Міністерство освіти і науки України

Ніжинський державний університет імені Миколи Гоголя

Навчально-науковий інститут природничо-математичних, медико-біологічних наук та інформаційних технологій

Кафедра інформаційних технологій, фізико-математичних та

економічних наук

Освітня програма: Комп’ютерні науки

Спеціальність:122 Компʼютерні науки

**КВАЛІФІКАЦІЙНА РОБОТА**

на здобуття освітнього ступеня *бакалавр*

**Реалізація СУБД «Дрібнооптова торгівля продуктами харчування»**

студентки Хоменко Анастасії Романівни

**Науковий керівник:**

Фетісов Валерій Сергійович,

кандидат економічних наук, доцент

**Рецензент:**

Лисенко Ірина Миколаївна,

кандидат фізико-математичних наук, доцент

**Допущено до захисту:** \_\_\_\_ \_\_\_\_\_\_ 2024 р.

Завідувач кафедри

проф. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Казачков І.В.

Ніжин − 2024

ЗМІСТ

[ЗМІСТ 2](#_Toc168406900)

[ВСТУП 4](#_Toc168406901)

[РОЗДІЛ 1.](#_Toc168406902)[ХАРАКТЕРИСТИКА ТОРГІВЛІ 6](#_Toc168406903)

[1.1 Основні поняття 6](#_Toc168406904)

[1.2 Форми торгівлі 6](#_Toc168406905)

[1.2.1 Роздрібна торгівля 7](#_Toc168406906)

[1.2.2 Оптова торгівля 7](#_Toc168406907)

[1.2.3 Дрібнооптова торгівля 8](#_Toc168406908)

[1.3 Стан дрібнооптової торгівлі в Україні до війни та під час війни 9](#_Toc168406909)

[РОЗДІЛ 2.](#_Toc168406910)[АВТОМАТИЗАЦІЯ ДРІБНООПТОВОЇ ТОРГІВЛІ 11](#_Toc168406911)

[2.1 Роль баз даних в автоматизації дрібнооптової торгівлі 11](#_Toc168406912)

[2.2 Процеси дрібнооптової торгівлі, які підлягають автоматизації 11](#_Toc168406913)

[2.3 Програмні рішення для автоматизації процесів дрібнооптової торгівлі 12](#_Toc168406914)

[2.3.1 Характеристика BAS ERP 13](#_Toc168406915)

[РОЗДІЛ 3.](#_Toc168406916)[ПРОБЛЕМА ЗАХИСТУ ІНФОРМАЦІЇ В БАЗАХ ДАНИХ 15](#_Toc168406917)

[3.1 Основні загрози 15](#_Toc168406918)

[3.2 Заходи для забезпечення захисту інформації 15](#_Toc168406919)

[РОЗДІЛ 4.](#_Toc168406920)[ХАРАКТЕРИСТИКА СУБД MYSQL. КЛІЄНТ HEIDISQL 17](#_Toc168406921)

[4.1 Загальний опис MySQL 17](#_Toc168406922)

[4.2 Основні особливості MySQL 17](#_Toc168406923)

[4.3 Переваги MySQL 18](#_Toc168406924)

[4.4 Клієнт HeidiSQL 19](#_Toc168406925)

[РОЗДІЛ 5.](#_Toc168406926)[ПРАКТИЧНА РЕАЛІЗАЦІЯ 20](#_Toc168406927)

[5.1 Встановлення необхідного програмного забезпечення 20](#_Toc168406928)

[5.1.1 Встановлення серверу MySQL 20](#_Toc168406929)

[5.1.2 Встановлення HeidiSQL 20](#_Toc168406930)

[5.2 Концептуальне моделювання даних 21](#_Toc168406931)

[5.2.1 Позначення 22](#_Toc168406932)

[5.2.2 Побудова ER-діаграми 22](#_Toc168406933)

[5.2.3 Опис реляційної моделі 24](#_Toc168406934)

[5.3 Реалізація запитів, збережених процедур та тригерів в HeidiSQL 30](#_Toc168406935)

[5.3.1 Створення запитів 30](#_Toc168406936)

[5.3.1.1 Список всіх замовлень з інформацією про клієнтів і товари 30](#_Toc168406937)

[5.3.1.2 Контактні дані постачальників та товари, які вони постачають 31](#_Toc168406938)

[5.3.1.3 Страховий запас всіх продуктів 32](#_Toc168406939)

[5.3.1.4 Список продуктів, кількість яких на складі менша за страховий запас 33](#_Toc168406940)

[5.3.1.5 Список протермінованих продуктів на сьогодні 34](#_Toc168406941)

[5.3.1.6 Список оплачених замовлень 35](#_Toc168406942)

[5.3.1.7 Список неоплачених замовлень 35](#_Toc168406943)

[5.3.2 Створення збережених процедур 36](#_Toc168406944)

[5.3.2.1 Підрахунок загальної суми оплати замовлень за день, тиждень та місяць 37](#_Toc168406945)

[5.3.2.2 Переоцінка цін на товари з урахуванням коефіцієнта інфляції 38](#_Toc168406946)

[5.3.3 Створення тригера для моніторингу запасів на складі 40](#_Toc168406947)

[5.4 Надання прав користувачам бази даних 42](#_Toc168406948)

[5.4.1 Процес надання прав користувачам 42](#_Toc168406949)

[5.4.2 Опис створених користувачів та їхніх прав 43](#_Toc168406950)

[ВИСНОВКИ 46](#_Toc168406951)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 48](#_Toc168406952)

[ДОДАТОК А 50](#_Toc168406953)

ВСТУП

**Актуальність теми.** Дрібнооптова торгівля є невід'ємною складовою сучасної економіки, забезпечуючи перехід товарів від виробника до кінцевого споживача. Вона охоплює широкий спектр галузей, від роздрібної торгівлі на ринках та у магазинах до оптової дистрибуції товарів масового споживання.

Популярність дрібнооптової торгівлі визначається не лише величезним обсягом товарообігу, але й важливістю цього сектору для забезпечення споживчого попиту. Вона відіграє ключову роль у формуванні цінової політики, доступності товарів для споживачів та ефективності ланцюга постачання в цілому.

В сучасному світі, де конкуренція на ринку постійно зростає, автоматизація стає важливим елементом успішного функціонування дрібнооптової торгівлі. Швидкість та ефективність виконання операцій, висока точність та надійність обробки даних, а також можливість аналізу та стратегічного планування стають ключовими факторами, які визначають успіх у цій сфері. Автоматизовані системи управління дозволяють забезпечити оптимальне використання ресурсів, ефективно відслідковувати та контролювати процеси, а також оперативно реагувати на зміни в умовах ринку.

Отже, з огляду на значущість та розповсюдженість дрібнооптової торгівлі, вивчення та впровадження сучасних підходів до автоматизації її процесів має великий практичний і науковий інтерес.

**Об’єкт дослідження** – бази даних.

**Предмет дослідження** – використання баз даних для автоматизації процесів дрібнооптової торгівлі продуктами харчування.

**Метою** кваліфікаційної роботи є ознайомлення з можливими способами застосування баз даних для автоматизації дрібнооптової торгівлі та реалізація цієї задачі.

Відповідно до вказаного об’єкту, предмету і для досягнення поставленої мети визначені такі **завдання** роботи:

* проаналізувати технічну, спеціальну літературу відповідно до теми кваліфікаційної роботи;
* встановити необхідне програмне забезпечення;
* знайти різноманітні ефективні способи реалізації потрібної задачі з використанням баз даних, вивчити можливості знайдених доступних прикладних рішень та особливості роботи з такими базами даних;
* реалізувати задачу автоматизації дрібнооптової торгівлі продуктами харчування за допомогою клієнта HeidiSQL;
* зробити висновки щодо ефективності та переваг впровадження баз даних у процеси дрібнооптової торгівлі.

Робота має **практичне значення**, оскільки результати дослідження можуть бути використані малими та середніми підприємствами в галузі дрібнооптової торгівлі продуктами харчування для підвищення ефективності управління запасами, оптимізації процесів закупівлі, підвищення якості обслуговування клієнтів та забезпечення точності та надійності даних.

РОЗДІЛ 1

ХАРАКТЕРИСТИКА ТОРГІВЛІ

* 1. Основні поняття

У широкому значенні «торгівля» — це вид підприємницької діяльності, пов'язаний з купівлею-продажем товарів. Провідною функцією торгівлі є обмін результатів праці (продукції, послуг) на гроші, тобто реалізація виробленої споживчої вартості. При цьому масштаби та структура обміну стають дедалі складнішими за рахунок складності величезних мас товарів, продукції, послуг, у тому числі іноземного походження.

Ключовими учасниками в процесі торгівлі є продавець та покупець, які взаємодіють між собою для обміну товарами або послугами на певні умови. Продавець – це особа або організація, яка пропонує товари або послуги для продажу. Вони виступають в ролі постачальника, який надає доступ до товарів або послуг, що вони мають на продаж. Покупець, зі свого боку, є особою або організацією, яка купує товари або послуги від продавця. Вони представляють кінцевого споживача, який має певні потреби та прагне задовольнити їх, купуючи товари чи послуги на ринку. Продавці та покупці взаємодіють між собою на основі угоди, яка визначає умови торгівлі, такі як ціна, кількість товарів, терміни оплати та умови доставки. Ця взаємодія формує основу для функціонування ринку та сприяє обміну товарами та послугами в економіці.

* 1. Форми торгівлі

Розглядаючи питання класифікації торгівлі, важливим аспектом є розуміння різниці між внутрішньою і зовнішньою торгівлею, хоча вони мають схожу економічну суть – посередництво в реалізації товарного капіталу. Внутрішня торгівля виявляється найбільш значущою за обсягом, становлячи до 80% усієї валової продукції економічно розвинутих країн. Тому далі буде приділено увагу формам організації товарообороту на внутрішньому ринку, зокрема оптовій і роздрібній торгівлі.

Оптова та роздрібна торгівля – це два основних види торгівлі, які відрізняються за обсягом продажів, характером клієнтів та специфікою товарів, які продаються. Вони взаємодіють один з одним у ланцюгу постачання та виробництва, забезпечуючи ефективну торговельну інфраструктуру та задоволення потреб різних категорій споживачів.

* + 1. Роздрібна торгівля

Роздрібна торгівля орієнтована на продаж товарів кінцевим споживачам у невеликих кількостях та за роздрібними цінами. Підвидами роздрібної торгівлі є магазини, кіоски, інтернет-магазини, палаткові торговці та ін..

Роздрібна торгівля виконує ряд ключових функцій, серед яких: формування асортименту товарів і послуг, розподіл великих і середніх партій товарів на одиничні екземпляри, збереження товарів, їх розміщення у торгових залах та встановлення цін. Крім цього, роздрібна торгівля забезпечує надання інформації споживачам про товари за допомогою реклами, надає консультаційні послуги та здійснює продаж товарів кінцевим споживачам через торговельну мережу. Важливою функцією роздрібної торгівлі є також стимулювання збуту і просування товарів. Роздрібна торгівля здійснюється через роздрібних торговців – фізичних або юридичних осіб, які купують товар у виробника або оптового продавця та продають його кінцевому споживачеві

* + 1. Оптова торгівля

Оптова торгівля орієнтована на продаж товарів у великих обсягах та за оптовими цінами і в основному вона спрямована на інші бізнеси, які використовують ці товари для подальшого перепродажу або виробництва. Підвидами оптової торгівлі є торгові посередники, промислові торговці, торговельні агенти, гуртові покупці та ін..

B оптовій торгівлі, яка не має на меті, на відміну від роздрібної, чітко вираженої реалізації попередньо закуплених товарів у кінцеве споживання, поділ на продавців і покупців залишається доволі умовним. Оптові підприємства, придбавши партію товару, одночасно стають його продавцем. Зворотна аналогія простежується в діяльності виробничих підприємств – реалізація виготовленого товару супроводжується закупівлею товарів сировинного характеру (постачання).

Завдяки своїм функціям, оптова торгівля стимулює підприємницьку діяльність у всіх її аспектах, сприяючи активізації підприємницької ініціативи у всіх сферах, формах та видів. Вона також бере активну участь у формуванні виробничого асортименту, спрямовуючи зусилля на підвищення загального рівня якості виробів та послуг. Оптова торгівля організовує та інвестиційно забезпечує процес переміщення матеріальних благ з сфери виробництва, шляхом впровадження нових каналів збуту та удосконалення існуючих.

Окрім цього, вона переймає на себе частину комерційного ризику, пов'язаного з життєвим циклом товарів і послуг на ринку, тим самим сприяючи ефективнішому управлінню ризиками для різних учасників ринкових відносин. Також, оптова торгівля формує оптимальну концепцію логістичного забезпечення виробництва, що базується на максимізації прибутку та урахуванні комерційних інтересів інших суб'єктів ринку.

Паралельно оптова торгівля виконує ряд функцій, спрямованих на задоволення потреб роздрібних торговельних підприємств та індивідуальних споживачів. Ці функції включають в себе вивчення попиту та кон'юнктури ринку, трансформацію виробничого асортименту, концентрацію матеріальних благ для ритмічного постачання торговельних підприємств, зберігання товарів, організацію доставки товарів, кредитування покупців та інформаційне обслуговування клієнтів.

* + 1. Дрібнооптова торгівля

Після розгляду понять оптової та роздрібної торгівлі, наступним логічним кроком є розгляд різновиду оптової торгівлі – дрібнооптової торгівлі. Цей аспект торгівлі визначається не лише розміром операцій, а й специфікою взаємодії з кінцевими споживачами. Дрібнооптова торгівля, часто вважається проміжним етапом між оптовою та роздрібною торгівлею, де товари закуповують великими партіями від виробників чи оптовиків, але продають в невеликих кількостях кінцевим споживачам.

Дрібнооптова торгівля зазвичай виникає відповідно до змін в ринкових умовах та потреб споживачів, які змушують підприємства шукати нові шляхи оптимізації своїх постачальницьких ланцюгів. Цей вид торгівлі стає відповіддю на різноманітність та індивідуальність попиту, який потребує гнучкості та швидкості реакції. Наприклад, зростання попиту на різні категорії товарів або виникнення нових ринків споживання можуть вимагати більш диференційованого підходу до постачання. Також розвиток торгових каналів, таких як інтернет-торгівля, може стимулювати формування дрібнооптових ринків, які сприяють ефективнішій та швидкій доставці товарів до кінцевих споживачів.

* 1. Стан дрібнооптової торгівлі в Україні до війни та під час війни

До війни в Україні, дрібнооптова торгівля відігравала значну роль у економіці країни. Малі та середні підприємства, а також місцеві торгові точки були активними учасниками ринку, забезпечуючи населення товарами першої необхідності. Проте, з початком військових дій та загострення конфлікту, ситуація на ринку змінилася.

Під час війни, дрібнооптова торгівля зазнала серйозних труднощів через обмеження в обсягах поставок, руйнування інфраструктури та зменшення попиту через економічні та соціальні труднощі. Багато підприємств зазнали втрат, а деякі були змушені припинити свою діяльність.

Перспективи розвитку дрібнооптової торгівлі після війни в Україні є обіцяючими, але потребують комплексного підходу та впровадження сучасних технологій. Автоматизація торгівельних процесів, таких як управління запасами, замовленнями та обліком клієнтів, стане ключовим етапом у відновленні та розвитку цього сектору. Це дозволить підприємствам ефективніше управляти своєю діяльністю, підвищити рівень обслуговування та зробити бізнес більш конкурентоспроможним у складних умовах ринкової економіки.

РОЗДІЛ 2

АВТОМАТИЗАЦІЯ ДРІБНООПТОВОЇ ТОРГІВЛІ

* 1. Роль баз даних в автоматизації дрібнооптової торгівлі

Автоматизація дрібнооптової торгівлі тісно пов'язана з використанням баз даних. Бази даних відкривають широкі можливості для оптимізації дрібнооптової торгівлі.

По-перше, вони забезпечують зберігання докладної інформації про товари, включаючи їх характеристики, ціни та описи, що дозволяє ефективно керувати асортиментом.

По-друге, бази даних допомагають вести облік клієнтів, зберігаючи дані про них, їх контактну інформацію та історію покупок, що полегшує взаємодію з клієнтами та аналіз їх поведінки.

Важливою функцією є моніторинг запасів, який дозволяє в режимі реального часу відстежувати наявність товарів на складі та забезпечує можливість прогнозувати потребу в додаткових поставках.

Крім того, бази даних автоматизують обробку замовлень, відстежуючи їх статус, здійснюючи оплату та організовуючи доставку, що сприяє підвищенню ефективності бізнес-процесів.

Врешті решт, бази даних забезпечують засоби для аналізу даних та генерації звітів, що допомагає у прийнятті обґрунтованих стратегічних рішень та визначенні подальших кроків у розвитку підприємства.

* 1. Процеси дрібнооптової торгівлі, які підлягають автоматизації

Дрібнооптова торгівля продуктами харчування представляє собою складний процес, що включає в себе ряд етапів, які можна оптимізувати та автоматизувати за допомогою сучасних інформаційних технологій.

Першим кроком є управління запасами. Створення бази даних для відстеження руху продуктів у магазині дозволяє ефективно контролювати наявність товарів та аналізувати попит на них. Це допомагає у плануванні закупівель, зменшуючи ризик нестачі або надміру товарів.

Другим важливим аспектом є управління постачальниками. Розробка бази даних для зберігання інформації про постачальників, їх ціни, умови поставок, дозволяє оптимізувати процес замовлення продуктів. Автоматичний вибір оптимального постачальника на основі умов та цін сприяє ефективному здійсненню закупівель та зниженню витрат.

