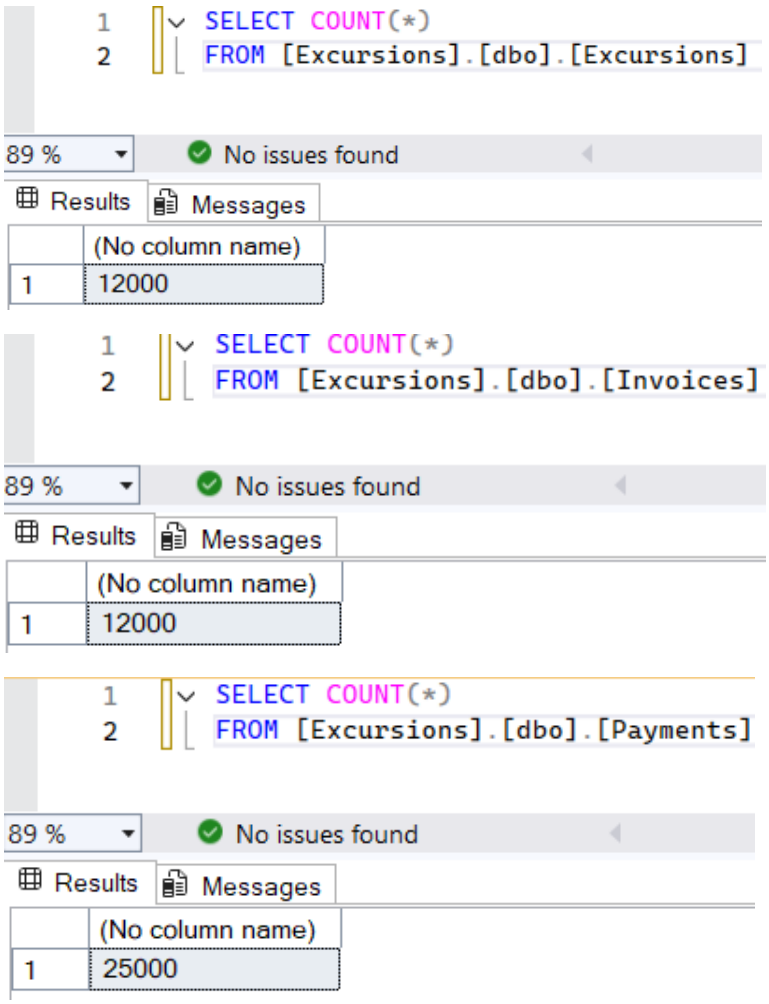
	(No column name)
1	50000

```
1  SELECT COUNT(*)
2  FROM [Excursions].[dbo].[Drivers]
```

89 % No issues found

Results Messages

	(No column name)
1	400



Все зійшлось по кількості. Дані записані у базу даних.

Після генерації тестових даних за допомогою SQL Data Generator було виконано додатковий SQL-запит для коректного обчислення загальної суми рахунків на основі даних таблиці платежів.

```
UPDATE i
SET i.total_amount = p.sum_amount
FROM Invoices i
JOIN (
    SELECT invoice_id, SUM(amount) AS sum_amount
    FROM Payments
    GROUP BY invoice_id
) p ON i.invoice_id = p.invoice_id;
```

Розділ 3. Реалізація ETL-процесів

Побудова Data Warehouse

Для забезпечення ефективного аналітичного опрацювання великих обсягів інформації та формування OLAP-звітів на основі операційної бази даних було спроєктовано та реалізовано сховище даних (Data Warehouse, DWH).

Операційна база даних Excursions, створена у попередніх розділах, належить до класу OLTP-систем і призначена для щоденного обліку екскурсій, клієнтів, транспортних засобів і фінансових операцій. Однак така структура не є оптимальною для виконання складних аналітичних запитів, агрегацій та побудови звітів за великий період часу.

З цією метою було розроблено сховище даних з використанням багатовимірної моделі та зіркоподібної схеми (Star Schema), яка є стандартом для OLAP-систем. Основною перевагою такої схеми є простота побудови аналітичних запитів, висока продуктивність та зручність інтеграції з SQL Server Analysis Services.

Сховище даних складається з 2 таблиць фактів, що містять числові показники для аналізу, та 5 вимірних таблиць, які описують контекст цих показників.

Фактові таблиці

1) Таблиця FactExcursion містить агреговані дані про проведені екскурсії та використовується для аналізу операційної діяльності компанії. Вона містить: зовнішні ключи на виміри екскурсій, дати, автобуса, шофера та замовника; тривалість та вартість екскурсії та кількість учасників. Дані цієї таблиці використовуються для аналізу завантаженості автобусів, роботи шоферів та популярності маршрутів.
2) Таблиця FactPayment використовується для формування фінансових звітів, контролю оплат та аналізу заборгованостей. В собі містить такі атрибути, як дату екскурсії, ідентифікатор екскурсії, замовника та платежу, суму платежу та їх кількість, залишок заборгованостей.

Виміри

DimDate містить календарні дані та забезпечує аналіз у часовому розрізі. Його атрибути: рік, квартал, місяць та день.

DimExcursion описує характеристики екскурсій. Його атрибути: назва екскурсії.

Buses містить інформацію про автобуси. Його атрибути: модель, номер та рік автобуса, а також кількість місць в ньому.

Drivers містить дані про шоферів. Його атрибути: ім'я та прізвище водія англійською мовою, стаж роботи та номер телефону.

Customers описує замовників екскурсій. Його атрибути: ім'я та прізвище замовника англійською мовою, тип (фізична чи юридична особа) , номер телефону та електронна пошта (не обов'язково).

В програмі SQL Server Management Studio створила Excursions_DWH (скрипт наведено в додатку).

