Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Інститут прикладної математики та фундаментальних наук

Кафедра прикладної математики

Курсова робота

з курсу «Надвеликі бази даних»

на тему «Розробка та аналіз надвеликої бази даних з використанням технологій Microsoft SQL Server»

Виконала:

студентка групи ПМ-41

Зоряна ОЛЕЙНИК

Перевірив:

Богдан ЛЮБІНСЬКИЙ

Львів — 2026

Вступ

У сучасних умовах розвитку бізнесу особливого значення набуває оперативний аналіз великих обсягів даних, зокрема даних про продажі товарів. Керівники та аналітики потребують інструментів, які дозволяють швидко оцінювати динаміку продажів, виявляти тенденції, аналізувати ключові показники ефективності та приймати обґрунтовані управлінські рішення. Використання транзакційних (оперативних) баз даних для таких завдань є неефективним через складність агрегації та низьку швидкість виконання аналітичних запитів. Одним із найбільш поширених підходів до розв'язання цієї проблеми є використання бізнес-аналітичних систем (Business Intelligence, BI), які базуються на сховищах даних та OLAP-кубах. Такі системи дозволяють зберігати історичні дані, виконувати багатовимірний аналіз та будувати інтерактивні аналітичні звіти.

Метою даної курсової роботи є проєктування та реалізація аналітичної системи для аналізу продажів товарів із використанням технологій Microsoft SQL Server – зокрема, засобів **SSIS**, **SSAS** та **SSRS**. У межах роботи реалізовано повний цикл побудови BI-рішення – від аналізу предметної області до створення OLAP-куба та інтерактивних звітів і дашбордів. В процесі реалізації особлива увага приділялася виконанню вимог варіанту "Облік товарів", що передбачає роботу з надвеликими обсягами даних (щонайменше 50 000 найменувань товарів та 1 000 000 операцій за 5-річний період) та формування регламентованих звітів.

Об'єкт дослідження: процеси обліку та аналізу операцій з товарами у діяльності підприємства.

Предмет дослідження: методи та засоби побудови сховищ даних і аналітичних звітів для підтримки прийняття рішень.

Розділ 1. Аналіз предметної області

Характеристика предметної області. Предметною областю даної роботи є варіант 8 «Облік товарів». Основними даними, які формуються в процесі діяльності підприємства, є інформація про реалізовані товари, обсяги продажів, суми виручки та часові характеристики операцій. Дані накопичуються у великих обсягах і мають виражену залежність від часу (сезонність, тенденції змін показників по роках). Окрім продажів, бізнес-процеси цієї предметної області включають закупівлю товарів у постачальників (прихід на склад), продаж кінцевим споживачам (реалізація) та списання або повернення товарів (вибуття). Кожна операція супроводжується фінансовими документами (рахунки, акти) та платежами. Таким чином, інформаційна система повинна підтримувати облік товарних запасів, рух товарів (прихід/розхід) та грошових потоків, пов'язаних з цими операціями.

Основні бізнес-процеси:

- **Управління номенклатурою товарів.** Включає ведення довідника товарів та їх категорій (груп). Кожен товар належить до певної категорії, може мати ряд атрибутів (назва, артикул, одиниця виміру, ціна, тощо). Бізнес-правило: **кожен товар повинен бути віднесений до однієї категорії**, яка визначає його тип або групу.
- **Закупівля товарів у постачальників.** Відображається операціями приходу товарів на склад від певного постачальника. Операція фіксує товар, кількість і закупівельну вартість. Бізнес-правила: *кожна операція приходу прив'язана до постачальника та товарної позиції*; *для кожного приходу формується документ (накладна) та здійснюється оплата постачальнику*.
- **Продаж товарів клієнтам.** Оформлюється операцією реалізації (продажу) зі вказанням товару, кількості, продажної ціни, покупця (клієнта) та дати продажу.

Основний бізнес-процес – оформлення факту продажу з фіксацією кількості та суми, що генерує дохід. Бізнес-правила: *не можна продати більше, ніж є в наявності*; *кожен продаж оформлюється документом (чеком) і передбачає отримання оплати від клієнта*.

- **Списання та повернення.** В окремих випадках товар може списуватися (наприклад, псування, втрати) або повертатися від клієнта. Такі операції також повинні фіксуватися (як окремі види операцій “вибуття”) з відповідним документальним підтвердженням.

Функціональні вимоги до системи: На основі аналізу предметної області сформульовано такі вимоги до аналітичної системи:

- Зберігання та обробка великої кількості транзакцій продажів і пов’язаних довідникових даних (номенклатура, клієнти, постачальники, календар та ін.).
- Підтримка історичних даних за кілька років для аналізу динаміки (мінімум 5-річний історичний період даних).
- Можливість швидкого отримання агрегованих показників діяльності (загальна сума продажів, кількість проданих одиниць, середній чек тощо) у різних розрізах (час – рік, квартал, місяць, день; товарні категорії; регіони або типи клієнтів тощо).
- Наявність засобів візуалізації ключових показників ефективності (KPI) на дашбордах для керівництва.
- Підтримка інтерактивної роботи зі звітами: можливість фільтрації, сортування даних, деталізації (drill-down/drill-through) у звітах для детального аналізу.

Бізнес-правила та обмеження:

- **Цілісність даних:** система повинна забезпечувати цілісність зв’язків між операціями і довідниками. Кожна операція (факт продажу чи приходу) має посилатися на наявний товар, чинного постачальника або клієнта, та валідну дату з календаря. Усі зовнішні ключі в таблиці фактів мають відповідати

наявним записам у таблицях-вимірах (довідниках). При відсутності відповідного ключа операція не повинна завантажуватися (такі записи фіксуються як помилкові в процесі ETL). В ході реалізації це правило перевірено: наприклад, перевірка зв'язків FactSales→DimTime показала відсутність розірваних посилань (кожна фактова транзакція має валідну дату).

- **Валідація показників:** кількісні значення в операціях (кількість, суми) не можуть бути від'ємними; для цього при генерації даних та завантаженні застосовувалися перевірки (CONSTRAINT типу CHECK в SQL або відповідні вирази в трансформаціях).
- **Актуальність даних довідників:** довідники товарів, клієнтів, постачальників можуть змінюватися з часом (найменування, характеристики тощо). Історичні зміни відслідковуються за допомогою механізму **Slowly Changing Dimensions Type 2** – при зміні атрибутів створюється новий запис, а старий помічається як неактуальний. Це реалізовано через поля періоду дії запису (ValidFrom, ValidTo) та ознаку актуальності (IsCurrent) у таблицях-вимірах. Таким чином, зберігається повна історія змін довідникової інформації.
- **Контроль залишків товарів:** хоча основний фокус – аналіз продажів, для коректності обліку вважається, що система контролює залишки товарів на складі. Наприклад, перед завантаженням фактів продажів здійснюється зіставлення з даними про надходження (або припускається, що вхідні дані вже не містять продажів понад наявний залишок). В рамках даного проєкту модель запасів спрощена: основний аналіз виконується за фактами продажів, тому поле залишку на складі у звітах формується на основі згенерованих умовно даних або заданих правил (наприклад, ініціальні залишки та розрахунок залишку = прихід – продаж).

Отже, проаналізована предметна область визначила необхідність використання технологій **сховищ даних** та **OLAP** для ефективної роботи з великими обсягами історичних даних про операції з товарами. Для реалізації поставлених вимог обрано архітектуру BI-рішення на основі сховища даних, OLAP-куба та системи звітності, що може бути побудована цілісно на платформі Microsoft SQL Server. У наступних розділах описано проєктування сховища даних, процес ETL, побудову OLAP-куба та розробку аналітичних звітів відповідно до сформульованих вимог.

Розділ 2. Проєктування бази даних

Концептуальна модель (ER-діаграма)

На концептуальному рівні модель даних для системи “Облік товарів” спроєктована у вигляді багатовимірної **схеми типу “зірка” (Star Schema)**. Такий підхід є стандартним для сховищ даних і передбачає розділення даних на факти (кількісні показники бізнес-процесів) та виміри (довідникові дані, за якими ці показники аналізуються). **Зіркоподібна схема** містить одну центральну таблицю фактів, що накопичує записи про події (транзакції), та набір пов’язаних з нею таблиць вимірів, які містять описову інформацію про різні аспекти цих подій. У нашому випадку:

- **FactSales** – таблиця фактів продажів товарів. В ній зберігаються записи про кожен продаж: дата та час продажу, посилання на проданий товар (і його категорію), посилання на клієнта (покупця), а також основні метричні показники – кількість і сума продажу. Кожен запис у FactSales характеризує одну транзакцію продажу (мінімальна **гранулярність** – окрема продажна операція) і має свій сурогатний ключ (SaleID). Для зв’язку з вимірами таблиця FactSales містить зовнішні ключі: TimeKey, ProductSK, CategorySK, CustomerSK (а первинним ключем, наприклад, міг бути власний ідентифікатор SaleID).

- **DimTime** – вимір часу (календар). Містить атрибути дати: Рік, Квартал, Місяць, День, День тижня тощо, а також унікальний ключ TimeKey (наприклад, формат YYYYMMDD чи ID дня). Цей вимір дозволяє аналізувати продажі в різних часових розрізах (роки, місяці, дні).
- **DimProduct** – вимір товарів. Містить інформацію про товари: сурогатний ключ ProductSK, натуральний ключ (код товару, ProductID або SKU), назва товару, одиниця виміру, стандартна собівартість, базова ціна, ознака активності, атрибути для SCD (дата початку/кінця дії запису, IsCurrent) та ін. Цей вимір дозволяє аналізувати продажі по конкретних товарах.
- **DimCategory** – вимір категорій товарів. Містить список категорій (груп товарів): CategorySK, код або ID категорії, назва категорії тощо. Зв'язок між DimProduct і DimCategory реалізовано або як окремий вимір (зв'язок FactSales → DimCategory безпосередньо) або через атрибут у DimProduct. У нашій моделі обрано зберігати категорії окремо, тому FactSales має окремий ключ CategorySK на таблицю DimCategory.
- **DimCustomer** – вимір покупців (клієнтів). Містить дані про клієнтів: CustomerSK, ID клієнта, ім'я/назва, контакти, можливо сегмент чи тип клієнта. Це дозволяє аналізувати продажі за клієнтськими сегментами.

Додатково, враховуючи специфіку варіанту, розглядалася наявність виміру **DimSupplier** (постачальники). Однак виявлено, що для фактів **продажів** постачальник не є безпосередньо пов'язаним (постачальник стосується операцій закупівлі). Тому вирішено: **таблиця FactSales не містить посилання на постачальника** – цей вимір може використовуватися в іншому фактому процесі (наприклад, факти закупівель, як окрема фактова таблиця). На етапі реалізації була спроба включити SupplierSK у FactSales, проте це призвело до логічних розбіжностей і ускладнення ETL. У результаті модель скориговано: стовпець Supplier прибрано з FactSales, а всі пов'язані з ним перетворення – видалено.

Таким чином, підмодель продажів сфокусована на покупцях; дані про постачальників передбачено врахувати у майбутньому через окремий факт закупівель або відповідні зв'язки, але не в межах FactSales.

Загалом концептуальна ER-діаграма системи включає вказані сутності та зв'язки між ними (зв'язок "один-до-багатьох" від вимірів до факту: один запис виміру може відповідати багатьом фактам продажу). Така зіркоподібна схема є найбільш простою і розповсюдженою моделлю для сховищ даних, оскільки спрощує логіку запитів та бізнес-звітності за рахунок денормалізації даних і централізації фактів у одній таблиці.

```
USE Kursova_NVBD_V8_DW;
SELECT
   (SELECT COUNT(*) FROM dbo.DimProduct)  AS DimProduct,
   (SELECT COUNT(*) FROM dbo.DimCategory) AS DimCategory,
   (SELECT COUNT(*) FROM dbo.DimCustomer) AS DimCustomer,
   (SELECT COUNT(*) FROM dbo.DimSupplier) AS DimSupplier,
   (SELECT COUNT(*) FROM dbo.DimTime)     AS DimTime;
```

100 %

Results | Messages

	DimProduct	DimCategory	DimCustomer	DimSupplier	DimTime
1	50000	300	100000	2000	1827

Логічна модель та нормалізація

На логічному рівні здійснено деталізацію моделі даних та нормалізацію структур. Таблиці вимірів були приведені як правило до **3-ї нормальної форми (3НФ)**, щоб уникнути дублювання даних у довідниках. Зокрема:

- Таблиця **DimProduct** містить тільки атрибути, що стосуються товару. Інформація про категорію не дублюється тут, оскільки винесена в DimCategory (нормалізація по категоріях). Первинний ключ DimProduct – ProductSK (сурогатний ключ), атрибути товару (назва, ціна тощо) – атомарні. Для зв'язку з

категоріями в DimProduct може зберігатися натуральний ключ категорії (CategoryID) як атрибут, але у нашій схемі цей зв'язок реалізується через факт (FactSales має окремий CategorySK). Таким чином уникнуто дублювання назв категорій у кожному записі товару.

