

Міністерство освіти та науки України

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики і програмної інженерії

**Звіт**

з дисципліни «Бази даних»

ЕКЗАМЕНАЦІЙНИЙ БІЛЕТ N° 24

**Виконав:**

*Студент II курсу*

*гр. ІП-33*

Соколов О. В.

**Екзаменатор:**

Катерина ЛІЩУК.

2024

**Опис предметного середовища**

Проєктується база даних охоронного підприємства. У базі даних міститься інформація про обʼєкти, які охороняються - це адреса, назва обʼєкта, тип сигналізації, відстань, також зберігаються дані про технічні характеристики систем сигналізації, які використовуються. Окрім того, зберігаються дані про наявні авто та їх характеристики, такі як марка, швидкість, та інше. В базі даних фіксуються всі спрацювання охоронної системи, та результати виїзду на місце події співробітників, дата і час, обʼєкт, хто виїздив на виклик і результат.

Для заданого предметного середовища необхідно виконати наступне завдання:

1) Розробити ER-модель для заданого предметного середовища (мін. 5-6 сутностей). Відношення повинно знаходитись шонайменше в ЗНФ

2) Згідно зі розробленої ER-моделі створити таблиці в БД засобами мови SOL. Передбачити необхідність наявності обмежень для підтримки посилальної цілісності, цілісності даних, допустимості значень, значень за замовченням. При створенні перевірочних обмежень використати апарат збережених процедур/функцій. Для підтримки цілісності створити щонайменше 2 тригера AFTER <відповідна дія> та тригер INSTEAD OF.

3) Необхідно передбачити можливість отримання звіту про роботу охоронного підприємства з вказанням інформації про клієнтів та кількості виїздів до них за останній рік. Для розвʼязання поставленої задачі використати курсори.

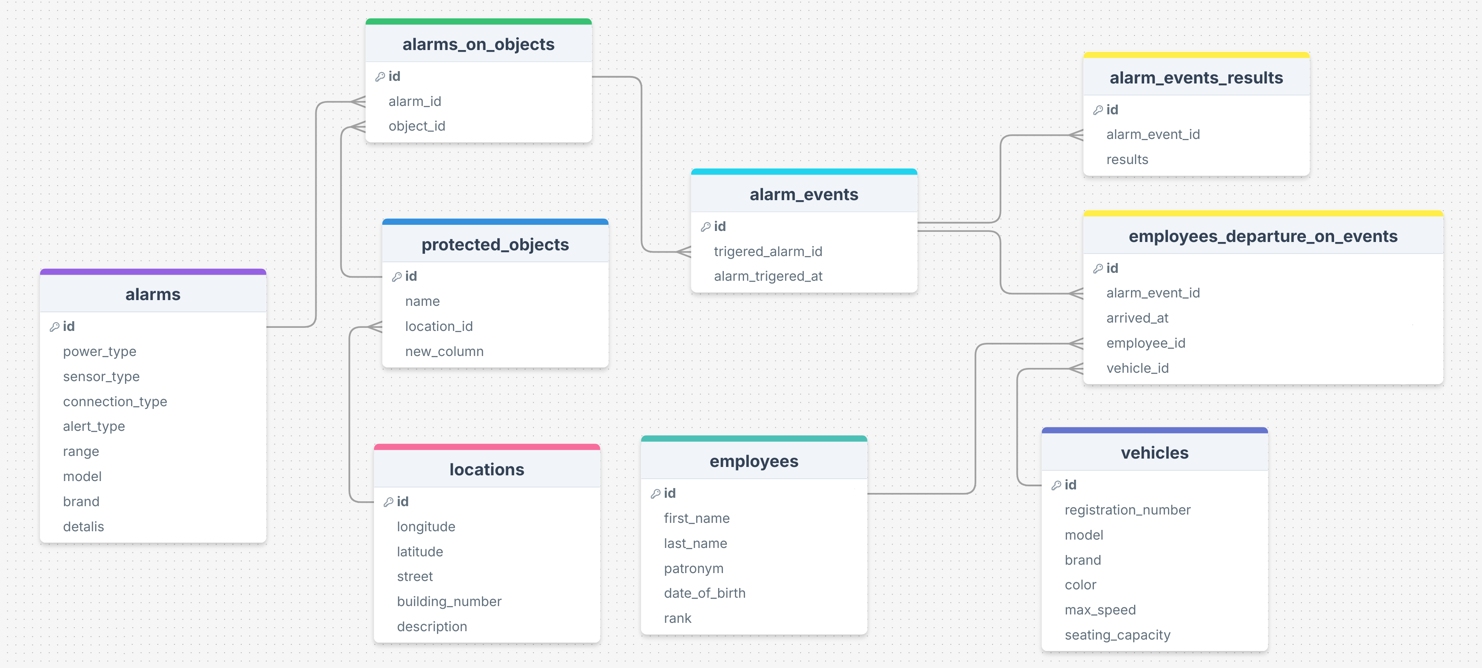
4) Створити запити на основі їх текстового формулювання:

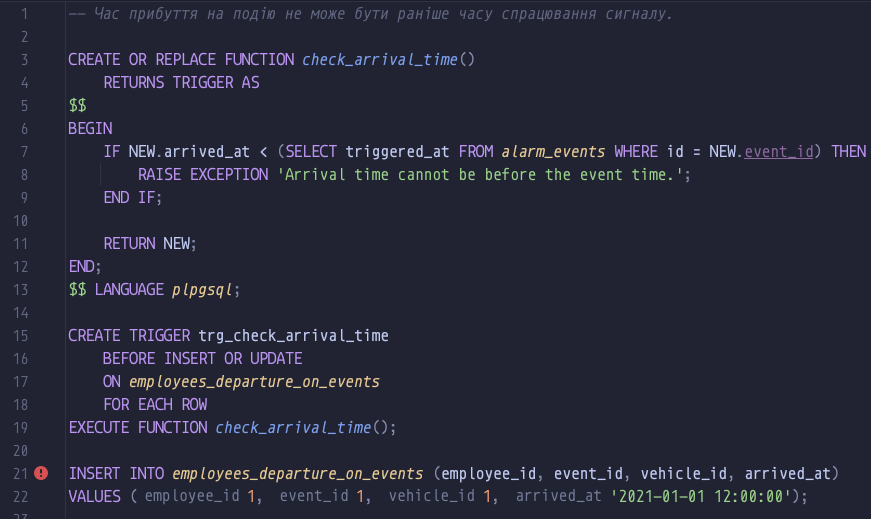
а) Обʼєкти, котрі розміщуються на проспекті Перемоги та на них за минулий рік виїжджали не менше 5 разів.

