НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

(повна назва інституту/факультету)

КАФЕДРА інформатики та програмної інженерії

(повна назва кафедри)

## КУРСОВА РОБОТА

з дисципліни «Бази даних»

(назва дисципліни)

на тему: “База даних для підтримки діяльності авторинку”

Студента 2 курсу ІП-13 групи спеціальності 121 «Інженерія

програмного забезпечення» Недельчева Є. О.

(прізвище та ініціали)

Керівник Ліщук Олександр Васильович

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна шкала

Кількість балів: Оцінка ECTS

Члени комісії

(підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

(підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

(підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

Київ – 2022 рік

## Національний технічний університет України

**«Київський політехнічний інститут імені Ігоря Сікорського»**

Факультет Інформатики та обчислювальної техніки

(повна назва)

Кафедра Інформатики та програмної інженерії

(повна назва)

Дисципліна Бази даних

Курс 2 Група ІП-13 Семестр 3

# З А В Д А Н Н Я

**НА КУРСОВУ РОБОТУ СТУДЕНТУ**

Недельчеву Євгену Олександровичу

(прізвище, ім’я, по батькові)

1. Тема роботи “ База даних для підтримки діяльності авторинку ” керівник роботи: Ліщук Олександр Васильович

(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

1. Строк подання студентом роботи 10.01.2023
2. Вихідні дані до роботи завдання на розробку бази даних підтримки для підтримки діяльності авторинку
3. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно

розробити

* 1. Аналіз предметного середовища
  2. Побудова ER-моделі
  3. Побудова реляційної схеми з ER-моделі
  4. Створення бази даних, у форматі обраної СУБД
  5. Створення користувачів бази даних
  6. Імпорт даних з використанням засобів СУБД в створену базу даних
  7. Створення мовою SQL запитів
  8. Оптимізація роботи запитів

1. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
2. Дата видачі завдання 08.11.2022

## КАЛЕНДАРНИЙ ПЛАН

|  |  |  |  |
| --- | --- | --- | --- |
| №  з/п | Назва етапів виконання курсового проекту | Строк виконання етапі проекту | Примітка |
| 1 | Аналіз предметного середовища | 29.12.2022 |  |
| 2 | Побудова ER-моделі | 29.12.2022 |  |
| 3 | Побудова реляційної схеми з ER-моделі | 29.12.2022 |  |
| 4 | Створення бази даних, у форматі обраної  системи управління базою даних | 01.01.2023 |  |
| 5 | Створення користувачів бази даних | 01.01.2023 |  |
| 6 | Імпорт даних з використанням засобів СУБД в створену базу даних | 02.01.2023 |  |
| 7 | Створення мовою SQL запитів | 04.01.2023 |  |
| 8 | Оптимізація роботи запитів | 04.01.2023 |  |
| 9 | Оформлення пояснювальної записки | 06.01.2023 |  |
| 10 | Захист курсової роботи | 11.01.2023 |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Студент**  Недельчев Є.О.

(підпис ) (прізвище та ініціали)

**Керівник роботи**  Ліщук О.В.

(підпис ) (прізвище та ініціали)

# Зміст

[ВСТУП 5](#_bookmark0)

[ОПИС ПРЕДМЕТНОГО СЕРЕДОВИЩА 6](#_bookmark1)

[ПОСТАНОВКА ЗАДАЧІ 7](#_bookmark2)

[ER-діаграма 9](#_bookmark3)

[Бізнес правила 9](#_bookmark4)

[Вибір сутностей 9](#_bookmark5)

[Набор атрибутів сутностей 10](#_bookmark6)

[РЕЛЯЦІЙНА МОДЕЛЬ БАЗИ ДАНИХ 14](#_bookmark7)

[Побудова необхідних відношень, визначення первинних та зовнішніх ключів.](#_bookmark8)

[................................................................................................................................. 14](#_bookmark8)

[РЕАЛІЗАЦІЯ БАЗИ ДАНИХ 15](#_bookmark9)

[Створення бази даних 15](#_bookmark10)

[Імпортування даних 19](#_bookmark11)

[СТВОРЕННЯ КОРИСТУВАЧІВ БАЗИ ДАНИХ 20](#_bookmark12)

[Розробник 20](#_bookmark13)

[Продавець 20](#_bookmark14)

[Покупець 20](#_bookmark15)

[SQL запити 21](#_bookmark16)

[Створення тригерів на таблиці 21](#_bookmark17)

[Створення процедур 25](#_bookmark18)

[Створення функцій 27](#_bookmark19)

[Створення представлень 29](#_bookmark20)

[Створення різних запитів 34](#_bookmark21)

[Створення індексів 42](#_bookmark22)

[ВИСНОВОК 43](#_bookmark23)

[ПЕРЕЛІК ПОСИЛАНЬ 44](#_bookmark24)

## ВСТУП

У сучасному світі бази даних відіграють важливу роль у зберіганні, організації та управлінні великими обсягами даних. Вони є важливою частиною сучасних обчислювальних систем, які використовуються в широкому діапазоні галузей і застосувань.

Бази даних дозволяють організаціям ефективно зберігати та отримувати великі обсяги структурованих даних, наприклад інформацію про клієнтів, записи про продажі та дані про запаси. Вони також надають інструменти для організації, обробки й аналізу даних, що дозволяє отримувати цінну інформацію та приймати обґрунтовані рішення.

Крім того, бази даних є центральним сховищем даних, що забезпечує їх послідовність, точність і актуальність. Вони також підтримують такі заходи безпеки, як автентифікація користувачів і шифрування даних для захисту конфіденційної інформації від несанкціонованого доступу.

У цій курсовій роботі представлена база даних для підтримки діяльності авторинку.

## ОПИС ПРЕДМЕТНОГО СЕРЕДОВИЩА

Авторинок – це місце, де люди можуть продати свій автомобіль, або придбати інший. Зазвичай авторинки знаходяться на великих відкритих площадках, де володарі можуть зручно розташувати свої авто. Зазвичай угоди про купівлю/продаж укладаються просто на місці.

Перед тим, як виставити свій автомобіль на продаж, продавці надають детальну інформацію про свої автівки, щоб покупці могли зробити правильний вибір. Покупці ж перед придбанням авто детально ознайомлюються з хар-ками бажаних авто.

Предметне середовище для авторинку містить різні об'єкти та сутності, які пов'язані з його функціонуванням. Основні об'єкти, які можуть бути включені до такого середовища:

* Автомобілі
* Характеристики авто (двигун, ходова частина)
* Володарі авто
* Покупці

## ПОСТАНОВКА ЗАДАЧІ

Метою даної роботи є розробка бази даних для підтримки діяльності кіностудії. Результатом проектування повинна бути БД з визначеною структурою, заповнена даними та оптимізована для потреб користувача. Необхідно створити об’єкти, які покращать роботу розробника та користувача.

Завданням курсової роботи є розробка бази даних і її використання для вирішення практичних задач.

При розробці бази даних необхідно враховувати:

* вимоги до функціональності (наявність усіх функцій, які необхідні для реалізації поставленої задачі);
* вимоги до цілісності даних;
* вимоги до мінімізації об’єму даних, що зберігаються;
* наявність багатокористувальницького режиму;
* вимоги до швидкодії.