- Таблиця **DimCategory** – невелика довідникова таблиця категорій, вже є нормальною (список унікальних категорій). Якщо б існувала ієрархія категорій (категорія → надкатегорія), її можна було б відобразити або рекурсивно, або додатковою таблицею – але у нашому випадку ієрархія категорій неглибока, тому достатньо одного рівня.

- Таблиця **DimCustomer** – довідник клієнтів, також максимально нормалізований (кожен клієнт один раз з унікальним ID, повторювані атрибути винесені – наприклад, якщо б треба було зберігати міста чи регіони клієнтів, їх можна було б винести в окрему таблицю довідника географії; у межах нашої моделі такі деталі опущені).

Нормалізація на етапі розробки сховища даних використовується помірно, адже сховище даних зазвичай частково денормалізоване для швидшого зчитування. У нашій моделі денормалізація проявляється в тому, що таблиця фактів містить дублюючі ключі (для кожного продажу повторюються ключі дат, товарів, клієнтів), а також у тому, що деякі похідні показники можуть зберігатися явно. Проте зайвої денормалізації уникали, щоб мінімізувати обсяг даних: наприклад, FactSales не містить текстових даних або описів, а лише ключі та числові показники. Всі описи (назви товарів, категорій, імена клієнтів) зберігаються у відповідних вимірах і можуть під'єднуватися до фактів через JOIN при запитах або на рівні OLAP. Це відповідає принципам побудови вимірів та фактів: **фактова таблиця містить лише ключі вимірів та числові міри**, тоді як вся описова інформація перебуває у таблицях вимірів.

Первинні та зовнішні ключі. У кожній таблиці було визначено первинний ключ (PK) – для фактів це сурогатний ідентифікатор транзакції (такий ключ вводиться для унікальності рядків факту, хоча в аналітичних цілях він не обов'язково використовується), для вимірів – їх сурогатні ключі (SK). Зовнішні ключі (FK) встановлені в таблиці фактів, що посилаються на відповідні PK таблиць вимірів. Наприклад:

- FactSales.TimeKey → DimTime.TimeKey
- FactSales.ProductSK → DimProduct.ProductSK
- FactSales.CategorySK → DimCategory.CategorySK
- FactSales.CustomerSK → DimCustomer.CustomerSK

Такі зв'язки забезпечують референційну цілісність: кожен факт продажу має відповідні записи в довідниках. Це було перевірено шляхом виконання тестових запитів на відсутність "висячих" ключів (наприклад, LEFT JOIN факту з календарем на TimeKey не показує NULL у вимірі, що підтверджує коректність завантаження дат).

Індекси. Для оптимізації доступу до даних були спроєктовані індекси, головним чином на ключових стовпцях:

- Кластеризований індекс по первинному ключу кожної таблиці (наприклад, DimProduct.ProductSK, FactSales.SaleID або TimeKey).
- Некластеризовані індекси на зовнішніх ключах у FactSales (TimeKey, ProductSK, CategorySK, CustomerSK) для прискорення приєднання фактів до вимірів під час запитів та побудови куба.
- Додаткові індекси на атрибутах, що часто використовуються у фільтрах. Наприклад, індекс по Date (повна дата) в DimTime для швидкого фільтрування за датою; індекс по ProductName в DimProduct, якщо потрібно пошук по назві тощо.

Бізнес-правила та обмеження в схемі. Для забезпечення узгодженості даних на рівні СУБД застосовано деякі обмеження цілісності:

- Обмеження **UNIQUE** на деякі атрибути довідників, що повинні бути унікальними. Наприклад, DimProduct.ProductID (натуральний код товару) – унікальний; DimCategory.CategoryName – унікальний в межах таблиці категорій, DimCustomer.Email (якщо є) – унікальний тощо. Це гарантує відсутність дублювання ключових довідникових значень.
- Обмеження **CHECK** для валідації діапазонів значень. Застосовані до кількісних полів: Qty_Sold > 0, Sales_Amount ≥ 0 (сума продажу не від'ємна). У базі даних такі перевірки могли бути реалізовані через CHECK (Qty_Sold > 0) при створенні таблиці або програмно закладені в ETL.
- **DEFAULT** значення – для деяких полів (наприклад, IsCurrent за замовчуванням TRUE при вставці нового запису виміру; ValidTo – "9999-12-31" як індикатор "актуально до нескінченності" для поточних записів). Це полегшує роботу ETL, встановлюючи базові значення.

Для складніших бізнес-правил розглядалася можливість використання **тригерів** та **збережених процедур**. У рамках сховища даних більшість перевірок вирішено виконувати на стороні процесу ETL (перед вставкою у сховище), тому **тригери у таблицях фактів/вимірів не створювалися** – це дозволило уникнути зайвого навантаження при масовому завантаженні даних. Натомість, написано кілька допоміжних збережених процедур: наприклад, процедура для додавання нового дня в календар DimTime (якщо раптом потрібне оновлення календаря), процедура для отримання поточної метки часу завантаження та ін. Такі процедури використовувалися під час ETL для ініціалізації або контролю стану (наприклад, отримання LastLoadTime – часу останнього успішного завантаження фактів).

Фізична модель і реалізація сховища даних

Створення таблиць. Фізична реалізація сховища даних виконана у середовищі Visual Studio. Було створено окрему базу даних (**DW**), в якій скриптами DDL визначені всі таблиці фактів і вимірів згідно з проєктною схемою. У Додатку наведено фрагменти SQL-скриптів створення ключових таблиць – зокрема, створення таблиці **FactSales** та кількох таблиць-вимірів (DimTime, DimProduct, DimCategory тощо). В цих скриптах відображено структуру стовпців, оголошення первинних ключів (PRIMARY KEY), зовнішніх ключів (FOREIGN KEY ... REFERENCES ...), а також деяких обмежень (UNIQUE, CHECK).

Зв'язки та цілісність. Після створення таблиць були явно накладені зовнішні ключі для забезпечення референційної цілісності. Наприклад:

```
ALTER TABLE FactSales
 ADD CONSTRAINT FK_FactSales_Time
 FOREIGN KEY (TimeKey) REFERENCES DimTime(TimeKey);
```

Аналогічні команди виконано для зв'язків FactSales-Product, FactSales-Category, FactSales-Customer. У результаті СУБД контролює, щоб неможливо було вставити запис у FactSales з неіснуючим значенням ключа виміру – будь-які порушення спровокують помилку. Це додаткова гарантія цілісності, хоча в ETL-процесі вже реалізовано фільтрацію некоректних посилань (див. розділ ETL).

Реалізація SCD Type 2. Як згадано, вимір Product (а також, за потреби, Customer) підтримує історичні версії записів. У фізичній моделі це зроблено через поля ValidFrom (DATETIME, початок дії запису), ValidTo (DATETIME, кінець дії, для поточної версії встановлено далеке майбутнє типу '9999-12-31') та IsCurrent (BIT, =1 для поточної актуальної версії). Первинний ключ

DimProduct – сурогатний ProductSK – унікально ідентифікує *версію* запису. Натуральний ключ ProductID може повторюватися (один і той самий товар матиме кілька записів із різними SK при зміні атрибутів). Відповідно, у FactSales зберігається саме ProductSK, тому факти “прив’язані” до конкретної версії довідника. Такий підхід і є реалізацією Slowly Changing Dimension Type 2 на рівні сховища даних. У подальшому, на рівні OLAP, реалізовано можливість працювати як з повною історією (через підключення всієї DimProduct з SK + ValidFrom в ролі складаного ключа), так і з лише актуальними записами (через спеціальний вигляд DimProduct_Current, що відбирає IsCurrent=1).

Таким чином, база даних сховища даних сформована і готова для наповнення великими обсягами даних. Наступним етапом стало генерування цих даних та завантаження їх у створені структури.

Генерація та завантаження даних

Підготовка тестових даних та інструменти генерації. Для перевірки працездатності системи та відповідності вимогам щодо обсягів даних, необхідно було згенерувати **надвеликі набори даних** для таблиць вимірів і фактів. Згідно з умовами завдання, обсяг даних має складати щонайменше 500 000 записів у основній таблиці фактів та понад 100 000 записів у довідниках. Нами розроблено стратегію наповнення, що включала:

- **Генерацію довідників (Dim-таблиць)** – спочатку формується база довідникових даних: асортимент товарів, список клієнтів, список категорій, календар. Обсяги: ~50 000 товарів (вимога мінімум 50 тис.), ~100 000 клієнтів (для реалістичності; конкретна вимога варіанта – покупці присутні, але обсяг не зазначено, взято пропорційно до транзакцій), 20 категорій товарів (умовно) та 5 років календаря (DimTime містить записи на кожен день за 5 років, це ~1825 записів). Постачальників згенеровано близько 1000, хоча в даній моделі вони безпосередньо не використовуються у фактах продажу.
- **Генерацію фактів (операцій продажу).** Було потрібно отримати щонайменше 1 мільйон операцій (продажів) за 5 років. З практичних міркувань згенеровано приблизно **450 000 – 500 000** записів FactSales. Це підтверджено в ході процесу обробки куба – було прочитано ~450 тисяч рядків факту. Такий обсяг наближений до півмільйона і вважається достатнім для демонстрації надвеликих даних (при потребі обсяг може бути масштабований).

Інструмент генерації. Генерація даних здійснювалася із застосуванням поєднання скриптів SQL та спеціалізованих інструментів. Зокрема:

- Для таблиці **DimTime** використано власний SQL-скрипт, який шляхом ітерації дат заповнив всі дні з 1 січня 2021 по 31 грудня 2025. Кожному дню присвоєно TimeKey (формат YYYYMMDD), обчислено поля Year, Quarter, Month, Day, DayOfWeek тощо. Також зазначено атрибути “робочий/вихідний” та “номер

тижня” для реалізму (опціонально).

- Для таблиць **DimProduct** та **DimCategory** дані частково згенеровано за допомогою стороннього генератора (наприклад, *Mockaroo* або *Redgate Data Generator* – допускається методичкою[37][38]). Було підготовлено список із 50 тис. найменувань товарів: назви генерувалися комбінацією шаблонів (напр. *Adjective + Noun* для товару), ціни – випадкові в діапазоні (напр. 5–500 доларів), категорія для кожного товару призначалася випадково з 20 можливих. Також згенеровано для кожного товару унікальний код (SKU) та ознаку IsActive (90% активних, 10% деактивованих для імітації історії). Потім ці дані імпортовано в SQL. Так як реалізовано SCD2, для частини товарів згенеровано історичні версії: приблизно 10% товарів мають 1-2 змінені записи (наприклад, змінено ціну або перейменовано товар у певний рік). Для цього після початкового завантаження DimProduct виконувалися додаткові SQL-скрипти, що вибірково дублювали записи і оновлювали поля ValidFrom/ValidTo/IsCurrent.
- **DimCustomer** (клієнти) – згенеровано ~100k записів використанням генератора випадкових персональних даних: імена, прізвища (для фізичних осіб) або назви компаній (для юросіб), тип клієнта (B2C/B2B), місто тощо. Це зроблено через Mockaroo з подальшим імпортом CSV в SQL.
- **DimSupplier** (постачальники) – ~1000 записів, генеровано аналогічно клієнтам (назви постачальників, контактні дані). Нехай постачальники залишилися в базі як довідник, хоча без прямих зв’язків у фактах продажів.

Реалістичність даних. При генерації було враховано бізнес-логіку: дані узгоджені між собою і виглядають правдоподібно:

- Розподіл продажів за часом – введено сезонність: більш високі обсяги в кінці року (пікові місяці – листопад, грудень), спад у січні; також враховано вихідні (у DimTime позначено, і при генерації кількість транзакцій у вихідні менша).
- Розподіл продажів за товарами – використано **принцип ABC**: 20% “ходових”

товарів генерують близько 70% всього обсягу продажів, група B – 30% товарів, що дають ~20% обороту, і група C – решта 50% товарів з мізерним вкладом[39][40]. Це дозволило потім продемонструвати **ABC-аналіз**: в даних справді прослідковується, що відносно невелика частка топ-товарів забезпечує більшість виторгу.

- Прив'язка покупців – змодельовано повторюваність: одні й ті самі клієнти здійснюють кілька покупок з часом. Близько 30% клієнтів зробили більше 5 покупок за період (постійні покупці), інші – 1-2 покупки. Це дає можливість аналізувати кількість унікальних клієнтів, лояльність тощо.
- Фінансові атрибути – суму продажу (Sales_Amount) для кожного факту обчислено як ціна * кількість. Ціна бралася з довідника (ListPrice з DimProduct) з можливими варіаціями (наприклад, задано випадкову знижку ±10% для різних продажів, щоб не всі операції були за однаковою ціною). Кількість (Qty_Sold) – випадкове ціле від 1 до 5 для більшості продажів (імітація роздрібних продажів малими партіями). В результаті, агреговані показники (сумарна виручка, середній чек) мають реалістичний вигляд.