b) Фірма, котра виробляє найнадійніші системи сигналізації (надійною є система сигналізації де у випадку виїзду екіпажу результатом є відсутність взлому).

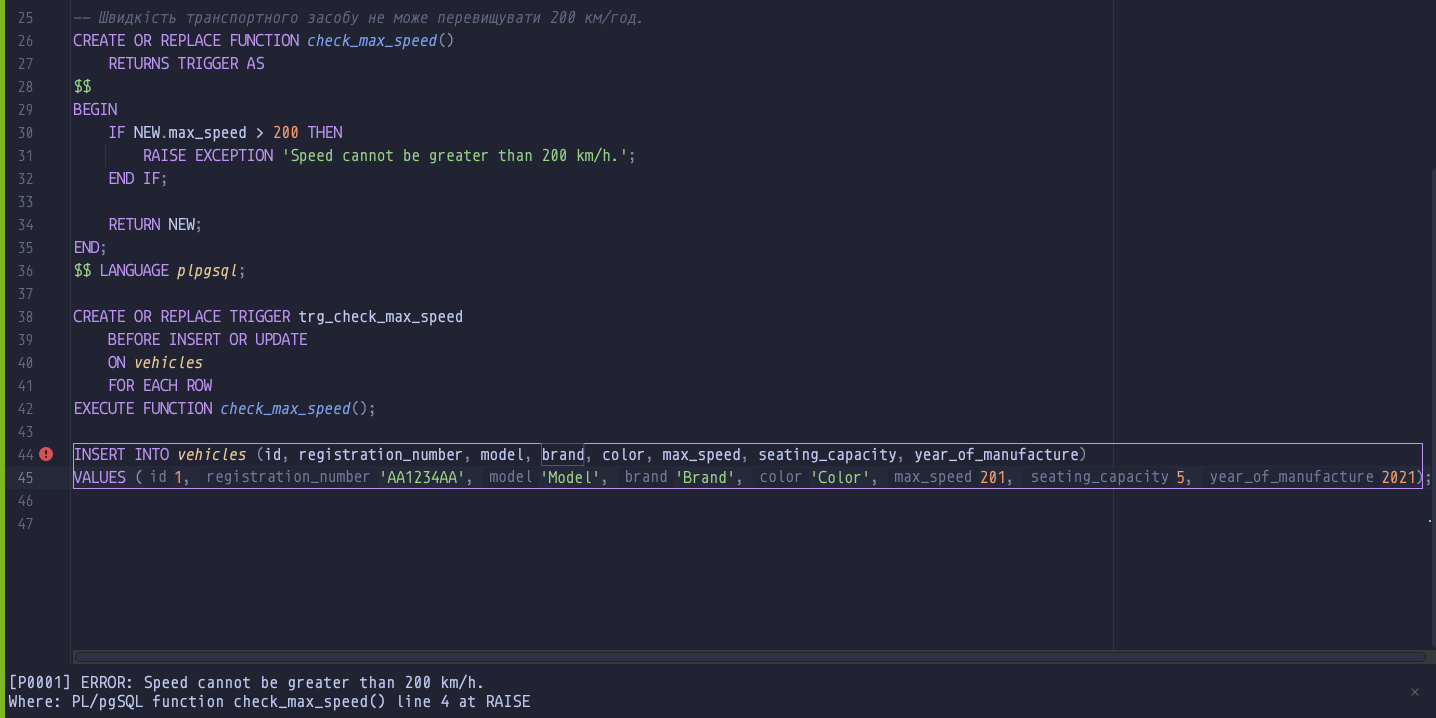
с) Номер автомобіля, на якому за останній рік виїжджали ни виклики найбільшу кількість разів.

d) Дні тижня, в які було найбільше виїздів за минулий місяць.