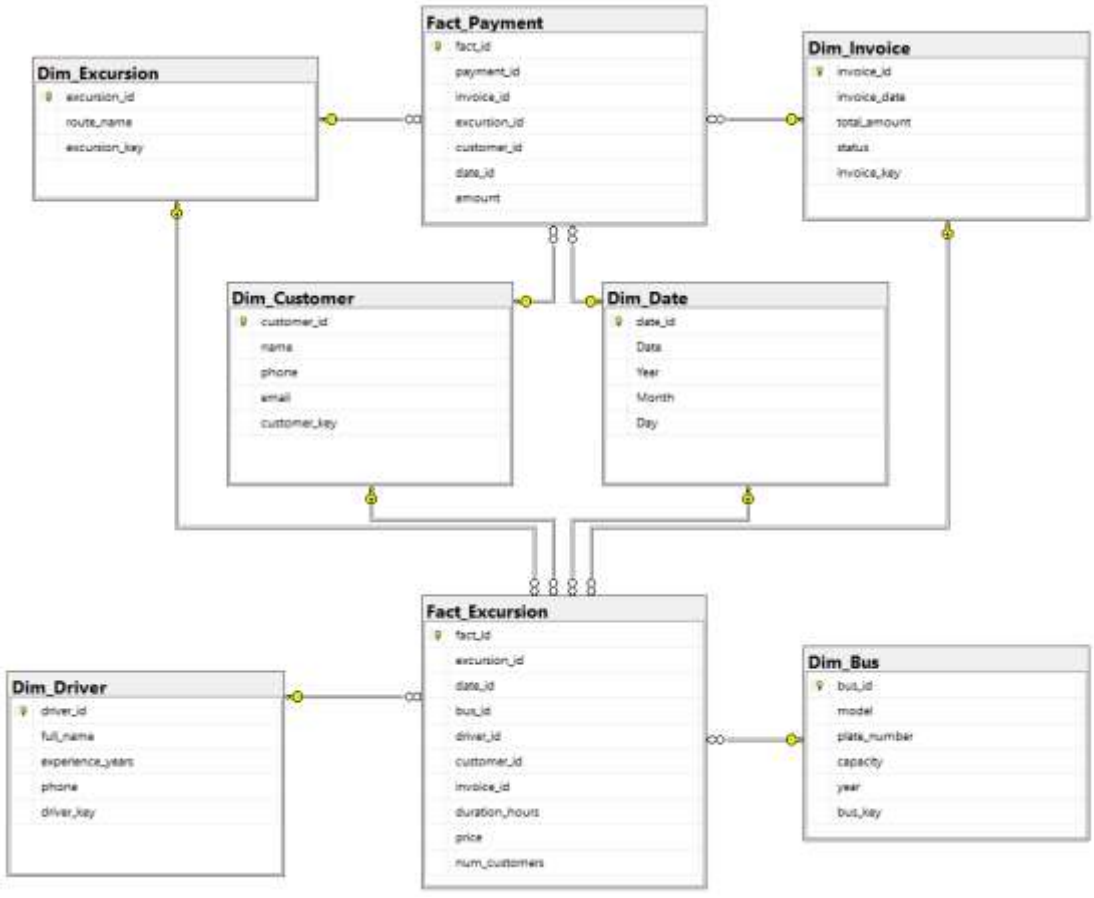


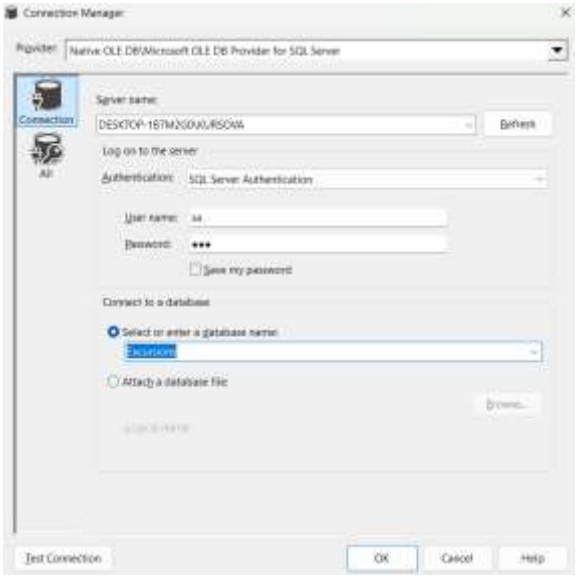
Рис.2 Структура Data Warehouse

Створення SSIS для реалізації ETL-процесу

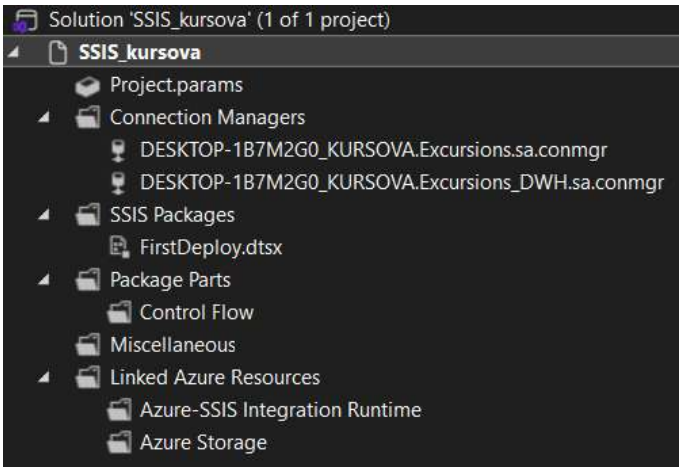
В Visual Studio встановлюємо SQL Server Integration Services Projects:

Створивши проект Integration Services Project налаштовуємо ETL процес:

Налаштовуємо з'єднання з бд Excursions та те саме проробляємо з Excursion_DWH:



Отримали таку структуру нашого проекту в Object Explorer:



Налаштовуємо потоки (ETL - процес)

Тепер для кожного виміру, який є у сховищі даних створюємо блок DataFlow. Кожному блоку даємо відповідну назву та проводимо зв'язки. Додаємо

перший блок для Dim_Excursion. Всередині блока встановлюємо OLE DB Source та OLE DB Destination, даємо відповідні назви та проводимо між ними зв'язок. В OLE DB Source вставляємо SQL-запит:

```
SELECT
    excursion_id,
    route_name
FROM Excursions
```

У OLE DB Destination встановлюємо наступні зв'язки та зберігаємо:

Input Column	Destination Column
<ignore>	excursion_id
route_name	route_name
excursion_id	excursion_key

Тепер зробимо DataFlow для Dim_Bus. У OLE DB Source вставляємо наступний SQL-запит:

```
SELECT
    bus_id,
    model,
    plate_number,
    capacity,
    year
FROM Buses
```

У OLE DB Destination встановлюємо наступні зв'язки та зберігаємо:

Input Column	Destination Column
<ignore>	bus_id
model	model
plate_number	plate_number
capacity	capacity
year	year
bus_id	bus_key

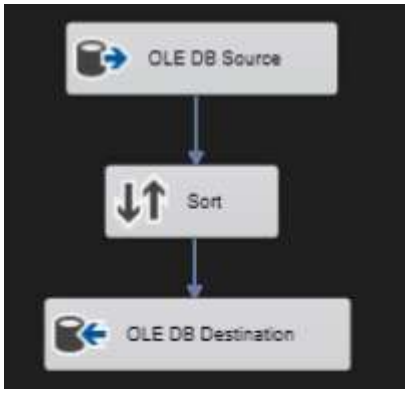
У OLE DB Destination в Dim_Driver, Dim_Customer та Dim_Invoice встановлюємо наступні зв'язки та зберігаємо:

Input Column	Destination Column
<ignore>	driver_id
full_name	full_name
experience_years	experience_years
phone	phone
driver_id	driver_key

Input Column	Destination Column
<ignore>	customer_id
name	name
phone	phone
email	email
customer_id	customer_key

Input Column	Destination Column
<ignore>	invoice_id
invoice_date	invoice_date
total_amount	total_amount
status	status
invoice_id	invoice_key

В DataFlow в Dim_Date налаштовуємо такий порядок та в OLE DB Source вставляємо запит та зв’язки:



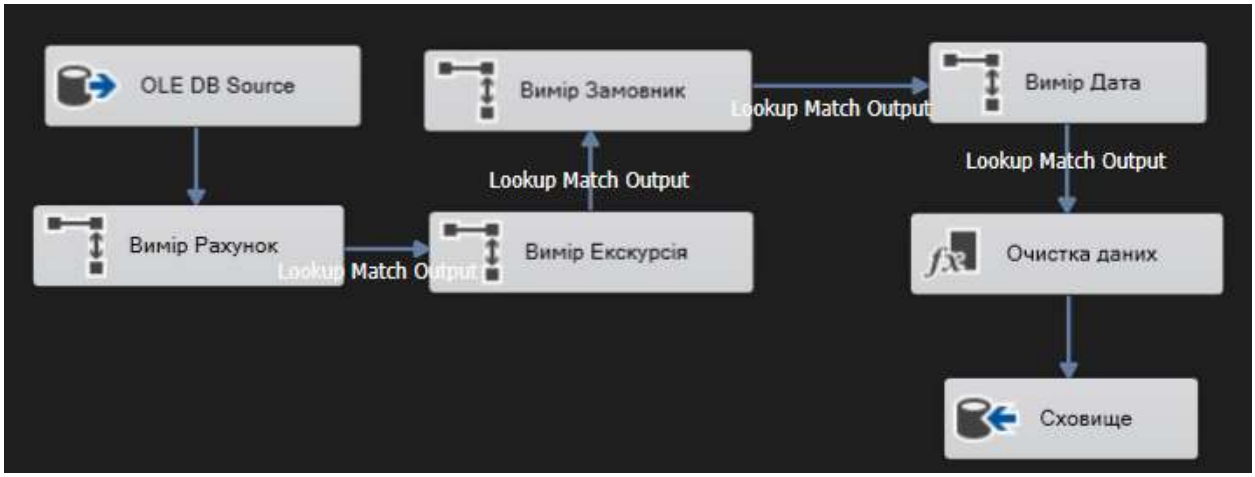

```
(
	Select
		start_date As [Date],
		YEAR(start_date) As [Year],
		MONTH(start_date) As [Month],
		DAY(start_date) As [Day]
	From Excursions
) Union (
	Select
		invoice_date As [Date],
		YEAR(invoice_date) As [Year],
		MONTH(invoice_date) As [Month],
		DAY(invoice_date) As [Day]
	From Invoices
) Union (
	Select
		payment_date As [Date],
		YEAR(payment_date) As [Year],
		MONTH(payment_date) As [Month],
		DAY(payment_date) As [Day]
	From Payments
)
```

Input Column	Destination Column
<ignore>	date_id
Date	Date
Year	Year
Month	Month
Day	Day

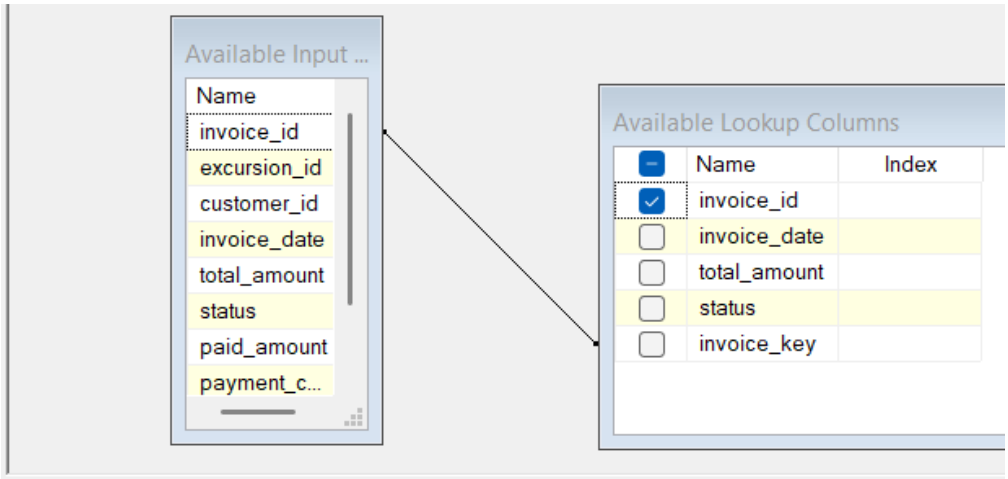
Переходимо до створення фактів

Ми закінчили розробку ETL процесу для вимірів. Тепер потрібно зробити для таблиць фактів. Вставляємо DataFlow блок та в середині блока встановлюємо джерело даних OLE DB Source, далі для усіх вимірів з якими зв'язана таблиця фактів ми вставляємо Lookup блоки, для очищення даних вставляємо блок Derived Column та OLE DB Destination щоб загрузати дані у фактів.