Додатково, варто звернути увагу на управління клієнтськими відносинами. Встановлення бази даних для зберігання інформації про клієнтів, їхні покупки та уподобання, дозволяє персоналізувати пропозиції для кожного клієнта. Це сприяє збільшенню лояльності клієнтів та зростанню обсягів продажів.

Крім того, важливим елементом є аналітика та звітність. Бази даних допомагають збирати та аналізувати дані про продажі, витрати та інші показники. Це дозволяє приймати обґрунтовані рішення щодо стратегій розвитку та забезпечує підвищення ефективності бізнесу в цілому.

* 1. Програмні рішення для автоматизації процесів дрібнооптової торгівлі

В сучасному світі підприємства все частіше звертаються до використання інформаційних технологій для оптимізації своїх бізнес-процесів. Автоматизація ділових операцій стає ключовим елементом у забезпеченні ефективності та конкурентоспроможності. В цьому контексті для дрібнооптової торгівлі, яка є складним процесом зі своїми унікальними потребами, важливо мати на увазі наявність програмних рішень, що можуть бути налаштовані під конкретні потреби та специфіку бізнесу. Існують різноманітні програмні платформи***,*** які надають широкий спектр функцій та можливостей для автоматизації та оптимізації процесів управління дрібнооптовою торгівлею.

Серед таких платформ найпопулярнішою в Україні наразі є BAS ERP (Business Automation Software Enterprise Resource Planning). Це інтегрована система, призначена для автоматизації управління та обліку на підприємствах різного масштабу та галузей діяльності. BAS ERP забезпечує ефективне управління бізнес-процесами, включаючи фінанси, логістику, виробництво, продажі, закупівлі, складські операції та інші аспекти діяльності підприємства.

* + 1. Характеристика BAS ERP

BAS ERP розроблена на основі технологій та рішень, що були впроваджені в системі 1С: Підприємство. Це дозволило створити продукт, який успадкував найкращі характеристики 1С, але водночас запропонував нові можливості та поліпшення. Основні зв’язки між BAS ERP та 1С: Підприємство включають кілька важливих аспектів.

По-перше, BAS ERP використовує технологічну платформу, подібну до тієї, що застосовується в 1С: Підприємство. Це забезпечує високу сумісність з існуючими рішеннями та дозволяє легко переходити з 1С на BAS ERP. Така спадкоємність технологій є ключовою перевагою, оскільки дозволяє підприємствам мінімізувати витрати та ризики, пов'язані з переходом на нову систему.

По-друге, розробники BAS ERP врахували багаторічний досвід впровадження та використання 1С: Підприємство в різних галузях. Це дало змогу створити продукт, який відповідає потребам сучасного бізнесу. Завдяки цьому, BAS ERP включає всі необхідні функціональні можливості для ефективного управління підприємством та автоматизації бізнес-процесів.

По-третє, BAS ERP підтримує можливість міграції даних з 1С: Підприємство, що значно спрощує процес переходу на нову систему без втрати важливої інформації. Це є критично важливим для підприємств, які мають великі обсяги даних та не можуть дозволити собі тривалі періоди простою або втрату даних під час міграції.

Нарешті, подібність інтерфейсу BAS ERP та 1С: Підприємство забезпечує користувачам можливість швидко освоїти нову систему завдяки знайомій логіці роботи та інтерфейсу. Це знижує витрати на навчання персоналу та підвищує ефективність впровадження нової системи.

BAS ERP є ідеальним рішенням для автоматизації дрібнооптової торгівлі. Вона дозволяє ефективно керувати всіма етапами торгового процесу, від закупівлі товарів до їх реалізації кінцевим споживачам. Основні переваги використання BAS ERP у цій сфері:

* система дозволяє точно відстежувати кількість товарів на складі, аналізувати попит та автоматично формувати замовлення на поповнення запасів;
* BAS ERP забезпечує ефективне управління взаємовідносинами з постачальниками, включаючи оцінку їх надійності, аналіз умов поставок та оптимізацію закупівельних процесі;
* система дозволяє вести облік всіх операцій з продажу, відстежувати виконання замовлень, управляти ціноутворенням та формувати аналітичні звіти;
* BAS ERP включає інструменти для управління клієнтською базою, персоналізації пропозицій та підвищення рівня обслуговування клієнтів;
* потужні аналітичні інструменти дозволяють керівництву підприємства отримувати повну інформацію про стан бізнесу та приймати обґрунтовані рішення на основі точних даних.

РОЗДІЛ 3

ПРОБЛЕМА ЗАХИСТУ ІНФОРМАЦІЇ В БАЗАХ ДАНИХ

Захист баз даних є критично важливим аспектом у сучасному інформаційному середовищі. Бази даних зберігають величезні обсяги інформації, яка може бути конфіденційною, фінансово значущою або мати іншу важливість для організацій та користувачів. Проблеми захисту баз даних стають все більш актуальними у зв'язку з зростанням обсягу даних, які зберігаються, і збільшенням числа загроз, спрямованих на несанкціонований доступ до них.

* 1. Основні загрози

Основні загрози для баз даних включають несанкціонований доступ, крадіжку даних, зміни даних, атаки типу «відмова в обслуговуванні» та інші види кібератак. Деякі з цих загроз можуть виникати через зовнішні атаки, в той час як інші можуть бути результатом внутрішніх загроз, таких як недбалість співробітників або навмисні зловживання. Несанкціонований доступ є найпоширенішою загрозою, яка включає доступ до даних без відповідних прав. Це може призвести до витоку конфіденційної інформації, її модифікації або знищення. Це в свою чергу може спричинити значні проблеми для організацій, включаючи фінансові втрати та порушення довіри клієнтів.

3.2 Заходи для забезпечення захисту інформації

Для забезпечення належного захисту баз даних використовуються різні методи та підходи. Аутентифікація є процесом перевірки особи користувача перед наданням доступу до бази даних. Це може бути реалізовано через паролі, біометричні дані або інші методи. Авторизація є процесом визначення прав доступу користувачів до різних частин бази даних, що дозволяє обмежувати дії користувачів відповідно до їх ролей і повноважень. Шифрування використовує криптографічні методи для захисту даних під час їх зберігання та передачі, забезпечуючи захист даних навіть у випадку їх перехоплення. Контроль доступу встановлює правила і політики, які регулюють доступ до бази даних на основі ролей, привілеїв і контексту. Аудит та моніторинг передбачають відстеження та запис дій користувачів і системи для виявлення та реагування на потенційні загрози, а також допомагають у виявленні аномалій та розслідуванні інцидентів безпеки. Резервне копіювання полягає в регулярному створенні копій бази даних для відновлення даних у випадку їх втрати або пошкодження.

Захист баз даних є невід'ємною частиною загальної стратегії інформаційної безпеки організацій. Впровадження відповідних заходів захисту допомагає мінімізувати ризики, пов'язані з несанкціонованим доступом, крадіжкою та втратою даних.

РОЗДІЛ 4

ХАРАКТЕРИСТИКА СУБД MYSQL. КЛІЄНТ HEIDISQL

* 1. Загальний опис MySQL

MySQL – це одна з найпопулярніших систем управління реляційними базами даних у світі. Відкрита та вільна у використанні, MySQL набула широкого розповсюдження завдяки своїй надійності, масштабованості та продуктивності. Розроблена компанією MySQL AB і згодом придбана Oracle Corporation, MySQL стала незамінним інструментом для веб-додатків, корпоративних систем та багатьох інших галузей.

* 1. Основні особливості MySQL

Популярність MySQL зумовлена багатьма її ключовими особливостями.

По-перше, MySQL розповсюджується під ліцензією GNU General Public License (GPL), що дозволяє користувачам вільно використовувати, змінювати та поширювати програму. Це забезпечує широку доступність і можливість налаштування системи під конкретні потреби. Крім того, існує комерційна версія MySQL, яка надає додаткові можливості підтримки та функціоналу, що є особливо корисним для підприємств, які потребують розширених можливостей та технічної підтримки.

По-друге, MySQL використовує архітектуру клієнт-сервер, де сервер обробляє запити, що надходять від одного або більше клієнтів. Це дозволяє централізовано зберігати дані та забезпечувати доступ до них з різних клієнтських додатків, що значно спрощує управління базами даних та підвищує їх доступність.

По-третє, MySQL здатна обробляти великі обсяги даних і підтримувати високу продуктивність навіть у найвимогливіших середовищах. Завдяки функціям масштабованості, таким як реплікація та шардінг, MySQL підходить як для невеликих додатків, так і для великих корпоративних систем, дозволяючи підприємствам масштабувати свої бази даних відповідно до зростання бізнесу.

По-четверте, система забезпечує надійний захист даних через механізми аутентифікації, шифрування та контролю доступу на основі ролей. Це гарантує, що тільки авторизовані користувачі можуть отримати доступ до даних, а також захищає дані від несанкціонованого доступу. Вбудовані можливості резервного копіювання та відновлення даних додатково гарантують збереження та цілісність інформації, що є важливим для бізнес-критичних додатків.

Однією з важливих технічних характеристик MySQL є підтримка різних типів таблиць. MySQL підтримує такі типи таблиць, як InnoDB, MyISAM, Memory, CSV та інші. InnoDB, наприклад, забезпечує транзакційну підтримку та цілісність даних, що є критично важливим для додатків, де потрібні транзакції та підтримка зв'язності даних. MyISAM, з іншого боку, може бути використаний для швидкого читання великих обсягів даних, що робить його ідеальним для додатків, які потребують швидкого доступу до даних.

MySQL широко використовується в різних галузях, включаючи веб-розробку, електронну комерцію, фінансові системи та телекомунікації. Веб-платформи, такі як Facebook, Twitter, YouTube, Google та WordPress, використовують MySQL для управління своїми базами даних, завдяки його швидкості, надійності та здатності легко інтегруватися з іншими технологіями.

* 1. Переваги MySQL
* СУБД оптимізована для швидкого виконання запитів і обробки великих обсягів даних;
* підтримка різних типів таблиць дозволяє вибирати оптимальний механізм зберігання для конкретних задач;
* вбудовані засоби захисту даних забезпечують надійне збереження інформації;
* підтримка реплікації та шардінгу дозволяє масштабувати систему відповідно до потреб бізнесу;
* вільне використання та можливість внесення змін у код сприяють широкому впровадженню та розвитку.
  1. Клієнт HeidiSQL

Для зручного користування MySQL використовуються різні клієнти, що забезпечують графічний інтерфейс для адміністрування баз даних. Одним з таких клієнтів є HeidiSQL. Це вільний відкритий клієнт для управління базами даних, розроблений німецьким програмістом А. Бекером (A. Becker) та кількома іншими розробниками, написаний на Delphi та призначений для операційної системи Windows. Він підтримує з'єднання та роботу з MySQL, MariaDB та Percona, а також Microsoft SQL Server, починаючи з версії 7.0.

HeidiSQL забезпечує потужні інструменти для редагування даних, включаючи можливості перегляду та редагування таблиць у зручному табличному форматі, що значно полегшує процес маніпуляції даними, а також підтримує роботу з великими базами даних і забезпечує високу продуктивність завдяки оптимізованому коду та ефективним алгоритмам обробки даних. До того ж HeidiSQL забезпечує безпечне з'єднання з серверами баз даних за допомогою SSL/TLS шифрування та підтримує механізми аутентифікації для захисту даних від несанкціонованого доступу.

Однією з важливих функцій HeidiSQL є можливість створення резервних копій баз даних та їх відновлення, що забезпечує збереження та цілісність даних. Крім того, HeidiSQL дозволяє легко експортувати та імпортувати дані у різні формати, такі як SQL-скрипти, CSV, XML та інші, що значно спрощує процес обміну даними між різними системами.

Не менш важливим є те що HeidiSQL підтримує підключення до кількох серверів одночасно. Це дозволяє користувачам керувати кількома базами даних з одного інтерфейсу і робить HeidiSQL універсальним і потужним інструментом для адміністраторів баз даних, які прагнуть ефективно керувати своїми даними та оптимізувати процеси управління базами даних.

РОЗДІЛ 5

ПРАКТИЧНА РЕАЛІЗАЦІЯ

* 1. Встановлення необхідного програмного забезпечення

Для виконання практичної частини дипломної роботи нам необхідно встановити сервер баз даних MySQL та клієнт HeidiSQL.

* + 1. Встановлення серверу MySQL

Для початку перейдемо на офіційний сайт MySQL: <https://dev.mysql.com/downloads/mysql/>

Далі виберемо операційну систему Windows і завантажимо інсталяційний файл.

Запустимо завантажений інсталяційний файл та пройдемо через майстра установки, вибираючи необхідні компоненти.

На етапі конфігурації, задамо пароль для користувача root.

Завершимо установку та переконаймося, що MySQL запущений.

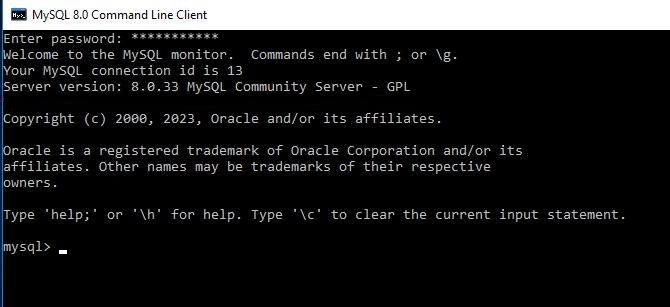


Рисунок 5.1 – Сервер MySQl

* + 1. Встановлення HeidiSQL

Для завантаження HeidiSQL перейдемо на офіційний сайт HeidiSQL: <https://www.heidisql.com/download.php>

Завантажимо інсталяційний файл для операційної системи Windows та запустимо його. Далі пройдемо через майстра установки, вибираючи стандартні опції.

Для підключення до MySQL за допомогою HeidiSQL створимо нове підключення:

1. Запустимо HeidiSQL.
2. Натиснемо "New" для створення нового підключення.
3. Введемо наступні дані:

Network type: MariaDB or MySQL (TCP/IP)

Hostname / IP: 127.0.0.1

User: root

Password: Пароль, встановлений для користувача root.

1. Натиснемо "Open" для підключення.

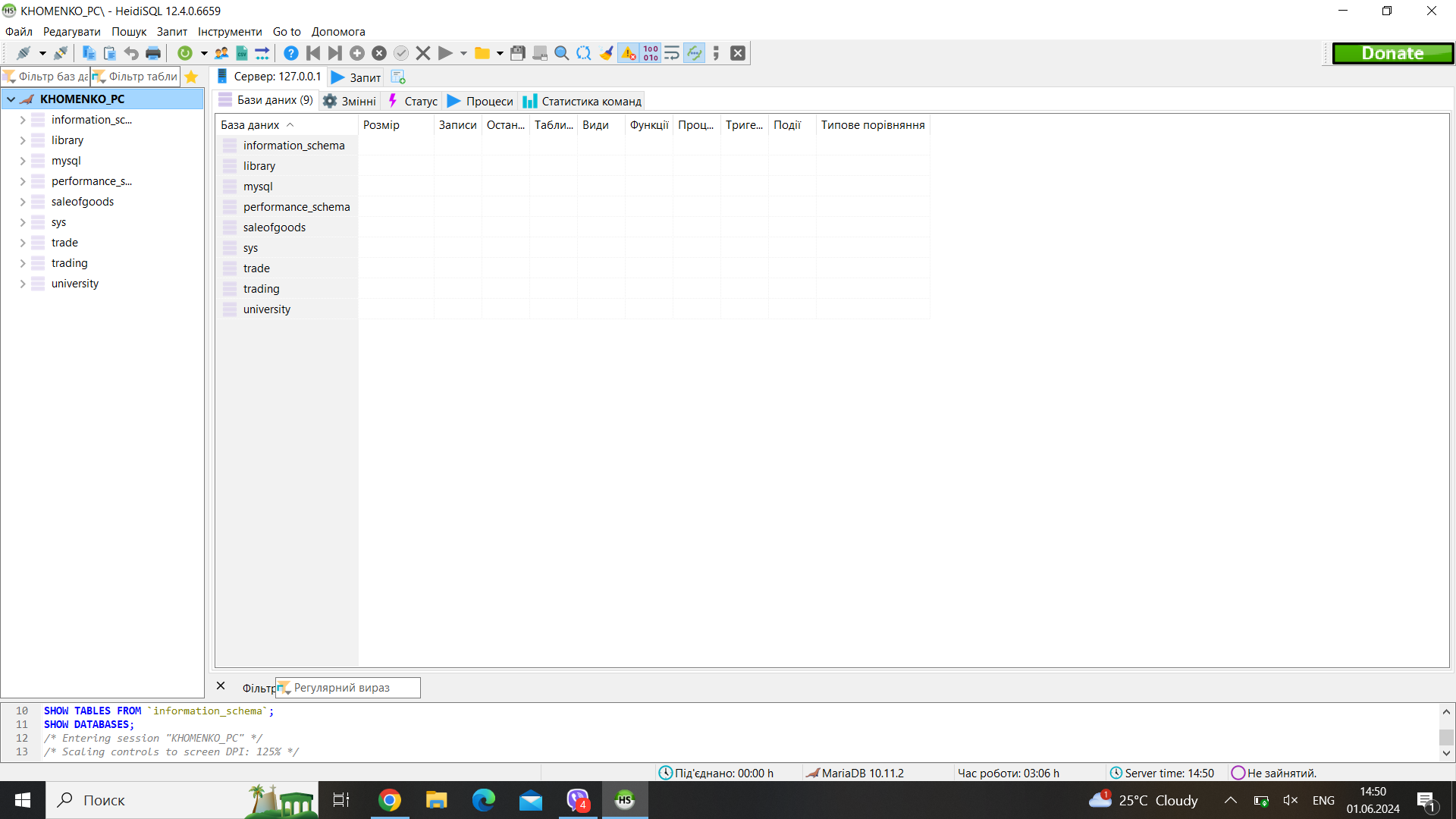


Рисунок 5.2 – Діалогове вікно HeidiSQl

* 1. Концептуальне моделювання даних

Моделювання, яке використовується для розробки схеми бази даних, називається концептуальним моделюванням даних. Основним інструментом при цьому є ER-модель, що означає модель "сутність-зв’язок" (англ. Entity-Relationship Diagram).