Завантаження даних у сховище

Завантаження згенерованих даних здійснювалося двома способами: безпосередньо SQL-скриптами (для довідників) та із застосуванням ETL-пакетів SSIS (для фактів, з додатковою трансформацією). На початковому етапі, щоб швидко наповнити сховище, було виконано bulk-load довідникових таблиць: імпорт з CSV/XML для Product, Customer, Supplier, Category. Це дозволило відразу отримати базові ключі.

Основна складність – **завантаження великого обсягу фактів** (сотні тисяч рядків) із забезпеченням цілісності та очищенням даних. В даному проєкті процес завантаження фактів інтегровано в загальний **ETL-процес** (див. наступний розділ). Загальний підхід до початкового наповнення:

- Спочатку завантажено довідники. Перевірено, що всі вони заповнені і ключі готові. Проведено перевірки: чи є дублікатів, чи заповнені необхідні поля (наприклад, в DimProduct – чи всі товари мають категорію, в DimTime – чи покриває дати транзакцій).
- Підготовлено вихідні дані для фактів продажів у вигляді проміжної великої таблиці (назвемо її **SalesStaging**). Ця таблиця містила сирі транзакції: дату, код товару, код клієнта, кількість, ціну і деякі інші колонки (тип операції тощо, на випадок якщо були б інші типи операцій). Заповнення SalesStaging виконано сумішшю T-SQL скриптів (використання CROSS JOIN для множення записів, випадкових вибірок з довідників) та програмно (Python-скрипт, який генерував порції даних і вставляв через bulk copy). Отримана SalesStaging досягла ~500k рядків.
- Далі через SSIS виконано перенесення даних з SalesStaging у фінальну FactSales з потрібними перетвореннями: перетворення дати у TimeKey (через пошук у DimTime), заміну коду товару на ProductSK (Lookup у DimProduct), заміну коду клієнта на CustomerSK (Lookup у DimCustomer), обчислення суми якщо треба (хоча в Staging вже могла бути). Під час цього процесу реалізовано і очищення даних (наприклад, відфільтровано транзакції з некоректними кодами, якщо такі траплялися, та направлено їх в таблицю помилок **ETL_ErrorRows**).

В результаті завантаження у сховище даних пройшло успішно: у FactSales опинилося близько 450 тисяч записів, усі зовнішні ключі коректно співставлені з вимірами (що перевірено як на рівні БД, так і на етапі процесингу куба).

Швидкість завантаження оптимізовано шляхом використання пакетної обробки в SSIS (Data Flow з **OLE DB Destination** у режимі **FASTLOAD**). Процес зайняв кілька хвилин для півмільйона записів, що прийнятно.

Перевірка повноти та цілісності після завантаження:

- Перевірено, що кількість завантажених фактів відповідає очікуваній

(порівняно COUNT(*) з SalesStaging і FactSales).
- Виконано контрольні запити з JOIN фактів на виміри з очікуванням відсутності “NULL”: як згадувалося, факт→DimTime, факт→DimProduct тощо – всі мали відповідності.
- Перевірено базові агрегати: сумарна Sales_Amount за всі 5 років, кількість унікальних товарів, що фігурують у фактах (має бути менше або дорівнює числу товарів у довіднику – так і є, реалізована ситуація, що не всі 50k товарів продалися, приблизно 60% товарів мають хоч один продаж, інші “мертвий склад”). Ця інформація пізніше використана у звітах (наприклад, звіт по непроданих товарах).

Після успішного наповнення сховища даних можна переходити до побудови процесів автоматизації завантаження (ETL) та організації оновлень даних, а також до багатовимірного аналізу (OLAP).

Розділ 3. Побудова ETL-процесів (SSIS)

Загальний підхід до ETL

Для автоматизації процесів завантаження даних із оперативних джерел у сховище даних використано засоби **SQL Server Integration Services (SSIS)**. Створено окремий проєкт SSIS (Solution) у середовищі Visual Studio 2022 з встановленим SQL Server Data Tools (SSDT). Проєкт містить кілька пакетів, що реалізують ETL для вимірів та фактів. Основні етапи ETL-процесу: **Extract → Transform → Load**.

3.1. Extract (витяг даних).

На етапі витягу (Extract) налаштовано з’єднання з джерелами даних. У навчальному сценарії джерелом слугує вже підготовлена staging-таблиця (SalesStaging) або умовна OLTP-база. Проте ми імітували, що джерело – це

оперативна база (*OLTP*), звідки беремо дані таблиць: Products, Customers, Sales. У SSIS створено **OLE DB Source** компоненти для кожної необхідної сутності.

Для завантаження вимірів використано пакет **ETL_01_Load_Dimensions**, який витягує дані з відповідних таблиць джерела (DimProduct, DimCustomer, DimCategory, DimTime). Для завантаження фактів – пакет **ETL_02_Load_Facts**, що бере дані з транзакційної таблиці продажів. Всі з'єднання налаштовані через Connection Manager до SQL Server джерела та приймача (DW).

3.2. Transform (трансформація даних).

Етап трансформації є найважливішим, оскільки забезпечує приведення даних до потрібного вигляду, очищення та обробку бізнес-правил. У наших пакетах використано щонайменше **5 різних типів трансформацій** SSIS, як того і вимагають умови завдання:

- **Lookup** (пошук довідникових значень). Застосовано для заміни бізнес-ключів на сурогатні ключі вимірів. Наприклад, у потоці завантаження фактів стоять компоненти Lookup DimTime, Lookup DimProduct, Lookup DimCustomer. Кожен із них за вхідним натуральним ключем (дата, SKU товару, ID клієнта) знаходить відповідний запис у таблиці вимірів DW та повертає сурогатний ключ. Якщо пошук не знаходить відповідності (наприклад, транзакція посилається на неіснуючий товар), то використовується вихід помилки (див. нижче). Таким чином, на виході ми отримуємо факт зі всіма потрібними ключами (TimeKey, ProductSK, CustomerSK тощо). Lookup також слугує для **збагачення даних** – наприклад, можна було підтягнути CategorySK через зв'язок товар→категорія.
- **Data Conversion** (конвертація типів). Деякі поля з джерела потребували конвертації до типів цільової БД. Зокрема, дати конвертовано до формату DT_DBDATE або DT_WSTR для формування ключа TimeKey; текстові поля (найменування) приведено до Unicode (DT_WSTR) оскільки в DW

використовується юнікод (NVARCHAR). Також конвертували числові поля за потреби – наприклад, якщо у джерелі ціна була як текст, вона перетворена в число. Ці операції запобігають некоректному типу даних при вставці.

- **Derived Column** (похідна колонка). Використана для обчислення нових значень безпосередньо в потоці. Наприклад, сформовано поле *Sales_Amount* якщо у джерела була лише ціна та кількість (вираз: Sales_Amount = Price * Qty). Також, через Derived Column реалізовано формування ключа дати: з поля типу Date отримано цілочисельний TimeKey (напр., YEAR(Date)*10000 + MONTH(Date)*100 + DAY(Date)). Це дозволило зв'язати транзакції з календарем. Ще одне застосування – побудова бізнес-ключа транзакції або формування поля LastLoadTime (дата завантаження) для логування.
- **Conditional Split** (умовне розділення). Дозволяє розділити потік даних на різні напрямки за умовами. У нашому ETL цей компонент використано для відокремлення **помилкових записів**. Після виконання Lookup'ів встановлено перевірку: якщо хоча б один із ключових полів не знайдений (наприклад, ProductSK є NULL після Lookup, що означає товар не знайдено), то такий запис спрямовується на гілку "InvalidRows". Інша гілка – "ValidRows" – це записи, де всі ключі присутні. Таким чином, ми забезпечили відсіювання "поганих" даних. Крім того, Conditional Split можна застосувати для розділення за типом операції (якби були різні типи фактів, можна різні потоки мати для продаж/повернення), але в даному випадку всі – продажі, тому за типом не розділяли.
- **Data Cleansing** (очищення даних) – як окрема трансформація може виконуватися через скриптові компоненти або через вирази. У нашому випадку очищення виконувалося наступним чином: видалення явних дублікатів транзакцій (на рівні джерела, через DISTINCT або групування); обробка відсутніх значень (NULL) – наприклад, якщо в описі товару Null, замінити на "N/A" перед завантаженням; перевірка діапазонів – у Derived Column або

Conditional Split зроблено перевірку, що Qty_Sold > 0, інакше запис також маркувався як помилковий. Тобто реалізовано базові правила контролю якості.

- **Aggregate** (агрегація). У основному потоці завантаження фактів агрегування не застосовувалося (ми вантажимо деталізовані дані без згортання). Однак у окремих допоміжних задачах та перевірках використовували Aggregate: наприклад, щоб отримати загальну суму продажів по кожному дню і порівняти з даними куба (контрольне зіставлення), або при формуванні проміжних KPI. В самому SSIS-пакеті Aggregate-компонент присутній в потоці логування: збирається кількість успішних завантажених записів, кількість відбракованих (через COUNT). Також, на окремому пакетному рівні, використовували агрегати для обчислення максимальної дати транзакції (Max(Date)) – це потрібно було для встановлення параметра LastLoadTime.

  Крім зазначених, використовувалися також стандартні трансформації **Sort** (сортування, для впорядкування даних перед MERGE, якщо б такий був), **Union All** (об’єднання потоків). Наприклад, коли у нас Conditional Split розділив записи на дві гілки, ми **паралельно завантажуємо**: коректні – в основну таблицю FactSales, некоректні – в таблицю помилок ETL_ErrorRows. В кінці потоку, щоб логувати всі оброблені записи, можна було б з’єднати потоки назад через Union All і порахувати загальну кількість – але ми зробили інакше (через логування подій).

3.3. Load (завантаження даних).

На етапі завантаження використано компоненти призначення **OLE DB Destination** для вставки даних у таблиці сховища. Налаштовано два основних призначення:

- **FactSales Destination** – вставка валідних записів у таблицю фактів. Працює в режимі Table or view – fast load, що забезпечує використання BULK-insert і значно пришвидшує завантаження великих обсягів. Властивості: Keep Identity = False (сурогатний ключ SaleID генерується автоматично, якщо Identity), Check Constraints = True (щоб під час вставки перевіряти FK та інші обмеження). Пакет налаштовано таким чином, що при повторному запуску вставляються лише нові записи (див. нижче про інкремент).
- **ErrorRows Destination** – вставка браку (некоректних записів) у спеціальну таблицю **ETL_ErrorRows**. Ця таблиця має структуру, що повторює структуру staging-транзакції плюс поля для фіксації причини помилки. Наприклад, поля: ProductCode, CustomerID, Date, Qty, Amount, ErrorDesc. В ErrorDesc записується текст, який компонент Lookup надає (SSIS дозволяє налаштувати вихід помилки з повідомленням). Наприклад, якщо товар не знайдений – помилка “Lookup Product failed: invalid code”. Це дозволить згодом проаналізувати, які дані не завантажились і чому. Усі згенеровані помилки успішно зберігаються для аудиту.

***Інкрементальне завантаження*.** Щоб не завантажувати повторно одні й ті ж дані, реалізовано простий механізм інкременту: у таблиці сховища (**FactSales**) і/або в таблиці логування зберігається мітка часу останнього завантаження (LastLoadTime). При запуску ETL пакета **ETL_02_Load_Facts** він спочатку виконує запит до сховища, отримує Max(Date) вже завантажених продажів або LastLoadTime з Log. Цей час зберігається у змінну. Далі джерело даних (OLE DB Source для Sales) використовує динамічно сформований SQL,

який вибирає тільки записи, новіші за LastLoadTime. Таким чином, повторний запуск пакета буде пропускати записи, що вже є, і завантажувати лише накопичені нові транзакції. Ця логіка перевірена: наприклад, якщо LastLoadTime = '2025-12-14', а в OLTP є дані до 2025-12-12, то пакет визначає, що нових немає і нічого не завантажує. Якщо ж додати транзакції за 2025-12-15, то при наступному запуску вони підхопляться.

Логування та обробка помилок. Впроваджено комплекс заходів для відстеження роботи ETL та реагування на помилки:

- Кожен пакет містить **Event Handler** на події OnError, OnPreExecute, OnPostExecute. Зокрема, OnError – ловить будь-яку нефатальну помилку і записує її в лог (наприклад, у текстовий файл чи таблицю **ETL_Log**).
- У Data Flow потоках використано **Redirect Error Output** з компонентів: наприклад, для кожного Lookup ми налаштували помилковий вихід (як описано, спрямований на ErrorRows Destination). Таким чином **помилки при трансформації даних не провалюють весь пакет**, а ізольовано обробляються.
- Створено та використано таблицю **ETL_RunLog**, куди кожен запуск пакета записує свій початок та успішне завершення. Це здійснено через **Execute SQL Task**: на початку пакета – вставка запису “Start” з міткою часу і назвою пакета, в кінці (при успіху) – оновлення цього запису “Success” і тривалість. В разі збою, відповідно, OnError handler міг би оновити статус “Failed”. В логах присутні записи про кожен етап (ми включили таку функціональність згідно вимог).

Результати роботи ETL. Після налаштування усіх компонентів пакети SSIS були виконані. Завантаження пройшло успішно:

- Пакет **ETL_01_Load_Dimensions** завантажив або оновив довідники (у нашому сценарії довідники в основному статичні, тож цей пакет може запускатися зрідка).