В процесі роботи над курсовою роботою повинні бути виконані наступні завдання:

* побудувати ER-модель, для чого необхідно:
  + детально проаналізувати предметне середовище;
  + сформулювати бізнес-правила, які будуть основою завдання обмежень при проектуванні та реалізації бази даних;
  + виявити необхідний набір сутностей;
  + визначити необхідний набір атрибутів для кожної сутності; - визначити зв'язки між об'єктами;
  + описати отриману ER-модель в одній з відомих нотацій;
  + розробити модель користувачів бази даних з описом їх прав;
* побудувати реляційну схему з ER-моделі, для чого необхідно:
  + побудувати набір необхідних відношень бази даних;
  + виділити первинні і зовнішні ключі у кожному з відношень;
  + привести отримані відношення до третьої нормальної формі;
  + визначити обмеження цілісності для спроектованих відношень;
* створити базу даних, що була спроектована, у форматі обраної системи управління базою даних (СУБД);
* створити користувачів бази даних, реалізувавши розроблену багатокористувальницьку модель;
* імпортувати дані з використанням засобів СУБД в створену базу даних;
* мовою SQL написати запити для визначених на етапі аналізу предметного середовища потреб користувачів;
* оптимізувати роботу запитів (продемонструвати роботу до і після оптимізації).

## ER-діаграма

Бізнес правила

Для правильної роботи БД необхідно ввести певні бізнес-правила:

1. Кожен автомобіль може бути одночасно виставлений на продаж лише один раз.
2. Кожен продавець може виставити на продаж будь-яку кількість автівок.
3. Кожен покупець може купити будь-яку кількість автівок.
4. Кожна автівка може мати декілька полісів страхування, в тому числі протерміновані, а може не мати жодного.

Вибір сутностей

Вибір сутностей для Бази Даних виглядатиме наступним чином:

* Car\_accidents
* Engine
* Transmission
* Insurance
* Sellers
* Cars
* Listings
* Car\_types
* Transaction
* Buyers

Набор атрибутів сутностей

Таблиця 3.1 – Сутності та їх атрибути.

|  |  |
| --- | --- |
| Сутність | Атрибути |
| Сar\_accidents | **accident\_id** car\_id event  date\_of\_event |
| Engine | **engine\_id** engine\_type horse\_power engine\_size |
| Transmission | **transmission\_id** transmission\_type type\_of\_drive |
| Insurance | **insurance\_id** car\_id provider policy\_number start\_date end\_date |

Продовження таблиці 3.1

|  |  |
| --- | --- |
| Cars | **car\_id** engine\_id transmission\_id type\_id  brand model color year mileage |
| Sellers | **seller\_id** first\_name last\_name email phone |
| Listing | **listing\_id** seller\_id car\_id price listing\_date |
| Car\_type | **type\_id** typename amount\_of\_doors amount\_of\_seats |

Продовження таблиці 3.1

|  |  |
| --- | --- |
| Transaction | **transaction\_id** listing\_id buyer\_id  price transaction\_date |
| Buyer | **buyer\_id** first\_name last\_name email phone |

Сутність Car\_accident буде пов’язана **багато до одного** із сутністю Car, адже кожна автівка може мати декілька ДТП.

Сутність Insurance буде пов’язана **багато до одного** із сутністю Car, адже кожна автівка може мати декілька страхових полісів.

Сутність Car буде пов’язана **багато до одного** із сутністю Engine, адже декілька автівок можуть мати однаковий тип двигуна.

Сутність Car буде пов’язана **багато до одного** із сутністю Transmission, адже декілька автівок можуть мати однаковий тип трансмісії.

Сутність Car буде пов’язана **багато до одного** із сутністю Car\_types, адже декілька автівок можуть мати однаковий тип кузова.

Сутність Car буде пов’язана **один до одного** із сутністю Listing, адже одна автівка може бути виставлена на продаж лише 1 раз.

Сутність Listing буде пов’язана **багато до одного** із сутністю Seller, адже кожен продавець може мати декілька об’яв про продаж авто.

Сутність Transaction буде пов’язана **один до одного** із сутністю Listing, адже кожна об’ява про продаж може мати лише одну угоду.

Сутність Transaction буде пов’язана **один до багатьох** із сутністю Buyer, адже кожен покупець може укласти декілька угод про купівлю авто.

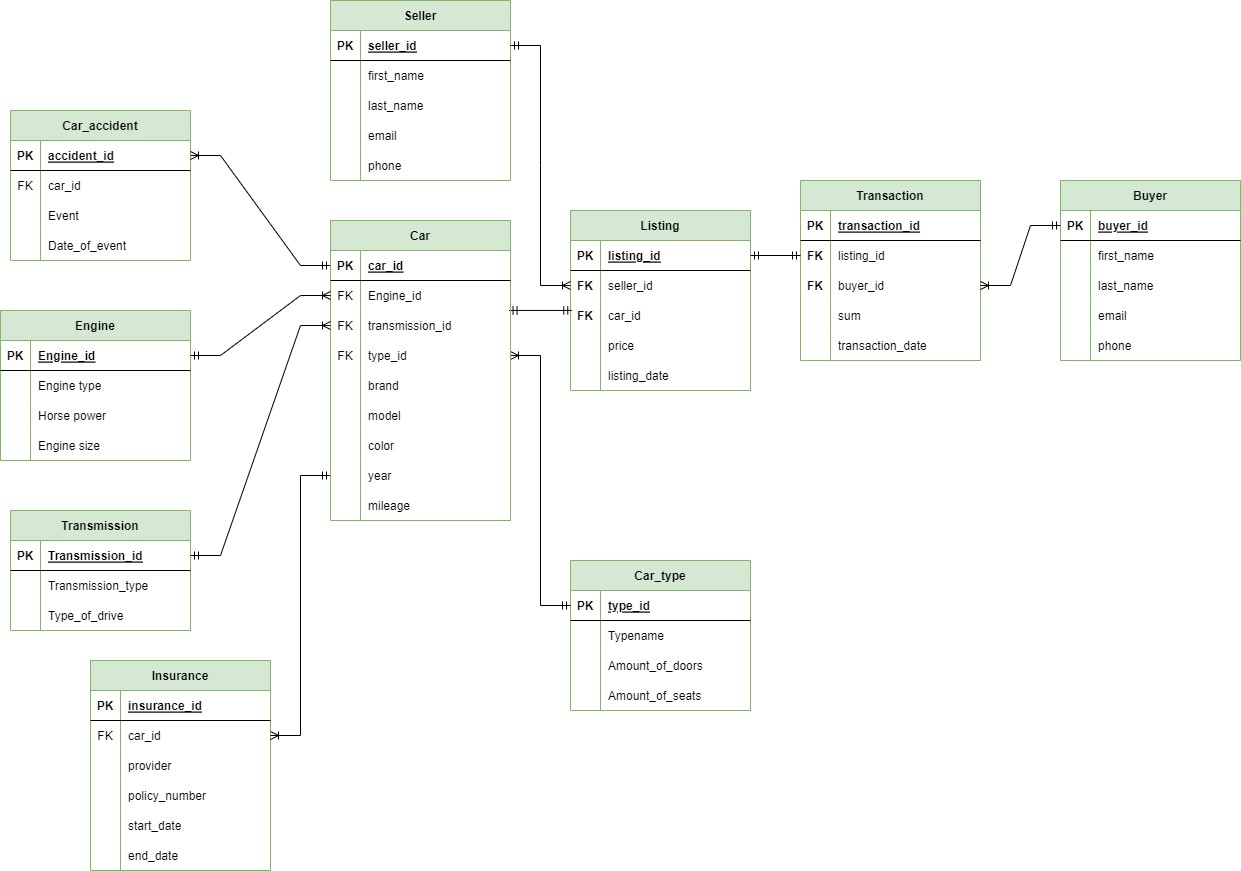


Рисунок 3.1 – ER-діаграма

## РЕЛЯЦІЙНА МОДЕЛЬ БАЗИ ДАНИХ

Побудова необхідних відношень, визначення первинних та зовнішніх ключів.