ER-модель складається з:

* Сутностей: Об'єкти або поняття, що зберігаються в базі даних (наприклад, клієнти, товари).
* Атрибутів: Властивості або характеристики сутностей (наприклад, ім'я клієнта, ціна товару).
* Зв'язків: Відношення між сутностями (наприклад, клієнт робить замовлення).

Графічне представлення ER-моделі, що відображає сутності, атрибути та зв'язки між ними називається ER-діаграмою.

* + 1. Позначення

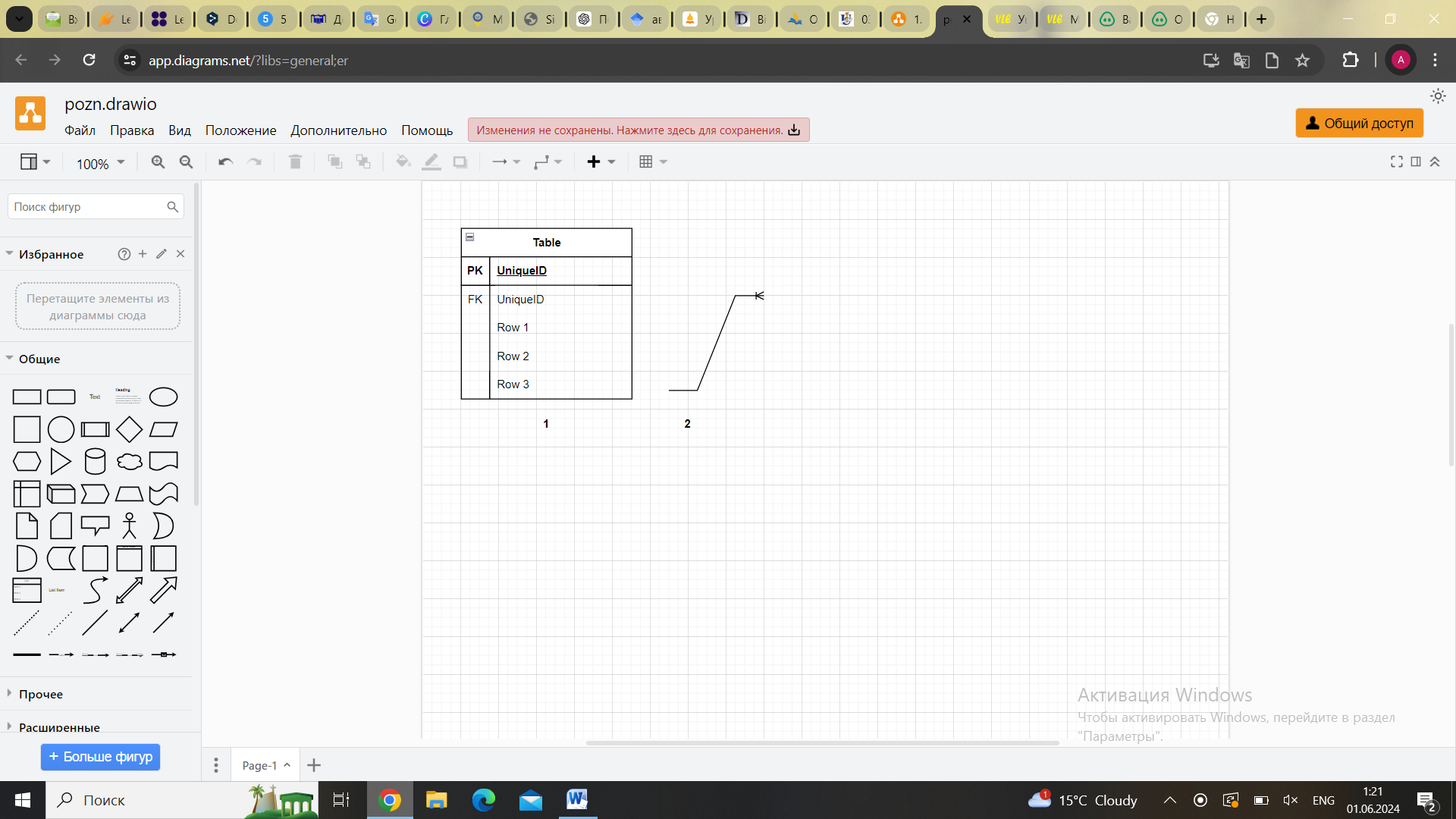


Рисунок 5.3 – Позначення

1. Сутність Table (Первинний ключ PK. Вторинний ключ FK. Обов’язкові атрибути Row 1, Row 2, Row 3).
2. Зв’язок «один-до-багатьох».
   * 1. Побудова ER-діаграми

Для побудови ER-діаграми будемо використовувати сайт: <https://app.diagrams.net/>

Використовуючи дані з пункту 5.2.1 маємо:

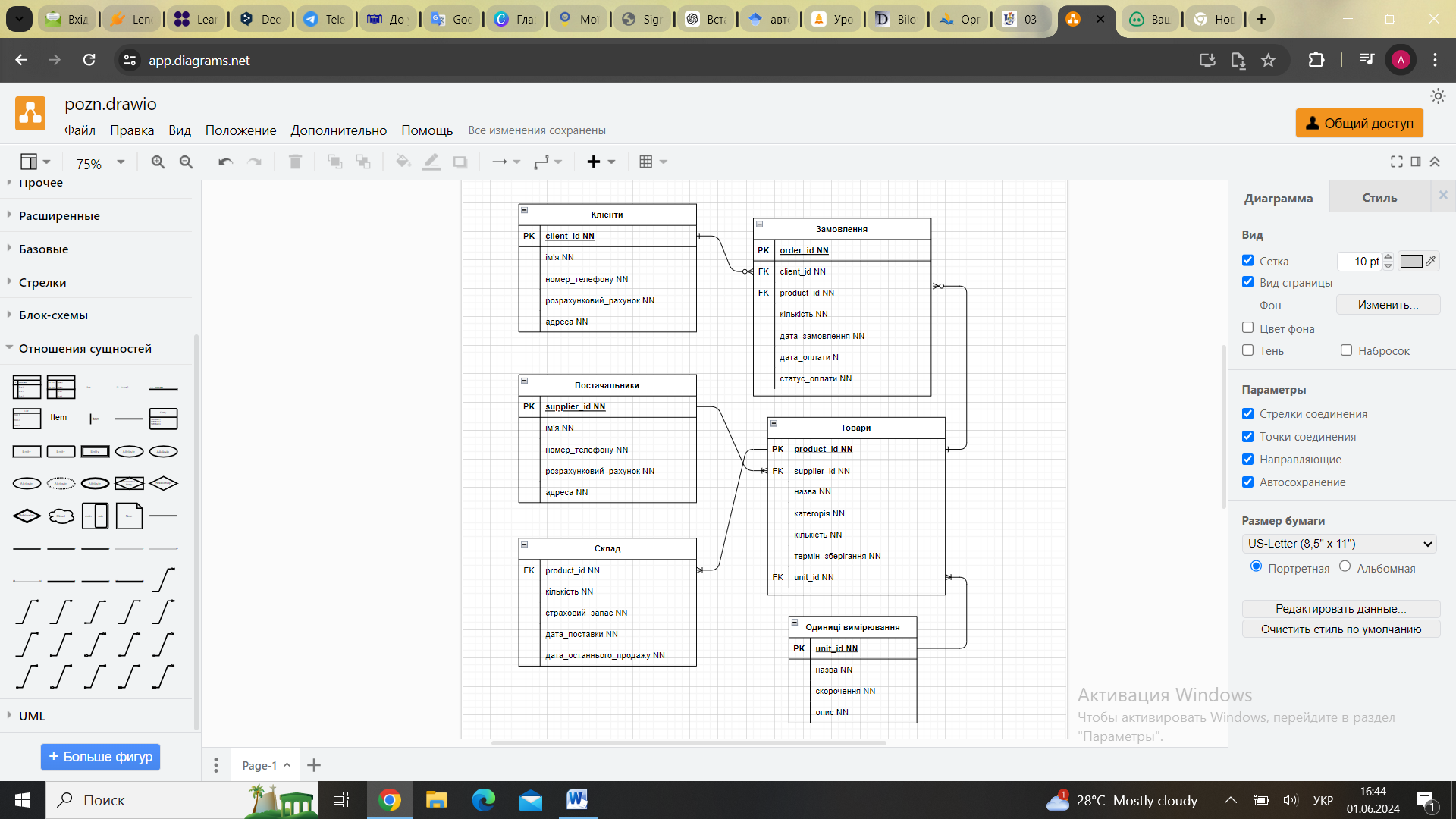


Рисунок 5.4 – ER-діаграма

Таблиці мають такий вигляд:

1. Таблиця «Товари» має містити такі дані: Id товару, назва продукту, категорія (наприклад, овочі, фрукти, м'ясо, тощо), кількість на складі, термін зберігання товару, постачальник, ціна за одиницю, Id одиниці вимірювання.
2. Таблиця «Клієнти» має містити такі дані: Id клієнта, Ім'я, номер телефону, розрахунковий рахунок, адреса (місто, вулиця, номер будинку).
3. Таблиця «Замовлення» має містити такі дані: Id замовлення, Id клієнта, Id товару, кількість, дата замовлення, дата оплати, статус оплати (оплачено або не оплачено).
4. Таблиця «Постачальники» має містити такі дані: Id постачальника, ім'я, номер телефону, розрахунковий рахунок, адреса (місто, вулиця, номер будинку).
5. Таблиця «Склад» має містити такі дані: Id товару, кількість, страховий запас, дата поставки, дата останнього продажу.
6. Таблиця «Одиниці вимірювання» має містити такі дані: Id одиниці вимірювання, скорочення, опис.
   * 1. Опис реляційної моделі

Товари

Таблиця 5.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Products  Відповідає типу сутності Товари | | | | | |
| Ключ | Ім’я атрибуту | Тип атрибуту | NOT NULL /NULL | FK:  ON DELETE  ON UPDATE | Пояснення (перехід від ER-моделі) |
| PK | **product\_id** | int(12) | NOT NULL |  | Атрибут простий, обов’язковий |
| FK | **supplier\_id** | int(12) | NOT NULL | ON DELETE: RESTRICT  ON UPDATE: CASCADE | Атрибут простий, обов’язковий |
|  | **name** | varchar(20) | NOT NULL |  | Атрибут простий, обов’язковий |
|  | **category** | varchar(20) | NOT NULL |  | Атрибут простий, обов’язковий |
|  | **price** | decimal(5,2) | NOT NULL |  | Атрибут простий, обов’язковий |
|  | **amount** | int(11) | NOT NULL |  | Атрибут простий, обов’язковий |
| FK | **unit\_id** | int(11) | NOT NULL | ON DELETE: RESTRICT  ON UPDATE: CASCADE | Атрибут простий, обов’язковий |
|  | **exp\_date** | date | NOT NULL |  | Атрибут простий, обов’язковий |
|  | **infl\_coeff** | decimal(5,2) | NOT NULL |  | Атрибут простий, обов’язковий |

Клієнти

Таблиця 5.2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Clients  Відповідає типу сутності Клієнти | | | | | |
| Ключ | Ім’я атрибуту | Тип атрибуту | NOT NULL /NULL | FK:  ON DELETE  ON UPDATE | Пояснення (перехід від ER-моделі) |
| PK | **client\_id** | int(12) | NOT NULL |  | Атрибут простий, обов’язковий |
|  | **name** | varchar(30) | NOT NULL |  | Атрибут простий, обов’язковий |
|  | **phone\_number** | varchar(20) | NOT NULL |  | Атрибут простий, обов’язковий |
|  | **account** | varchar(30) | NOT NULL |  | Атрибут простий, обов’язковий |
|  | **address** | varchar(60) | NOT NULL |  | Атрибут простий, обов’язковий |

Замовлення

Таблиця 5.3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Orders  Відповідає типу сутності Замовлення | | | | | |
| Ключ | Ім’я атрибуту | Тип атрибуту | NOT NULL /NULL | FK:  ON DELETE  ON UPDATE | Пояснення (перехід від ER-моделі) |
| PK | **order\_id** | int(12) | NOT NULL |  | Атрибут простий, обов’язковий |
| FK | **client\_id** | int(12) | NOT NULL | ON DELETE: CASCADE  ON UPDATE: CASCADE | Атрибут простий, обов’язковий |
| FK | **product\_id** | int(12) | NOT NULL | ON DELETE: RESTRICT  ON UPDATE: CASCADE | Атрибут простий, обов’язковий |
|  | **amount** | int(10) | NOT NULL |  | Атрибут простий, обов’язковий |
|  | **order\_date** | date | NOT NULL |  | Атрибут простий, обов’язковий |
|  | **payment\_date** | date | NULL |  | Атрибут простий, обов’язковий |
|  | **paid** | tinyint(1) | NOT NULL |  | Атрибут простий, обов’язковий |

Постачальники

Таблиця 5.4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Suppliers  Відповідає типу сутності Постачальники | | | | | |
| Ключ | Ім’я атрибуту | Тип атрибуту | NOT NULL /NULL | FK:  ON DELETE  ON UPDATE | Пояснення (перехід від ER-моделі) |
| PK | **supplier\_id** | int(12) | NOT NULL |  | Атрибут простий, обов’язковий |
|  | **name** | varchar(30) | NOT NULL |  | Атрибут простий, обов’язковий |
|  | **phone\_number** | varchar(20) | NOT NULL |  | Атрибут простий, обов’язковий |
|  | **account** | varchar(30) | NOT NULL |  | Атрибут простий, обов’язковий |
|  | **address** | varchar(60) | NOT NULL |  | Атрибут простий, обов’язковий |

Склад

Таблиця 5.5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Warehouse  Відповідає типу сутності Склад | | | | | |
| Ключ | Ім’я атрибуту | Тип атрибуту | NOT NULL /NULL | FK:  ON DELETE  ON UPDATE | Пояснення (перехід від ER-моделі) |
| PK | **product\_id** | int(12) | NOT NULL |  | Атрибут простий, обов’язковий |
|  | **amount** | int(12) | NOT NULL |  | Атрибут простий, обов’язковий |
|  | **reserve** | int(12) | NOT NULL |  | Атрибут простий, обов’язковий |
|  | **deliver\_date** | date | NOT NULL |  | Атрибут простий, обов’язковий |
|  | **date\_of\_last\_sale** | date | NOT NULL |  | Атрибут простий, обов’язковий |

Одиниці вимірювання

Таблиця 5.6

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Units\_of\_measurement  Відповідає типу сутності Одиниці вимірювання | | | | | |
| Ключ | Ім’я атрибуту | Тип атрибуту | NOT NULL /NULL | FK:  ON DELETE  ON UPDATE | Пояснення (перехід від ER-моделі) |
| PK | **unit\_id** | int(12) | NOT NULL |  | Атрибут простий, обов’язковий |
|  | **name** | varchar(20) | NOT NULL |  | Атрибут простий, обов’язковий |
|  | **abbreviation** | varchar(20) | NOT NULL |  | Атрибут простий, обов’язковий |
|  | **description** | varchar(50) | NOT NULL |  | Атрибут простий, обов’язковий |

* 1. Реалізація запитів, збережених процедур та тригерів в HeidiSQL
     1. Створення запитів

Запити — це основний засіб взаємодії з базою даних. Запити використовуються для виконання різних операцій, таких як вибірка, вставка, оновлення та видалення даних.

Далі буде представлено запити, які були розроблені для ефективного управління складськими запасами та замовленнями в системі дрібнооптової торгівлі продуктами харчування. Ці запити допомагають забезпечити контроль над наявністю товарів, виконанням замовлень та взаємодією з постачальниками.

* + - 1. Список всіх замовлень з інформацією про клієнтів і товари

Цей запит надає інформацію про всі замовлення разом з деталями про клієнтів і товари, які входять до кожного замовлення. Це допомагає відстежувати виконання замовлень та взаємодію з клієнтами.