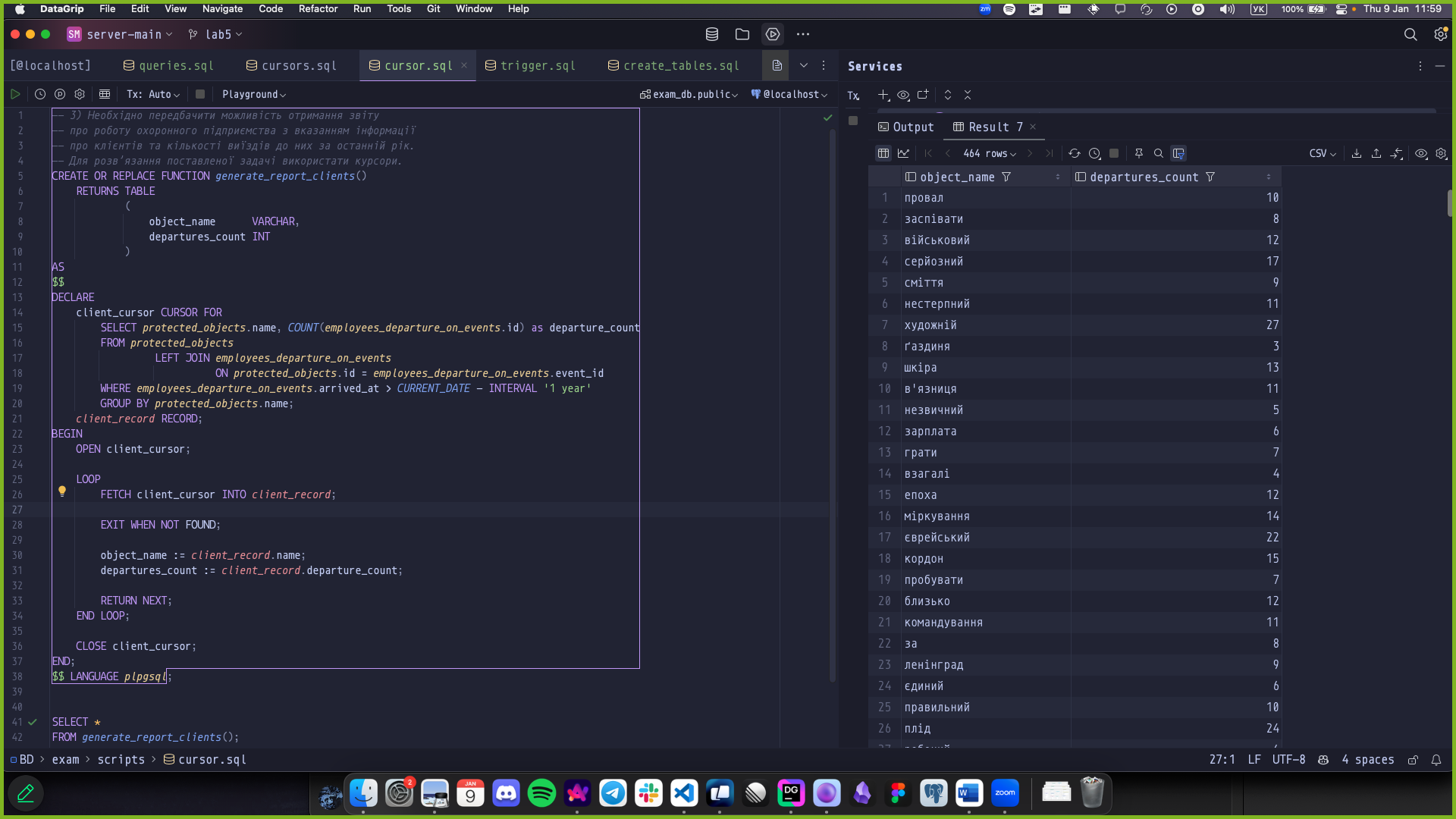
1) ER-модель

2) Тригери на перевірку значень