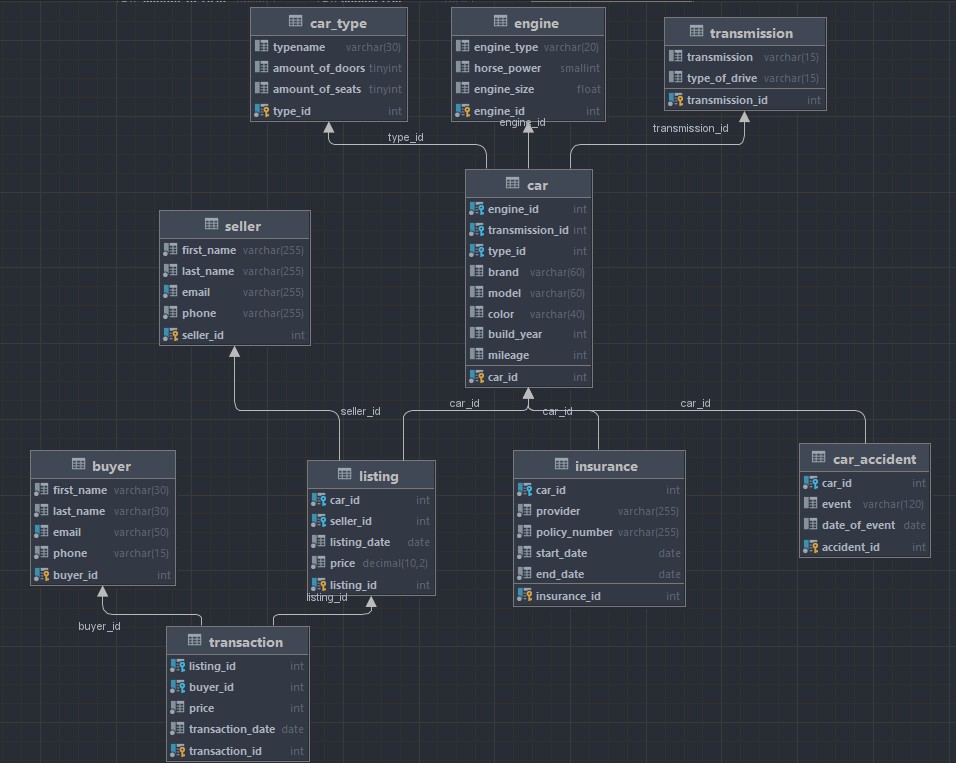


Рисунок 4.1 – Реляційна схема бази даних

На даній схемі видно, що база даних знаходиться у 3 нормальній формі, адже всі поля таблиць декомпозовані, також всі атрибути таблиць функціонально повно залежать від первинного ключа, кожен неключовий атрибут не є транзитивно залежним від первинного ключа.

1. Обов’язкові атрибути таблиць мають обмеження NOT NULL, для запобіганню помилок при роботі з даними.
2. Забезпечуються каскадні дії при видаленні зовнішних ключів однієї з таблиць (ON DELETE CASCADE).

## РЕАЛІЗАЦІЯ БАЗИ ДАНИХ

Створення бази даних

DROP DATABASE IF EXISTS car\_market; CREATE DATABASE car\_market;

use car\_market;

SET foreign\_key\_checks = 0; DROP TABLE IF EXISTS buyer; DROP TABLE IF EXISTS seller;

DROP TABLE IF EXISTS `engine`; DROP TABLE IF EXISTS transmission; DROP TABLE IF EXISTS car\_type; DROP TABLE IF EXISTS car\_accident; DROP TABLE IF EXISTS insurance; DROP TABLE IF EXISTS car;

DROP TABLE IF EXISTS listing;

DROP TABLE IF EXISTS `transaction`; SET foreign\_key\_checks = 1;

CREATE TABLE buyer (

buyer\_id INT NOT NULL AUTO\_INCREMENT,

first\_name VARCHAR(30) NOT NULL, last\_name VARCHAR(30) NOT NULL, email VARCHAR(50) NOT NULL, phone VARCHAR(15) NOT NULL,

PRIMARY KEY (buyer\_id)

);

ALTER TABLE buyer ADD CONSTRAINT email\_unique UNIQUE (email);

CREATE TABLE seller (

seller\_id INT NOT NULL AUTO\_INCREMENT,

first\_name VARCHAR(255) NOT NULL, last\_name VARCHAR(255) NOT NULL, email VARCHAR(255) NOT NULL, phone VARCHAR(255) NOT NULL,

PRIMARY KEY (seller\_id)

);

ALTER TABLE seller ADD CONSTRAINT email\_unique UNIQUE (email);

CREATE TABLE `engine` (

engine\_id INT NOT NULL AUTO\_INCREMENT,

engine\_type VARCHAR(20), horse\_power SMALLINT, engine\_size FLOAT, PRIMARY KEY (engine\_id)

);

CREATE TABLE transmission (

transmission\_id INT NOT NULL AUTO\_INCREMENT, transmission VARCHAR(15),

type\_of\_drive VARCHAR(15), PRIMARY KEY (transmission\_id)

);

CREATE TABLE car\_type (

type\_id INT NOT NULL AUTO\_INCREMENT,

typename VARCHAR(30), amount\_of\_doors TINYINT, amount\_of\_seats TINYINT, PRIMARY KEY (type\_id)

);

CREATE TABLE car (

car\_id INT NOT NULL AUTO\_INCREMENT,

engine\_id INT NOT NULL, transmission\_id INT NOT NULL, type\_id INT NOT NULL, brand VARCHAR(60),

model VARCHAR(60),

color VARCHAR(40),

build\_year INT, mileage INT, PRIMARY KEY (car\_id),

FOREIGN KEY (engine\_id) REFERENCES `engine` (engine\_id) ON DELETE CASCADE,

FOREIGN KEY (transmission\_id) REFERENCES transmission (transmission\_id) ON DELETE CASCADE,

FOREIGN KEY (type\_id) REFERENCES car\_type (type\_id) ON DELETE CASCADE

);