Запит на мові SQL:

SELECT

Orders.order\_id,

Orders.order\_date AS Дата\_замовлення,

Clients.name AS Клієнт,

Products.name AS Товар,

Orders.amount AS Кількість,

Orders.amount \* Products.price AS Загальна\_сума

FROM

Orders

JOIN

Clients ON Orders.client\_id = Clients.client\_id

JOIN

Products ON Orders.product\_id = Products.product\_id;

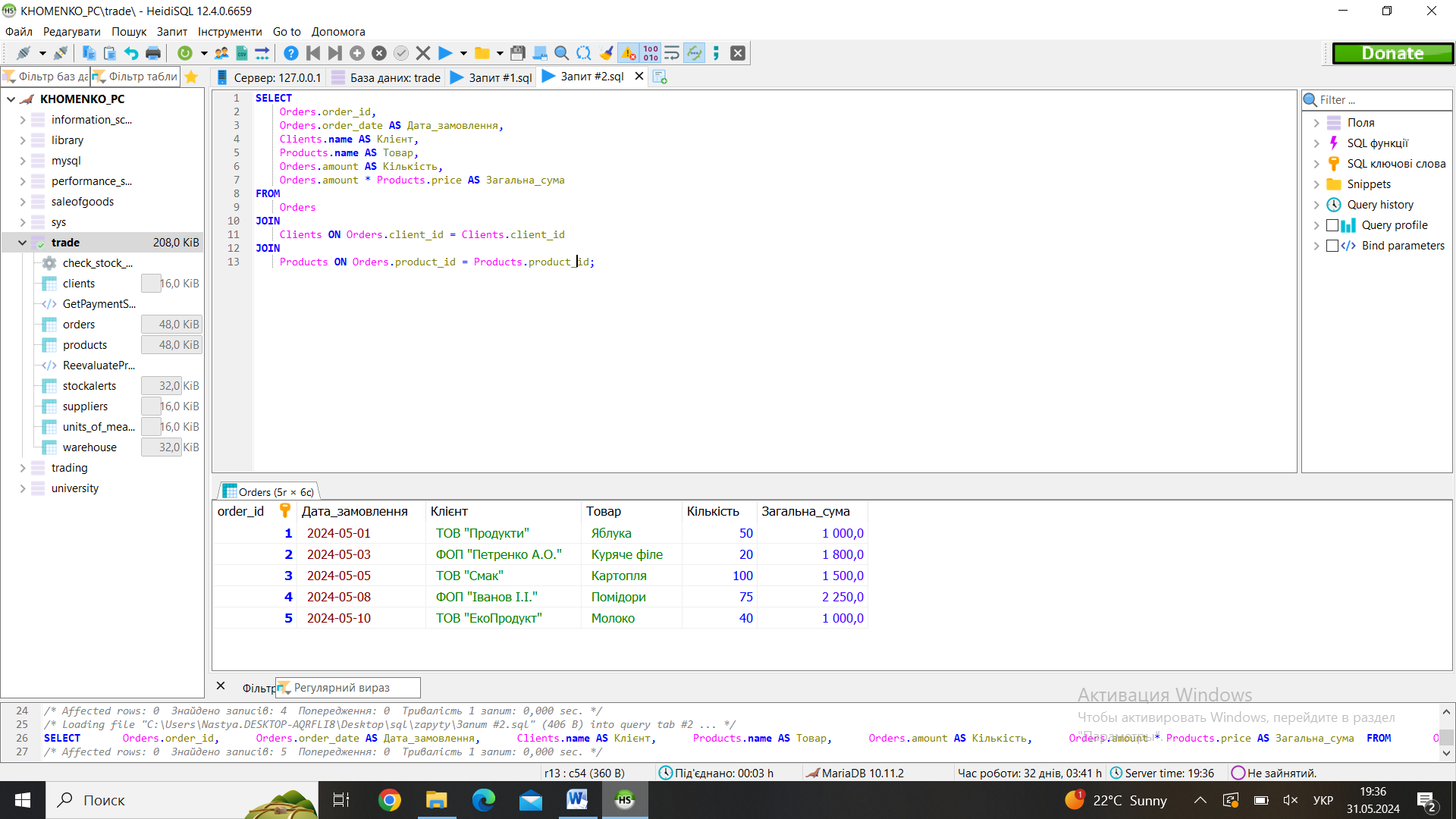


Рисунок 5.6 – Результат виконання запиту «Список всіх замовлень з інформацією про клієнтів і товари» в HeidiSQL

* + - 1. Контактні дані постачальників та товари, які вони постачають

Цей запит дозволяє отримати контактну інформацію про постачальників та перелік продуктів, які вони постачають. Це полегшує управління постачанням та забезпечує безперебійне постачання необхідних товарів.

Запит на мові SQL:

SELECT

Suppliers.supplier\_id,

Suppliers.name AS Постачальник,

Suppliers.phone\_number AS Номер\_телефону,

Suppliers.address AS Адреса,

Products.product\_id,

Products.name AS Товар,

Products.category AS Категорія

FROM

Suppliers

JOIN

Products ON Suppliers.supplier\_id = Products.supplier\_id;

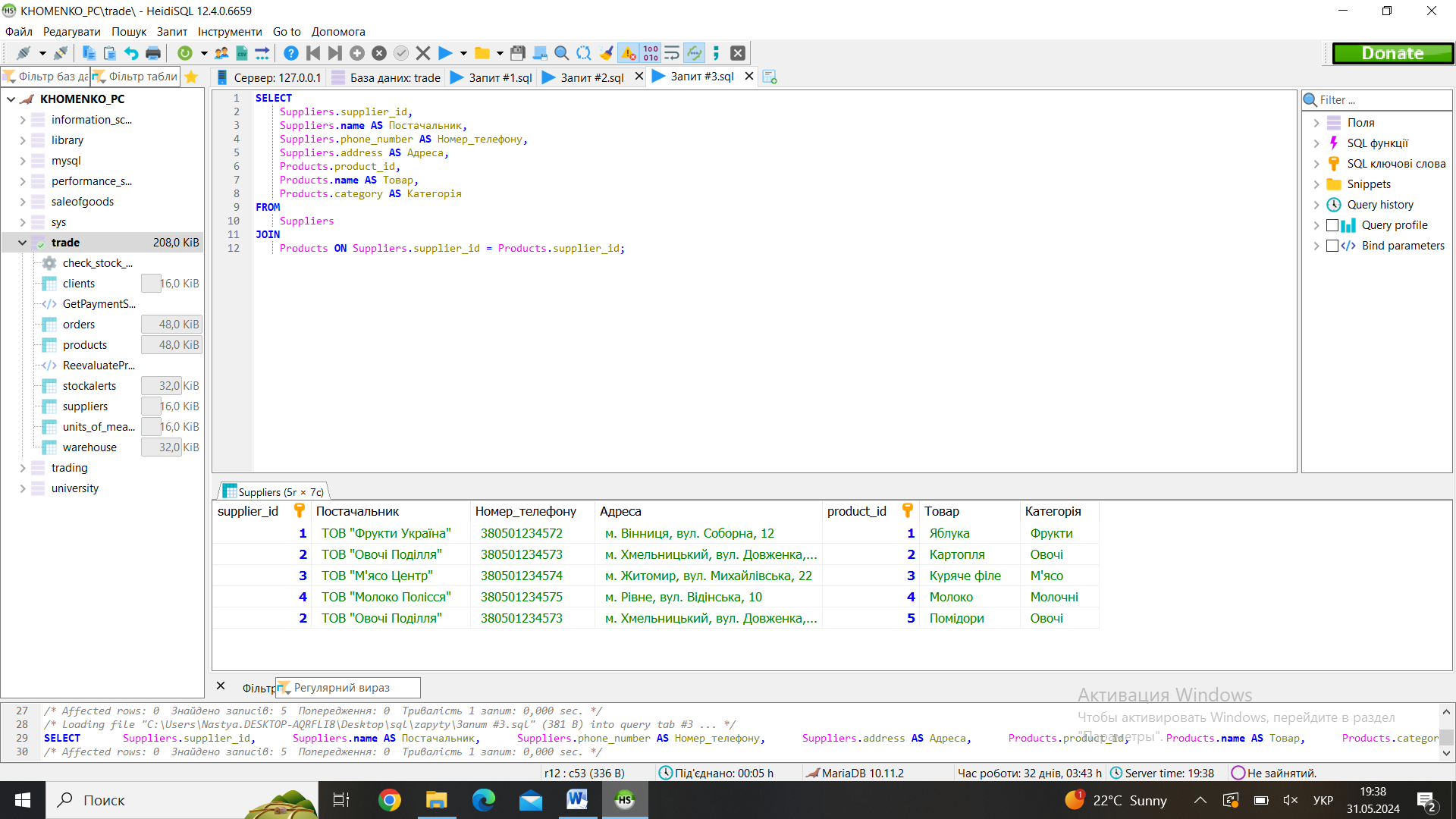


Рисунок 5.7 – Результат виконання запиту «Контактні дані постачальників та товари, які вони постачають» в HeidiSQL

* + - 1. Страховий запас всіх продуктів

Цей запит надає інформацію про страховий запас для всіх продуктів на складі. Це допомагає контролювати рівень запасів та приймати обґрунтовані рішення щодо поповнення товарів.

Запит на мові SQL:

SELECT

Products.name AS Товар,

Warehouse.reserve AS Страховий\_запас

FROM

Warehouse

JOIN

Products ON Warehouse.product\_id = Products.product\_id;

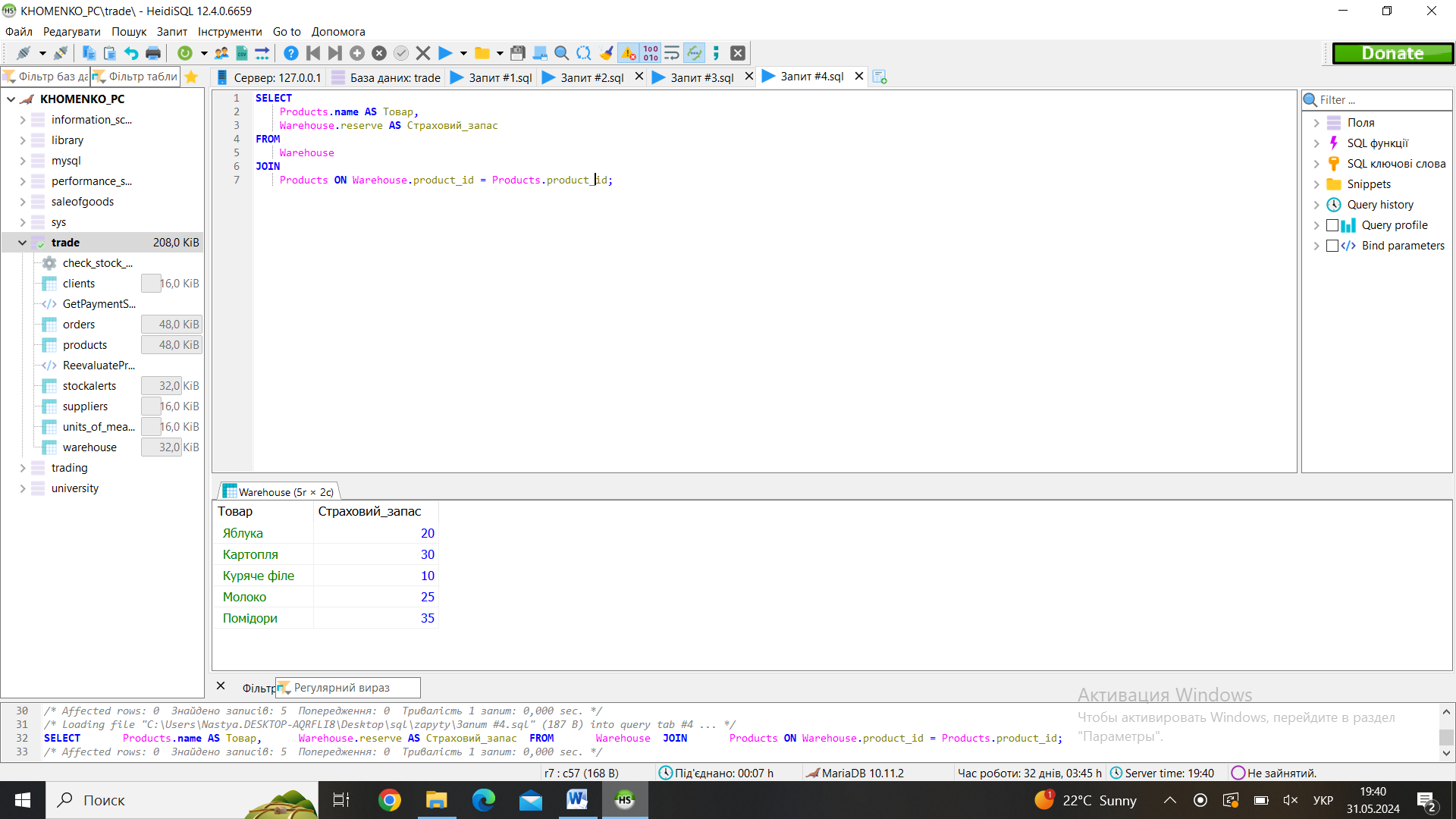


Рисунок 5.8 – Результат виконання запиту «Страховий запас всіх продуктів» в HeidiSQL

* + - 1. Список продуктів, кількість яких на складі менша за страховий запас

Цей запит дозволяє виявити продукти, кількість яких на складі менша за встановлений страховий запас. Це допомагає уникнути дефіциту товарів і своєчасно поповнювати запаси.

Запит на мові SQL:

SELECT

Products.product\_id,

Products.name AS Назва,

Warehouse.amount AS Кількість,

Warehouse.reserve AS Страховий\_запас

FROM

Warehouse

JOIN

Products ON Warehouse.product\_id = Products.product\_id

WHERE

Warehouse.amount < Warehouse.reserve;

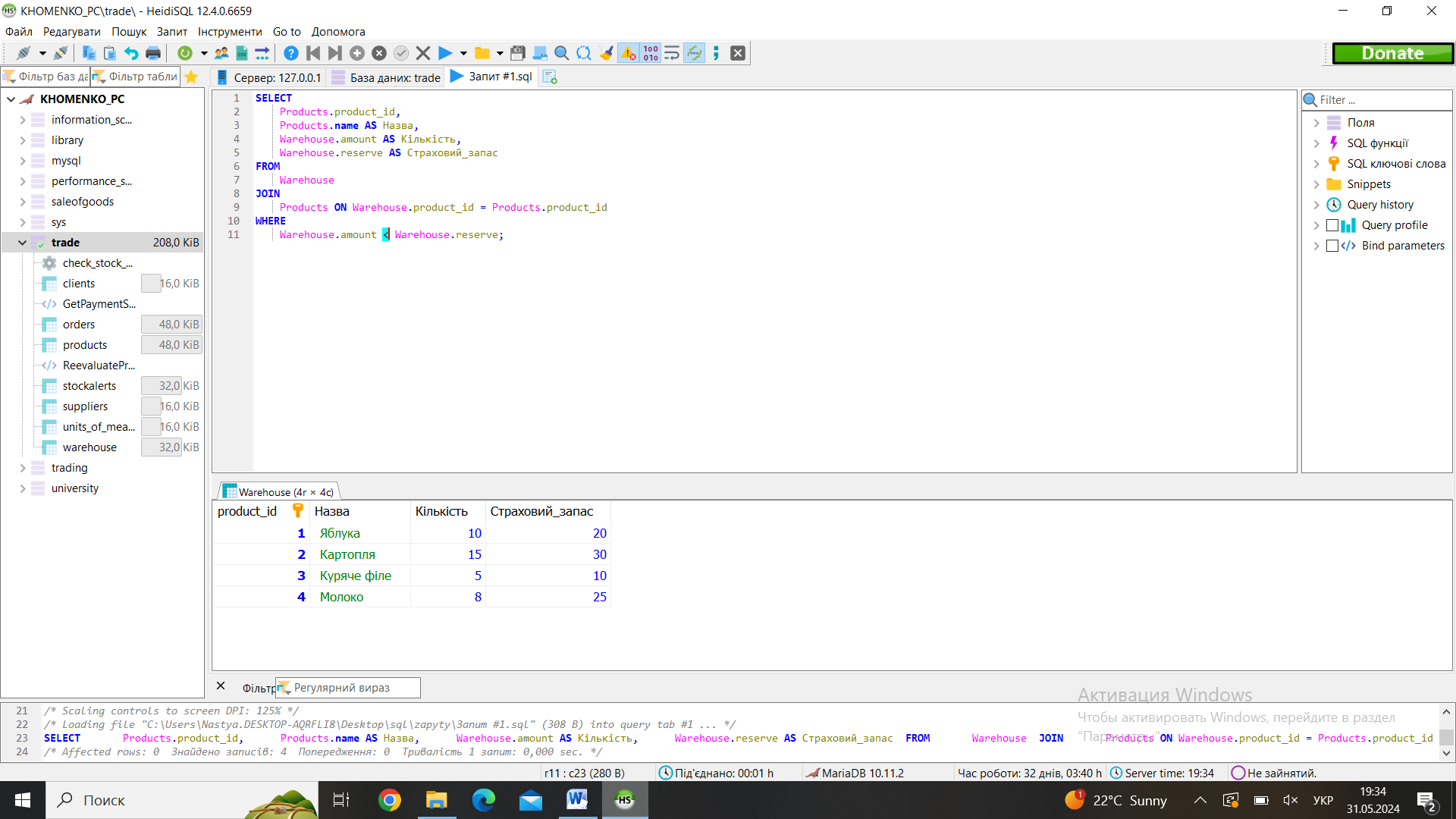


Рисунок 5.5 – Результат виконання запиту «Список продуктів, кількість яких на складі менша за страховий запас» в HeidiSQL

* + - 1. Список протермінованих продуктів на сьогодні

Цей запит дозволяє виявити продукти, термін придатності яких минув на сьогоднішній день. Це допомагає забезпечити своєчасне вилучення протермінованих товарів з продажу.