Для таблиці Fact_Playment створюємо наступну структуру:

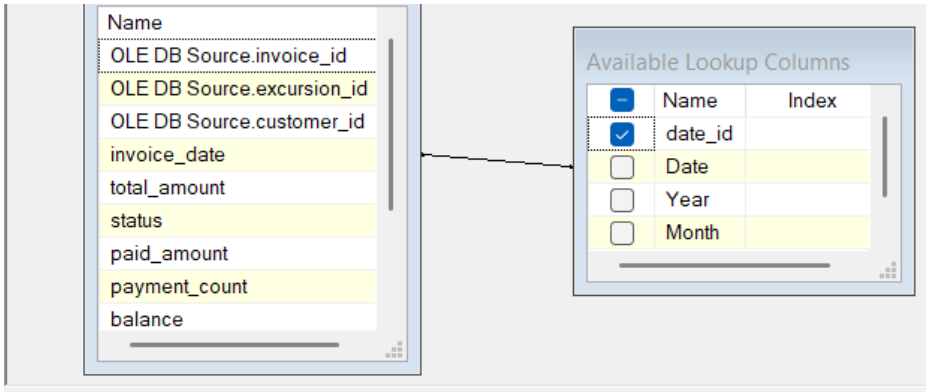


У Data Source вставляємо запит, який подається у додатках. У кожному блоці Lookup ставимо галочку для поля з закінченням ..._id у другій таблиці, а від поля з закінченням …_id у першій таблиці проводимо зв'язок то поля з закінченням …_key у другій таблиці.



Lookup Column	Lookup Operation	Output Alias
invoice_id	<add as new column>	invoice_id

Таке робимо для кожного Lookup, а для Виміру Дата встановлюємо зв'язок між invoice_date та Date. У блоці Derived Column зробимо наступне очищення:



Lookup Column	Lookup Operation	Output Alias
date_id	<add as new column>	date_id

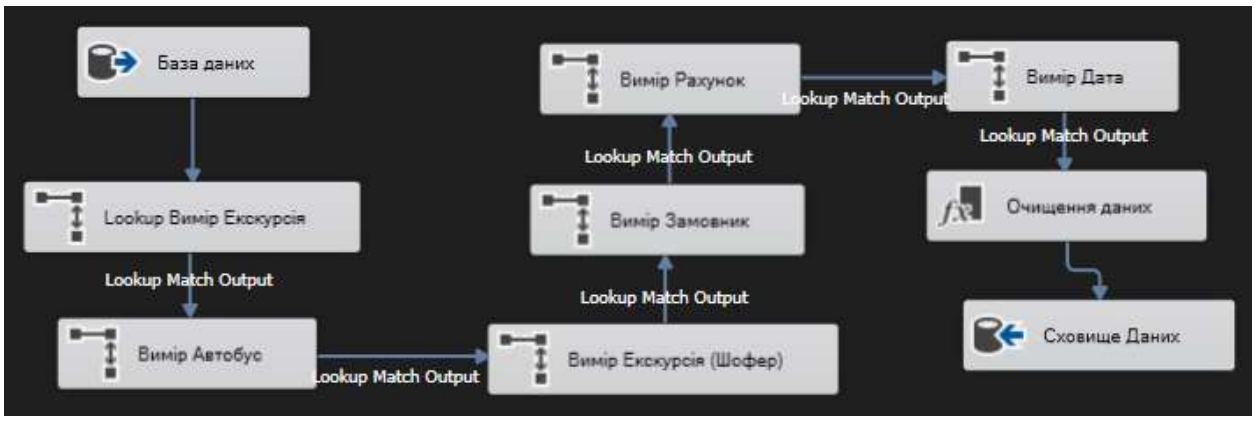
Derived Column Name	Derived Column	Expression
cleaned_total_amount	<add as new column>	total_amount < 0 ? 0 : total_amount
has_payments	<add as new column>	payment_count > 0 ? 1 : 0

У блоці OLE DB Destination встановлюємо наступні зв'язки для таблиці фактів:

Input Column	Destination Column
<ignore>	fact_id
<ignore>	payment_id
Вимір Рахунок.invoice_id	invoice_id
Вимір Екскурсія.excursion_id	excursion_id
Вимір Замовник.customer_id	customer_id
date_id	date_id
cleaned_total_amount	amount

Робимо те саме для таблиці Fact_Excursion:

Структура DataFlow:



Робимо усе те саме, що і для попередньої таблиці (зв’язки у Lookup).

Робимо очистку даних:

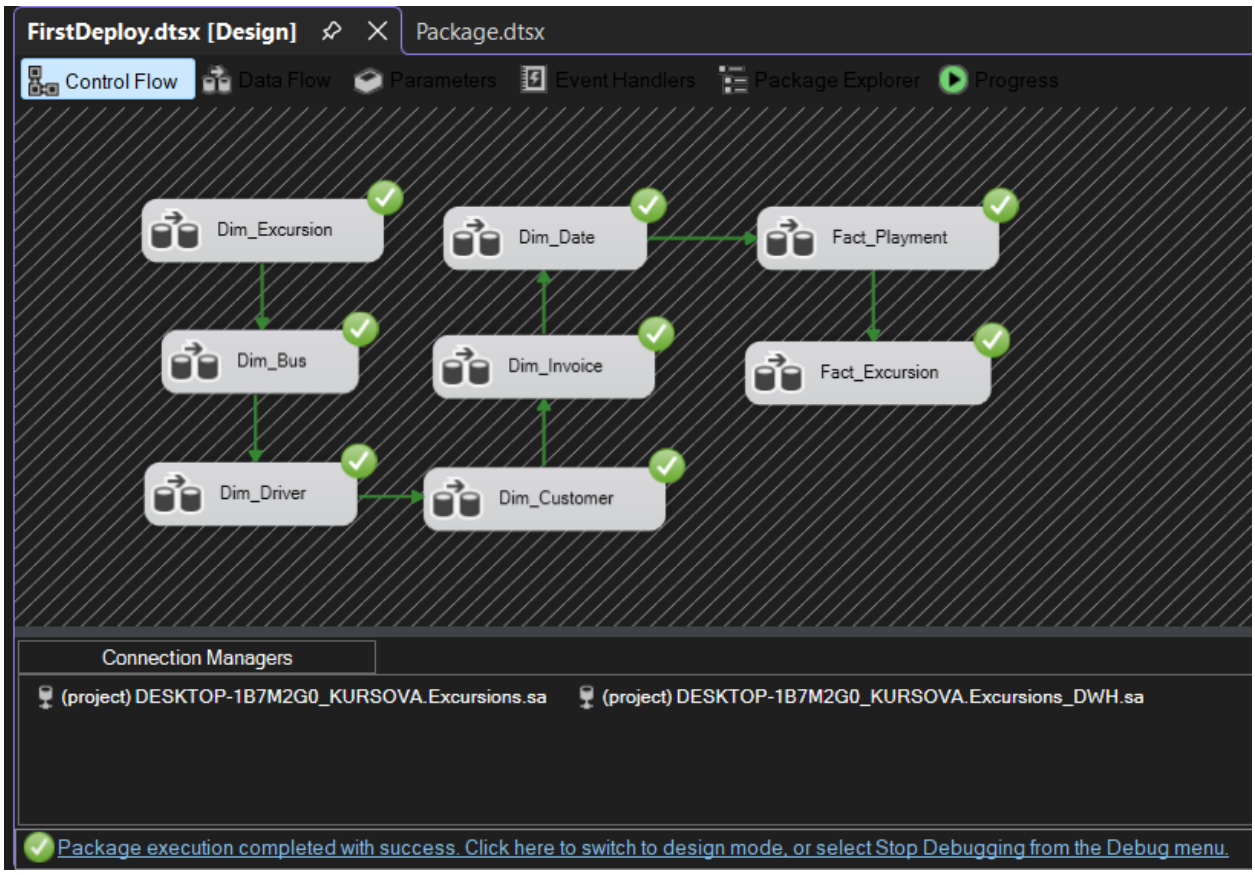
Derived Column Name	Derived Column	Expression
Num_cust_NOT_NULL	<add as new column>	REPLACENULL(num_customers,1)
price_NOT_NULL	<add as new column>	REPLACENULL(price,100)
duration_NOT_NULL	<add as new column>	REPLACENULL(duration_hours,2)