CREATE TABLE insurance (

insurance\_id INT NOT NULL AUTO\_INCREMENT, car\_id INT NOT NULL,

provider VARCHAR(255) NOT NULL,

policy\_number VARCHAR(255) NOT NULL, start\_date DATE NOT NULL, end\_date DATE NOT NULL, PRIMARY KEY (insurance\_id),

FOREIGN KEY (car\_id) REFERENCES car (car\_id) ON DELETE CASCADE

);

CREATE TABLE car\_accident (

accident\_id INT NOT NULL AUTO\_INCREMENT, car\_id INT NOT NULL,

`event` VARCHAR(120), date\_of\_event DATE, PRIMARY KEY (accident\_id),

FOREIGN KEY (car\_id) REFERENCES car (car\_id) ON DELETE CASCADE

);

CREATE TABLE listing (

listing\_id INT NOT NULL AUTO\_INCREMENT, car\_id INT NOT NULL,

seller\_id INT NOT NULL, listing\_date DATE NOT NULL, price DECIMAL(10, 2) NOT NULL,

PRIMARY KEY (listing\_id),

FOREIGN KEY (car\_id) REFERENCES car (car\_id) ON DELETE CASCADE,

FOREIGN KEY (seller\_id) REFERENCES seller (seller\_id) ON DELETE CASCADE

);

CREATE TABLE `transaction` (

transaction\_id INT NOT NULL AUTO\_INCREMENT, listing\_id INT NOT NULL,

buyer\_id INT NOT NULL, price INT NOT NULL, transaction\_date DATE NOT NULL, PRIMARY KEY (transaction\_id),

FOREIGN KEY (listing\_id) REFERENCES listing (listing\_id) ON DELETE CASCADE,

FOREIGN KEY (buyer\_id) REFERENCES buyer (buyer\_id) ON DELETE CASCADE

);

Імпортування даних

Для імпортування даних у була використана можливість завантаження даних за .csv файлу. Було створено .csv файли для кожної з таблиць.

Перелік основних файлів зображено на рисунку:

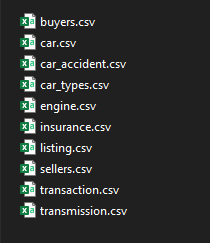


Рисунок 5.1 – Створені файли для заповнення Для завантаження даних були використані запити виду:

LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\buyers.csv'

INTO TABLE buyer

FIELDS TERMINATED BY ',' ENCLOSED BY '"'

LINES TERMINATED BY '\n' IGNORE 1 ROWS;

СТВОРЕННЯ КОРИСТУВАЧІВ БАЗИ ДАНИХ

Розробник

create role if not exists developer; grant all on car\_market.\* to developer;

create user if not exists dev@localhost identified by 'admin'; grant developer to dev@localhost;

Продавець

create role if not exists seller;

grant select on car\_market.\* to seller;

grant create, update, delete on car\_market.listing to seller; create user if not exists seller@localhost identified by 'seller'; grant seller to seller@localhost;

Покупець

create role if not exists seller;

grant select on car\_market.\* to seller;

grant create, update, delete on car\_market.listing to seller; create user if not exists seller@localhost identified by 'seller'; grant seller to seller@localhost;

# SQL запити

Створення тригерів на таблиці

Тригер, який не дозволяю зменшувати пробіг для занесених в базу даних автівок.

DROP TRIGGER IF EXISTS restrict\_mileage\_decrease; DELIMITER $$

CREATE TRIGGER restrict\_mileage\_decrease BEFORE UPDATE ON car

FOR EACH ROW BEGIN

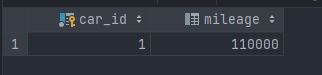
IF NEW.mileage < OLD.mileage THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Cannot decrease

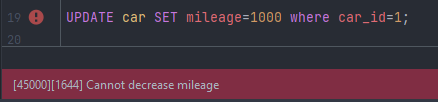
mileage'; END IF;

END$$ DELIMITER ;

Отримаємо пробіг авто з id=1:



При спробі зменшити його пробіг отримаємо помилку:



Тригер, який унеможливить видалення продавців, в яких є хоча б одна активна об’ява.

DELIMITER $$

DROP TRIGGER IF EXISTS restrict\_seller\_delete; CREATE TRIGGER restrict\_seller\_delete BEFORE DELETE ON seller

FOR EACH ROW BEGIN

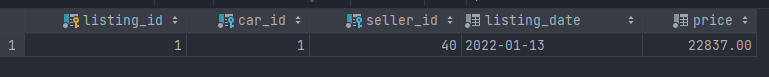
IF (SELECT COUNT(\*) FROM listing WHERE seller\_id = OLD.seller\_id) > 0 THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Cannot delete seller with active listings';

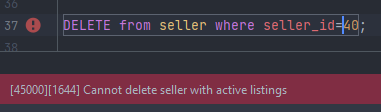
END IF;

END$$ DELIMITER ;

Переконаємося, що в продавця з id=1 є хоча б одна активна об’ява:



При спробі його видалити маємо помилку:



Тригер, який унеможливить створення записів про угоди, дата яких більш рання за дату подачі об’яви про продаж.

DELIMITER $$

DROP TRIGGER IF EXISTS transaction\_date\_trigger; CREATE TRIGGER transaction\_date\_trigger

BEFORE INSERT ON `transaction` FOR EACH ROW

BEGIN

IF NEW.transaction\_date < (SELECT l.listing\_date FROM listing l WHERE l.listing\_id = NEW.listing\_id) THEN

SIGNAL SQLSTATE '45000'

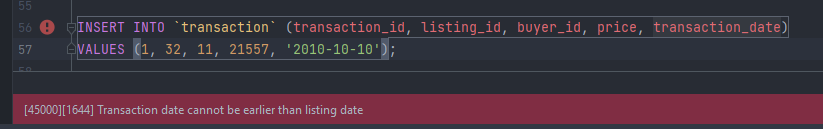
SET MESSAGE\_TEXT = 'Transaction date cannot be earlier than listing date'; END IF;

END$$ DELIMITER ;

Виконаємо запит та побачимо дату об’яви з id=1:



Спробуємо створити угоду, дата якої задовго до дати зверху:



Тригер, який не дозволить подавати декілька об’яв на одну й ту саму машину.

DELIMITER $$

DROP TRIGGER IF EXISTS prevent\_duplicate\_listings; CREATE TRIGGER prevent\_duplicate\_listings

BEFORE INSERT ON listing FOR EACH ROW

BEGIN

DECLARE duplicate INT;

SELECT COUNT(\*) INTO duplicate FROM listing WHERE car\_id =

NEW.car\_id;

IF duplicate > 0 THEN SIGNAL SQLSTATE '45000'

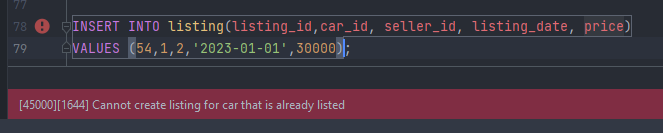
SET MESSAGE\_TEXT = 'Cannot create listing for car that is already listed'; END IF;

END$$ DELIMITER ;

Переконаємося, що запит про авто з id=1 існує:



Спробуємо створити ще одну об’яву на цю автівку:



Створення процедур

Процедура, яка виводить середню ціну по ринку на автівки заданої марки

DELIMITER $$

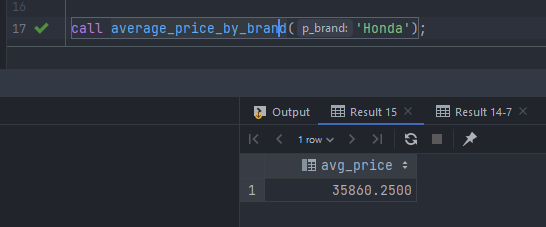
CREATE PROCEDURE average\_price\_by\_brand(IN p\_brand VARCHAR(60)) BEGIN

SELECT AVG(`transaction`.price) AS avg\_price FROM car

INNER JOIN listing ON car.car\_id = listing.car\_id

INNER JOIN `transaction` ON listing.listing\_id = `transaction`.listing\_id WHERE car.brand = p\_brand;

END$$ DELIMITER ;



Процедура, яка перевіряє, чи дійсний страховий поліс, айді якого було передано в процедуру як параметр.