Запит на мові SQL:

SELECT

Products.name AS Товар,

Products.category AS Категорія,

Products.exp\_date AS Термін\_зберігання

FROM

Products

WHERE

Products.exp\_date < CURDATE();

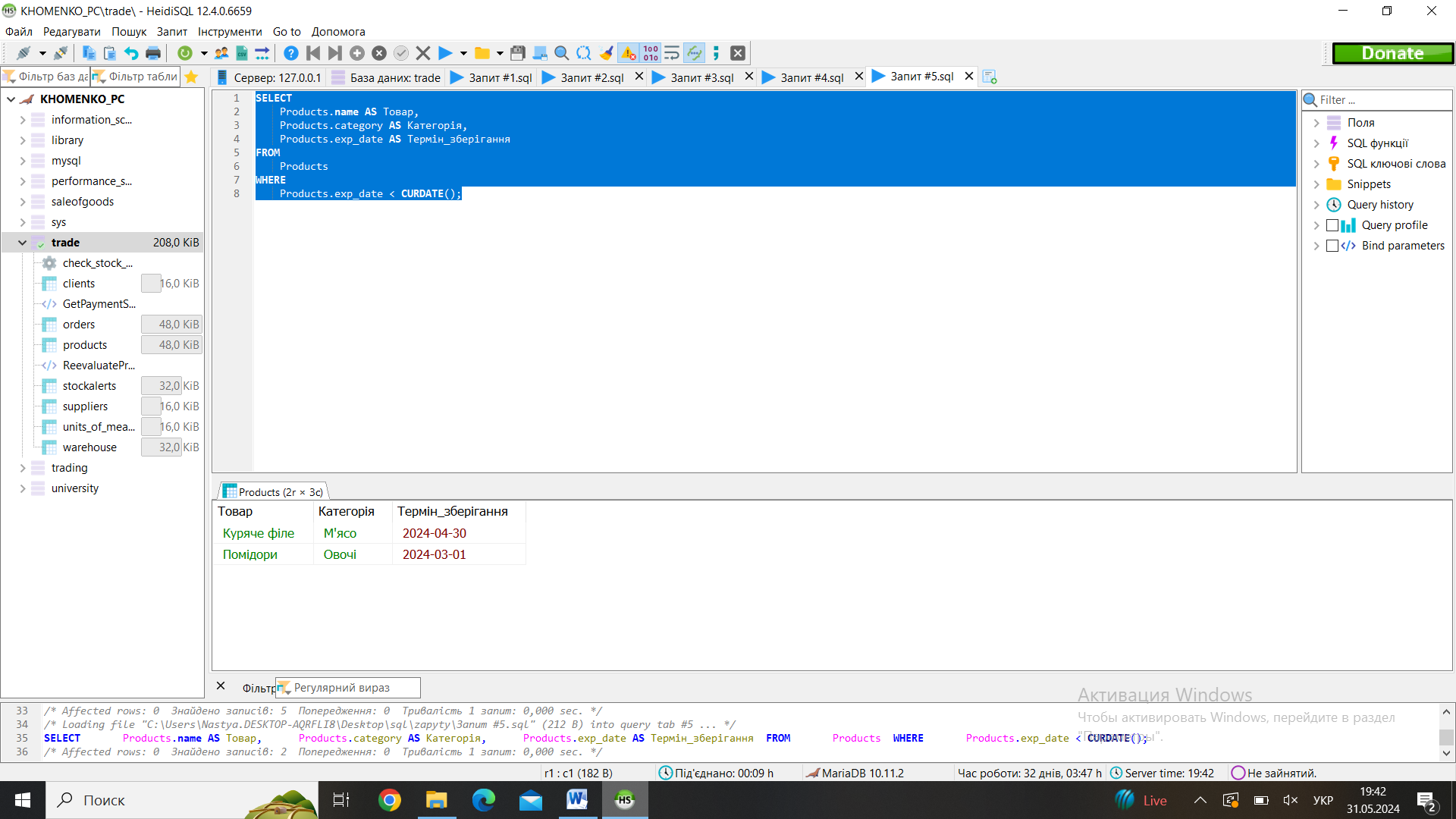


Рисунок 5.9 – Результат виконання запиту «Список протермінованих продуктів на сьогодні» в HeidiSQL

* + - 1. Список оплачених замовлень

Цей запит надає інформацію про всі замовлення, які були оплачені. Це допомагає відстежувати фінансові потоки та виконання замовлень.

Запит на мові SQL:

SELECT

Orders.order\_id AS Order\_id,

Orders.order\_date AS Дата\_замовлення,

Orders.payment\_date AS Дата\_оплати,

Clients.name AS Клієнт,

Products.name AS Товар,

Orders.amount AS Кількість,

Orders.paid AS Статус\_оплати

FROM

Orders

JOIN

Clients ON Orders.client\_id = Clients.client\_id

JOIN

Products ON Orders.product\_id = Products.product\_id

WHERE

Orders.paid = TRUE;

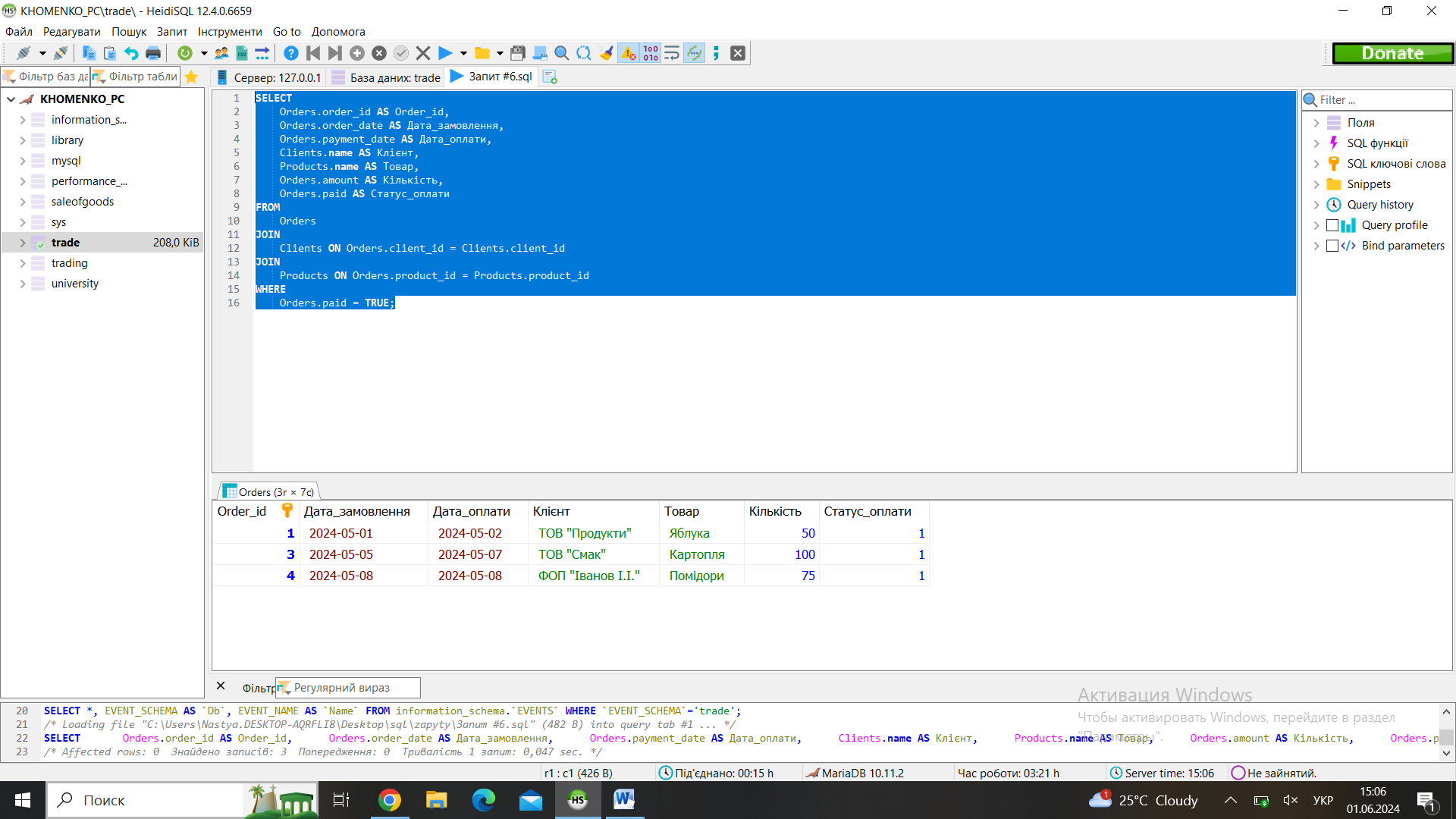


Рисунок 5.10 – Результат виконання запиту «Список оплачених замовлень» в HeidiSQL

* + - 1. Список неоплачених замовлень

Цей запит дозволяє отримати інформацію про всі замовлення, які ще не були оплачені. Це допомагає контролювати заборгованості клієнтів та вживати необхідних заходів для їх погашення.

Запит на мові SQL:

SELECT

Orders.order\_id AS Order\_id,

Orders.order\_date AS Дата\_замовлення,

Orders.payment\_date AS Дата\_оплати,

Clients.name AS Клієнт,

Products.name AS Товар,

Orders.amount AS Кількість,

Orders.paid AS Статус\_оплати

FROM

Orders

JOIN

Clients ON Orders.client\_id = Clients.client\_id

JOIN

Products ON Orders.product\_id = Products.product\_id

WHERE

Orders.paid = FALSE;

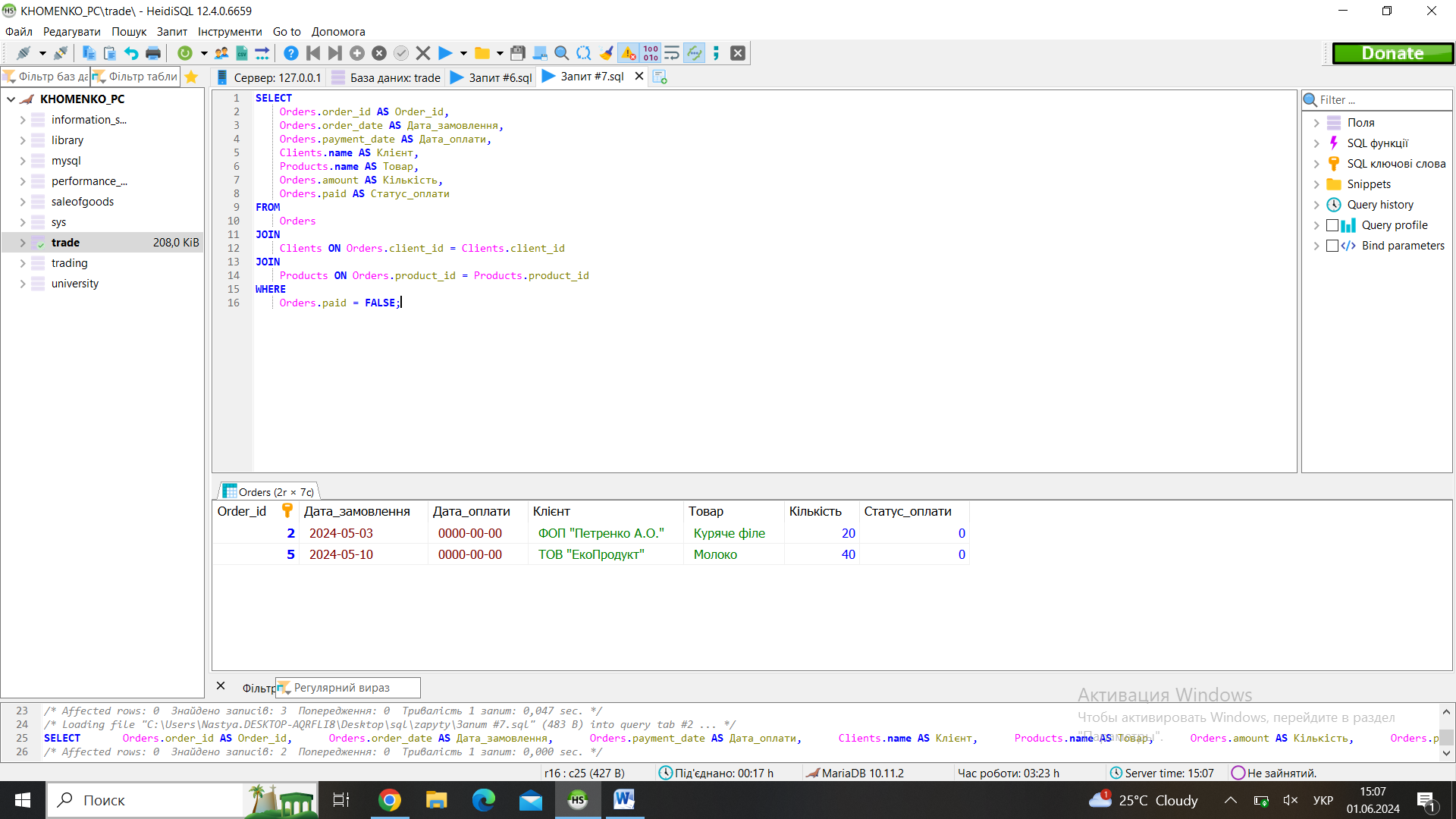


Рисунок 5.11 – Результат виконання запиту «Список неоплачених замовлень» в HeidiSQL

* + 1. Створення збережених процедур

Збережені процедури — це набори SQL-інструкцій, збережених у базі даних, які можуть бути виконані як єдина операція. Вони дозволяють зберігати складну логіку на стороні сервера бази даних, що покращує продуктивність і безпеку.

Переваги збережених процедур:

* процедури можна викликати багаторазово без необхідності переписувати SQL-код;
* код виконується на сервері бази даних, що зменшує обсяг переданих даних між сервером і клієнтом;
* користувачі можуть отримати доступ до збережених процедур без прямого доступу до таблиць бази даних;
  + - 1. Підрахунок загальної суми оплати замовлень за день, тиждень та місяць

Процедура приймає два параметри: дату початку і дату кінця, та виводить загальну суму оплати замовлень за день, тиждень та місяць. Це забезпечує зручний спосіб отримання фінансових звітів для аналізу доходів підприємства.

Опис збереженої процедури:

Збережена процедура під назвою GetPaymentSummary приймає два параметри:

* startDate (DATE): дата початку періоду;
* endDate (DATE): дата кінця періоду.

Опис логіки роботи:

Перший запит обчислює загальну суму оплати за кожен день у заданому діапазоні дат, групуючи результати за днями (DATE(payment\_date)).

Другий запит обчислює загальну суму оплати за кожен тиждень у заданому діапазоні дат, групуючи результати за роком і тижнем (YEAR(payment\_date) і WEEK(payment\_date)).

Третій запит обчислює загальну суму оплати за кожен місяць у заданому діапазоні дат, групуючи результати за роком і місяцем (YEAR(payment\_date) і MONTH(payment\_date)).

Вивід результатів:

Результати виконання збереженої процедури GetPaymentSummary будуть складатися з трьох окремих наборів даних: загальна сума оплати за день, загальна сума оплати за тиждень, і загальна сума оплати за місяць. Кожен з цих наборів буде містити відповідні поля та суми для зазначених інтервалів.

Виклик процедури:

Для виклику процедури використовується наступний SQL-запит з передачею необхідних параметрів startDate та endDate:

CALL GetPaymentSummary('2024-01-01', '2024-12-31');