Зв’язки у останньому блоці встановлюємо наступні:

Input Column	Destination Column
<ignore>	fact_id
date_id	date_id
duration_NOT_NULL	duration_hours
Lookup Вимір Екскурсія.excursion_id	excursion_id
Num_cust_NOT_NULL	num_customers
price_NOT_NULL	price
Вимір Автобус.bus_id	bus_id
Вимір Екскурсія (Шофер).driver_id	driver_id
Вимір Замовник.customer_id	customer_id
Вимір Рахунок.invoice_id	invoice_id

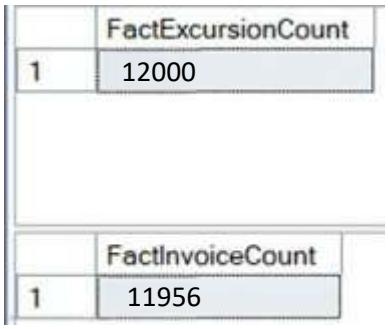
Запускаємо на виконання та перевіряти результат.

Запуск ETL-процесу



Перевіряємо в SSMS:

```
SELECT COUNT(*) AS FactExcursionCount FROM Fact_Excursion;
SELECT COUNT(*) AS FactInvoiceCount FROM Fact_Payment;
```



	FactExcursionCount
1	12000

	FactInvoiceCount
1	11956

ETL-процес виконано успішно. Дані завантажено у сховище даних і вони готові до подальшого аналізу.

Розділ 4. Побудова OLAP-куба та аналітичні звіти

Наступний крок - побудова OLAP-куба на основі DWH для побудови звітів.

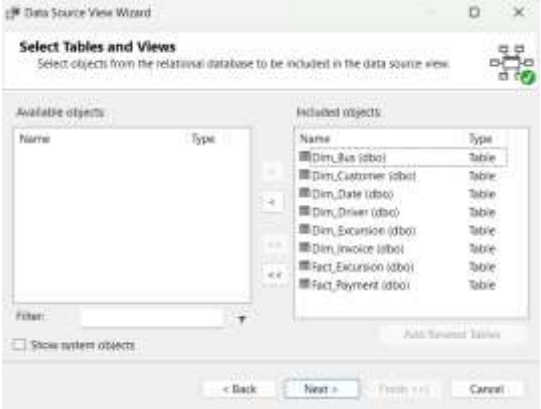
Створення SSAS проекту та підключення до DWH:

Спочатку встановлюємо Microsoft Analysis Services:

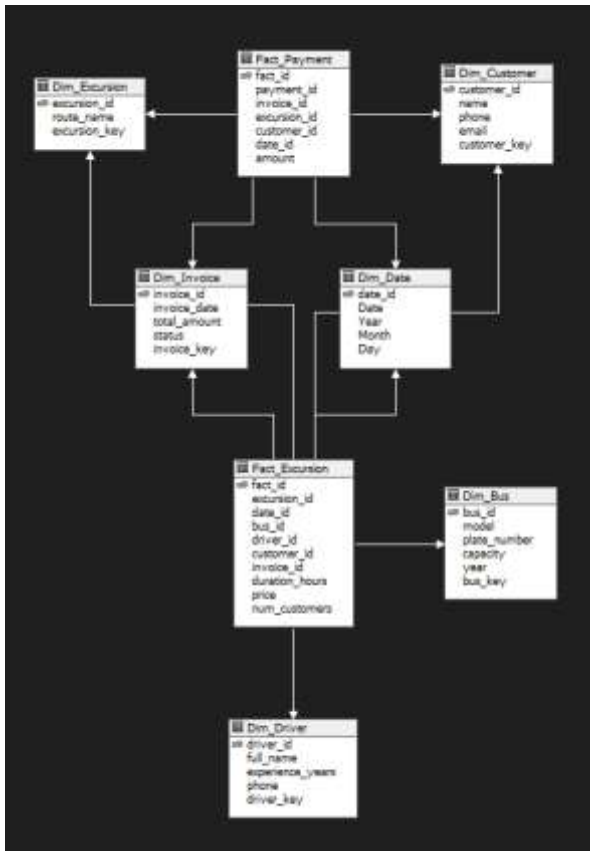
Відкриваємо Visual Studio та вибираємо Analysis Services Multidimensional Project та називаємо його SSAS. Встановлюємо з'єднання до нашого сховища даних.

Тепер потрібно додати Data Sources Views. У Data Sources View Wizard вибираємо наше сховище даних.

Завантажуємо необхідні таблиці в сховище:

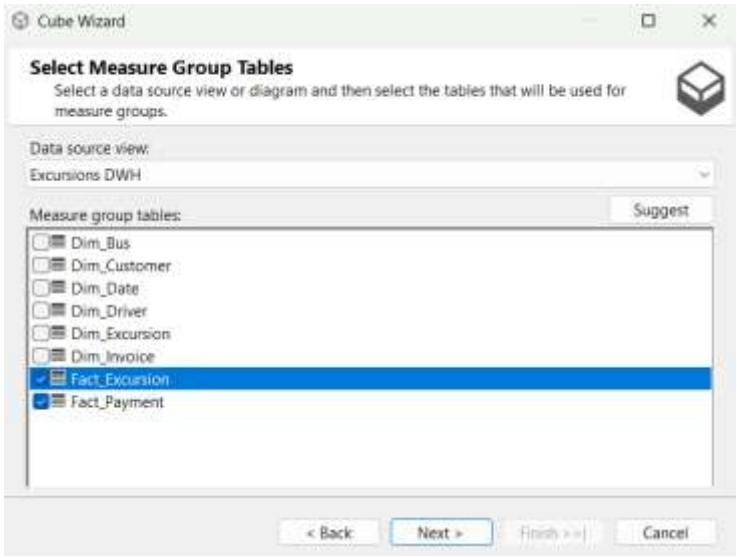


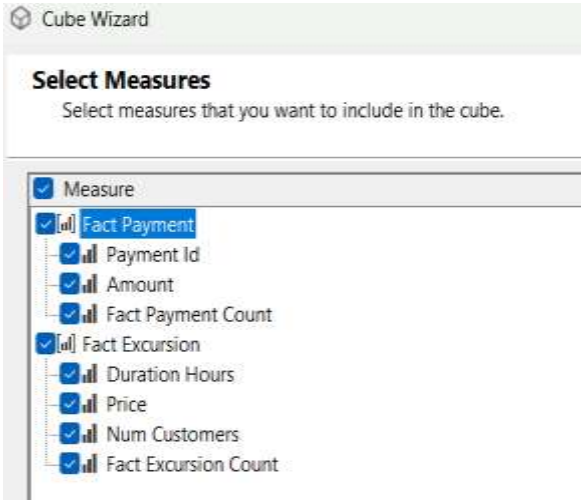
Налаштовуємо перший вимір:
Структуру сховища даних у нашому проекті:



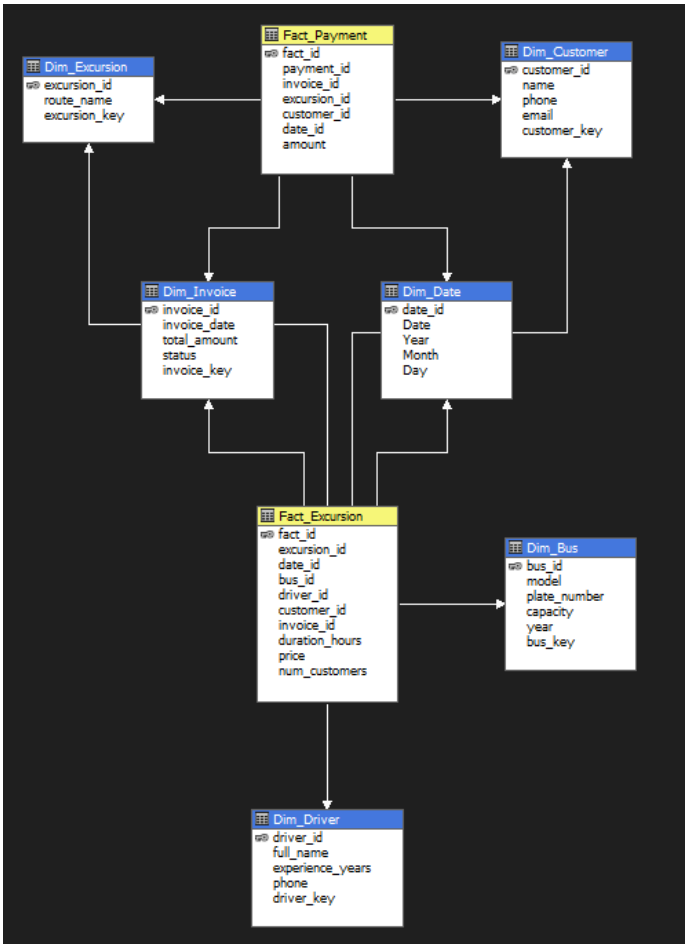
Побудова OLAP-куба:

Щоб побудувати куб у Solution Explorer вибираємо Add cube в розділі Cubes. Відкривається Cube Wizard. У ньому вибираємо Use existing tables. А на наступній вкладці ставимо галочку на таблицях фактів Fact_Excursion та Fact_Payment, таким чином вибираємо їх основними таблицями.