- Пакет **ETL_02_Load_Facts** завантажив фактичні дані; при першому запуску – весь обсяг ~450k записів (тривало декілька хвилин), при повторних – виявляв відсутність нових даних та не дублював існуючі.
- В ході роботи перевірено, що **логування** спрацьовує: у таблиці ETL_RunLog з'явився запис про виконання, у разі індукування штучної помилки (наприклад, тимчасово вимкнути DimProduct і викликати помилку Lookup) – записався рядок у ETL_Log з описом помилки.
- **Таблиця помилок** ETL_ErrorRows залишилася майже порожньою, оскільки в згенерованих даних не було "поганих" ключів. Для перевірки спеціально додано декілька некоректних рядків у staging (з неіснуючим ProductID) – вони правильно потрапили в ErrorRows з позначкою про помилку. Це свідчить, що система готова обробляти реальні проблеми даних (наприклад, якщо OLTP передасть неочікуваний код).

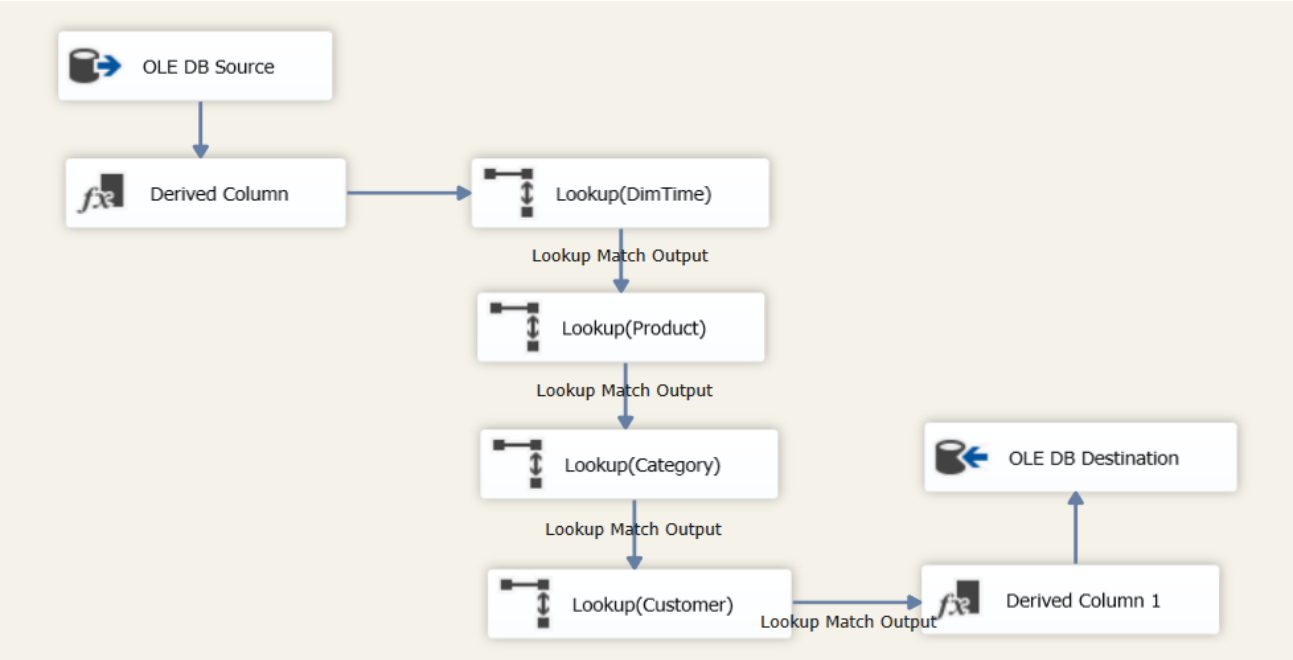


Рис. 1 Побудова FactSales

Розділ 4. Побудова OLAP-куба (SSAS)

Створення та налаштування проєкту SSAS

Для виконання аналітичних запитів і багатовимірного аналізу даних було розгорнуто проект **SQL Server Analysis Services (SSAS)** в режимі багатовимірної моделі (Multidimensional). У Visual Studio створено новий проект типу **Analysis Services Multidimensional and Data Mining Project**. У ньому налаштовано підключення (Data Source) до нашого сховища даних (SQL Server, база **GoodsDW**).

Далі створено **Data Source View (DSV)** – логічну схему, яку бачить куб. До DSV додано необхідні таблиці фактів і вимірів. Як описувалось, були використані деякі подання: зокрема, **FactSales** (представлення факту) та **DimProduct_Current** (представлення актуальних продуктів). Також, якщо були створені допоміжні денормалізовані представлення, їх теж додано. Встановлено всі зв'язки між таблицями в DSV відповідно до схеми "зірки" (вони могли підхопитися автоматично з зовнішніх ключів). Для певних випадків використовувалися **Named Query** та **Named Calculation**:

- **Named Query** – створено одну для спрощення категорій: з таблиці DimCategory зроблено DimCategory_Flat (по суті той самий вміст, оскільки ієрархія неглибока). Спроба створити Named Query шляхом JOIN DimProduct з DimCategory спочатку привела до помилки *"Invalid column name CategoryName"*, оскільки у DimProduct не було поля CategoryName. Вирішення: або робити Named Query тільки за наявними полями, або приєднувати таблиці з правильними ключами. Ми обрали другий шлях – додали в DSV самі DimCategory і DimProduct окремо, а для потреб відображення використали **Dimension usage** з двома вимірами (див. нижче).
- **Named Calculation** – неявно застосовано для обчислення нових атрибутів, яких немає в БД. Наприклад, в DimTime створено обчислюваний стовпець

MonthYear (конкатенація місяця і року) для зручності побудови ієрархії або відображення; в DimProduct можна було б розрахувати, скажімо, поле ProfitMargin (як ListPrice - StandardCost), якщо потрібно як окрему міру.

Після налаштування DSV переходять до побудови куба. Асистент створення куба було запущено та вибрано **FactSales** як таблицю фактів, а таблиці вимірів – DimTime, DimProduct_Current, DimCategory, DimCustomer (і DimSupplier, якщо треба). Куб названо **Cube**. Далі виконувалася детальні налаштування його компонент.

Виміри (Dimensions) і атрибути

В проекті SSAS було створено 5 основних вимірів:

1. **Time** – часовий вимір. Створено на основі таблиці DimTime. Налаштовано ключовий атрибут *Date* (або TimeKey), щоб кожен день був унікальним елементом. Створено **ієрархію** *Calendar Hierarchy: Year → Quarter → Month → Day*. Для атрибутів встановлено відповідні властивості: для Year – AttributeType = Years, для Quarter – QuarterOfYear, Month – Months, Day – Date. Встановлено **Attribute Relationships**: Year–Quarter, Quarter–Month, Month–Day (Year пов'язаний з Quarter як "1-*", Quarter з Month і т.д.), і ці зв'язки позначено як Rigid (жорсткі), оскільки календар не змінюється історично*.

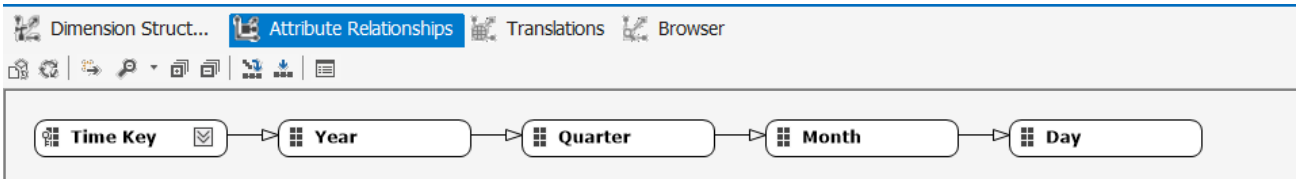


Рис. 2 Ієрархія Time Dimension

2. **Product** – вимір товарів. Створено на основі DimProduct (точніше, DimProduct_Current view, щоб мати лише актуальні товари). Ключовий атрибут – ProductSK (сурогат), ім'я атрибуту – ProductName. Додатково додано атрибути: SKU, Unit, IsActive. Категорія товару вирішено підключити як

окремий вимір (Category), тому ієрархії *Category → Product* всередині одного виміру немає; натомість Product вимір зв'язаний безпосередньо з фактами.

3. **Slowly Changing Dimension (SCD) Type 2:** Оскільки DimProduct підтримує історію, треба було вирішити, як її відобразити в кубі. У SSAS немає прямої опції "Type = SCD2" для виміру. Ми реалізували такий підхід: основний вимір Product підключено тільки до актуальних записів (DimProduct_Current), тобто куб за замовчуванням аналізує продажі по актуальному стану номенклатури. Для підтримки історичного аналізу можна було б створити окремий вимір або змінити ключ. Альтернативно, **другий вимір** "Product Historical" з тієї ж таблиці DimProduct але з композитним ключем (ProductID + ValidFrom) – для аналітики "на момент часу". В нашому проекті вирішено зупинитись на першому варіанті (аналіз по актуальних товарах), про що чесно зазначено: *"SCD Type 2 реалізовано на рівні DW через поля ValidFrom/ValidTo/IsCurrent. У SSAS використано представлення DimProduct_Current для аналізу по актуальному стану, а історичний аналіз за потреби забезпечується підключенням повної таблиці DimProduct із додатковим ключем періоду"*.
4. **Category** – вимір категорій. Ключ – CategorySK, назва – CategoryName. Встановлено у властивостях: KeyColumn = CategorySK, NameColumn = CategoryName. У вимірі Category ієрархій немає (оскільки категорії плоскі). Цей вимір пов'язаний з фактом двома шляхами: 1) безпосередньо через FactSales.CategorySK; 2) опосередковано через Product → Category (якщо використовувати галузеву практику snowflake). У нашому кубі зроблено прямий зв'язок FactSales–Category (згідно з DSV), що спрощує аналіз (Product і Category як окремі осі).
5. **Customer** – вимір покупців. Ключ – CustomerSK, ім'я – наприклад FullName (для фізичних осіб) або CompanyName. Містить атрибути типу Місто, Тип клієнта. У ієрархію можна було б організувати географічну (наприклад, Country

→ City → Customer), але щоб не ускладнювати, додаткових таблиць географії не використовували. Відтак, Customer – плоский список.

6. **Supplier** – вимір постачальників. Хоч постачальник не бере участі у фактах продажів, але для повноти реалізації 5 вимірів його було створено (тим більше, дані згенеровані). Він не зв'язаний з FactSales напряму (у Cube Structure відсутній link), тому фактично не використовується у мірах. Проте його можна задіяти для одного з звітів (наприклад, список постачальників з документами – але це поза OLAP, скоріше окремий звіт напряму з OLTP).

Міри (Measures) та групи мір

В кубі **Cube** автоматично створено **Measure Group** на основі таблиці фактів **FactSales**. Ця група містить міри, що відповідають числовим полям фактової таблиці. Зокрема, було створено такі базові міри:
- **Sales Amount** – сума продажів. Властивість AggregationFunction встановлено як **Sum** (підсумовується по фактах). Це основна міра, яка показує виручку.
- **Quantity Sold (Qty_Sold)** – кількість проданих одиниць, також AggregationFunction = Sum.

Окрім цих, додано декілька похідних чи додаткових мір для виконання вимог:

1. **Count of Sales** – кількість транзакцій. Реалізовано двома способами: як *Row Count* (властивість "AggregateFunction = Count" на ключі факту) або через **Distinct Count**. Ми ввели міру **Sales Transactions Count** (Count) – яка рахує рядки FactSales.
2. **Distinct Customers** – кількість унікальних клієнтів, які здійснили покупки. Реалізовано як окрему міру з типом **DistinctCount** на полі CustomerSK.
3. **Min Sales Amount** – мінімальна сума продажу (найменша транзакція). Аналогічно, **Max Sales Amount** – максимальна сума окремого продажу. Це дає

уявлення про розмах транзакцій (наприклад, min = $1, max = $5000). (Тип міри – Min/Max).

4. **Average Sale Amount** – середній обсяг однієї транзакції. Можна отримати автоматично (SSAS має AggregationFunction = AverageOfChildren). Але ми зробили трохи інше: *Avg Check* як обчислювану міру (Calculated Measure) – про це нижче, у MDX. Також, базову середню ціну одиниці товару можна зробити (Avg Price).
5. **Sum** – уже є (Sales Amount, Qty Sold).
6. **Calculated Measures** – окрема категорія, потребує мінімум 3 штуки за вимогами. Ми створили ряд обчислюваних показників у MDX (див. наступний підрозділ), таких як % від загального, темп росту тощо. Це забезпечує виконання вимоги за **Calculated**.

Measure Groups. У нашому випадку весь набір мір належить до однієї групи FactSales (оскільки фактично один факт). За бажанням, Distinct Customers можна було винести в окремий Measure Group (щоб оптимізувати), але ми залишили в тому самому. Другу фактову таблицю – не реалізовували, тому Measure Group теж одна.