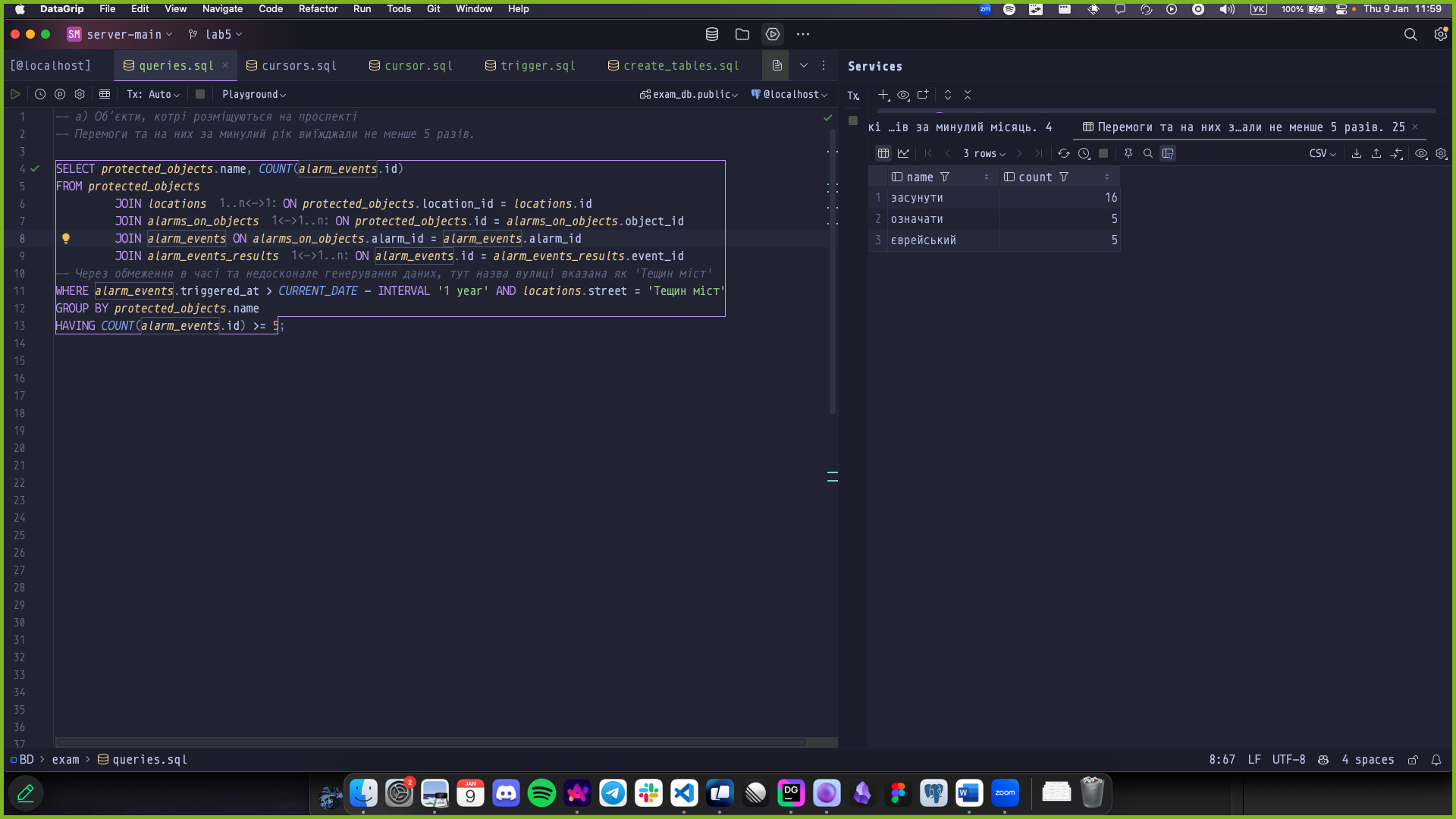


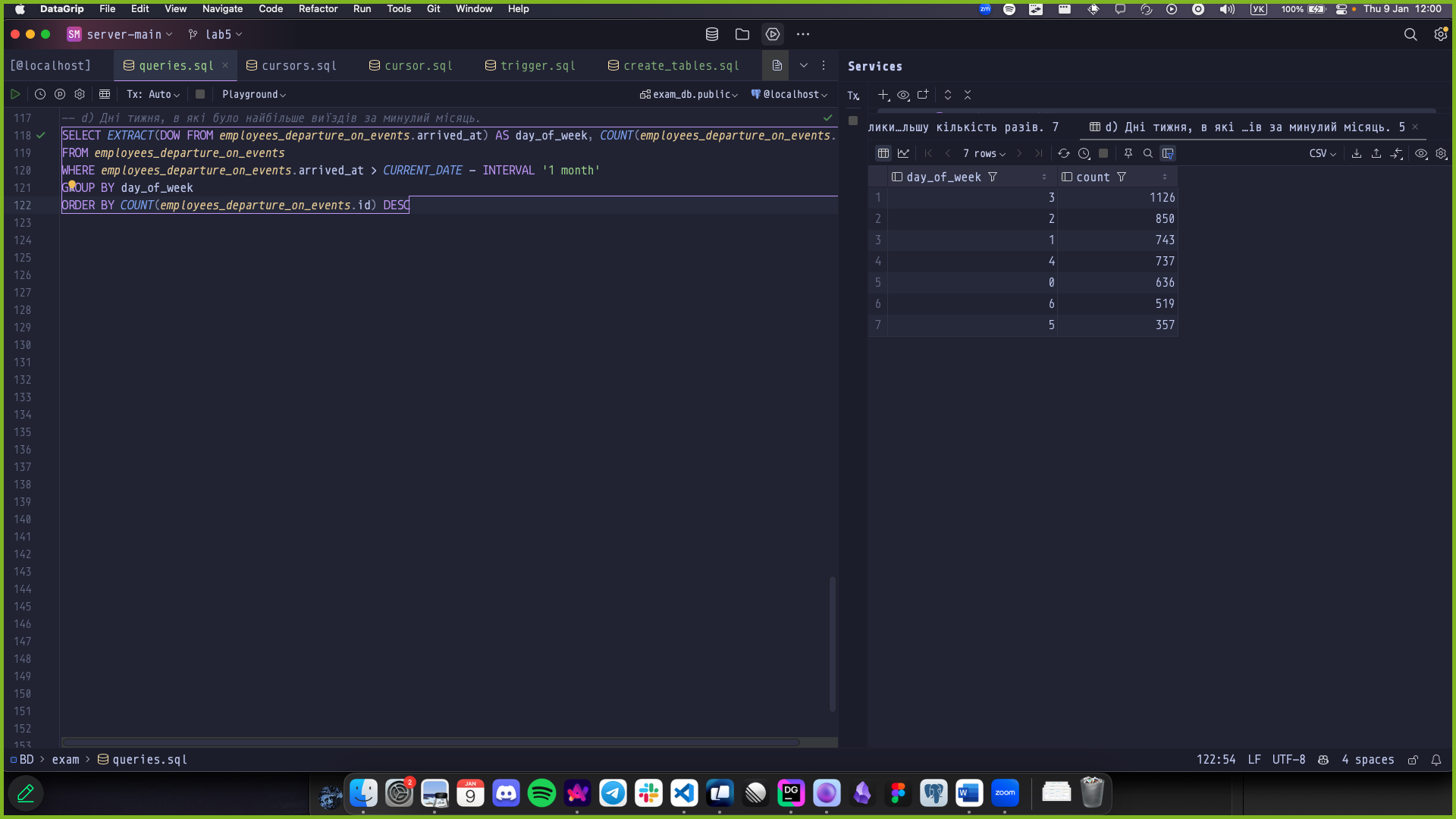