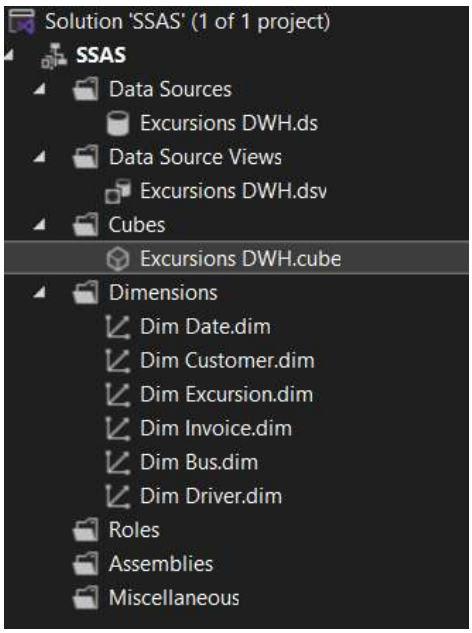




Тепер створений куб:



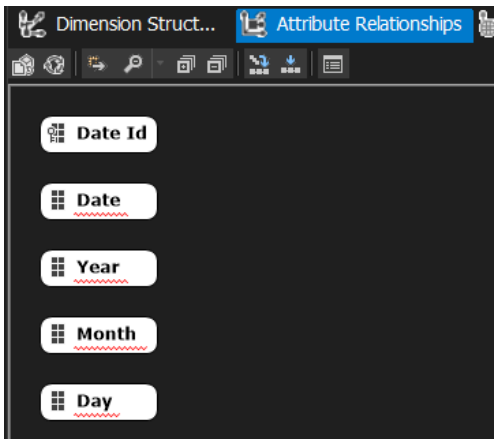
Структура проекту:



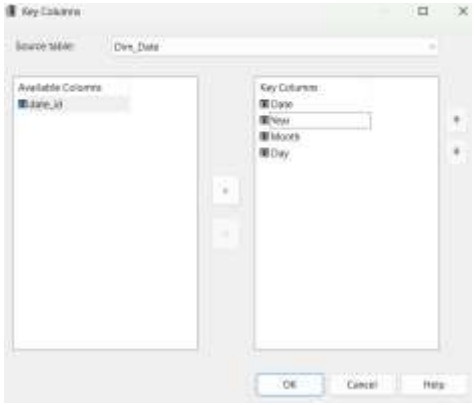
Виміри підтягнулися автоматично, але їх потрібно ще налаштувати. Оскільки в кожному вимірі підтягнулися лише поля з ID у вкладку з атрибутами, то ми повині в усіх вимірах перетягнути в атрибути решту полів з Data Sources View окрім поля з KEY.

Після цього нам потрібно налаштувати ієрархію у вимірі Dim_Date. Ієрархія має вигляд: Data > Year > Month > Day. Для цього додаємо атрибути у вкладку Hierarchies:

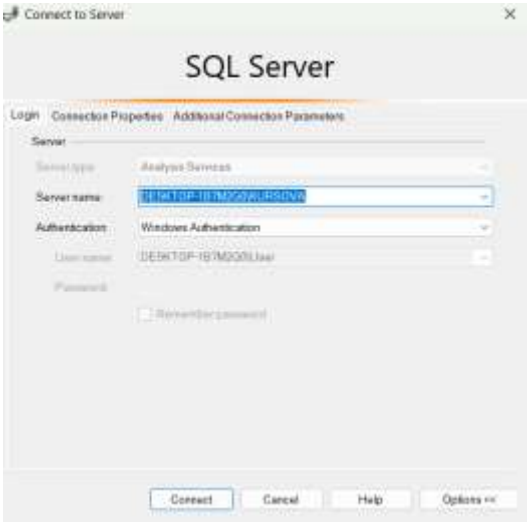
У вкладці Attribute Relationships встановлюємо наступні зв’язки для атрибутів:

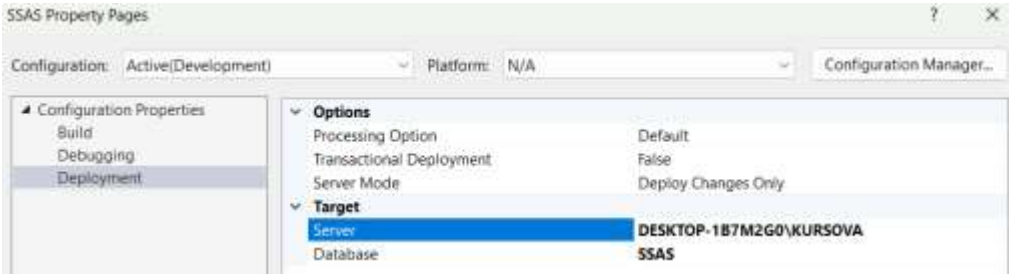


Тепер для кожного атрибута у вимірі Dim_Date встановлюємо у properties в KeyColumns вибираємо усе що для даного атрибута стоїть нище по ієрархії а в NameColumn вибираємо самий атрибут. Так для Date у KeyColumns вибираємо: Date, Year, Month та Day. Для Year вибираємо: Year, Month та Day. Для Month вибираємо: Month та Day, а для Day вибираємо лише Day. Тепер наш куб можна деплоїти.

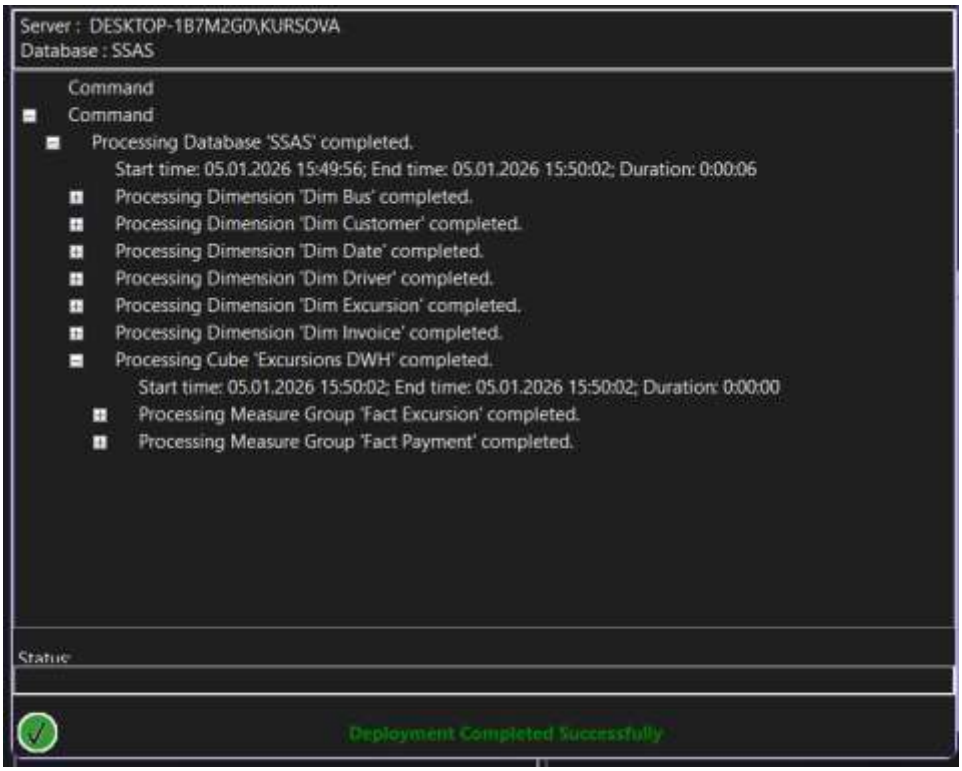


Заходимо в SQL Server Management Studio та копіюємо назву, щоб вставити далі:





Деплоїмо



Побудова звітів

Тепер починаємо побудову звітів. У варіанті предметної області подаються обов’язкові звіти. Але перед їх побудовою потрібно вибрати середовище, яке

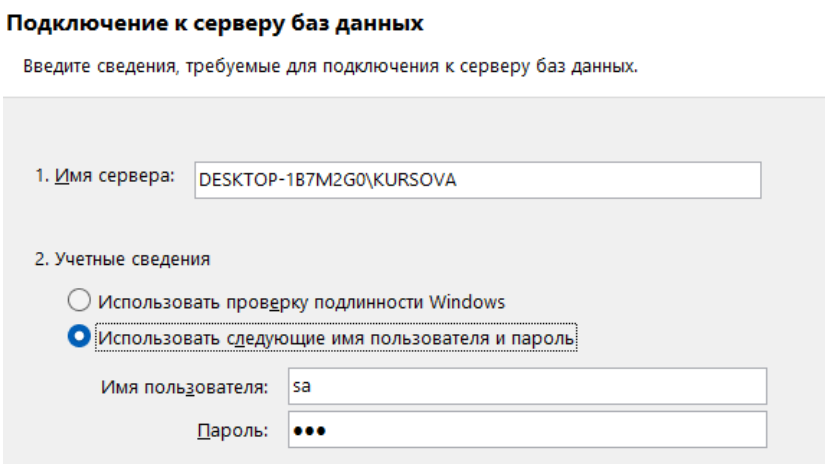
дає можливість побудувати звіти. Компанія Microsoft пропонує декілька своїх варіантів для побудови звітів.

Перший варіант: у Visual Studio можна створити проект Report Server Project і у ньому зробити з'єднання до нашого куба та побудувати звіти. Це простий варіант, але він дозволяє побудувати лише звіти у вигляді таблиць і матриць, побудова діаграм та інших візуалізацій відсутня.

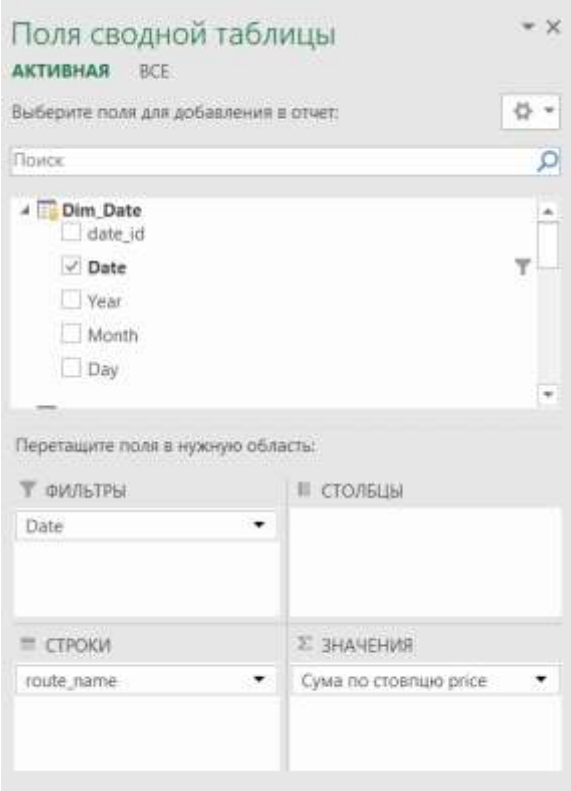
Другий же варіант – побудова звітів у **Excel**. Тут також можна встановити з'єднання до нашого куба через SQL та сформувати звіти за даними. Excel має перевагу у великій кількості типів візуалізації, таких як діаграми, графіки, умовне форматування, а також зручні інструменти для обробки та аналізу даних, наприклад, зведені таблиці та Power Pivot. Це робить Excel гнучким і доступним інструментом для швидкого створення як табличних, так і графічних звітів.

Я для побудови звітів у даній роботі вибрала Excel. Для цього необхідно відкрити програму, перейти у вкладку Дані та обрати Отримати дані > З інших джерел → з сервера SQL Server. Після цього вказується назва серверу і можна підключитися до нього для подальшого створення зведених таблиць та діаграм. Такий підхід дозволяє наочно представити інформацію та виконати глибокий аналіз даних без потреби додаткових програмних інструментів.

Підключення Excel до SQL Server (SSMS база даних)



Вибираю базу даних та створюю зведену таблицю.



Звіт 1 Список екскурсій за період з вартістю

Звіт дозволяє переглянути перелік екскурсій за вибраний період (в нашому випадку грудень 2019 року) із зазначенням вартості.

День - екскурсія	Ціна
2019-01-01	**4528,84**
Тур до Одеси	4528,84
2019-01-02	**18617,36**
Екскурсія до Карпат	4016,72
Екскурсія Кам'янця-Подільського	1189,72
Подорож Закарпаття	3486,48
Подорож Карпат	6204,12
Тур до Одеси	3720,32
2019-01-03	**14010,16**
Екскурсія до Варшави	605,08
Екскурсія до Києва	1803,92
Екскурсія до Луцька	5849,52
Подорож до Києва	5751,64

Звіт 2 Графік використання автобусів

Звіт дозволяє побачити який автобус на яку екскурсію поїхав в вибраний період. Можна переглядати і певний день і тиждень, і місяць.

Date	2026-02-20
Автобус+Екскурсія	**Кількість рейсів**
GS7178HA	**4**
Екскурсія до Варшави	4
IT3662UE	**4**
Екскурсія до Карпат	4
ML8179IB	**4**
Подорож до Києва	4
ST3821HF	**4**
Екскурсія до Карпат	4
TZ8463YR	**4**
Подорож Закарпаття	4
VU7252LS	**4**
Тур до Кам'янця-Подільського	4
XH9895CU	**4**
Тур Чернівців	4
XW5214KA	**4**
Тур до Закарпаття	4
Всього	**32**

Далі Фінансовий звіт

Можна переглядати всі періоди, як за один день, так і за рік і за місяць.

Результат заробітку з екскурсій за 1 день (5 січня 2019):

Date	2019-01-05
Назва екскурсії	**Сума по стовпцю price**
Подорож до Києва	7 624,68 ₴
Подорож Львова	4 010,08 ₴
Тур Києва	5 379,92 ₴
Всього	**17 014,68 ₴**

Тепер за певний місяць, наприклад грудень 2025:

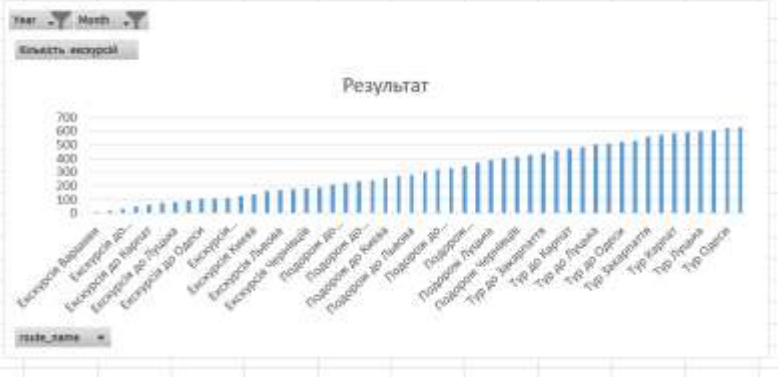
Date	(несколько элементов)
Назва екскурсії	**Сума по стовпцю price**
Екскурсія Варшави	4 484,12 ₴
Екскурсія до Варшави	3 807,44 ₴
Екскурсія до Закарпаття	5 067,60 ₴
Екскурсія до Кам'янця-Подільського	16 117,36 ₴
Екскурсія до Карпат	19 411,60 ₴
Екскурсія до Києва	14 146,00 ₴
Екскурсія до Луцька	7 393,92 ₴
Екскурсія до Львова	5 174,04 ₴
Екскурсія до Одеси	22 770,40 ₴
Екскурсія до Чернівців	12 245,28 ₴
Екскурсія Закарпаття	20 540,16 ₴
Екскурсія Кам'янця-Подільського	17 060,88 ₴
Екскурсія Карпат	979,80 ₴
Екскурсія Києва	2 878,48 ₴
Екскурсія Луцька	19 055,84 ₴
Екскурсія Чернівців	7 492,16 ₴
Подорож Варшави	6 918,24 ₴
Подорож до Варшави	13 719,80 ₴
Подорож до Закарпаття	14 066,04 ₴
Подорож до Кам'янця-Подільського	14 996,80 ₴
Подорож до Карпат	19 048,16 ₴
Подорож до Києва	14 673,52 ₴
Подорож до Луцька	7 914,72 ₴
Подорож до Львова	12 733,12 ₴
Подорож до Одеси	22 152,32 ₴
Подорож до Чернівців	27 677,28 ₴
Подорож Закарпаття	8 044,48 ₴
Подорож Кам'янця-Подільського	6 337,28 ₴
Подорож Карпат	5 016,44 ₴
Подорож Києва	13 514,00 ₴
Подорож Луцька	22 958,16 ₴
Подорож Львова	16 500,40 ₴
Подорож Чернівців	10 300,20 ₴
Тур до Закарпаття	11 661,08 ₴
Тур до Кам'янця-Подільського	13 545,60 ₴
Тур до Львова	11 781,52 ₴
Тур до Одеси	22 193,48 ₴
Тур до Чернівців	3 576,28 ₴
Тур Закарпаття	19 812,48 ₴
Тур Карпат	1 461,16 ₴
Тур Києва	6 503,40 ₴
Тур Луцька	1 839,92 ₴
Тур Львова	7 990,20 ₴
Тур Одеси	571,88 ₴
Тур Чернівців	6 142,20 ₴
Всього	**522 275,24 ₴**