DELIMITER $$

CREATE PROCEDURE insurance\_in\_effect(IN p\_insurance\_id INT) BEGIN

SELECT policy\_number, CASE

WHEN NOW() BETWEEN start\_date AND end\_date THEN 'In effect' ELSE 'Expired'

END AS status FROM insurance

WHERE insurance\_id = p\_insurance\_id; END$$

DELIMITER ;

Створення функцій

Функція, яка повертає суму, на яку конкретний продавець здійснив угоди.

DELIMITER $$

DROP FUNCTION IF EXISTS seller\_sales; CREATE FUNCTION seller\_sales(p\_seller\_id INT)

RETURNS DECIMAL(10, 2) DETERMINISTIC

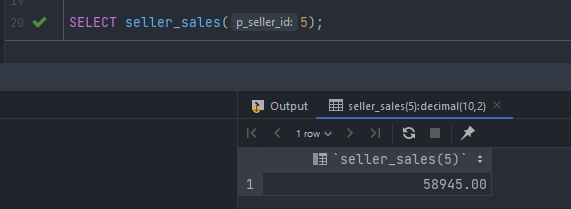
BEGIN

DECLARE v\_sales DECIMAL(10, 2); SELECT COALESCE(SUM(t.price), 0.0)

INTO v\_sales FROM `transaction` t

INNER JOIN listing l ON t.listing\_id = l.listing\_id WHERE l.seller\_id = p\_seller\_id;

RETURN v\_sales; END$$ DELIMITER ;



Функція, яка повертає інформаціє при двигун конкретної автівки DELIMITER $$

DROP FUNCTION IF EXISTS engine\_data\_by\_car;

CREATE FUNCTION engine\_data\_by\_car(car\_id INT) RETURNS VARCHAR(100) DETERMINISTIC

BEGIN

DECLARE engine\_data VARCHAR(100);

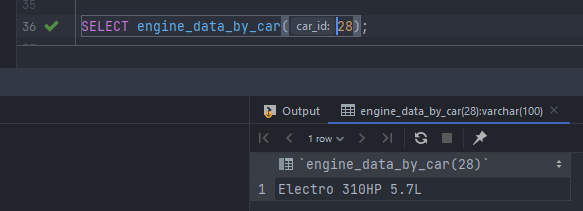
SELECT CONCAT(e.engine\_type, ' ', e.horse\_power, 'HP ', e.engine\_size, 'L') INTO engine\_data

FROM `engine` e

INNER JOIN car c ON e.engine\_id = c.engine\_id WHERE c.car\_id = car\_id;

RETURN engine\_data; END$$

DELIMITER ;



Створення представлень

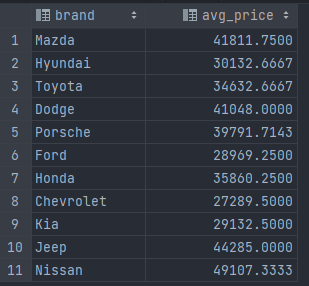
Представлення, яке демонструє середню ціну автівок по їхнім маркам: DROP VIEW IF EXISTS avg\_price\_by\_brand;

CREATE VIEW avg\_price\_by\_brand AS

SELECT car.brand, AVG(`transaction`.price) AS avg\_price FROM car

INNER JOIN listing ON car.car\_id = listing.car\_id

INNER JOIN `transaction` ON listing.listing\_id = `transaction`.listing\_id GROUP BY car.brand;



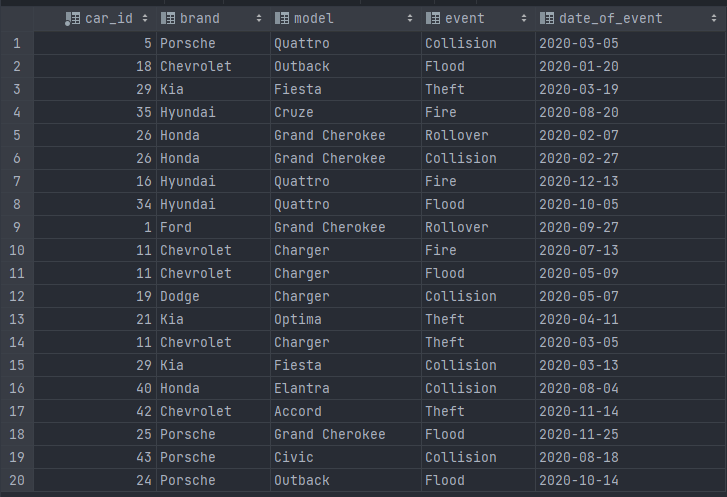
Представлення, яке демонструє «пригоди» кожної автівки: DROP VIEW IF EXISTS accidents\_by\_car;

CREATE VIEW accidents\_by\_car AS

SELECT car.car\_id, car.brand, car.model, car\_accident.event, car\_accident.date\_of\_event

FROM car

INNER JOIN car\_accident ON car.car\_id = car\_accident.car\_id;



Представлення, яке демонструє топ-продавців виходячи з кількості проданих ними автівок:

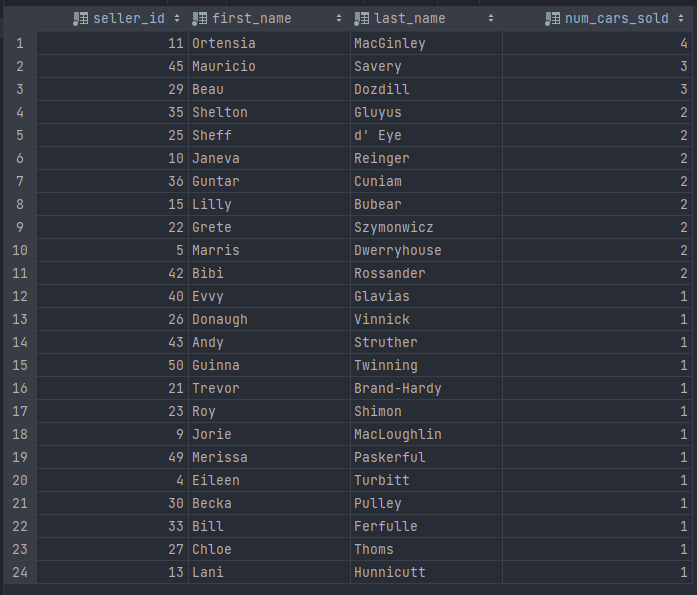
DROP VIEW IF EXISTS top\_sellers; CREATE VIEW top\_sellers AS