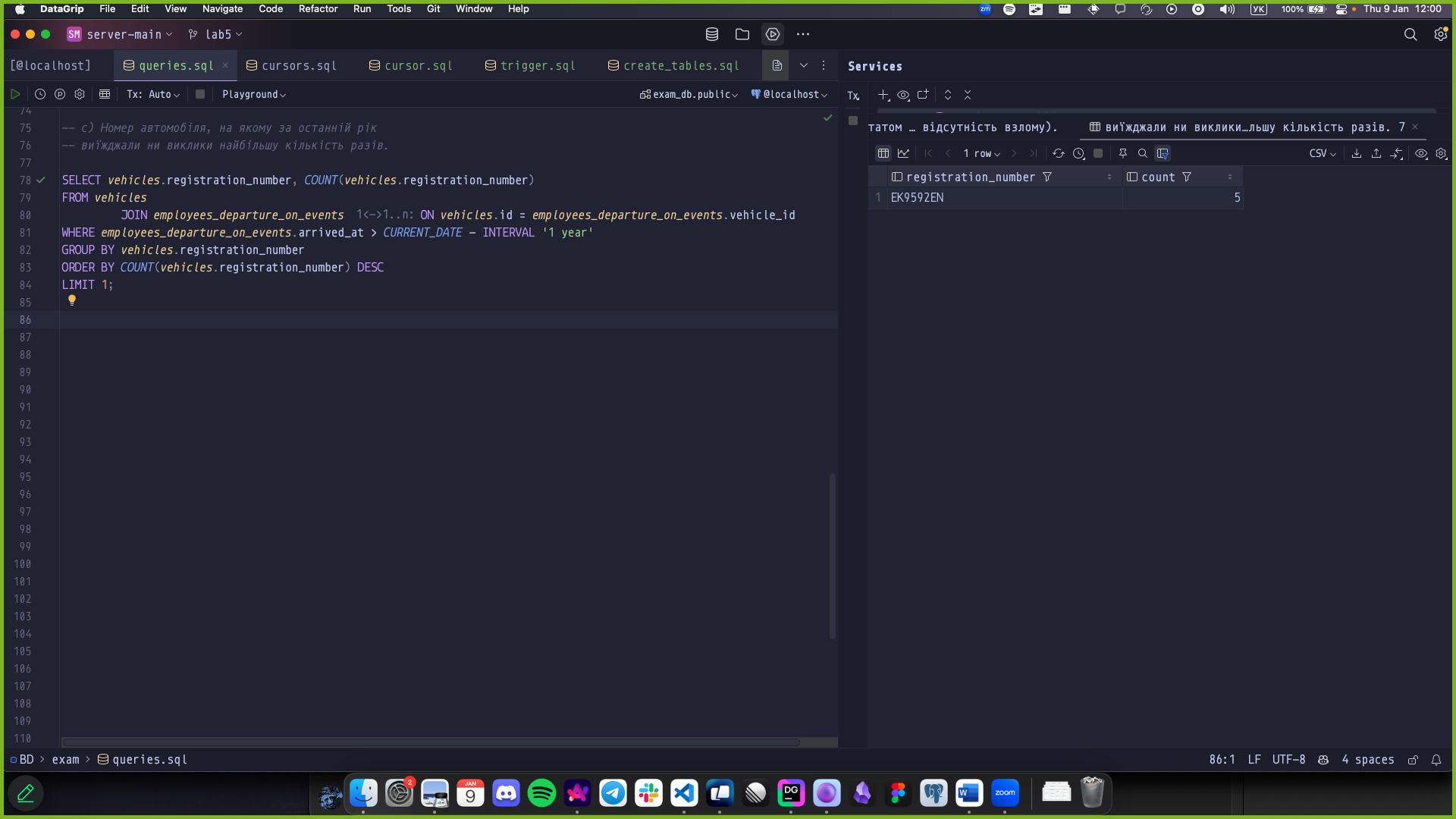
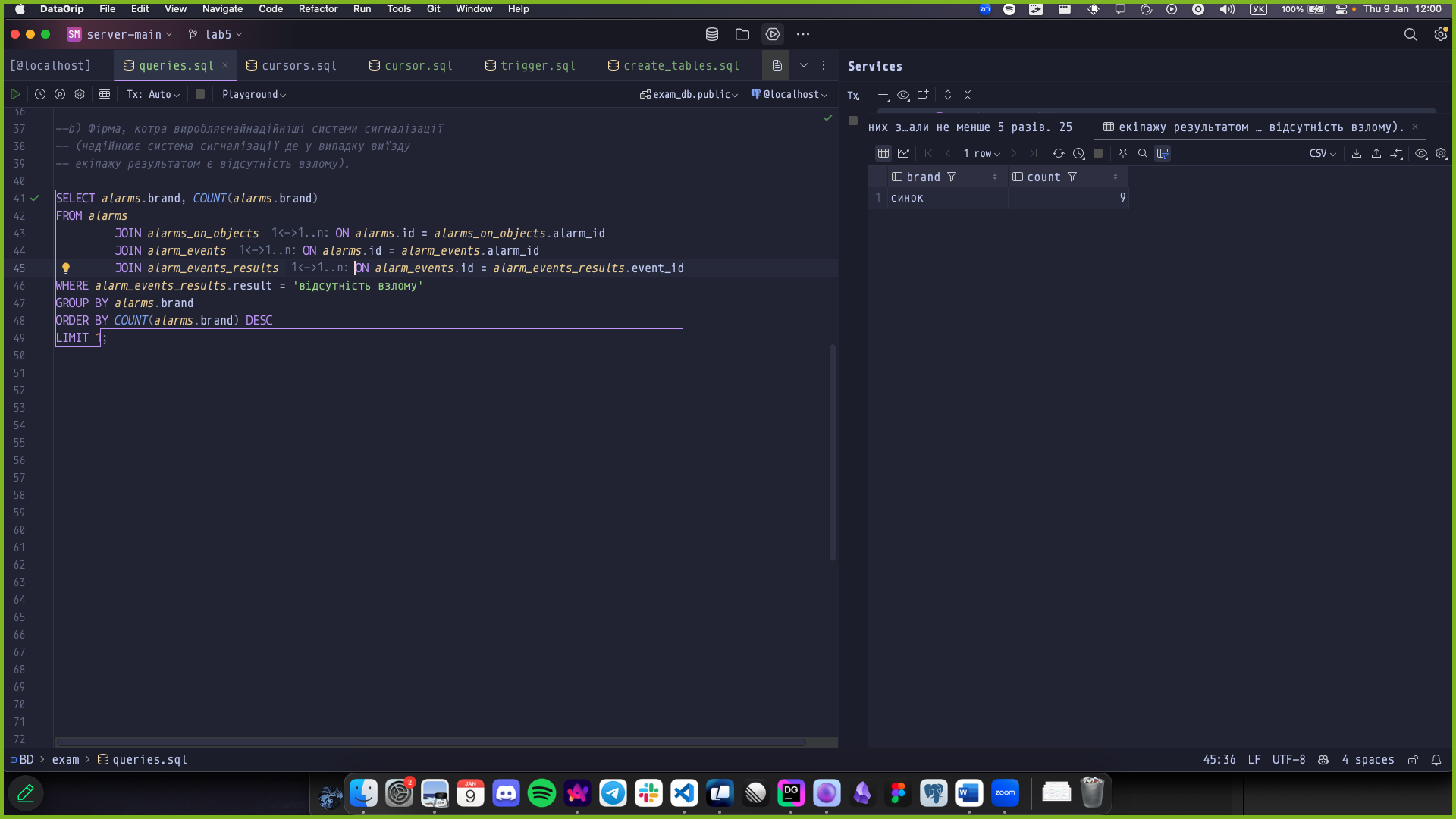