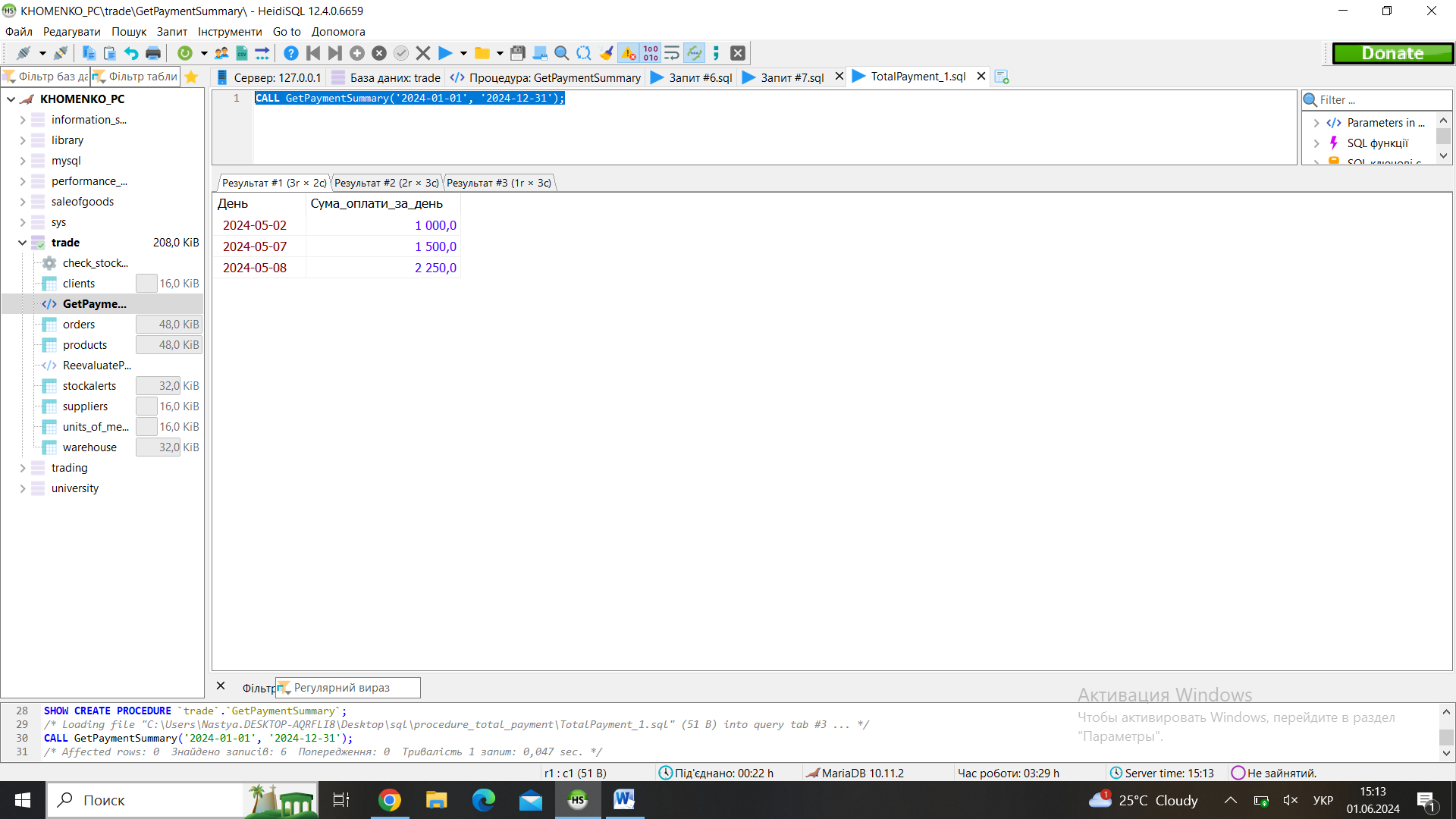


Рисунок 5.12 – Загальна сума оплати за день

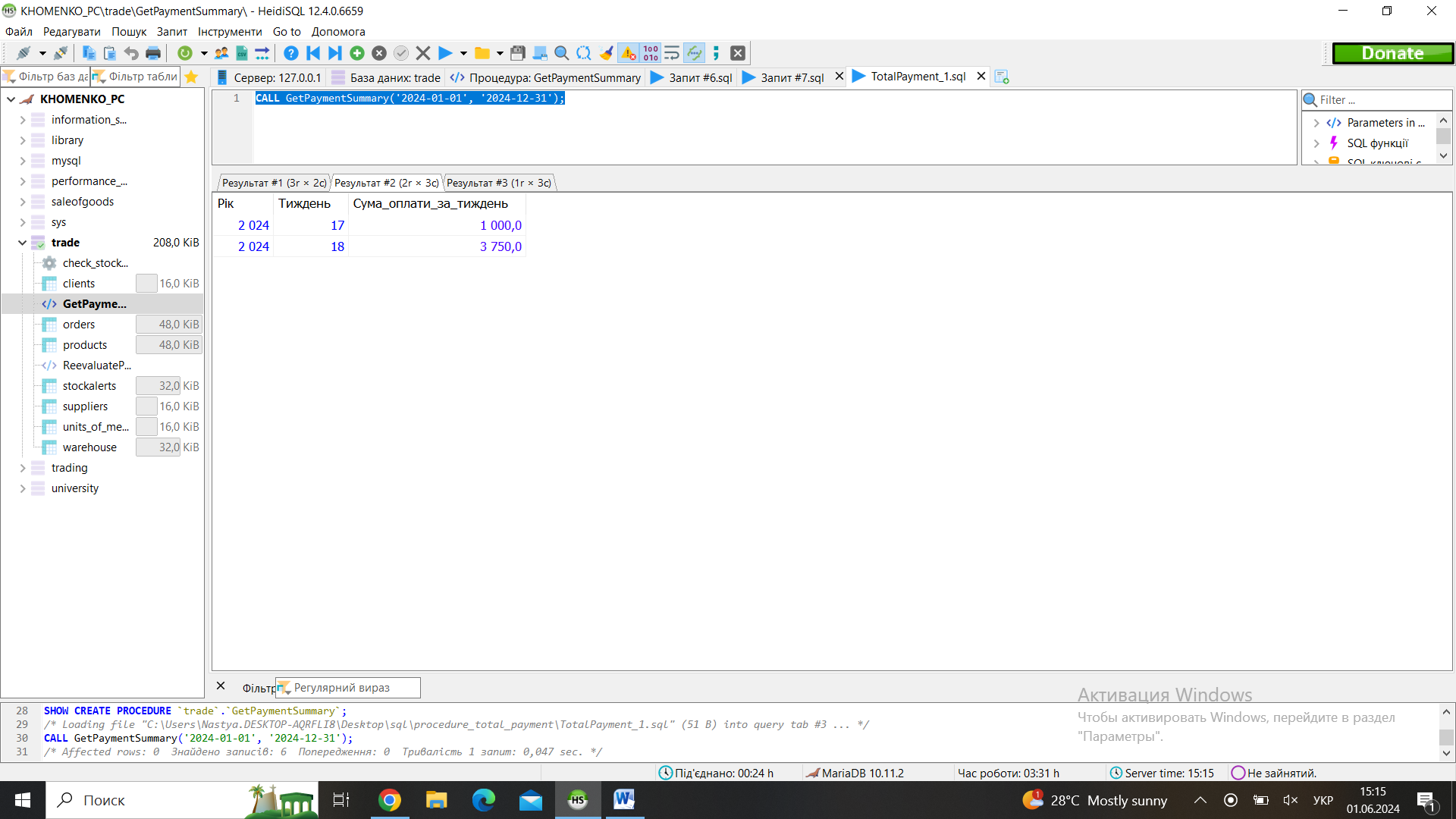


Рисунок 5.13 – Загальна сума оплати за тиждень

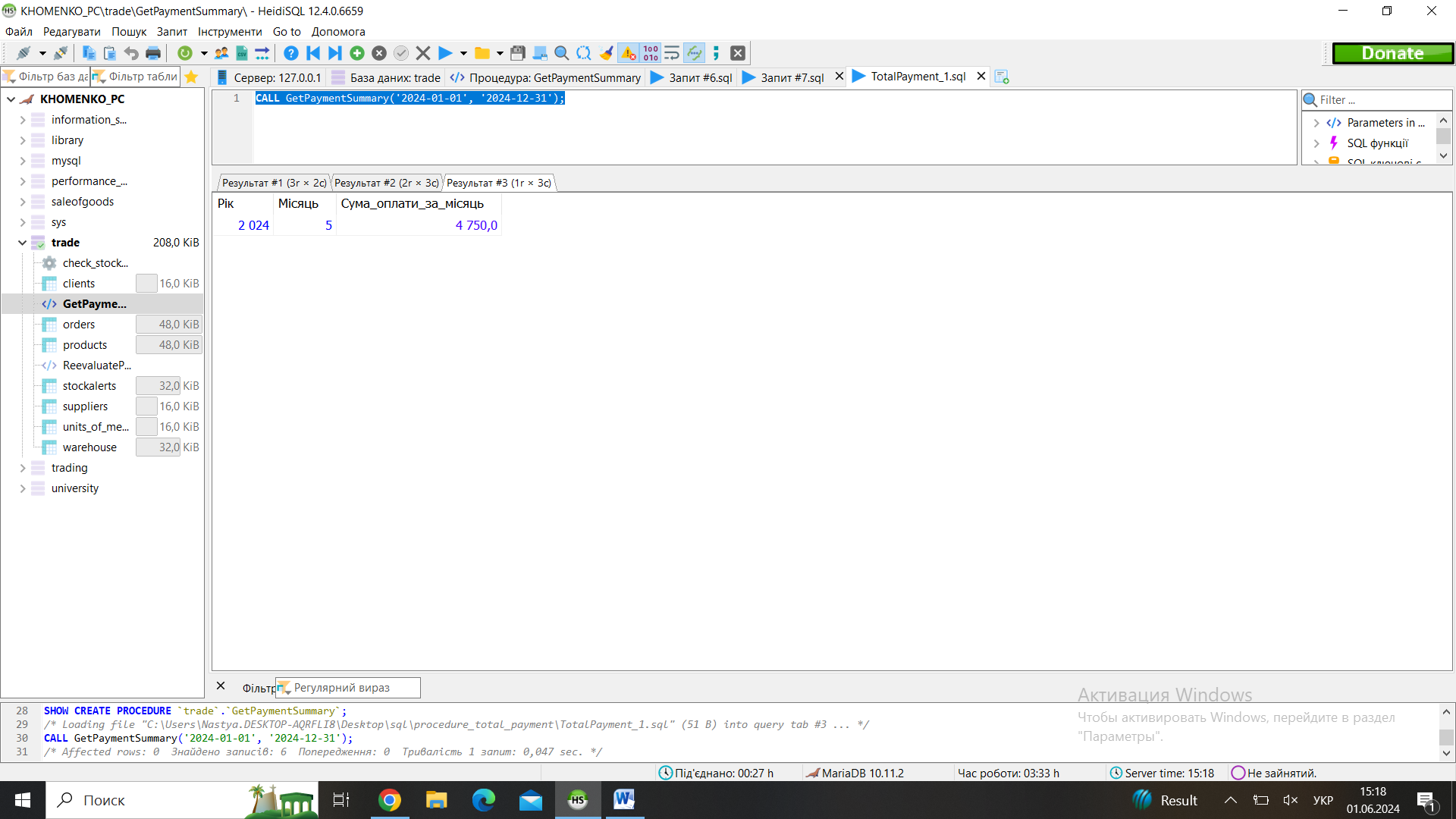


Рисунок 5.14 – Загальна сума оплати за місяць

* + - 1. Переоцінка цін на товари з урахуванням коефіцієнта інфляції

В умовах сучасної економіки**,** переоцінка цін на товари з урахуванням коефіцієнта інфляції є важливим інструментом для підприємств. Інфляція зменшує купівельну спроможність грошей і збільшує витрати на виробництво та реалізацію продукції, що робить переоцінку необхідною для підтримання реалістичної оцінки вартості активів. Це дозволяє уникнути викривлення фінансових показників, оптимізувати податкове навантаження, покращити управління запасами та прийняття стратегічних рішень. Регулярна переоцінка забезпечує підприємствам адаптацію до економічних змін, підвищує ефективність управління ресурсами та підтримує фінансову стабільність у нестабільних економічних умовах.

Щоб організувати переоцінку товарів з урахуванням коефіцієнта інфляції, можна створити збережену процедуру, яка буде виконувати переоцінку цін на товари. Ця процедура буде приймати параметри для коефіцієнта інфляції та способу переоцінки (по категоріям, по всім товарам, по окремому товару), а також буде враховувати чи потрібно збільшити або зменшити ціну.

Параметри процедури:

* inflation\_coefficient: коефіцієнт інфляції для переоцінки;
* category\_name: назва категорії для переоцінки (NULL, якщо не використовується);
* product\_id: Id продукту для переоцінки (NULL, якщо не використовується);
* all\_products: BOOLEAN параметр, який вказує, чи треба переоцінити всі товари;
* increase\_price: BOOLEAN параметр, який вказує, чи потрібно збільшити або зменшити ціну.

Логіка процедури:

Спочатку визначається коефіцієнт для збільшення або зменшення ціни. Потім процедура перевіряє, які товари потрібно переоцінити:

* якщо all\_products = TRUE, переоцінюються всі товари;
* якщо category\_name не NULL, переоцінюються товари зазначеної категорії;
* якщо product\_id не NULL, переоцінюється товар з зазначеним Id.

Виклик процедури:

Для виклику процедури можна використовувати наступні SQL-запити з передачею необхідних параметрів:

CALL ReevaluatePrices(1.05, NULL, NULL, TRUE, TRUE);

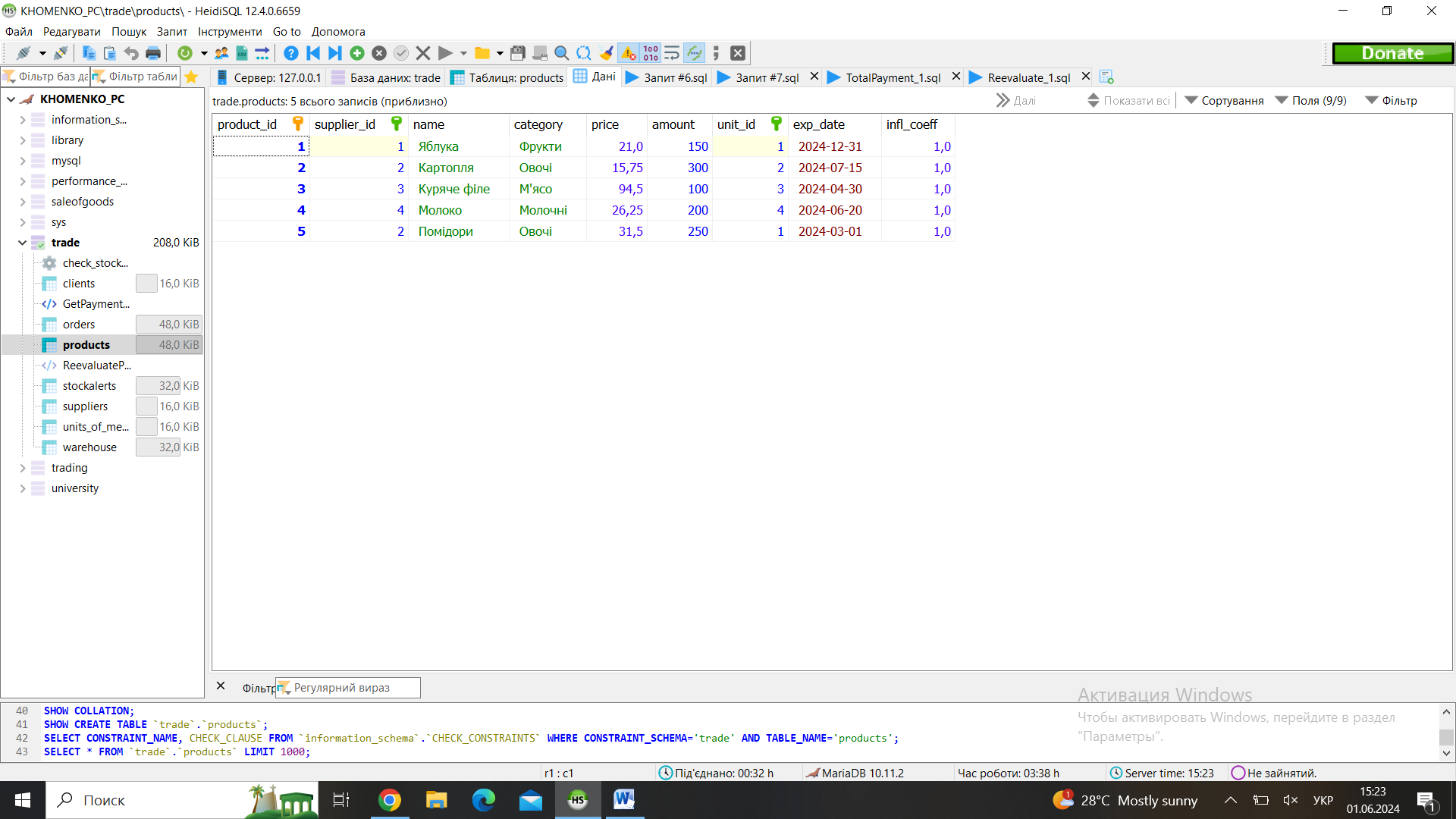


Рисунок 5.15 – Переоцінка всіх товарів із збільшенням цін на 5%

CALL ReevaluatePrices(1.06, 'овочі', NULL, FALSE, FALSE);

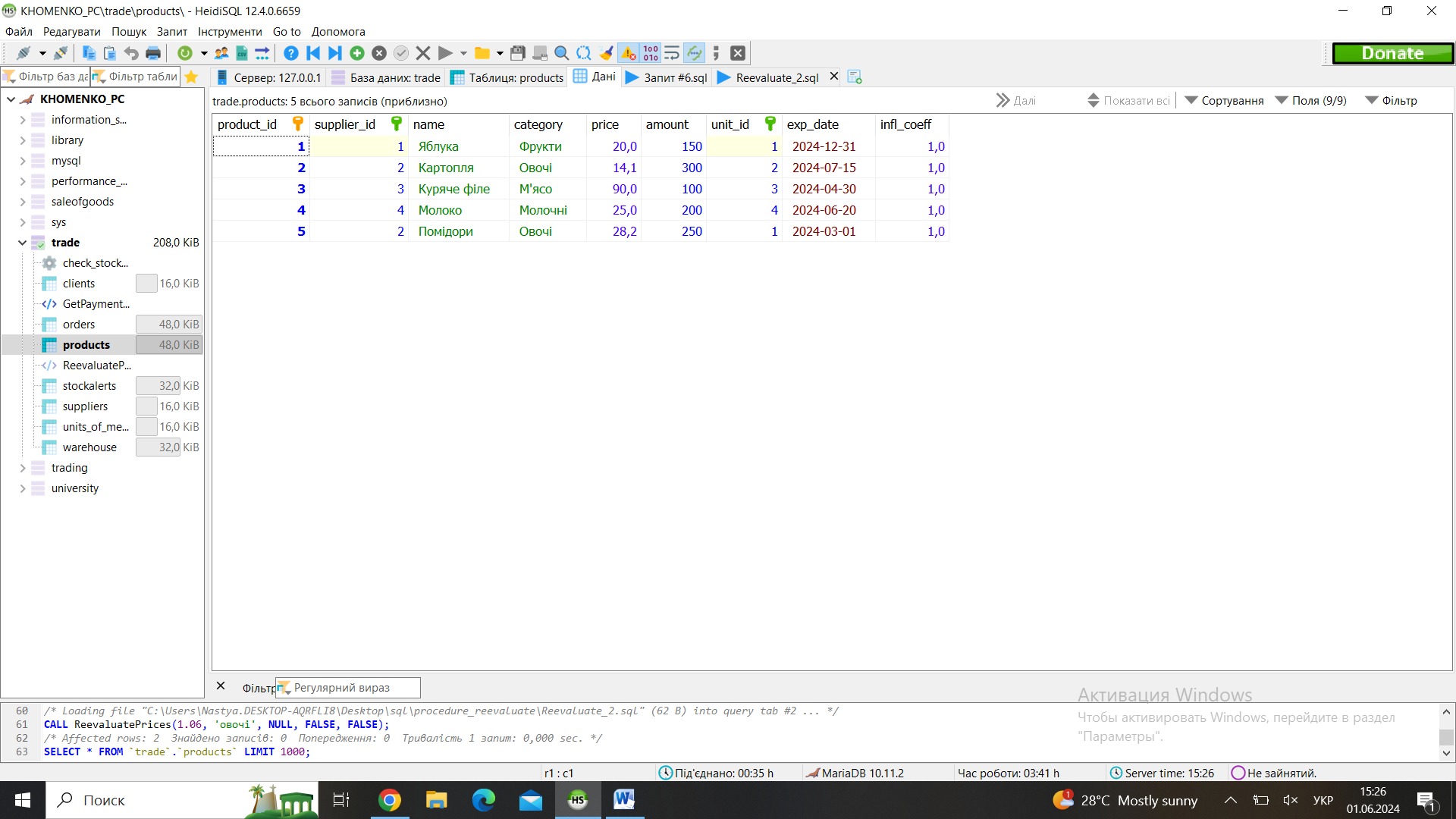


Рисунок 5.16 – Переоцінка всіх товарів категорії "овочі" із зменшенням цін на 6%

CALL ReevaluatePrices(1.02, NULL, 5, FALSE, TRUE);