SELECT seller.seller\_id, seller.first\_name, seller.last\_name, COUNT(`transaction`.transaction\_id) AS num\_cars\_sold FROM seller

INNER JOIN listing ON seller.seller\_id = listing.seller\_id

INNER JOIN `transaction` ON listing.listing\_id = `transaction`.listing\_id GROUP BY seller.seller\_id

ORDER BY num\_cars\_sold DESC;



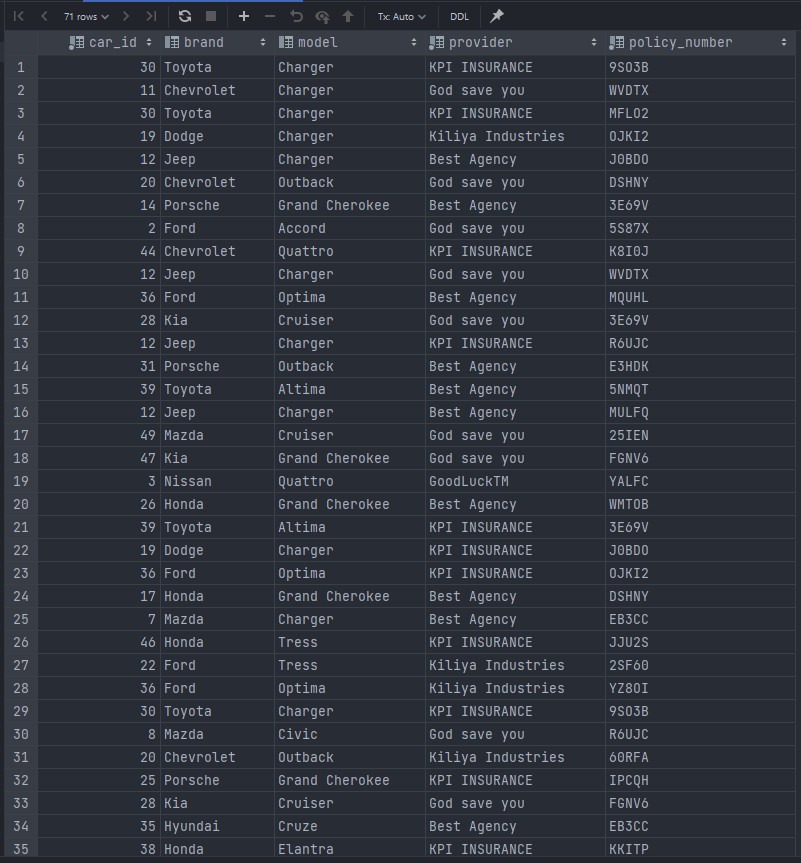
Представлення, яке демонструє інформацію про кожен страховий поліс кожної автівки:

DROP VIEW IF EXISTS cars\_with\_insurance; CREATE VIEW cars\_with\_insurance AS

SELECT car.car\_id, car.brand, car.model, insurance.provider, insurance.policy\_number

FROM car

INNER JOIN insurance ON car.car\_id = insurance.car\_id;



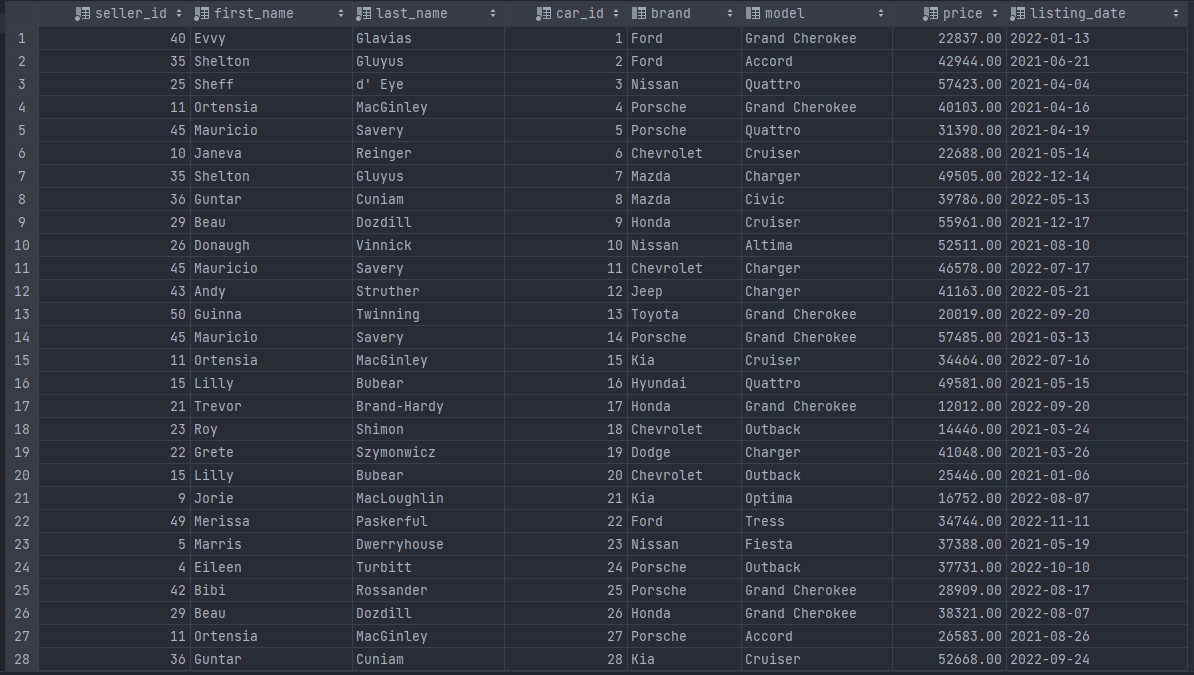
Представлення для відображення об’яв від кожного з продавців:

DROP VIEW IF EXISTS car\_listings\_by\_seller; CREATE VIEW car\_listings\_by\_seller AS

SELECT seller.seller\_id, seller.first\_name, seller.last\_name, car.car\_id, car.brand, car.model, listing.price, listing.listing\_date

FROM seller

INNER JOIN listing ON seller.seller\_id = listing.seller\_id INNER JOIN car ON listing.car\_id = car.car\_id;



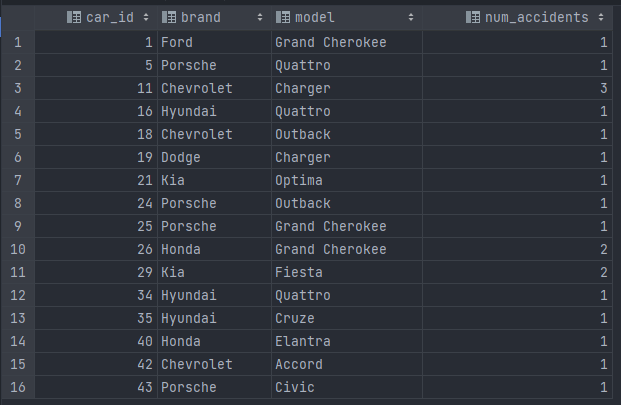
Створення різних запитів

Цей запит демонструє список автівок, з якими відбулися «неприємності»