3) Звіт за допомогою курсорів



4) Запити





Написаний код під час екзамену:  
  
CREATE SCHEMA IF NOT EXISTS public;

CREATE TABLE employees (

id SERIAL PRIMARY KEY,

first\_name VARCHAR(50) NOT NULL,

last\_name VARCHAR(50) NOT NULL,

patronymic VARCHAR(50),

birth\_date DATE NOT NULL,

phone\_number VARCHAR(20) NOT NULL,

email VARCHAR(50) NOT NULL,

CONSTRAINT ck\_birth\_date CHECK (birth\_date <= CURRENT\_DATE),

CONSTRAINT uq\_phone\_number UNIQUE (phone\_number),

CONSTRAINT uq\_email UNIQUE (email)

);

CREATE TABLE vehicles (

id SERIAL PRIMARY KEY,

registration\_number VARCHAR(8) NOT NULL,

model VARCHAR(50) NOT NULL,

brand VARCHAR(50) NOT NULL,

color VARCHAR(50) NOT NULL,

max\_speed INT,

seating\_capacity INT,

year\_of\_manufacture INT NOT NULL,

CONSTRAINT ck\_year\_of\_manufacture CHECK (

year\_of\_manufacture >= 1886

AND year\_of\_manufacture <= EXTRACT(

YEAR

FROM

CURRENT\_DATE

)

),

CONSTRAINT uq\_registration\_number UNIQUE (registration\_number),

CONSTRAINT ck\_seating\_capacity CHECK (seating\_capacity BETWEEN 1 AND 50)

);

CREATE TABLE locations (

id SERIAL PRIMARY KEY,

longitude DECIMAL(9, 6) NOT NULL,

latitude DECIMAL(9, 6) NOT NULL,

street VARCHAR(50),

building\_number VARCHAR(10),

description TEXT

);

CREATE TYPE power\_type AS ENUM ('accumulator', 'wired', 'combined');

CREATE TYPE sensor\_type AS ENUM ('movement', 'glass\_breakage', 'door\_opening', 'temperature', 'vibration');

CREATE TYPE connection\_type AS ENUM ('wired', 'wifi', 'radio');

CREATE TYPE alert\_type AS ENUM ('sms', 'email', 'phone\_call', 'app\_notification');

CREATE TABLE alarms (

id SERIAL PRIMARY KEY,

power power\_type NOT NULL,

sensor sensor\_type NOT NULL,

connection connection\_type NOT NULL,

alert alert\_type NOT NULL,

max\_range DECIMAL(5, 2),

model VARCHAR(50) NOT NULL,

brand VARCHAR(50) NOT NULL,

details TEXT

);

CREATE TABLE protected\_objects (

id SERIAL PRIMARY KEY,

name VARCHAR(50) NOT NULL,

location\_id INT NOT NULL,

CONSTRAINT fk\_location\_id FOREIGN KEY (location\_id) REFERENCES locations (id) ON DELETE CASCADE

);

CREATE TABLE alarms\_on\_objects (

id SERIAL PRIMARY KEY,

alarm\_id INT NOT NULL,

object\_id INT NOT NULL,

CONSTRAINT fk\_alarm\_id FOREIGN KEY (alarm\_id) REFERENCES alarms (id) ON DELETE CASCADE,

CONSTRAINT fk\_object\_id FOREIGN KEY (object\_id) REFERENCES protected\_objects (id) ON DELETE CASCADE

);

CREATE TABLE alarm\_events (

id SERIAL PRIMARY KEY,

alarm\_id INT NOT NULL,

triggered\_at TIMESTAMPTZ NOT NULL,

CONSTRAINT fk\_alarm\_id FOREIGN KEY (alarm\_id) REFERENCES alarms (id) ON DELETE CASCADE,

CONSTRAINT fk\_triggered\_at CHECK (triggered\_at <= CURRENT\_TIMESTAMP)

);

CREATE TABLE alarm\_events\_results (

id SERIAL PRIMARY KEY,

event\_id INT NOT NULL,

result TEXT NOT NULL,

CONSTRAINT fk\_event\_id FOREIGN KEY (event\_id) REFERENCES alarm\_events (id) ON DELETE CASCADE

);

CREATE TABLE employees\_departure\_on\_events (

id SERIAL PRIMARY KEY,

employee\_id INT NOT NULL,

event\_id INT NOT NULL,

vehicle\_id INT NOT NULL,

arrived\_at TIMESTAMPTZ NOT NULL,

CONSTRAINT fk\_employee\_id FOREIGN KEY (employee\_id) REFERENCES employees (id) ON DELETE CASCADE,

CONSTRAINT fk\_event\_id FOREIGN KEY (event\_id) REFERENCES alarm\_events (id) ON DELETE CASCADE,

CONSTRAINT fk\_vehicle\_id FOREIGN KEY (vehicle\_id) REFERENCES vehicles (id) ON DELETE CASCADE

)

-- Час прибуття на подію не може бути раніше часу спрацювання сигналу.