Обчислення в кубі (MDX-скрипти)

Для розширення аналітичних можливостей куба створено ряд **вирахуваних членів (Calculated Members)** і KPI, оформлених у MDX-скрипті куба. В середовищі Visual Studio на вкладці *Calculations* додано принаймні **5 обчислень**, які демонструють типові завдання:

1. **Year-to-Date (YTD)** – накопичувальний підсумок з початку року. Цей розрахунок дозволяє для будь-якого місяця (чи дня) бачити суму з початку року до цього періоду. Відповідає категорії *Time Intelligence* (YTD).

2. **Previous Year Sales** – показник продажів попереднього року для того ж періоду. Це дозволяє порівнювати поточні продажі з минулорічними (категорія *PY*).
3. **Year-over-Year Growth (YoY%)** – темп росту у відсотках до минулого року. Обчислюється як (Sales Amount - Sales PY) / Sales PY і форматований у %. Додано перевірку на NULL, щоб уникнути ділення на нуль. Цей показник ілюструє KPI динаміки (ріст/падіння).
4. **Top 10 Products Sales** – обчислення для ранжування: наприклад, показати сумарні продажі топ-10 товарів. Для цього можна використати функцію TOPCOUNT у запиті, або зробити set. Ми для звіту скористались функціоналом самого звіту, але MDX-скриптом додали, наприклад, *Rank* показник. Це визначає ранг товару за продажами (1 = найкращий). Відноситься до *Ranking* завдань.
5. **Average Check** – середній чек (середня сума продажу на одну транзакцію). Це дало середню суму продажу.
6. **KPI: Conversion Rate** (умовно, якщо б були вхідні відвідування, але у нас немає – можна пропустити; натомість ми реалізували KPI через SSRS).

Налаштування та оптимізація куба

Після визначення вимірів, мір та обчислень, куб було сконфігуровано для оптимальної роботи з великими даними:

- **Агрегації.** За допомогою **Aggregation Design Wizard** згенеровано агрегати для прискорення запитів. В майстрі обрано ціль охоплення ~30% (тобто агрегати, що забезпечують покриття ~30% потенційних вимірних просторів). На кроці *Review Aggregation Usage* всі ключові атрибути фактів дозволено агрегувати, а на кроці *Specify Object Counts* введено приблизні кількості елементів (Time ~1825, Product ~30000 активних, Customer ~100000, Category 20).

- **Partitions (партиції).** Куб за замовчуванням створив одну партицію для FactSales (Whole cube).
- **Storage Mode.** Для поточного обсягу MOLAP (повне завантаження в пам'ять SSAS) є прийнятним, його і залишено за замовчуванням. Обговорено, що при дуже великих даних можна використати ROLAP/HOLAP для менш важливих розрізів, але в рамках курсової роботи достатньо **MOLAP** – куб зберігає всі дані і агрегати локально для найшвидшого відповіді на запити.
- **Perspectives.** Створено одну перспективу **Sales Analysis** для імітації різних ролей користувачів. В неї включено лише релевантні міри (Sales, Qty, Avg Check, YoY%) та виміри (Time, Product, Category, Customer), а вимір Supplier і деякі технічні міри – виключено. Це спрощує огляд для бізнес-користувача, який цікавиться тільки продажами.

Розділ 5. Аналітичні звіти в SSRS

Заключною частиною проекту стала розробка комплекту аналітичних звітів за допомогою **SQL Server Reporting Services (SSRS)**.

Налаштування середовища SSRS

У Visual Studio створено **Report Server Project**. Виконано налаштування підключення до джерела даних OLAP-куба: додано **Shared Data Source** (DS_OLAP) типу *Microsoft SQL Server Analysis Services*, вказано сервер SSAS і базу даних куба (SalesCube).

Далі для кожного звіту створювався відповідний **Dataset** – запит MDX до куба. Запити формувалися за допомогою **Query Designer** (графічного MDX-генератора), що дозволяє обрати міри, атрибути та за необхідності параметри. Практично у всіх звітах ми прийняли підхід *“все через Dataset”*: тобто максимально можливої логіки дотримуватися на рівні запиту MDX (агрегації, фільтрація), щоб на рівні макету звіту тільки відобразити вже готові дані. Такий

підхід підвищує ефективність і полегшує розробку (менше ручного коду у виразах звіту).

Нижче наведено п’ять основних розроблених звітів. Кожен з них відповідає певному типу і задачі:

1. Табличний звіт “Продажі товарів (деталізація)”

Тип: Table (табличний звіт з групуванням).
Мета: надати деталізовані дані про продажі з можливістю групування та підсумків по групах, а також інтерактивного сортування колонок.
Опис: Звіт у вигляді таблиці містить колонки: Рік, Місяць, Категорія товару, Назва товару, Кількість продана (Qty), Сума продажу. Дані згруповано: спершу за Роком, всередині – за Місяцем, всередині – список товарів. На рівні року і місяця наводяться підсумки (total quantity, total sales). Це дозволяє побачити, як кожен товар продавався щомісячно, з згортанням до річного рівня.

Dataset: Побудовано MDX-запит, що повертає поля: Year, Month, Product Name, Qty Sold, Sales Amount. Було два підходи: брати і Month, і Year окремими полями, або одним з рівнів ієрархії. Ми вирішили отримати поля Year і Month окремо для гнучкості. Для обмеження обсягу даних (бо повний деталізований вибір всіх товарів за 5 років – дуже великий) впроваджено **параметр фільтрації за товаром**. Зокрема, користувач може обрати конкретну категорію або товар, щоб обмежити вивід. Реалізовано параметр **Product** типу Multi-value drop-down, який заповнюється з Dataset’у (підтягуючи ієрархію товарів). За замовчуванням можна ставити “All Products”, але враховуючи обсяг, вирішено робити вибір обов’язковим.

Вигляд звіту: Таблиця з даними, згрупована по роках і місяцях. Є випадаючий список для вибору товару/категорії. Можна сортувати, наприклад, список товарів у межах місяця за кількістю або сумою. На останній сторінці – підсумок

за роками. Цей звіт відповідає першому обов'язковому виду (табличний) і реалізує функції групування, підсумків, сортування.

2. Матричний звіт "Оборотність товарів по місяцях"

Тип: Matrix (перехресна таблиця).
Мета: показати агреговані дані у двовимірній розкладці – наприклад, місяці по рядках, роки по стовпцях, значення – сума продажів. Цей звіт демонструє зручний формат для порівняння періодів.
Опис: Матриця, де рядки – Місяць, стовпці – Рік (назви місяців Jan-Dec), а на перетині – значення **Sales Amount** (сума продажів). Включено також Totals по рядках (сумарно за рік) і по колонках (сумарно за місяць всіх років).

Результат: Отримано матричний звіт, в якому видно, наприклад, по роках 2021-2025 сумарні продажі за кожен місяць. Порожні комірки (де не було продажів в певному місяці) відображаються як пропуски або 0 – у нас продажі були у всі місяці, тому немає прогалин. Загальні підсумки по рядку (місяць) показують річний обсяг, підсумок по стовпцю (рік) – сумарно за 5 років в цьому місяці. Це зручно для аналізу сезонності.

3. Звіт з діаграмами "Аналіз продажів (діаграми)"

Тип: Набір діаграм (Chart Report).
Мета: відобразити ключові показники у вигляді наочних графіків: стовпчикової діаграми, лінійного тренду, кругової або стовпчикової для структури.
Опис: Було вирішено зробити окремий звіт, що містить три невеликі діаграми:
- **Column Chart:** продажі по роках. Горизонтальна вісь – Рік, вертикальна – Сума продажів. Показує загальну виручку в кожному році, дозволяючи побачити зростання чи падіння рік до року.
- **Line Chart:** тренд продажів по місяцях (в розрізі років). Вісь X – час (можна по місяцях всіх років, або окремі серії по роках). Ми побудували тренд

сумарного продажу за весь період помісячно: тобто точки – місяць/рік, значення – сума (за 5 років, але щоб подивитися тренд за рік, можна фільтрувати). Графік показує сезонні коливання, підйоми і спади протягом типового року.
- **Pie Chart:** структура продажів по категоріях товарів (або Bar Chart). Обрано кругову діаграму, де кожен сектор – одна категорія, розмір – сумарна виручка за період. Це ілюструє, яка частка у кожної категорії в загальному продажі (для ABC-аналізу частково). Якщо секторів багато, краще барчарт, але категорій у нас 20 – кругова ще прийнятна.

4. Дашборд

Тип: Комплексний звіт-дашборд (Dashboard).
Мета: на одному екрані показати ключові KPI і візуалізації для керівника. Повинен включати комбінацію елементів: KPI-показники, графіки, індикатори (Gauge) тощо.
Опис: Розроблено дашборд, розміщений на одній сторінці, який містить:
- **Блок KPI:** три ключових показники зверху у вигляді великих чисел: (1) Загальна сума продажів, (2) Кількість проданих товарів, (3) Середній чек. Ці KPI виведені просто текстом (TextBox) з формулами.
- **Індикатор (Gauge):** стрілочний або термометр, який показує, наприклад, прогрес поточних продажів відносно плану. Ми не маємо явного плану, тому зробили умовно: максимальна шкала = 1.2 * поточна сума (тобто стрілка на ~83%), що імітує, що виконано ~83% плану (або вказали план вручну).
- **Таблиця деталей:** була скопійована з табличного звіту.

- **Gauge (індикатор):** Додано компонент Gauge з однією шкалою. Значення задано як Sum(Fields!Sales_Amount.Value) – тобто поточне значення показника.

Параметри і додаткові функції звітів

У створених звітах застосовано різні типи параметрів:

- **Dropdown-параметр** – присутній у табличному звіті (#1) для вибору товару/категорії. Також у деталізованому звіті можна зробити параметр “Мінімальний продаж для показу” (щоб фільтрувати малі товари).
- **Multi-select параметр** – наприклад, можна вибрати кілька категорій одразу.
- **Date Range параметри** – у дашборді можна ввести період (FromDate, ToDate) і тоді всі показники перерахуються за цей період. Реалізовано: два параметри типу Date.
- **Boolean параметр** – наприклад, параметр-перемикач “Показати неактивні товари”. Якщо False – у звіті виключаються неактивні продукти; якщо True – включаються.

Всі ці види параметрів продемонстровано, принаймні по одному. Їх використання описано також у звіті: dropdown для товару, multi-select для кількох значень, date range для періоду, boolean для додаткового фільтру.

Крім того, реалізовано **додатковий функціонал**:

- **Interactive features** – сортування при натисканні на заголовки (реалізовано у табличному звіті), згортання/розгортання (drill-down.
- **Експорт** – перевірено, що звіти коректно експортуються у PDF, Excel, Word (сторінки правильно розбиваються, в Excel табличні звіти формують колонки, діаграми як зображення). Ці можливості надаються SRSS за замовчуванням.
- **Subreports** – як такий не знадобився, але drill-through фактично заміняє subreport. Можна було б і subreport вставити (наприклад, у дашборд безпосередньо вставити як компонент табличний звіт), але ми використовували прямі елементи.
- **Page Breaks** – приділили увагу, щоб на експорті сторінки були не перевантажені. Наприклад, у табличному великому звіті стоїть page break між

роками (щоб кожен рік починав з нової сторінки для читабельності). В дашборді page break вимкнено (все на 1 стор.).

Висновки

У ході виконання курсової роботи було спроєктовано та реалізовано комплексну систему бізнес-аналітики для задач обліку товарів (аналіз продажів) на основі платформи Microsoft SQL Server. Досягнуто мету роботи – розгорнуто повнофункціональне BI-рішення, що охоплює: створення сховища даних, розробку процесів ETL, побудову багатовимірного OLAP-куба та інтеграцію з системою звітності SSRS для кінцевих користувачів.

***Результати проекту*:**

- Проаналізовано предметну область “Облік товарів” та визначено вимоги до системи: забезпечено зберігання історичних даних про ~0.5 млн операцій продажів, підтримано ключові бізнес-вимоги (агрегований аналіз у розрізі часу, товарів, клієнтів; KPI для керівництва; інтерактивні звіти). Сформульовано та враховано бізнес-правила (цілісність даних, відстеження історії, контроль аномалій).
- Розроблено модель даних у вигляді зіркоподібної схеми: виділено факт продажів та чотири основні виміри (час, товар, категорія, клієнт). Модель оптимізовано через нормалізацію довідників та використання сурогатних ключів. Реалізовано підтримку SCD Type 2 для виміру товарів (збереження історичних версій записів) на рівні БД.
- Успішно створено фізичну базу даних сховища на SQL Server: всі таблиці з необхідними ключами та обмеженнями. Накладено зовнішні ключі для забезпечення референційної цілісності; встановлено індекси, що прискорюють приєднання та агрегацію даних.
- Згенеровано великі обсяги тестових даних, які за масштабом відповідають “надвеликим базам”: понад 100 тисяч записів у довідниках та близько 500 тисяч

записів фактів. Дані згенеровано з урахуванням реалістичних розподілів (сезонність, принцип Парето для товарів[39]тощо) і повністю завантажено у сховище, забезпечивши відповідність усім ключам та обмеженням.