SELECT car.car\_id, car.brand, car.model, COUNT(car\_accident.accident\_id) AS num\_accidents

FROM car

INNER JOIN car\_accident ON car.car\_id = car\_accident.car\_id GROUP BY car.car\_id;

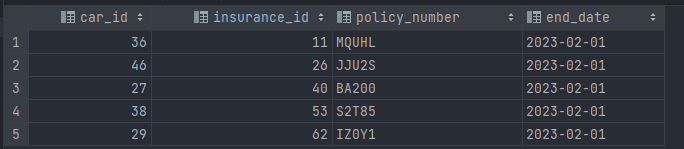


Цей запит демонструє усі автівки, страховий поліс яких закінчиться протягом наступного місяця:

SELECT car.car\_id, insurance\_id, insurance.policy\_number, insurance.end\_date FROM car

INNER JOIN insurance ON car.car\_id = insurance.car\_id

WHERE end\_date BETWEEN NOW() AND DATE\_ADD(NOW(), INTERVAL 1 MONTH);



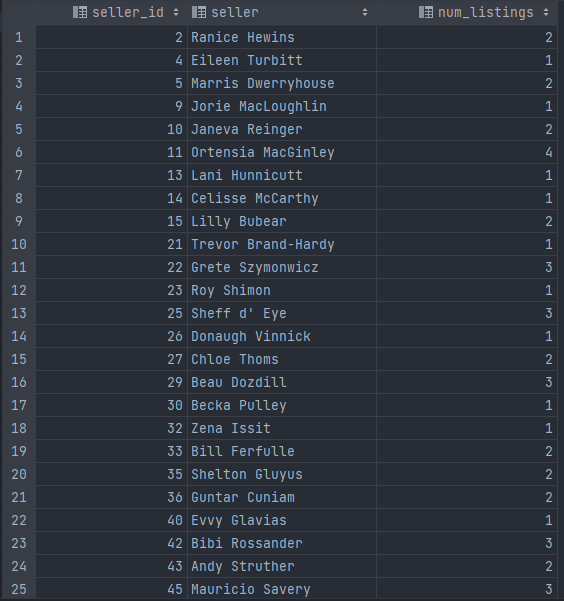
Цей запит демонструє кількість об’яв від кожного з продавців

SELECT seller.seller\_id,

concat(seller.first\_name, ' ', seller.last\_name) as seller, COUNT(listing.listing\_id) AS num\_listings

FROM seller

INNER JOIN listing ON seller.seller\_id = listing.seller\_id GROUP BY seller.seller\_id;



Цей запит демонструє кількість угод, укладених протягом минулого року:

SELECT COUNT(\*) as `Amount of transactions during past year` FROM `transaction`

WHERE transaction\_date BETWEEN DATE\_SUB(NOW(), INTERVAL 1 YEAR) AND NOW();

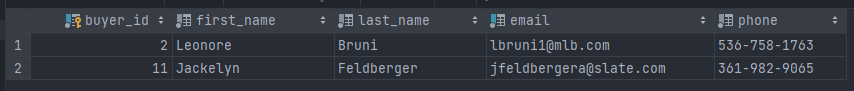


Цей запит показує інформацію про покупців, які укладали угоди с продавцем під id=5:

SELECT buyer.\* FROM buyer

INNER JOIN `transaction` ON buyer.buyer\_id = `transaction`.buyer\_id INNER JOIN listing ON `transaction`.listing\_id = listing.listing\_id

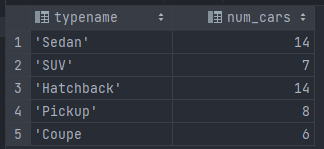
WHERE listing.seller\_id = 5;



Цей запит демонструє кількість автівок кожного типу:

SELECT car\_type.typename, COUNT(car.car\_id) AS num\_cars FROM car\_type

LEFT JOIN car ON car\_type.type\_id = car.type\_id GROUP BY car\_type.typename;

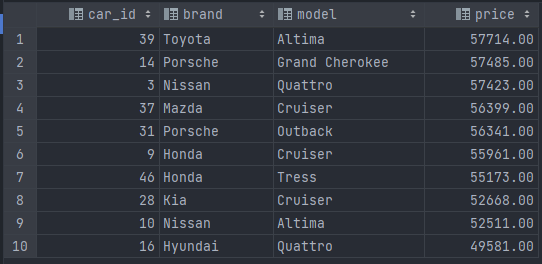


Цей запит виводить інформацію про 10 найдорожчих автомобілів

SELECT car.car\_id, brand, model, price FROM car

INNER JOIN listing ON car.car\_id = listing.car\_id ORDER BY price DESC

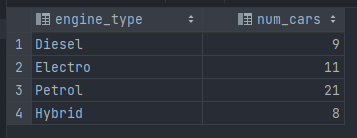
LIMIT 10;



Цей запит демонструє кількість двигунів різних типів в автівках, які виставлені на продаж:

SELECT engine.engine\_type, COUNT(car.car\_id) AS num\_cars FROM engine

LEFT JOIN car ON engine.engine\_id = car.engine\_id GROUP BY engine.engine\_type;

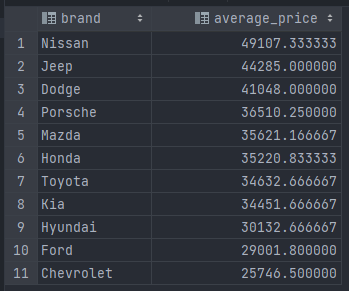


Цей запит виводить середню ціну усіх автівок по марках:

SELECT car.brand, AVG(price) AS average\_price FROM car

INNER JOIN listing ON car.car\_id = listing.car\_id GROUP BY car.brand

ORDER BY (average\_price) DESC;



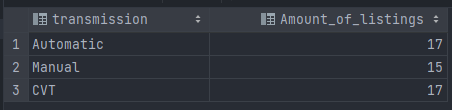
Цей запит демонструє кількість типів КПП, які встановлені на автівки, що виставлені на продаж:

SELECT transmission.transmission, COUNT(transmission.transmission\_id) AS Amount\_of\_listings

FROM transmission

INNER JOIN car ON transmission.transmission\_id = car.transmission\_id INNER JOIN listing ON car.car\_id = listing.car\_id

GROUP BY transmission.transmission;

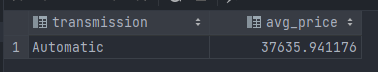


Цей запит виводить середню ціну автівок з автоматичною коробкою передач:

SELECT transmission, AVG(price) AS avg\_price FROM car

INNER JOIN transmission ON car.transmission\_id = transmission.transmission\_id

INNER JOIN listing ON car.car\_id = listing.car\_id WHERE transmission.transmission = 'Automatic';