CREATE OR REPLACE FUNCTION check\_arrival\_time()

RETURNS TRIGGER AS

$$

BEGIN

IF NEW.arrived\_at < (SELECT triggered\_at FROM alarm\_events WHERE id = NEW.event\_id) THEN

RAISE EXCEPTION 'Arrival time cannot be before the event time.';

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER trg\_check\_arrival\_time

BEFORE INSERT OR UPDATE

ON employees\_departure\_on\_events

FOR EACH ROW

EXECUTE FUNCTION check\_arrival\_time();

INSERT INTO employees\_departure\_on\_events (employee\_id, event\_id, vehicle\_id, arrived\_at)

VALUES (1, 1, 1, '2021-01-01 12:00:00');

-- Швидкість транспортного засобу не може перевищувати 200 км/год.

CREATE OR REPLACE FUNCTION check\_max\_speed()

RETURNS TRIGGER AS

$$

BEGIN

IF NEW.max\_speed > 200 THEN

RAISE EXCEPTION 'Speed cannot be greater than 200 km/h.';

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER trg\_check\_max\_speed

BEFORE INSERT OR UPDATE

ON vehicles

FOR EACH ROW

EXECUTE FUNCTION check\_max\_speed();

INSERT INTO vehicles (id, registration\_number, model, brand, color, max\_speed, seating\_capacity, year\_of\_manufacture)

VALUES (1, 'AA1234AA', 'Model', 'Brand', 'Color', 201, 5, 2021);

-- 3) Необхідно передбачити можливість отримання звіту

-- про роботу охоронного підприємства з вказанням інформації

-- про клієнтів та кількості виїздів до них за останній рік.

-- Для розвʼязання поставленої задачі використати курсори.

CREATE OR REPLACE FUNCTION generate\_report\_clients()

RETURNS TABLE

(

object\_name VARCHAR,

departures\_count INT

)

AS

$$

DECLARE

client\_cursor CURSOR FOR

SELECT protected\_objects.name, COUNT(employees\_departure\_on\_events.id) as departure\_count

FROM protected\_objects

LEFT JOIN employees\_departure\_on\_events

ON protected\_objects.id = employees\_departure\_on\_events.event\_id

WHERE employees\_departure\_on\_events.arrived\_at > CURRENT\_DATE - INTERVAL '1 year'

GROUP BY protected\_objects.name;

client\_record RECORD;

BEGIN

OPEN client\_cursor;

LOOP

FETCH client\_cursor INTO client\_record;

EXIT WHEN NOT FOUND;

object\_name := client\_record.name;

departures\_count := client\_record.departure\_count;

RETURN NEXT;

END LOOP;

CLOSE client\_cursor;

END;

$$ LANGUAGE plpgsql;

SELECT \*

FROM generate\_report\_clients();  
  
-- а) Обʼєкти, котрі розміщуються на проспекті

-- Перемоги та на них за минулий рік виїжджали не менше 5 разів.

SELECT protected\_objects.name, COUNT(alarm\_events.id)

FROM protected\_objects

JOIN locations ON protected\_objects.location\_id = locations.id

JOIN alarms\_on\_objects ON protected\_objects.id = alarms\_on\_objects.object\_id

JOIN alarm\_events ON alarms\_on\_objects.alarm\_id = alarm\_events.alarm\_id

JOIN alarm\_events\_results ON alarm\_events.id = alarm\_events\_results.event\_id

-- Через обмеження в часі та недосконале генерування даних, тут назва вулиці вказана як 'Тещин міст'

WHERE alarm\_events.triggered\_at > CURRENT\_DATE - INTERVAL '1 year'

AND locations.street = 'Тещин міст'

GROUP BY protected\_objects.name

HAVING COUNT(alarm\_events.id) >= 5;

--b) Фірма, котра виробляє найнадійніші системи сигналізації

-- (надійноює система сигналізації де у випадку виїзду

-- екіпажу результатом є відсутність взлому).

SELECT alarms.brand, COUNT(alarms.brand)

FROM alarms

JOIN alarms\_on\_objects ON alarms.id = alarms\_on\_objects.alarm\_id

JOIN alarm\_events ON alarms.id = alarm\_events.alarm\_id

JOIN alarm\_events\_results ON alarm\_events.id = alarm\_events\_results.event\_id

WHERE alarm\_events\_results.result = 'відсутність взлому'

GROUP BY alarms.brand

ORDER BY COUNT(alarms.brand) DESC

LIMIT 1;

-- с) Номер автомобіля, на якому за останній рік

-- виїжджали ни виклики найбільшу кількість разів.

SELECT vehicles.registration\_number, COUNT(vehicles.registration\_number)

FROM vehicles

JOIN employees\_departure\_on\_events ON vehicles.id = employees\_departure\_on\_events.vehicle\_id

WHERE employees\_departure\_on\_events.arrived\_at > CURRENT\_DATE - INTERVAL '1 year'

GROUP BY vehicles.registration\_number

ORDER BY COUNT(vehicles.registration\_number) DESC

LIMIT 1;

-- d) Дні тижня, в які було найбільше виїздів за минулий місяць.

SELECT EXTRACT(DOW FROM employees\_departure\_on\_events.arrived\_at) AS day\_of\_week,

COUNT(employees\_departure\_on\_events.id)

FROM employees\_departure\_on\_events

WHERE employees\_departure\_on\_events.arrived\_at > CURRENT\_DATE - INTERVAL '1 month'

GROUP BY day\_of\_week

ORDER BY COUNT(employees\_departure\_on\_events.id) DESC

Код, написаний для генерації даниї мовою python із використанням бібліотеки faker:

from calendar import c

from os import write

from faker import Faker

from faker\_vehicle import VehicleProvider

import csv

fake = Faker("uk\_UA")

fake.add\_provider(VehicleProvider)

def generate\_vehicles(filename, count):

vehicles = []

for i in range(count):

vehicle = {

"id": i+1,

"registration\_number": fake.license\_plate(),

"model": fake.vehicle\_model(),

"brand": fake.vehicle\_make(),

"color": fake.color\_name(),

"max\_speed": fake.random\_int(min=100, max=300),

"seating\_capacity": fake.random\_int(min=1, max=50),

"year\_of\_manufacture": fake.random\_int(min=1886, max=2021)