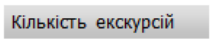
І ще зробимо звіт аналізу популярності маршрутів:

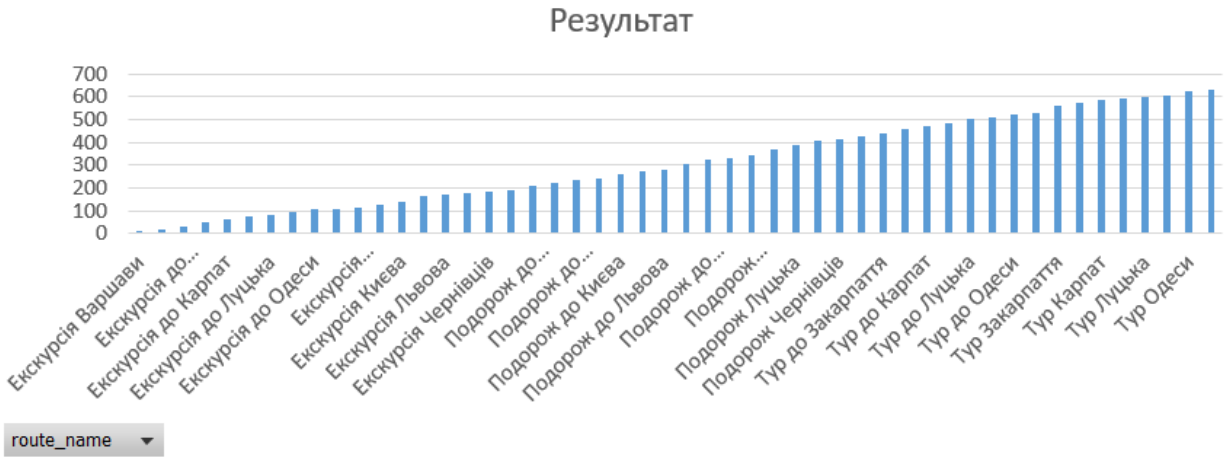
Зліва наведена частина таблиці маршрутів за жовтень 2025 року і їхню кількість, а вже справа для наочності побудована стовпчаста діаграма:

Year	2025
Month	10

Назва екскурсії	Кількість екскурсій
Екскурсія Варшави	12
Екскурсія до Варшави	20
Екскурсія до Закарпаття	28
Екскурсія до Кам'янця-Подільського	48
Екскурсія до Карпат	60
Екскурсія до Києва	76
Екскурсія до Луцька	84
Екскурсія до Львова	96
Екскурсія до Одеси	104
Екскурсія Закарпаття	108
Екскурсія Кам'янця-Подільського	116
Екскурсія Карпат	128
Екскурсія Києва	140
Екскурсія Луцька	164
Екскурсія Львова	172
Екскурсія Одеси	180
Екскурсія Чернівців	184







Висновки

В результаті розроблена інформаційна система для автоматизації діяльності екскурсійної компанії з використанням технологій платформи Microsoft SQL Server. Було реалізовано повний цикл створення OLAP-куба: від проектування операційної бази даних, через побудову сховища даних (Data Warehouse) та реалізацію ETL-процесів, до розгортання аналітичного куба в SSAS та побудови звітів за допомогою Excel.

Короткий хід дій:

Проектування бази даних: Було спроектовано реляційну базу даних «Excursions», яка відповідає вимогам предметної області та приведена до третьої нормальної форми (3НФ). База містить 6 взаємопов'язаних таблиць (Excursions, Buses, Drivers, Customers, Invoices, Payments), що забезпечує цілісність даних, уникнення надмірності та підтримку ключових бізнес-правил. Для автоматизації фінансових операцій створено тригери, які автоматично формують рахунки при додаванні екскурсій та оновлюють статуси оплати при здійсненні платежів.

Генерація реалістичних тестових даних: За допомогою інструменту Redgate SQL Data Generator сформовано великий обсяг даних: 12 000 записів про екскурсії, 500 автобусів, 400 водіїв дані за період понад 5 років. Дані були згенеровані з урахуванням реальних номерів телефонів, email, державні номери автобусів, що дозволило імітувати роботу системи в умовах, близьких до реальних.

Data Warehouse: Далі було створено сховище даних Excursions_DWH зі зіркоподібною схемою.. У ньому реалізовано 2 таблиці фактів (Fact_Excursion, Fact_Payment) та 5 вимірних таблиць (Dim_Date, Dim_Excursion, Dim_Bus, Dim_Driver, Dim_Customer, Dim_Invoice), що забезпечує швидкий та ефективний доступ до агрегованих даних для OLAP-аналізу.

SSIS: У середовищі Visual Studio було розроблено Integration Services проект, який автоматизує процес завантаження, трансформації та очищення даних з операційної бази в сховище. Кожен потік даних (Data Flow) було налаштовано з урахуванням зв'язків між вимірами та фактами, що дозволило коректно перенести та агрегувати інформацію. Успішне виконання пакету SSIS підтвердило повне та точне завантаження даних у DWH.

OLAP-куб в SSAS: на основі сховища даних було розгорнуто багатовимірний OLAP-куб у SQL Server Analysis Services. У кубі налаштовано всі виміри з атрибутами та ієрархіями, що дозволяє проводити гнучкий аналіз даних за різними критеріями. Куб успішно деплойлено та готовий до використання в аналітичних звітах.

Побудова звітів у Excel: для аналізу було використано Microsoft Excel з підключенням до OLAP-куба через зведені таблиці. Одну таблицю можна використовувати для різних періодів в часі.

Розроблену систему можна використовувати в реальній екскурсійній компанії для автоматизації обліку екскурсій, управління транспортом та персоналом, контролю фінансів, а також для отримання аналітичних звітів, необхідних для планування. Система як раз дозволяє скоротити час на формуванні звітів, але можна їх і доповнити іншими потрібними даними, а також проаналізувати великий об’єм інформації, наприклад для контролю завантаженості автобусів та шоферів.

В перспективі розвитку системи для більш наочної візуалізації можна реалізувати аналітичні звіти в Power BI.

Отже, маємо повністю реалізовану систему управління надвеликою базою даних для екскурсійної компанії, яка відповідає сучасним вимогам до інформаційних систем: масштабованість, продуктивність, аналітична спрямованість та зручність використання.

Список використаних джерел

1. Microsoft SQL Server Analysis Services (SSAS) Documentation.
2. Visual Studio 2022 та 2026 року https://visualstudio.microsoft.com/vs/community/
3. https://codestoresolutions.com/web-app-development/difference-between-microsoft-ssrs-ssas-and-ssis/
4. Для побудови зведених таблиць в Excel: https://support.microsoft.com/uk-ua/office/%D1%81%D1%82%D0%B2%D0%BE%D1%80%D0%B5%D0%BD%D0%BD%D1%8F-%D0%B7%D0%B2%D0%B5%D0%B4%D0%B5%D0%BD%D0%BE%D1%97-%D1%82%D0%B0%D0%B1%D0%BB%D0%B8%D1%86%D1%96-%D0%B4%D0%BB%D1%8F-%D0%B0%D0%BD%D0%B0%D0%BB%D1%96%D0%B7%D1%83-%D0%B4%D0%B0%D0%BD%D0%B8%D1%85-%D0%B0%D1%80%D0%BA%D1%83%D1%88%D0%B0-a9a84538-bfe9-40a9-a8e9-f99134456576