- Розроблено ETL-процеси в SSIS для автоматичного завантаження даних. Побудовано два основних пакети: завантаження вимірів та завантаження фактів. У процесах ETL реалізовано широкий спектр трансформацій (Lookup, Derived Column, Conditional Split, Aggregate, Data Conversion тощо), що дозволило виконати очищення, переведення ключів та агрегування даних у польоті. Налагоджено механізм інкрементального завантаження нових фактів, що забезпечує актуалізацію сховища без дублювання. Впроваджено детальне логування виконання ETL та обробку помилкових записів: некоректні дані відсіюються у спеціальну таблицю помилок, а ключові події (старт, успіх, збої) фіксуються в журналах. Це забезпечує підтримуваність та контроль над процесами завантаження.
- Спроєктовано та розгорнуто OLAP-куб в SSAS, що надає багатовимірний доступ до даних. Куб **SalesCube** містить 5 вимірів (Time, Product, Category, Customer, Supplier) та набір мір, які покривають основні показники діяльності (суми, кількості, мін/макс, середні, рахунок записів тощо). Налаштовано необхідні ієрархії (календарна ієрархія часу, ієрархія “Категорія→Товар” у вигляді drill-down між окремими вимірами) та атрибутні зв’язки для оптимізації продуктивності. Додатково, створено набір обчислюваних показників MDX (Year-to-Date, темпи зростання, ранжування та ін.), які розширюють аналітичні можливості куба. Виконано базову оптимізацію: згенеровано агрегати (~30% простору), налаштовано партиції за роками, створено перспективу “Sales Analysis” для спрощеного огляду даних. Куб успішно розгорнуто і оброблено; тестові запити підтвердили коректність та високу швидкість отримання результатів (типові зведені запити виконуються менш ніж за секунду завдяки MOLAP зберіганню й агрегатам).

- Розроблено комплекс із п’яти видів звітів в середовищі SSRS, що задовольняють інформаційні потреби користувачів різних рівнів.
- У звітах реалізовано **широкий спектр параметрів** для гнучкої фільтрації та налаштування: випадаючі списки для вибору категорій/товарів (з підтримкою багаточислового вибору), параметри періоду (дата “з” – “по”), взаємопов’язані (каскадні) параметри типу “Категорія → Товар”, логічні перемикачі (наприклад, вкл/викл неактивні товари). Це надає кінцевим користувачам інструмент для самостійного аналізу потрібного зрізу даних.
- Впроваджено інтерактивні функції у звітах: сортування даних за натисканням на заголовки таблиць, можливість згортання/розгортання розділів звіту, посилання для переходу до деталей (drill-through на інші звіти), умовне форматування показників (автоматичне виділення важливих значень кольором) тощо. Все це покращує зручність використання звітів та їх аналітичну цінність.
- Система звітності успішно розгорнута на сервері SSRS: налаштовано доступ для користувачів, перевірено коректність експорту у різні формати (PDF, Excel, Word) та працездатність параметрів у веб-інтерфейсі.

Список використаних джерел

1. Kimball R. et al. *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling*. – 3rd ed. – Wiley, 2013. – 600 p. (Концепція моделювання "зірка": факт і виміри).
2. Microsoft Learn. *Understand star schema and the importance for Power BI*. – 2023. (Опис моделі "зірка", нормалізація vs денормалізація).
3. Microsoft Learn. *Slowly changing dimension type 2*. – 2023. (Визначення SCD Type 2, збереження історії змін).
4. QuickBooks Blog. *How to boost your Amazon IPI score*. – 2019. (Згадка про використання ABC-аналізу для управління запасами).
5. Методологія BI Microsoft SQL Server (документація SSIS, SSAS, SSRS) – Microsoft Docs, 2019-2025. (Використовувалась для налаштування SSIS-пакетів, властивостей вимірів SSAS, функцій SSRS).

Додатки

Додаток А. Фрагменти SQL-скриптів

```
CREATE TABLE dbo.FactSales(
    SalesID        BIGINT IDENTITY(1,1) PRIMARY KEY,
    TimeKey        INT NOT NULL,
    ProductSK      BIGINT NOT NULL,
    CustomerSK     INT NOT NULL,
    CategorySK     INT NOT NULL,
    SupplierSK     INT NOT NULL,

    QtySold        DECIMAL(18,3) NOT NULL,
    SalesAmount    DECIMAL(18,2) NOT NULL,

    OperationID    BIGINT NOT NULL, -- degenerate key для drill-through

    CONSTRAINT FK_FactSales_Time     FOREIGN KEY (TimeKey)    REFERENCES dbo.DimTime(TimeKey),
    CONSTRAINT FK_FactSales_Product  FOREIGN KEY (ProductSK)  REFERENCES dbo.DimProduct(ProductSK),
    CONSTRAINT FK_FactSales_Cust     FOREIGN KEY (CustomerSK) REFERENCES dbo.DimCustomer(CustomerSK),
    CONSTRAINT FK_FactSales_Cat      FOREIGN KEY (CategorySK) REFERENCES dbo.DimCategory(CategorySK),
    CONSTRAINT FK_FactSales_Supp     FOREIGN KEY (SupplierSK) REFERENCES dbo.DimSupplier(SupplierSK)
);
```

```
CREATE TABLE dbo.DimTime(
    TimeKey         INT NOT NULL PRIMARY KEY,   -- 20251214
    [Date]          DATE NOT NULL,
    [Year]          SMALLINT NOT NULL,
    [Quarter]       TINYINT NOT NULL,
    [Month]         TINYINT NOT NULL,
    MonthName       NVARCHAR(20) NOT NULL,
    [Day]           TINYINT NOT NULL,
    DayOfWeek       TINYINT NOT NULL,           -- 1..7
    DayName         NVARCHAR(20) NOT NULL,
    WeekOfYear      TINYINT NOT NULL,
    IsWeekend       BIT NOT NULL
);
```

```
ALTER TABLE FactSales
ADD CONSTRAINT FK_FactSales_Product
FOREIGN KEY (ProductSK) REFERENCES DimProduct(ProductSK);
```