}

vehicles.append(vehicle)

with open(filename, "w") as file:

writer = csv.DictWriter(file, fieldnames=vehicles[0].keys())

writer.writeheader()

writer.writerows(vehicles)

def generate\_employees(filename, count):

employees = []

for i in range(count):

employee = {

"id": i+1,

"first\_name": fake.first\_name(),

"last\_name": fake.last\_name(),

"patronymic": fake.first\_name(),

"birth\_date": fake.date\_of\_birth(minimum\_age=18, maximum\_age=65),

"phone\_number": fake.phone\_number(),

"email": fake.email()

}

employees.append(employee)

with open(filename, "w") as file:

writer = csv.DictWriter(file, fieldnames=employees[0].keys())

writer.writeheader()

writer.writerows(employees)

def generate\_locations(filename, count):

locations = []

for i in range(count):

location = {

"id": i + 1,

"longitude": fake.longitude(),

"latitude": fake.latitude(),

"street": fake.street\_name(),

"building\_number": fake.building\_number(),

"description": fake.text(),

}

locations.append(location)

with open(filename, "w") as file:

writer = csv.DictWriter(file, fieldnames=locations[0].keys())

writer.writeheader()

writer.writerows(locations)

def generate\_alarms(filename, count):

alarms = []

for i in range(count):

alarm = {

"id": i+1,

"power": fake.random\_element(elements=("accumulator", "wired", "combined")),

"sensor": fake.random\_element(elements=("movement", "glass\_breakage", "door\_opening", "temperature", "vibration")),

"connection": fake.random\_element(elements=("wired", "wifi", "radio")),

"alert": fake.random\_element(elements=("sms", "email", "phone\_call", "app\_notification")),

"max\_range": fake.random\_int(min=1, max=100),

"model": fake.word(),

"brand": fake.word(),

"details": fake.text()

}

alarms.append(alarm)

with open(filename, "w") as file:

writer = csv.DictWriter(file, fieldnames=alarms[0].keys())

writer.writeheader()

writer.writerows(alarms)

def generate\_protected\_objects(filename, count):

protected\_objects = []

for i in range(count):

protected\_object = {

"id": i + 1,

"name": fake.word(),

"location\_id": fake.random\_int(min=1, max=count),

}

protected\_objects.append(protected\_object)

with open(filename, "w") as file:

writer = csv.DictWriter(file, fieldnames=protected\_objects[0].keys())

writer.writeheader()

writer.writerows(protected\_objects)

def generate\_alarms\_on\_objects(filename, count):

alarms\_on\_objects = []

for i in range(count):

alarm\_on\_object = {

"id": i + 1,

"alarm\_id": fake.random\_int(min=1, max=count),

"object\_id": fake.random\_int(min=1, max=count),

}

alarms\_on\_objects.append(alarm\_on\_object)

with open(filename, "w") as file:

writer = csv.DictWriter(file, fieldnames=alarms\_on\_objects[0].keys())

writer.writeheader()

writer.writerows(alarms\_on\_objects)

def generate\_alarm\_events(filename, count):

alarm\_events = []

for i in range(count):

alarm\_event = {

"id": i + 1,

"alarm\_id": fake.random\_int(min=1, max=count),

"triggered\_at": fake.date\_time\_this\_year(),

}

alarm\_events.append(alarm\_event)

with open(filename, "w") as file:

writer = csv.DictWriter(file, fieldnames=alarm\_events[0].keys())

writer.writeheader()

writer.writerows(alarm\_events)

def generate\_alarm\_events\_results(filename, count):

alarm\_events\_results = []

alarm\_event\_result\_options = ["відсутність взлому", "виявлено взлом", "виявлено відкриті двері", "виявлено відкрите вікно",

"виявлено відкриту дверцята", "виявлено відкриту люк", "виявлено відкриту кришку", "виявлено відкритий люк",

"виявлено підпал", "виявлено витік газу", "виявлено витік води", "виявлено витік палива", "виявлено витік рідини",

"крадіжка", "пожежа", "проникнення", "проникнення в приміщення", "проникнення на територію", "проникнення на об'єкт",

]

for i in range(count):

alarm\_event\_result = {

"id": i + 1,

"event\_id": fake.random\_int(min=1, max=count),

"result": fake.random\_element(elements=alarm\_event\_result\_options),

}

alarm\_events\_results.append(alarm\_event\_result)

with open(filename, "w") as file:

writer = csv.DictWriter(file, fieldnames=alarm\_events\_results[0].keys())

writer.writeheader()

writer.writerows(alarm\_events\_results)

def generate\_employees\_departure\_on\_events(filename, count):

employees\_departure\_on\_events = []

for i in range(count):

employee\_departure\_on\_event = {

"id": i + 1,

"employee\_id": fake.random\_int(min=1, max=count),

"event\_id": fake.random\_int(min=1, max=count),

"vehicle\_id": fake.random\_int(min=1, max=count),

"arrived\_at": fake.date\_time\_this\_year(),

}

employees\_departure\_on\_events.append(employee\_departure\_on\_event)

with open(filename, "w") as file:

writer = csv.DictWriter(file, fieldnames=employees\_departure\_on\_events[0].keys())

writer.writeheader()

writer.writerows(employees\_departure\_on\_events)

if \_\_name\_\_ == "\_\_main\_\_":

count = 10000

generate\_vehicles("vehicles.csv", count)

generate\_employees("employees.csv", count)

generate\_alarms("alarms.csv", count)

generate\_locations("locations.csv", count)

generate\_protected\_objects("protected\_objects.csv", count)

generate\_alarms\_on\_objects("alarms\_on\_objects.csv", count)

generate\_alarm\_events("alarm\_events.csv", count)

generate\_alarm\_events\_results("alarm\_events\_results.csv", count)

generate\_employees\_departure\_on\_events("employees\_departure\_on\_events.csv", count)