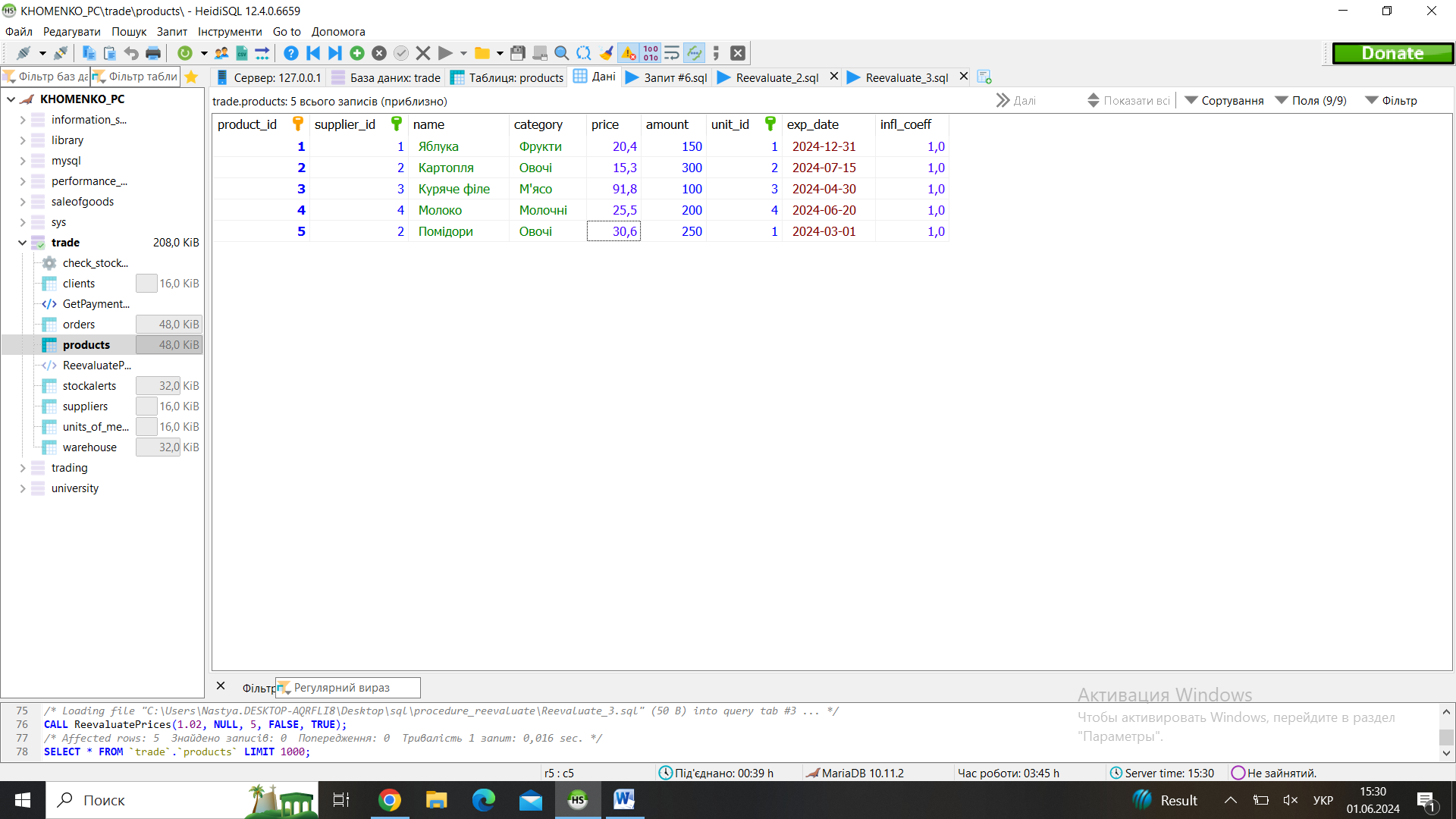


Рисунок 5.17 – Переоцінка окремого товару з ID 5 із збільшенням ціни на 2%

* + 1. Створення тригера для моніторингу запасів на складі

Тригери — це спеціальні види збережених процедур, які автоматично виконуються при настанні певних подій у базі даних, таких як вставка, оновлення або видалення записів у таблиці. Тригери використовуються для автоматичного забезпечення цілісності даних та реалізації бізнес-логіки.

Переваги тригерів:

* виконання дій у відповідь на зміни в базі даних відбувається автоматично;
* тригери можуть перевіряти умови та забезпечувати правильність даних при вставці, оновленні або видаленні;
* тригери можуть автоматично записувати історію змін у базі даних.

Для нашої бази даних створимо тригер, який забезпечує автоматичний контроль за кількістю продуктів на складі та своєчасне повідомлення про необхідність поповнення запасів. Це допомагає уникнути дефіциту товарів та забезпечує безперебійне функціонування підприємства.

Спочатку створимо таблицю StockAlerts для збереження повідомлень про недостатній запас продуктів з наступними атрибутами:

* alert\_id: унікальний ідентифікатор повідомленн;.
* product\_id: ідентифікатор продукту з таблиці Products;
* product\_name: назва продукту з таблиці Products;
* current\_quantity: поточна кількість продукту на складі;
* threshold\_quantity: страховий запас продукту;
* alert\_date: дата і час створення повідомлення.

Тепер створимо тригер, який буде спрацьовувати після оновлення таблиці Warehouse:

Якщо кількість продукту (NEW.кількість) стає меншою за страховий запас (NEW.страховий\_запас), тригер вставляє новий запис в таблицю StockAlerts з інформацією про продукт, його поточну кількість, страховий запас і дату створення повідомлення.

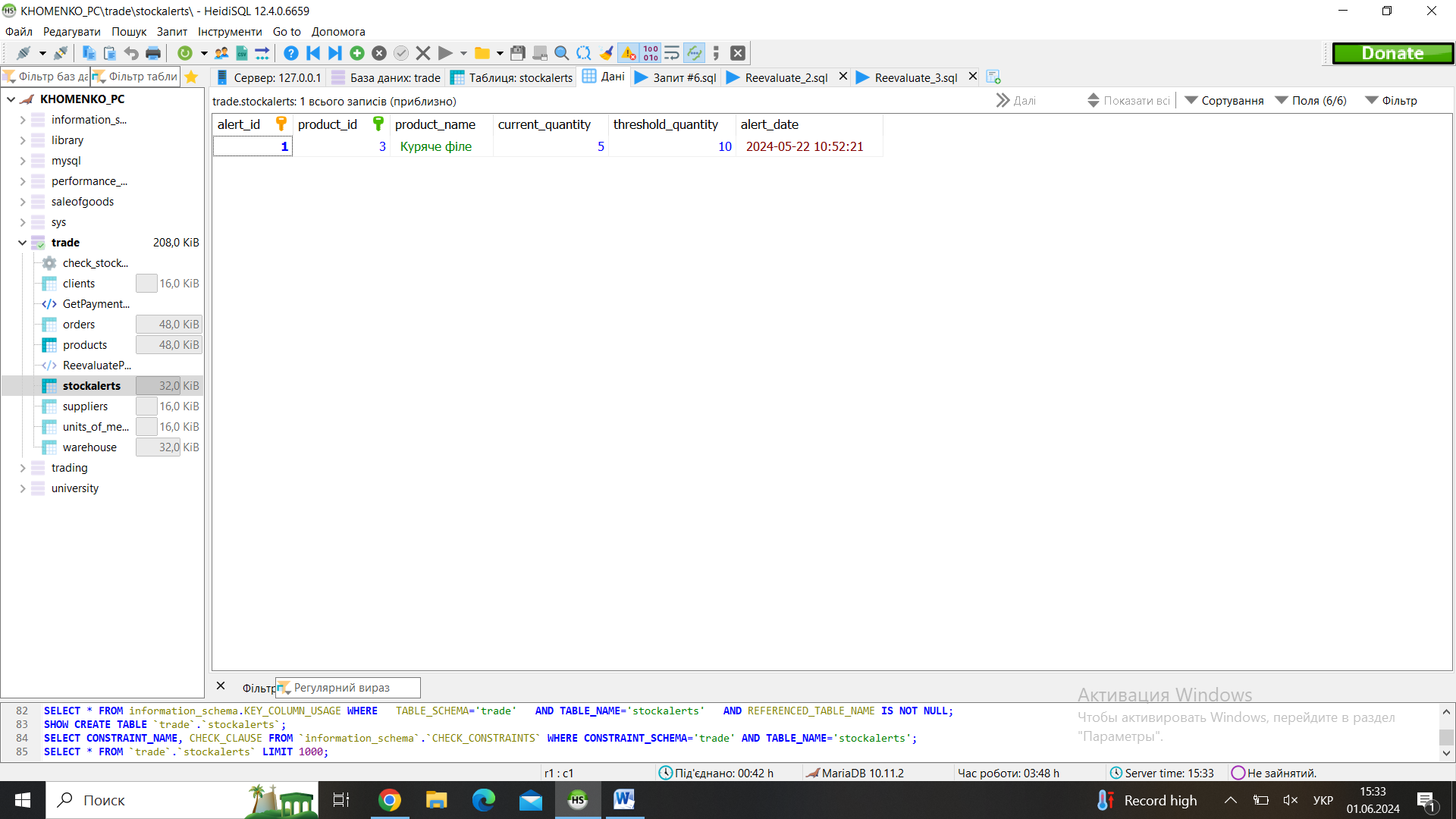


Рисунок 5.18 – Запис в таблиці StockAlerts після оновлення даних в таблиці Warehouse

* 1. Надання прав користувачам бази даних

Надання прав користувачам бази даних є важливим етапом у забезпеченні безпеки та ефективного управління інформаційними системами. Чітке розмежування прав доступу допомагає захистити дані від несанкціонованого доступу, підтримує їх цілісність, сприяє ефективному управлінню ресурсами та забезпечує можливість проведення аудиту.

Створимо кілька користувачів для нашої бази даних та налаштуємо для них різні рівні доступу, що допоможе продемонструвати практичні аспекти управління правами доступу в реальних умовах.

* + 1. Процес надання прав користувачам

У верхньому меню виберемо Інструменти -> Менеджер користувачів. Після цього у вікні менеджера користувачів натиснемо Додати користувача.

Введемо ім'я нового користувача у полі User name та пароль для нового користувача у полях Password і Confirm password.

У полі Host вкажемо % (для доступу з будь-якої IP-адреси). Для обмеженого доступу можна ввести конкретну IP-адресу або домен.

У правій частині вікна виберемо необхідний рівень доступу. Після вибору прав натиснемо «Зберегти».

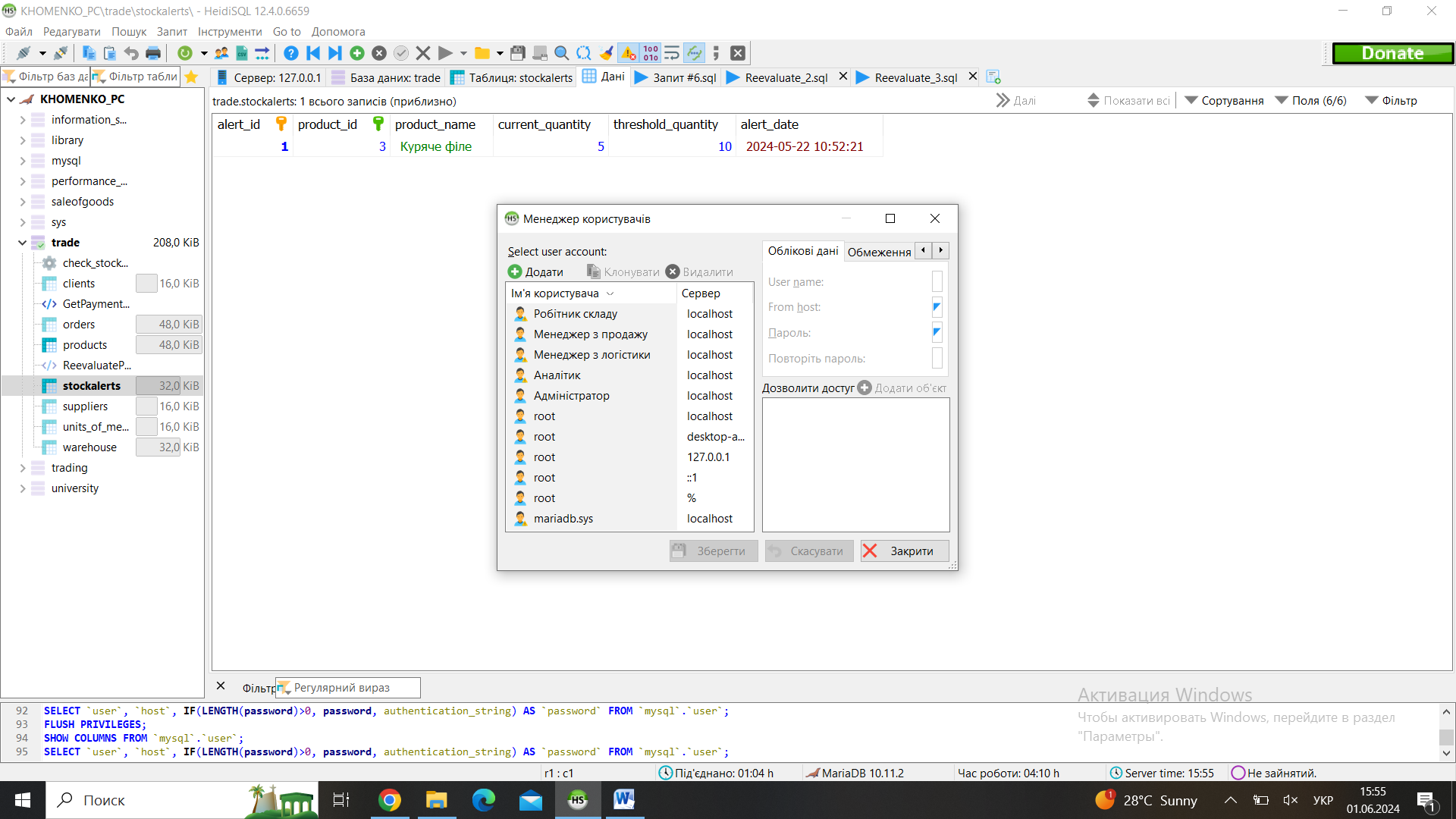


Рисунок 5.19 – Менеджер користувачів

* + 1. Опис створених користувачів та їхніх прав

Таблиця 5.7

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Користувач | Призначення | Права доступу | Таблиці до яких є доступ | Пояснення прав |
| Адміністратор бази даних | Управління всією базою даних, включаючи створення, зміну структури таблиць та управління користувачами. | ALL PRIVILEGES | Вся БД | Може створювати, змінювати та видаляти таблиці та бази даних.  Може призначати права іншим користувачам. |
| Менеджер з продажу | Управління замовленнями, продуктами, клієнтами та звітністю. | SELECT, INSERT, UPDATE | Orders, Products, Customers | Можливість перегляду та редагування замовлень та інформації про клієнтів і продукти. |
| Робітник складу | Управління запасами на складі, контроль за кількістю товарів та створення звітів про поставки. | SELECT, INSERT, UPDATE | Warehouse | Можливість перегляду та редагування інформації про наявність товарів на складі. |
| Менеджер із логістики | Управління поставками, координація логістичних процесів, забезпечення доставки товарів. | SELECT, INSERT, UPDATE | Warehouse, Orders | Можливість перегляду та редагування інформації про складські запаси та замовлення. |
| Аналітик | Аналіз даних, створення звітів та статистичних даних для підтримки бізнес-рішень. | SELECT | Всі таблиці | Можливість тільки переглядати дані для аналізу. |

ВИСНОВКИ

Під час роботи над дипломним проектом було розглянуто низку теоретичних питань:

* поняття дрібнооптової торгівлі;
* автоматизація дрібнооптової торгівлі за допомогою баз даних;
* характеристика серверу MySQL та клієнта HeidiSQL.