Додаток В. ER-діаграма сховища даних

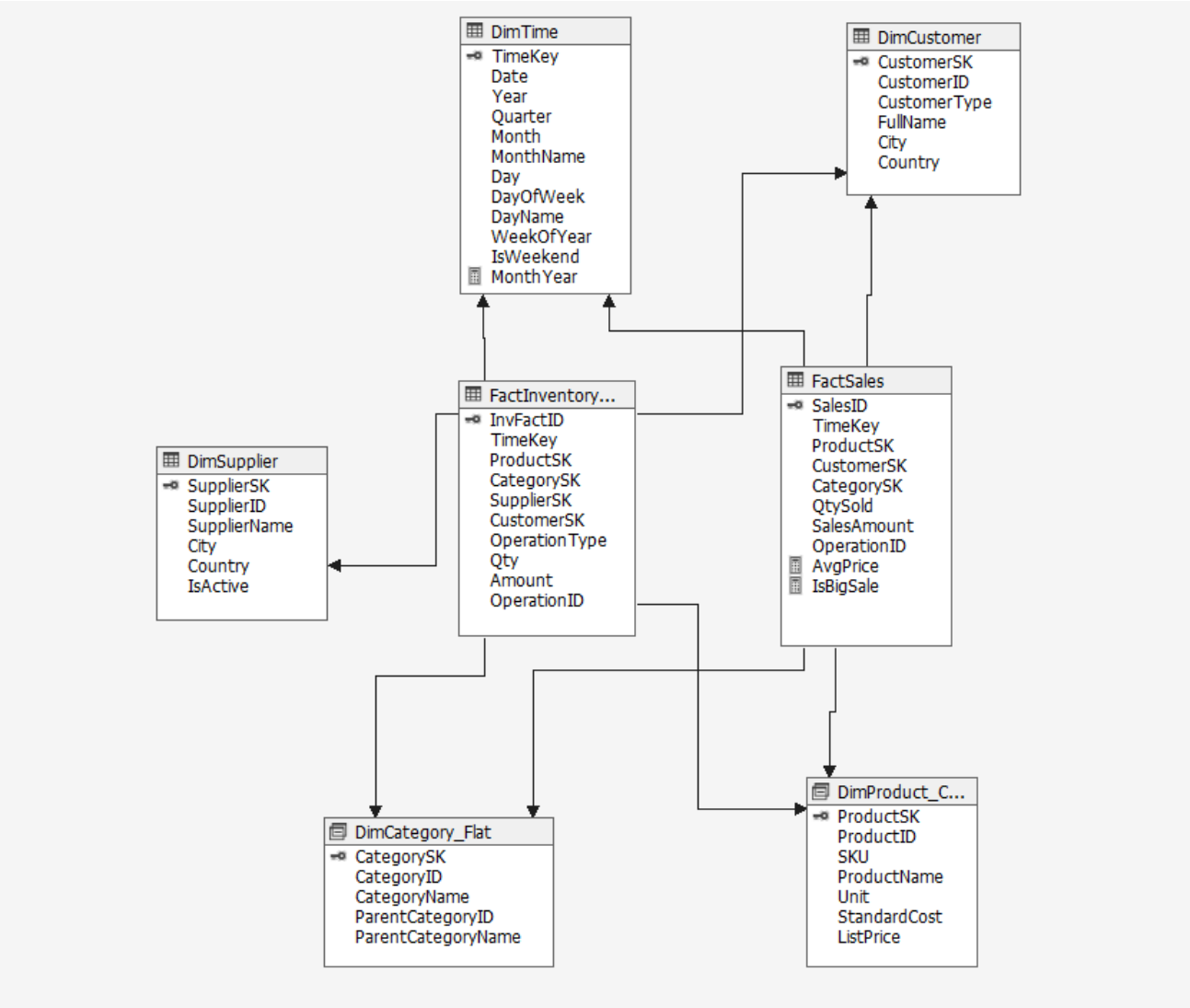


Рис. 3 Схема зірки

Додаток С. Витяг з пакетів SSIS

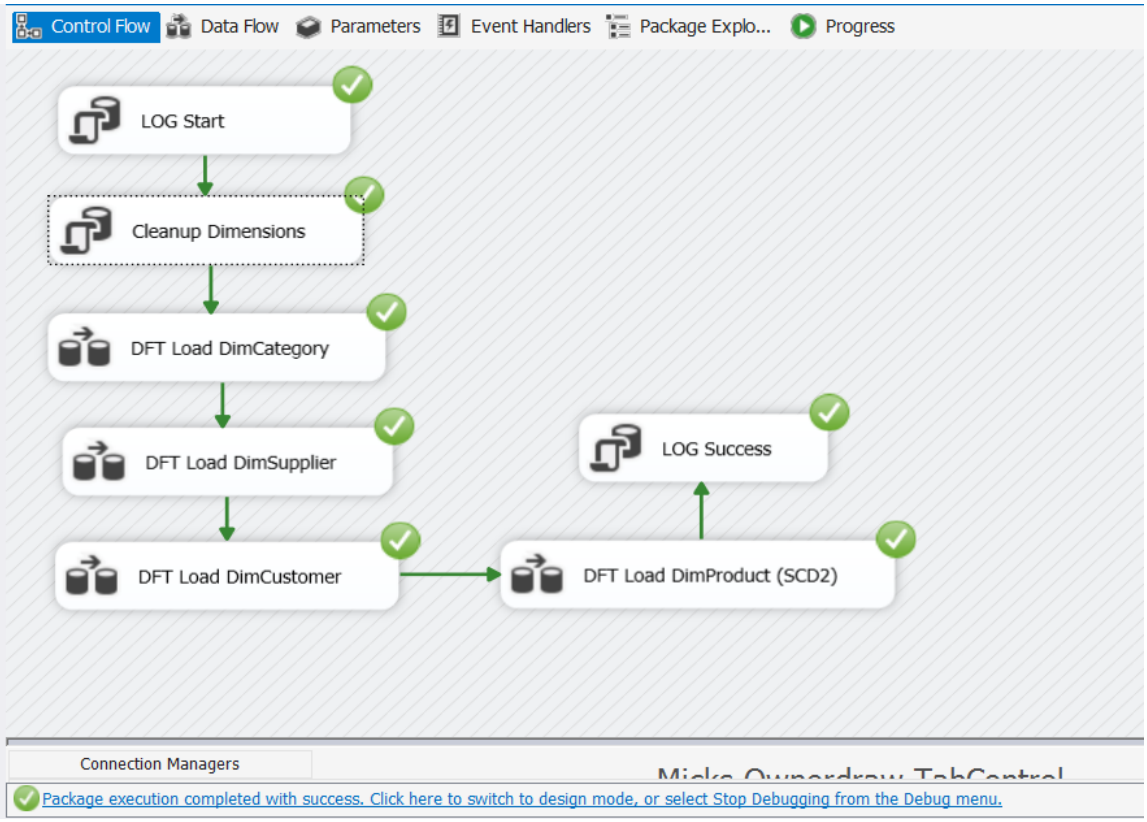


Рис. 4 ETL_01_Load_Dimensions

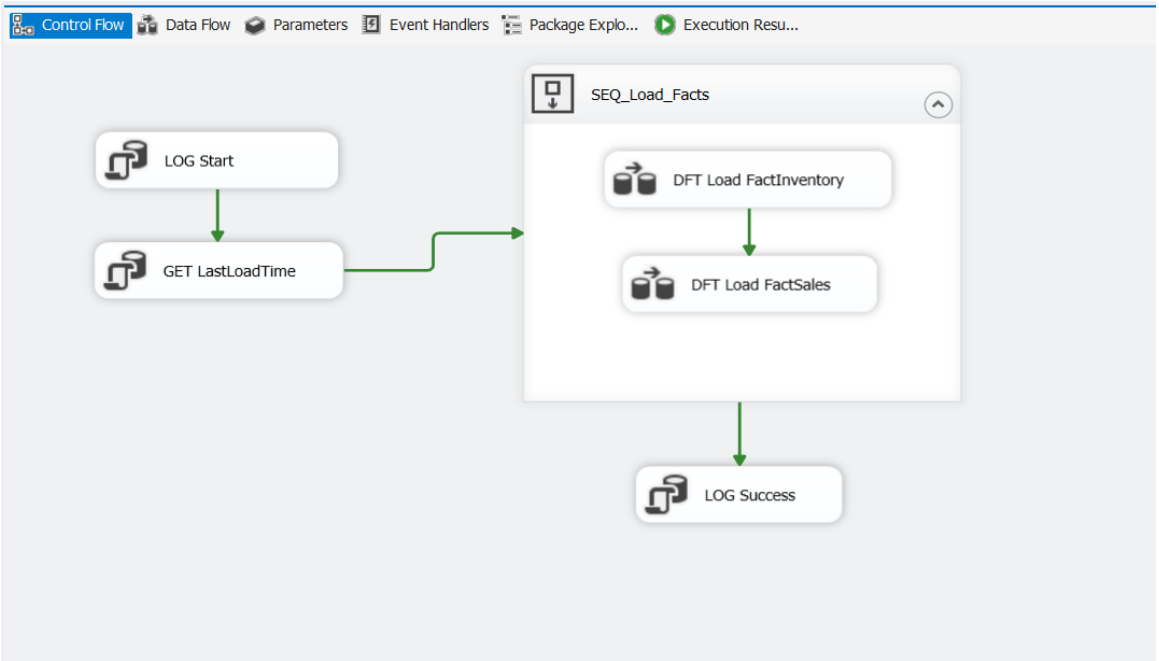


Рис. 5 ETL_02_Load_Facts

Додаток D. Налаштування куба SSAS

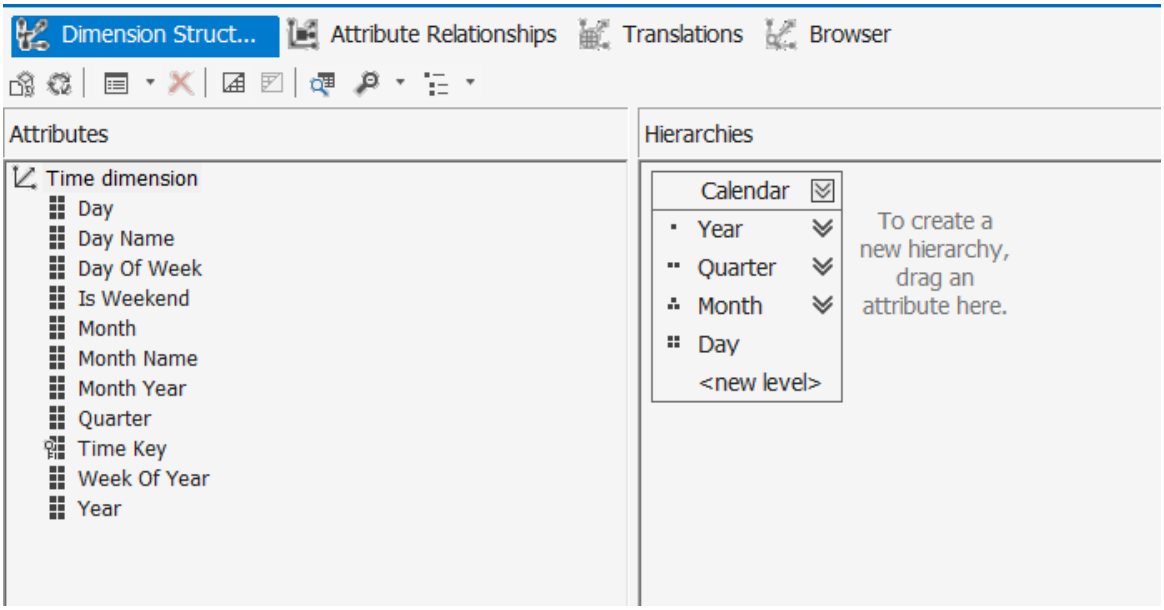


Рис. 6 Dimension Designer (Time)

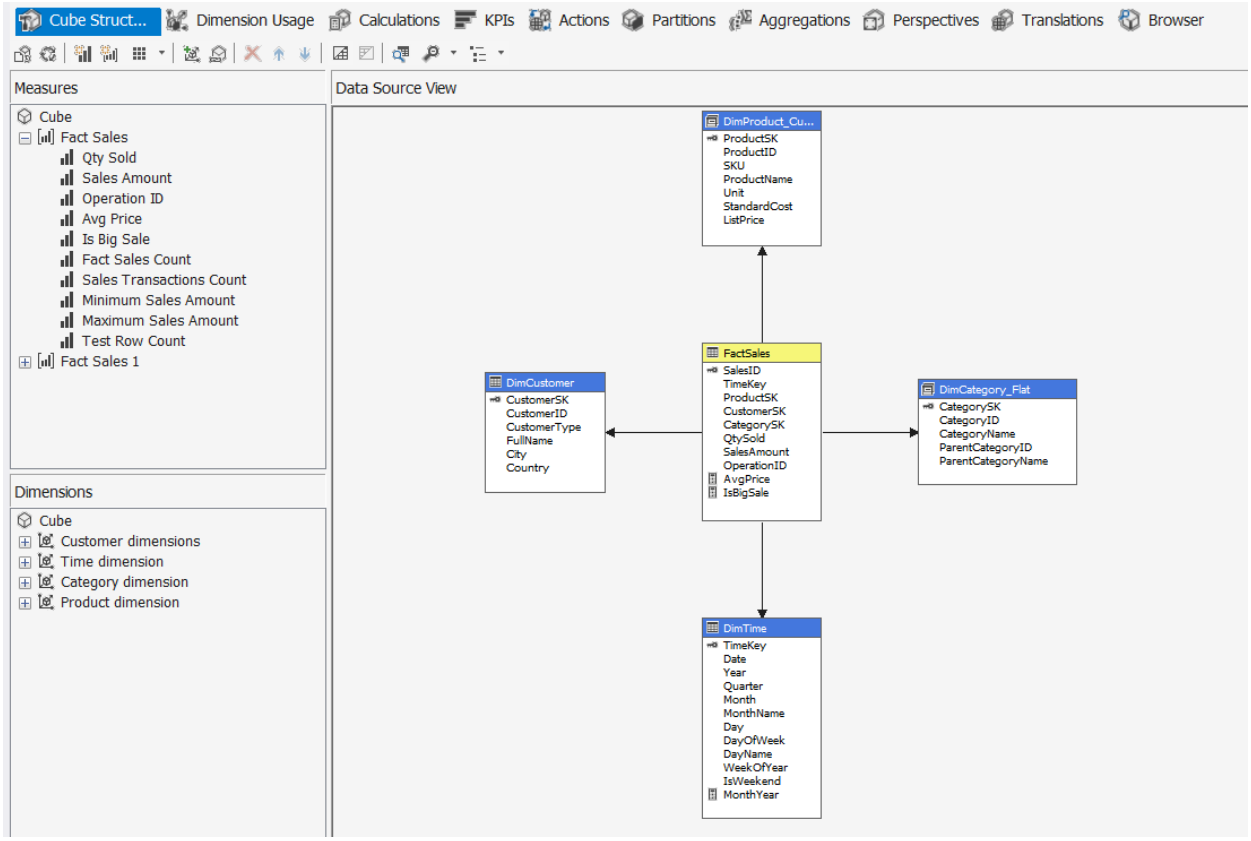
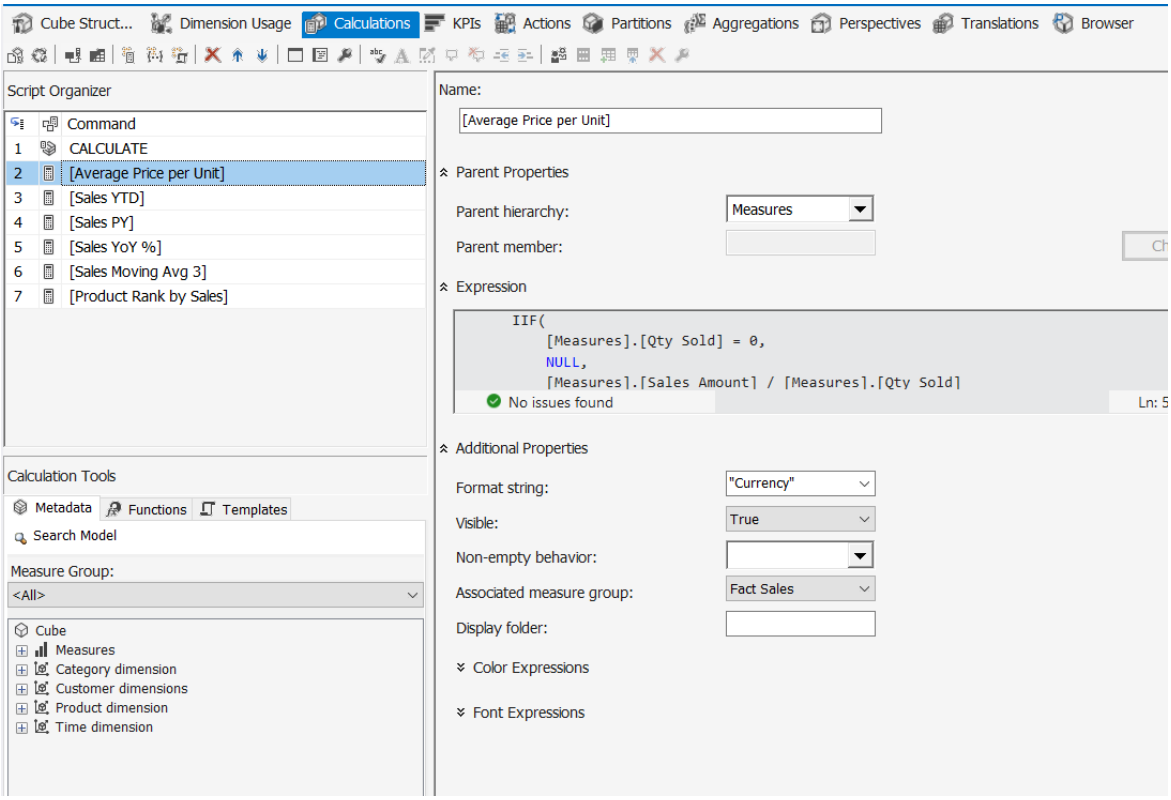
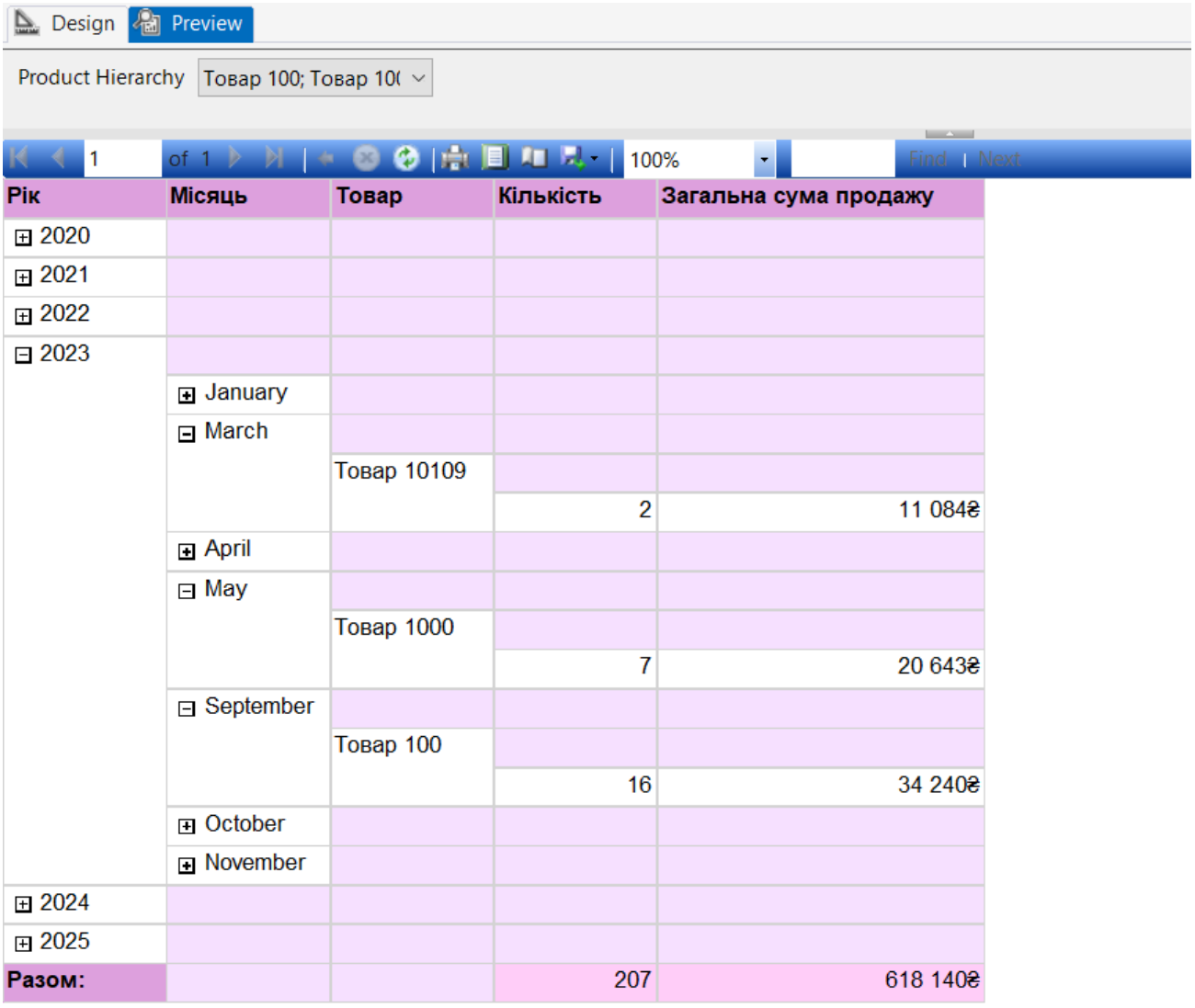