Додаток

SQL-скрипт для бази даних Excursions

```
CREATE DATABASE Excursions;
GO


USE Excursions;
GO

CREATE TABLE Buses (
    bus_id INT IDENTITY(1,1) PRIMARY KEY,
    model VARCHAR(50) NOT NULL,
    plate_number VARCHAR(15) NOT NULL UNIQUE,
    capacity INT CHECK (capacity > 0),
    [year] INT CHECK ([year] >= 1990)
);
GO

CREATE TABLE Drivers (
    driver_id INT IDENTITY(1,1) PRIMARY KEY,
    full_name VARCHAR(100) NOT NULL,
    experience_years INT CHECK (experience_years >= 0),
    phone VARCHAR(20) UNIQUE
);
GO

CREATE TABLE Customers (
    customer_id INT IDENTITY(1,1) PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    phone VARCHAR(20),
    email VARCHAR(100) UNIQUE
);
GO

CREATE TABLE Excursions (
    excursion_id INT IDENTITY(1,1) PRIMARY KEY,
    route_name VARCHAR(100) NOT NULL,
    start_date DATE NOT NULL,
    duration_hours INT CHECK (duration_hours > 0),
    price DECIMAL(10,2) CHECK (price >= 0),
    bus_id INT NOT NULL,
    driver_id INT NOT NULL,
    customer_id INT NOT NULL,
    CONSTRAINT FK_Excursions_Buses FOREIGN KEY (bus_id) REFERENCES Buses(bus_id),
    CONSTRAINT FK_Excursions_Drivers FOREIGN KEY (driver_id) REFERENCES
Drivers(driver_id),
    CONSTRAINT FK_Excursions_Customers FOREIGN KEY (customer_id) REFERENCES
Customers(customer_id)
);
GO

CREATE TABLE Invoices (
    invoice_id INT IDENTITY(1,1) PRIMARY KEY,
    excursion_id INT NOT NULL UNIQUE,
    invoice_date DATE NOT NULL,
    total_amount DECIMAL(10,2) CHECK (total_amount >= 0),
    status VARCHAR(20)
        CHECK (status IN ('оплачено', 'частково', 'не оплачено')),
    CONSTRAINT FK_Invoices_Excursions FOREIGN KEY (excursion_id)
        REFERENCES Excursions(excursion_id)
);
GO
```

```
CREATE TABLE Payments (
    payment_id INT IDENTITY(1,1) PRIMARY KEY,
    invoice_id INT NOT NULL,
    payment_date DATE NOT NULL,
    amount DECIMAL(10,2) CHECK (amount > 0),
    CONSTRAINT FK_Payments_Invoices FOREIGN KEY (invoice_id)
        REFERENCES Invoices(invoice_id)
);
GO
```

Тригери

```
USE Excursions;
GO

GO
CREATE TRIGGER trg_UpdateInvoiceAfterPayment
ON Payments
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
    SET NOCOUNT ON;

    UPDATE i
    SET
        total_amount = ISNULL(p.sum_amount, 0),
        status =
            CASE
                WHEN ISNULL(p.sum_amount, 0) = 0 THEN 'не оплачено'
                WHEN ISNULL(p.sum_amount, 0) < e.price THEN 'частково'
                ELSE 'оплачено'
            END
    FROM Invoices i
    JOIN Excursions e ON i.excursion_id = e.excursion_id
    LEFT JOIN (
        SELECT invoice_id, SUM(amount) AS sum_amount
        FROM Payments
        GROUP BY invoice_id
    ) p ON i.invoice_id = p.invoice_id
    WHERE i.invoice_id IN (
        SELECT invoice_id FROM inserted
        UNION
        SELECT invoice_id FROM deleted
    );
END;
GO


GO
CREATE TRIGGER trg_CreateInvoiceAfterExcursion
ON Excursions
AFTER INSERT
AS
BEGIN
    INSERT INTO Invoices (excursion_id, invoice_date, total_amount, status)
    SELECT
        excursion_id,
        GETDATE(),
        0,
        'не оплачено'
    FROM inserted;
END;
GO
```

SQL-скрипт для бази даних Excursions_DWH:

```
CREATE DATABASE Excursions_DWH;
GO

USE Excursions_DWH;
GO

CREATE TABLE Dim_Date (
    date_id INT PRIMARY KEY IDENTITY(1,1),
    Date DATE NULL,
    Year INT NULL,
    Month INT NULL,
    Day INT NULL
);
GO

CREATE TABLE Dim_Excursion (
    excursion_id INT PRIMARY KEY IDENTITY(1,1),
    route_name VARCHAR(100),
    excursion_key INT NULL
);
GO

CREATE TABLE Dim_Bus (
    bus_id INT PRIMARY KEY IDENTITY(1,1),
    model VARCHAR(50),
    plate_number VARCHAR(15),
    capacity INT,
    [year] INT,
    bus_key INT NULL
);
GO

CREATE TABLE Dim_Driver (
    driver_id INT PRIMARY KEY IDENTITY(1,1),
    full_name VARCHAR(100),
    experience_years INT,
    phone VARCHAR(20),
    driver_key INT NULL
);
GO

CREATE TABLE Dim_Customer (
    customer_id INT PRIMARY KEY IDENTITY(1,1),
    name VARCHAR(100),
    phone VARCHAR(20),
    email VARCHAR(100),
    customer_key INT NULL
);
GO

CREATE TABLE Dim_Invoice (
    invoice_id INT PRIMARY KEY IDENTITY(1,1),
    invoice_date DATE,
    total_amount DECIMAL(10,2),
    status VARCHAR(20) CHECK (status IN ('оплачено','частково','не оплачено')),
    invoice_key INT NULL
);
GO
```

Таблиці фактів:

```
CREATE TABLE Fact_Excursion (
```

```
    fact_id INT PRIMARY KEY IDENTITY(1,1),
    excursion_id INT NULL,
    date_id INT NULL,
    bus_id INT NULL,
    driver_id INT NULL,
    customer_id INT NULL,
    invoice_id INT NULL,
    duration_hours INT NULL,
    price DECIMAL(10,2) NULL,
    num_customers INT NULL,
    FOREIGN KEY (excursion_id) REFERENCES Dim_Excursion(excursion_id),
    FOREIGN KEY (date_id) REFERENCES Dim_Date(date_id),
    FOREIGN KEY (bus_id) REFERENCES Dim_Bus(bus_id),
    FOREIGN KEY (driver_id) REFERENCES Dim_Driver(driver_id),
    FOREIGN KEY (customer_id) REFERENCES Dim_Customer(customer_id),
    FOREIGN KEY (invoice_id) REFERENCES Dim_Invoice(invoice_id)
);
GO

-- Факт оплати
CREATE TABLE Fact_Payment (
    fact_id INT PRIMARY KEY IDENTITY(1,1),
    payment_id INT NULL,
    invoice_id INT NULL,
    excursion_id INT NULL,
    customer_id INT NULL,
    date_id INT NULL,
    amount DECIMAL(10,2) NULL,
    FOREIGN KEY (invoice_id) REFERENCES Dim_Invoice(invoice_id),
    FOREIGN KEY (excursion_id) REFERENCES Dim_Excursion(excursion_id),
    FOREIGN KEY (customer_id) REFERENCES Dim_Customer(customer_id),
    FOREIGN KEY (date_id) REFERENCES Dim_Date(date_id)
);
GO
```

Запит в OLE DB SOURCE в Fact_Payment

```
SELECT
    i.invoice_id,
    i.excursion_id,
    e.customer_id,
    i.invoice_date,

    CASE
        WHEN i.total_amount < 0 THEN 0
        WHEN i.total_amount > 1000000 THEN 1000000 -- обмеження для викидів
        ELSE i.total_amount
    END AS total_amount,

    CASE
        WHEN COALESCE(SUM(p.amount), 0) = 0 THEN 'не оплачено'
        WHEN COALESCE(SUM(p.amount), 0) >= i.total_amount THEN 'оплачено'
        ELSE 'частково'
    END AS status,

    -- Агрегація платежів з перевіркою
    CASE
        WHEN COALESCE(SUM(p.amount), 0) < 0 THEN 0
        WHEN COALESCE(SUM(p.amount), 0) > i.total_amount THEN i.total_amount
        ELSE COALESCE(SUM(p.amount), 0)
    END AS paid_amount,

    COUNT(p.payment_id) AS payment_count,

    CASE
        WHEN (i.total_amount - COALESCE(SUM(p.amount), 0)) < 0 THEN 0
```

```
        ELSE i.total_amount - COALESCE(SUM(p.amount), 0)
    END AS balance

FROM Invoices i
JOIN Excursions e ON i.excursion_id = e.excursion_id
LEFT JOIN Payments p ON i.invoice_id = p.invoice_id

-- Фільтрація некоректних даних
WHERE i.invoice_date <= GETDATE()
  AND i.invoice_date >= '2019-01-01'
  AND i.total_amount IS NOT NULL
  AND i.excursion_id IS NOT NULL

GROUP BY
    i.invoice_id,
    i.excursion_id,
    e.customer_id,
    i.invoice_date,
    i.total_amount,
    i.status
```