На основі аналізу дрібнооптової торгівлі було визначено основні функціональні вимоги для автоматизації цього процесу. Для їх реалізації була виконана значна робота з використанням клієнта HeidiSQL. Практична складова дипломної роботи містить:

* розробку бази даних;
* створення групи SQL-запитів;
* розробка збережених процедур;
* створення тригеру;
* організація групи користувачів з різними правами доступу.

Дипломна робота демонструє, як використання SQL-запитів, збережених процедур та тригерів може значно підвищити ефективність управління запасами, оптимізувати процеси закупівлі та продажу, а також забезпечити контроль за доступом до даних через організацію користувачів з різними рівнями прав доступу. Це дозволяє підвищити точність, швидкість та надійність роботи підприємства, що є критично важливим у конкурентному середовищі сучасного бізнесу.

Крім того, дослідження підкреслює важливість впровадження сучасних технологій в управлінні дрібнооптовою торгівлею, що сприяє зменшенню витрат часу та ресурсів на рутинні операції, а також покращує загальну продуктивність та ефективність бізнесу.

Основні положення дипломної роботи були висвітлені у доповіді на конференції «Молодь у науці», а також опубліковані у статті збірника «Вісник студентського наукового товариства».

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Касянчук Н.В., Ткачук Л.М. Захист інформації в базах даних // Вінницький національний технічний університет. 2019.

URL: <https://core.ac.uk/download/pdf/196301781.pdf>

1. Грищук Н. І. Захист інформації в базах даних на прикладі MySQL // Матеріали Міжнародної науково-практичної конференції «Наука у контексті сучасних глобалізаційних процесів», 2017. С. 55-57
2. Мулеса О. Ю., Варга Я. В. Інформаційні системи та реляційні бази даних : навч. посібник. Ужгород, 2023. 132 с.
3. Хоменко А. Р. Керування правами доступу користувачів до бази даних в HeidiSQl // Матеріали Всеукраїнської науково-практичної конференції «Математичні, природничі та комп’ютерні науки та науки про управління, технології, навчання: науково-практичні рішення та підходи молодих науковців», 2023.

URL: <https://phm.cuspu.edu.ua/ojs/index.php/SNYS/article/view/2047>

1. Біловодська О.А. Маркетингова політика розподілу. Київ: Знання, 2011. 495 с.
2. Пасічник В. В., Резніченко В. А. Організація баз даних і знань. Київ: Видавнича група BHV, 2006. 384 с.
3. Апопій В.В., Міщук І.П., Ребицький В.М., Рудницький С.І., Хом`як Ю.М. Організація торгівлі: Підручник. 2-ге видання., перероб. та доп. Київ: Центр учбової літератури, 2018. 632 с.
4. Урок на тему «Форми торгівлі».

URL: <https://naurok.com.ua/urok-na-temu-formi-torgivli-50551.html> (дата звернення: 04.05.2024)

1. BAS (Business Automation Software)

URL: <https://www.bas-soft.eu/> (дата звернення: 05.05.2024)

1. DuBois P. MySQL. The Complete Reference. McGraw-Hill Osborne Media, 2003. 912 p.
2. Connolly T. M., Begg C. E. Database Systems: A Practical Approach to Design, Implementation, and Management. Pearson, 2014. 1440 p.
3. Download HeidiSQL.

URL: <https://www.heidisql.com/download.php> (дата звернення: 07.05.2024)

1. Download MySQL Community Server.

URL: <https://dev.mysql.com/downloads/mysql/> (дата звернення: 07.05.2024)

1. draw.io

URL: <https://app.diagrams.net/> (дата звернення: 08.05.2024)

1. Date K.J. An introduction to database systems. Addison-Wesley, 2003. 1024 p.

ДОДАТОК А

SQL-файл бази даних для дрібнооптової торгівлі продуктами харчування

-- Dumping database structure for trade

CREATE DATABASE IF NOT EXISTS `trade` /\*!40100 DEFAULT CHARACTER SET cp1251 COLLATE cp1251\_ukrainian\_ci \*/;

USE `trade`;

-- Dumping structure for таблиця trade.clients

CREATE TABLE IF NOT EXISTS `clients` (

`client\_id` int(12) NOT NULL,

`name` varchar(30) NOT NULL DEFAULT '',

`phone\_number` varchar(20) NOT NULL DEFAULT '',

`account` varchar(30) NOT NULL DEFAULT '',

`address` varchar(60) NOT NULL DEFAULT '',

PRIMARY KEY (`client\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=cp1251 COLLATE=cp1251\_ukrainian\_ci;

-- Dumping data for table trade.clients: ~5 rows (приблизно)

INSERT INTO `clients` (`client\_id`, `name`, `phone\_number`, `account`, `address`) VALUES

(1, 'ТОВ "Продукти"', '380501234567', 'UA1234567890123456789', 'м. Київ, вул. Хрещатик, 10'),

(2, 'ФОП "Петренко А.О."', '380501234568', 'UA9876543210987654321', 'м. Львів, вул. Городоцька, 15'),

(3, 'ТОВ "Смак"', '380501234569', 'UA1122334455667788990 ', 'м. Одеса, вул. Дерибасівська, 5'),

(4, 'ФОП "Іванов І.І."', '380501234570', 'UA2233445566778899001', 'м. Харків, просп. Науки, 30'),

(5, 'ТОВ "ЕкоПродукт"', '380501234571 ', 'UA3344556677889900122', 'м. Дніпро, вул. Центральна, 20');

-- Dumping structure for процедура trade.GetPaymentSummary

DELIMITER //

CREATE PROCEDURE `GetPaymentSummary`(IN startDate DATE, IN endDate DATE)

BEGIN

-- Загальна сума за день

SELECT

DATE(payment\_date) AS День,

SUM(Products.price \* Orders.amount) AS Сума\_оплати\_за\_день

FROM

Orders

JOIN

Products ON Orders.product\_id = Products.product\_id

WHERE

Orders.payment\_date BETWEEN startDate AND endDate

GROUP BY

DATE(payment\_date);

-- Загальна сума за тиждень

SELECT

YEAR(payment\_date) AS Рік,

WEEK(payment\_date) AS Тиждень,

SUM(Products.price \* Orders.amount) AS Сума\_оплати\_за\_тиждень

FROM

Orders

JOIN

Products ON Orders.product\_id = Products.product\_id

WHERE

Orders.payment\_date BETWEEN startDate AND endDate

GROUP BY

Рік, Тиждень;

-- Загальна сума за місяць

SELECT

YEAR(payment\_date) AS Рік,

MONTH(payment\_date) AS Місяць,

SUM(Products.price \* Orders.amount) AS Сума\_оплати\_за\_місяць

FROM

Orders

JOIN

Products ON Orders.product\_id = Products.product\_id

WHERE

Orders.payment\_date BETWEEN startDate AND endDate

GROUP BY

Рік, Місяць;

END//

DELIMITER ;

-- Dumping structure for таблиця trade.orders

CREATE TABLE IF NOT EXISTS `orders` (

`order\_id` int(12) NOT NULL,

`client\_id` int(12) NOT NULL,

`product\_id` int(12) NOT NULL,

`amount` int(10) NOT NULL,

`order\_date` date NOT NULL,

`payment\_date` date DEFAULT NULL,

`paid` tinyint(1) unsigned zerofill NOT NULL DEFAULT 0,

PRIMARY KEY (`order\_id`),

KEY `FK\_\_clients` (`client\_id`),

KEY `FK\_order\_products` (`product\_id`),

CONSTRAINT `FK\_\_clients` FOREIGN KEY (`client\_id`) REFERENCES `clients` (`client\_id`) ON DELETE NO ACTION ON UPDATE NO ACTION,

CONSTRAINT `FK\_order\_products` FOREIGN KEY (`product\_id`) REFERENCES `products` (`product\_id`) ON DELETE NO ACTION ON UPDATE NO ACTION

) ENGINE=InnoDB DEFAULT CHARSET=cp1251 COLLATE=cp1251\_ukrainian\_ci;

-- Dumping data for table trade.orders: ~5 rows (приблизно)

INSERT INTO `orders` (`order\_id`, `client\_id`, `product\_id`, `amount`, `order\_date`, `payment\_date`, `paid`) VALUES

(1, 1, 1, 50, '2024-05-01', '2024-05-02', 1),

(2, 2, 3, 20, '2024-05-03', '0000-00-00', 0),

(3, 3, 2, 100, '2024-05-05', '2024-05-07', 1),

(4, 4, 5, 75, '2024-05-08', '2024-05-08', 1),

(5, 5, 4, 40, '2024-05-10', '0000-00-00', 0);

-- Dumping structure for таблиця trade.products

CREATE TABLE IF NOT EXISTS `products` (

`product\_id` int(12) NOT NULL,

`supplier\_id` int(12) NOT NULL,

`name` varchar(20) NOT NULL DEFAULT '',

`category` varchar(20) NOT NULL DEFAULT '',

`price` decimal(5,2) NOT NULL DEFAULT 0.00,

`amount` int(11) NOT NULL DEFAULT 0,

`unit\_id` int(11) NOT NULL DEFAULT 0,

`exp\_date` date NOT NULL,

`infl\_coeff` decimal(5,2) NOT NULL DEFAULT 1.00,

PRIMARY KEY (`product\_id`),

KEY `FK\_\_suppliers` (`supplier\_id`),

KEY `FK\_products\_units\_of\_measurement` (`unit\_id`),

CONSTRAINT `FK\_\_suppliers` FOREIGN KEY (`supplier\_id`) REFERENCES `suppliers` (`supplier\_id`) ON DELETE NO ACTION ON UPDATE NO ACTION,

CONSTRAINT `FK\_products\_units\_of\_measurement` FOREIGN KEY (`unit\_id`) REFERENCES `units\_of\_measurement` (`unit\_id`) ON DELETE NO ACTION ON UPDATE NO ACTION

) ENGINE=InnoDB DEFAULT CHARSET=cp1251 COLLATE=cp1251\_ukrainian\_ci;

-- Dumping data for table trade.products: ~5 rows (приблизно)

INSERT INTO `products` (`product\_id`, `supplier\_id`, `name`, `category`, `price`, `amount`, `unit\_id`, `exp\_date`, `infl\_coeff`) VALUES

(1, 1, 'Яблука', 'Фрукти', 20.40, 150, 1, '2024-12-31', 1.00),

(2, 2, 'Картопля', 'Овочі', 15.30, 300, 2, '2024-07-15', 1.00),

(3, 3, 'Куряче філе', 'М\'ясо', 91.80, 100, 3, '2024-04-30', 1.00),

(4, 4, 'Молоко', 'Молочні', 25.50, 200, 4, '2024-06-20', 1.00),

(5, 2, 'Помідори', 'Овочі', 30.60, 250, 1, '2024-03-01', 1.00);

-- Dumping structure for процедура trade.ReevaluatePrices

DELIMITER //

CREATE PROCEDURE `ReevaluatePrices`(

IN `inflation\_coefficient` DECIMAL(5,2),

IN `category\_name` VARCHAR(255),

IN `product\_id` INT,

IN `all\_products` BOOLEAN,

IN `increase\_price` BOOLEAN

)

BEGIN

DECLARE factor DECIMAL(5,2);

IF increase\_price THEN

SET factor = inflation\_coefficient;

ELSE

SET factor = 1 / inflation\_coefficient;

END IF;

-- Переоцінка по всім товарам

IF all\_products THEN

UPDATE Products

SET price = price \* factor;

END IF;

-- Переоцінка по категоріям

IF category\_name IS NOT NULL THEN

UPDATE Products

SET price = price \* factor

WHERE category = category\_name;

END IF;

-- Переоцінка по окремому товару

IF product\_id IS NOT NULL THEN

UPDATE Products

SET price = price \* factor

WHERE product\_id = product\_id;

END IF;

END//

DELIMITER ;

-- Dumping structure for таблиця trade.stockalerts

CREATE TABLE IF NOT EXISTS `stockalerts` (

`alert\_id` int(11) NOT NULL AUTO\_INCREMENT,

`product\_id` int(11) DEFAULT NULL,

`product\_name` char(50) DEFAULT NULL,

`current\_quantity` int(11) DEFAULT NULL,

`threshold\_quantity` int(11) DEFAULT NULL,

`alert\_date` timestamp NULL DEFAULT current\_timestamp(),

PRIMARY KEY (`alert\_id`),

KEY `product\_id` (`product\_id`),

CONSTRAINT `stockalerts\_ibfk\_1` FOREIGN KEY (`product\_id`) REFERENCES `products` (`product\_id`)

) ENGINE=InnoDB AUTO\_INCREMENT=2 DEFAULT CHARSET=cp1251 COLLATE=cp1251\_ukrainian\_ci;

-- Dumping data for table trade.stockalerts: ~1 rows (приблизно)

INSERT INTO `stockalerts` (`alert\_id`, `product\_id`, `product\_name`, `current\_quantity`, `threshold\_quantity`, `alert\_date`) VALUES

(1, 3, 'Куряче філе', 5, 10, '2024-05-22 07:52:21');

-- Dumping structure for таблиця trade.suppliers

CREATE TABLE IF NOT EXISTS `suppliers` (

`supplier\_id` int(12) NOT NULL,

`name` varchar(30) NOT NULL DEFAULT '',

`phone\_number` varchar(20) NOT NULL DEFAULT '',

`account` varchar(30) NOT NULL DEFAULT '',

`address` varchar(60) NOT NULL DEFAULT '',

PRIMARY KEY (`supplier\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=cp1251 COLLATE=cp1251\_ukrainian\_ci;

-- Dumping data for table trade.suppliers: ~5 rows (приблизно)

INSERT INTO `suppliers` (`supplier\_id`, `name`, `phone\_number`, `account`, `address`) VALUES

(1, 'ТОВ "Фрукти Україна"', '380501234572', 'UA4455667788990011223', 'м. Вінниця, вул. Соборна, 12'),

(2, 'ТОВ "Овочі Поділля"', '380501234573', 'UA5566778899001122334', 'м. Хмельницький, вул. Довженка, 18'),

(3, 'ТОВ "М\'ясо Центр"', '380501234574', 'UA6677889900112233445 ', 'м. Житомир, вул. Михайлівська, 22'),

(4, 'ТОВ "Молоко Полісся"', '380501234575 ', 'UA7788990011223344556', 'м. Рівне, вул. Відінська, 10'),

(5, 'ТОВ "Овочі Півдня"', '380501234576', 'UA8899001122334455667', 'м. Миколаїв, вул. Шевченка, 14');

-- Dumping structure for таблиця trade.units\_of\_measurement

CREATE TABLE IF NOT EXISTS `units\_of\_measurement` (

`unit\_id` int(12) NOT NULL,

`name` varchar(20) NOT NULL DEFAULT '',

`abbreviation` varchar(20) NOT NULL DEFAULT '',

`description` varchar(50) NOT NULL DEFAULT '',

PRIMARY KEY (`unit\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=cp1251 COLLATE=cp1251\_ukrainian\_ci;

-- Dumping data for table trade.units\_of\_measurement: ~5 rows (приблизно)

INSERT INTO `units\_of\_measurement` (`unit\_id`, `name`, `abbreviation`, `description`) VALUES

(1, 'Кілограм', 'кг', 'Одиниця маси'),

(2, 'Центнер', 'ц', 'Одиниця маси, рівна 100 кг'),

(3, 'Штука', 'шт', 'Одиниця кількості'),

(4, 'Літр', 'л', 'Одиниця об\'єму'),

(5, 'Упаковка', 'уп', 'Одиниця кількості в пакуванні');

-- Dumping structure for таблиця trade.warehouse

CREATE TABLE IF NOT EXISTS `warehouse` (

`product\_id` int(12) NOT NULL,

`amount` int(12) NOT NULL,

`reserve` int(12) NOT NULL,

`delivery\_date` date NOT NULL,

`date\_of\_last\_sale` date NOT NULL,

KEY `FK\_\_products` (`product\_id`),

CONSTRAINT `FK\_\_products` FOREIGN KEY (`product\_id`) REFERENCES `products` (`product\_id`) ON DELETE NO ACTION ON UPDATE NO ACTION

) ENGINE=InnoDB DEFAULT CHARSET=cp1251 COLLATE=cp1251\_ukrainian\_ci;

-- Dumping data for table trade.warehouse: ~4 rows (приблизно)

INSERT INTO `warehouse` (`product\_id`, `amount`, `reserve`, `delivery\_date`, `date\_of\_last\_sale`) VALUES

(1, 10, 20, '2024-04-25', '2024-05-02'),

(2, 15, 30, '2024-04-28', '2024-05-06'),

(3, 5, 10, '2024-04-20', '2024-05-04'),

(4, 8, 25, '2024-04-22', '2024-05-10'),

(5, 250, 35, '2024-04-30', '2024-05-08');

-- Dumping structure for тригер trade.check\_stock\_after\_update

SET @OLDTMP\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='STRICT\_TRANS\_TABLES,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_AUTO\_CREATE\_USER,NO\_ENGINE\_SUBSTITUTION';

DELIMITER //

CREATE TRIGGER check\_stock\_after\_update

AFTER UPDATE ON Warehouse

FOR EACH ROW

BEGIN

IF NEW.amount < NEW.reserve THEN

INSERT INTO StockAlerts (product\_id, product\_name, current\_quantity, threshold\_quantity)

VALUES (NEW.product\_id, (SELECT name FROM Products WHERE product\_id = NEW.product\_id), NEW.amount, NEW.reserve);

END IF;

END//

DELIMITER ;

SET SQL\_MODE=@OLDTMP\_SQL\_MODE;