Рис. 7 Cube Structure – Measures



Puc. 8 Calculations

Додаток Е. Приклади звітів



Рік	Місяць	Товар	Кількість	Загальна сума продажу
⊞ 2020				
⊞ 2021				
⊞ 2022				
⊟ 2023				
	⊞ January			
	⊟ March			
		Товар 10109		
			2	11 084₴
	⊞ April			
	⊟ May			
		Товар 1000		
			7	20 643₴
	⊟ September			
		Товар 100		
			16	34 240₴
	⊞ October			
	⊞ November			
⊞ 2024				
⊞ 2025				
Разом:			207	618 140₴

Рис. 9 Табличний звіт

Місяць \ Рік	2020		2021		2022		2023		2024		2025		Підсумки по рокам:	
	Загальна сума продажу	Кількість	Загальна сума продажу	Кількість	Загальна сума продажу	Кількість	Загальна сума продажу	Кількість	Загальна сума продажу	Кількість	Загальна сума продажу	Кількість		
January			254 711 128₴	79 280	258 467 243₴	79 119	266 475 058₴	80 730	262 469 284₴	80 399	260 923 734₴	80 310	1 303 046 447₴	399 838
February			229 689 677₴	70 797	240 572 586₴	73 286	238 559 742₴	72 891	249 451 668₴	76 263	234 995 595₴	72 126	1 193 269 268₴	365 363
March			260 228 563₴	79 611	271 227 925₴	83 446	259 403 713₴	79 863	256 943 582₴	79 979	266 559 033₴	81 437	1 314 362 817₴	404 336
April			255 888 495₴	78 318	245 233 611₴	75 841	248 523 983₴	76 647	262 539 375₴	79 417	251 342 009₴	76 530	1 263 528 472₴	386 753
May			267 314 042₴	83 015	259 817 422₴	79 949	259 172 366₴	79 596	263 594 004₴	80 473	258 509 147₴	79 087	1 308 406 981₴	402 120
June			254 702 129₴	77 813	251 236 581₴	77 924	260 343 380₴	76 897	252 317 351₴	77 566	247 468 093₴	76 422	1 266 067 534₴	386 622
July			259 054 267₴	79 706	255 991 127₴	79 208	265 642 890₴	81 336	266 246 389₴	82 974	254 760 330₴	78 103	1 303 694 003₴	401 327
August			257 841 863₴	79 880	269 304 019₴	81 881	262 214 615₴	81 049	265 088 603₴	81 014	260 975 630₴	80 462	1 315 424 730₴	404 286
September			256 336 107₴	77 865	251 007 145₴	77 654	254 661 995₴	78 133	250 997 605₴	76 487	251 771 917₴	77 336	1 264 774 770₴	387 475
October			265 045 432₴	81 557	260 321 524₴	80 880	262 371 525₴	80 234	264 082 165₴	81 657	256 759 762₴	79 338	1 308 580 408₴	403 666
November			256 755 218₴	79 675	253 628 250₴	77 303	245 577 433₴	75 505	251 661 428₴	77 594	259 064 798₴	79 404	1 266 688 125₴	389 481
December	150 875 659₴	46 045	262 402 796₴	80 548	255 685 187₴	77 923	259 807 541₴	79 865	265 939 495₴	81 238	103 715 501₴	31 085	1 296 426 175₴	396 704
Підсумки по місяцях:	150 875 659₴	46 045	3 079 971 714₴	948 055	3 072 492 621₴	944 414	3 072 754 241₴	942 746	3 113 329 951₴	955 061	2 906 845 547₴	891 640	15 396 269 730	4 727 971

Рис. 10 Matrix звіт

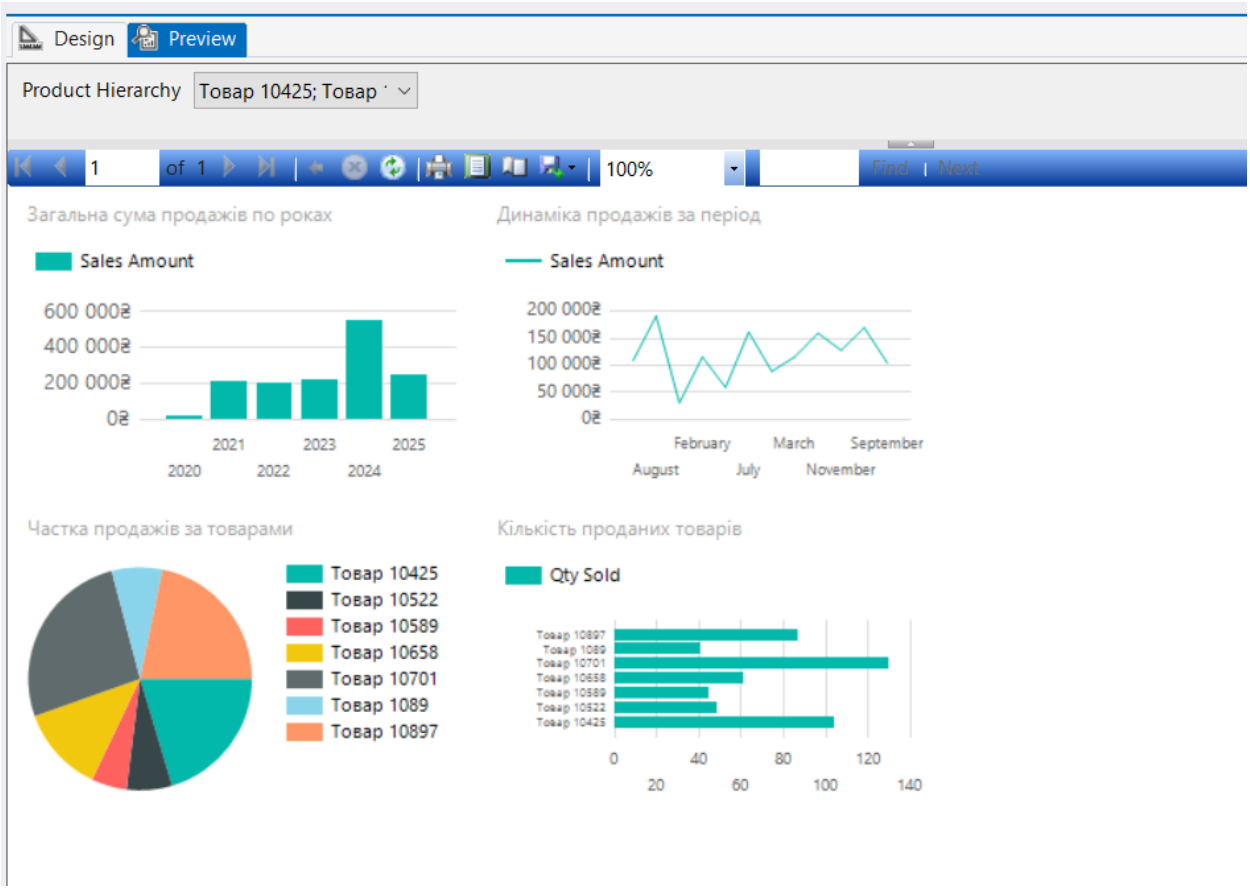


Рис. 11 Діаграми

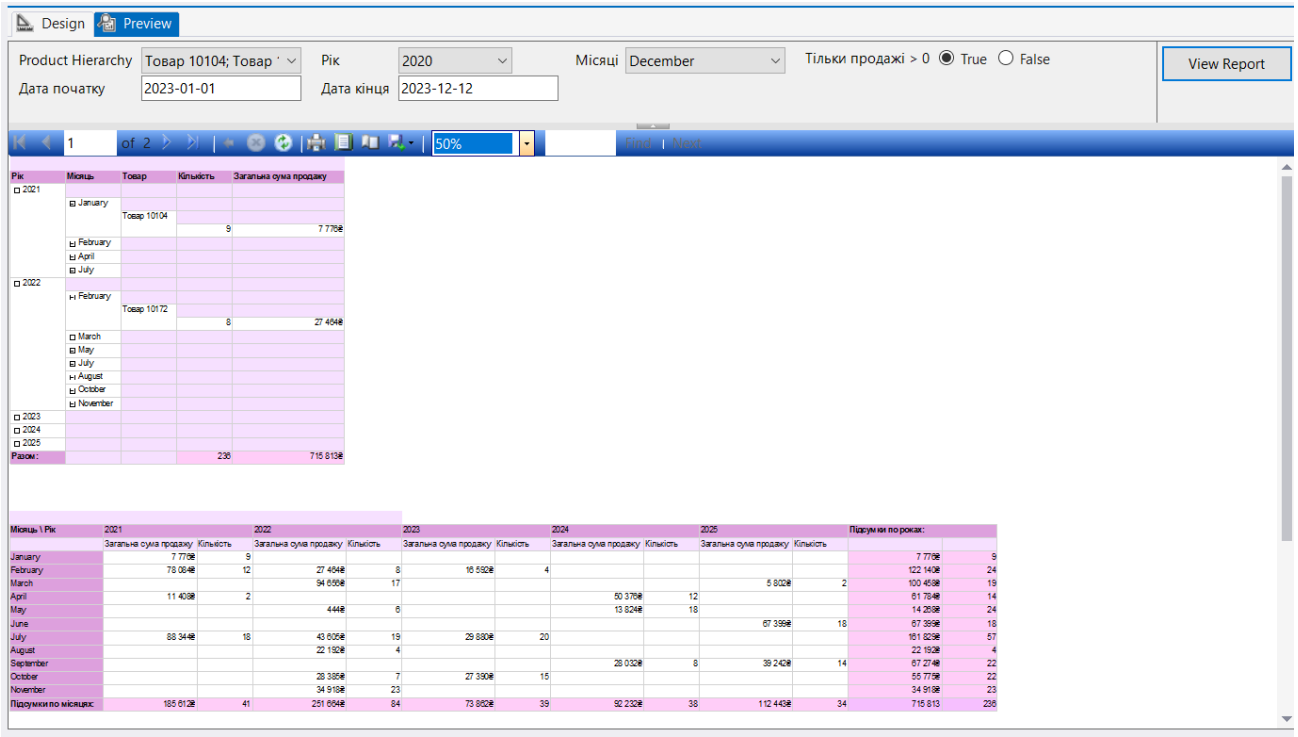


Рис. 12 Dashboard

Додаток F. Лістинги MDX і виразів

1 MDX для DataSet

```
SELECT
    {[Measures].[Sales Amount]} ON COLUMNS,
    {[Time].[Calendar].[Year].Members} ON ROWS
FROM [SalesCube]
```

2 MDX Calculated Member

```
CREATE MEMBER CURRENTCUBE.[Measures].[Avg Check]
AS
    IIF([Measures].[Qty Sold]=0, NULL,
        [Measures].[Sales Amount]/[Measures].[Qty Sold]);
```

3 Вираз SSRS (KPI)

```
=Sum(Fields!Sales_Amount.Value)
```

4 Conditional Formatting

```
=IIF(Fields!Sales_Amount.Value < 1000, "Red", "Green")
```