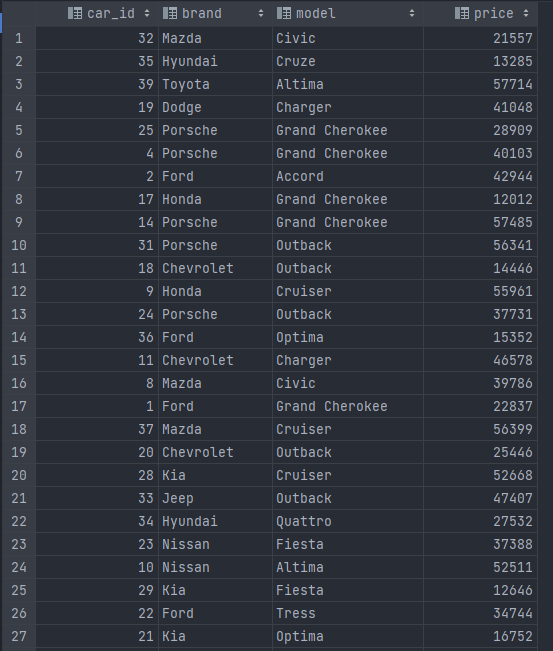


Цей запит демонструє продані автівки та їхні ціни:

SELECT car.car\_id, car.brand, car.model, `transaction`.price FROM car

INNER JOIN listing ON car.car\_id = listing.car\_id

INNER JOIN `transaction` ON listing.listing\_id = `transaction`.listing\_id;



Цей запит виводить продавців, які продали хоча б одну автівку:

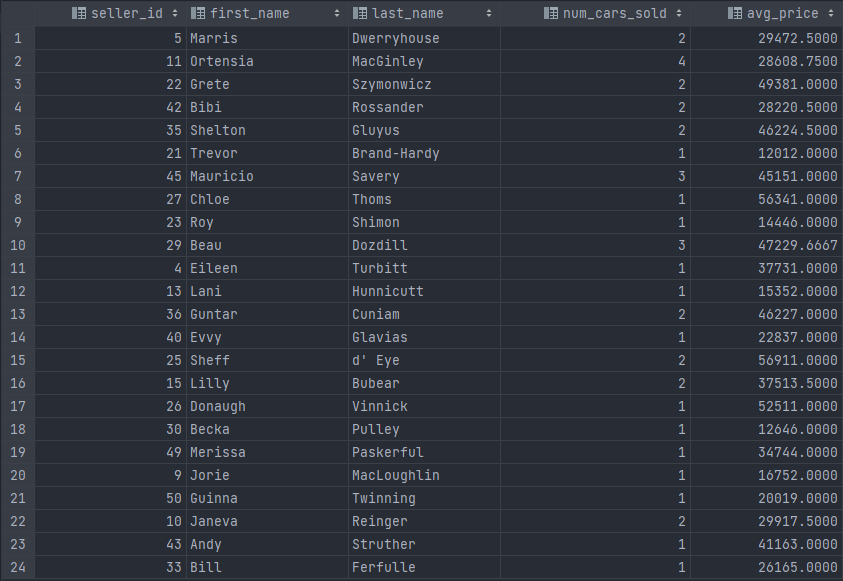
SELECT seller.seller\_id, seller.first\_name, seller.last\_name,

COUNT(`transaction`.transaction\_id) AS num\_cars\_sold, AVG(`transaction`.price) AS avg\_price

FROM seller

INNER JOIN listing ON seller.seller\_id = listing.seller\_id

INNER JOIN `transaction` ON listing.listing\_id = `transaction`.listing\_id GROUP BY seller.seller\_id;



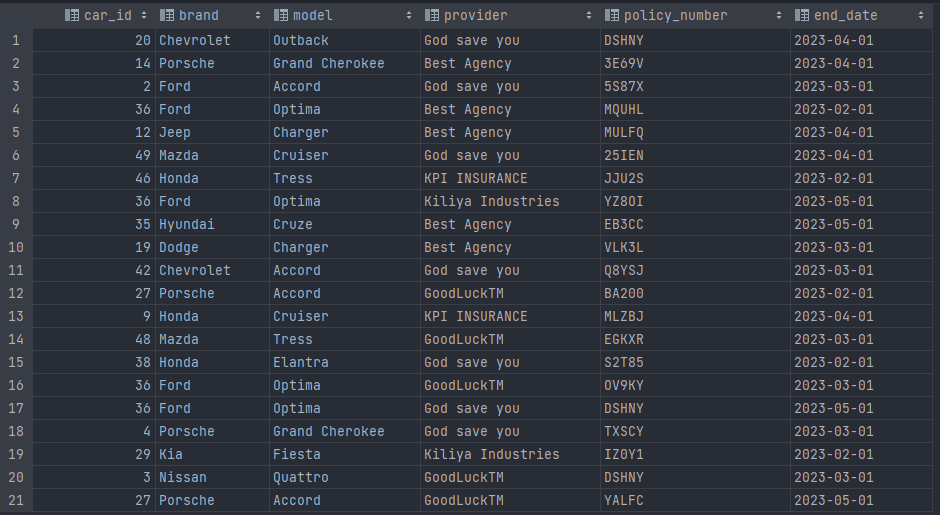
Цей запит виводить список автівок, страховий поліс на поточний момент є чинним:

SELECT car.car\_id, car.brand, car.model, insurance.provider, insurance.policy\_number, insurance.end\_date

FROM car

INNER JOIN insurance ON car.car\_id = insurance.car\_id

WHERE CURDATE() BETWEEN insurance.start\_date AND insurance.end\_date;



Цей запит виводить найбільш популярний тип автівок:

SELECT car\_type.typename, COUNT(car\_type.typename) AS num\_cars FROM car\_type

INNER JOIN car ON car\_type.type\_id = car.type\_id GROUP BY car\_type.typename

ORDER BY num\_cars DESC LIMIT 1;



Створення індексів

Зробимо запит по полю з електроною адресою:

explain select \* from buyer where email='gslowann@time.com';



Тепер створимо індекс і зробимо запит ще раз: create index buyer\_email on buyer(email);

explain select \* from buyer where email='gslowann@time.com';



Як можна побачити, перший запит виконувався 38ms, а запит з використанням індексів було виконано за 27ms, що є значно кращим результатом. У великій базі даних ця різниця може бути критичною.

## ВИСНОВОК

Перед виконанням завдання по створенню бази даних по підтримці діяльності автомаркету було спроєктовану майбутню базу даних, виокремленні основні сутності та зв’язки між ними. Були сформульовані бізнес-правила та вимоги, які потрібно було реалізувати у роботі.

Отже, у даній роботі було успішно розроблено та впроваджено базу даних для підтримки діяльності автомаркету. Було створено 10 різних таблиць та визначені зв’язки між ними за допомогою зовнішніх ключів і каскадних дій. Були закріплені навички створювання та використовування різних об’єктів бази даних, такі як тригери, функції та процедури, створення складних запитів, яке несуть практичне значення для потенційних користувачів сховища даних

## ПЕРЕЛІК ПОСИЛАНЬ

1. https://dev.mysql.com/doc/
2. <https://www.jetbrains.com/help/datagrip/meet-the-product.html>
3. https://[www.w3schools.com/mysql/](http://www.w3schools.com/mysql/)