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 об’єкти, які охороняються - це адреса, назва об’єкта, 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, об’єкт, 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 розв’язання поставленої задачі використати курсори.

4) Створити запити на основі їх текстового формулювання:

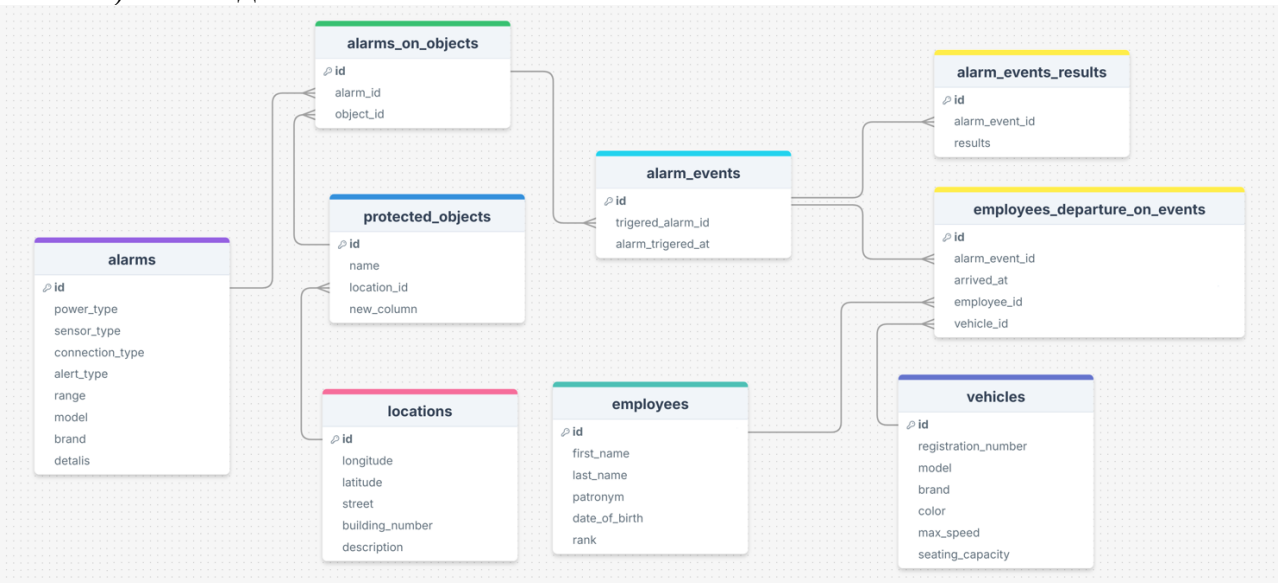
a) Об’єкти, котрі розміщуються на проспекті Перемоги та на них за минулий рік виїжджали не менше 5 разів.

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

c) 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



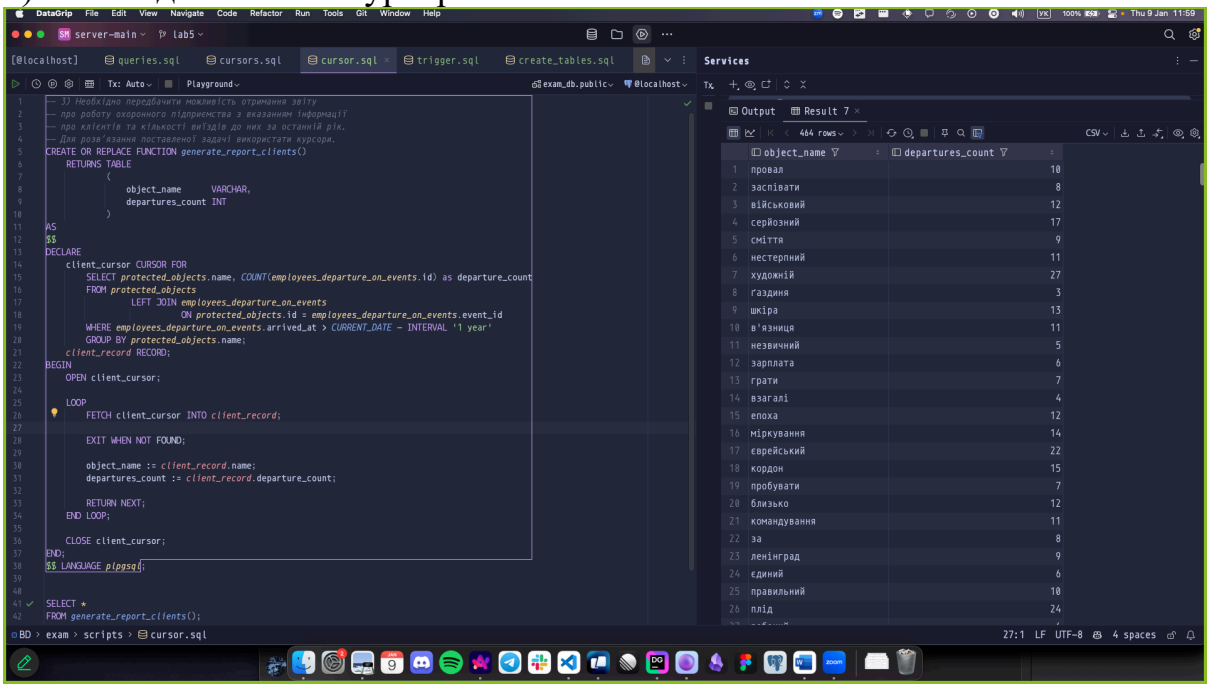
2) Тригери на перевірку значень

```
1   -- Час прибуття на подію не може бути раніше часу спрацювання сигналу.
2
3   CREATE OR REPLACE FUNCTION check_arrival_time()
4       RETURNS TRIGGER AS
5   $$
6   BEGIN
7       IF NEW.arrived_at < (SELECT triggered_at FROM alarm_events WHERE id = NEW.event_id) THEN
8           RAISE EXCEPTION 'Arrival time cannot be before the event time.';
9       END IF;
10
11      RETURN NEW;
12  END;
13  $$ LANGUAGE plpgsql;
14
15  CREATE TRIGGER trg_check_arrival_time
16      BEFORE INSERT OR UPDATE
17      ON employees_departure_on_events
18      FOR EACH ROW
19  EXECUTE FUNCTION check_arrival_time();
20
21  INSERT INTO employees_departure_on_events (employee_id, event_id, vehicle_id, arrived_at)
22  VALUES ( employee_id 1,  event_id 1,  vehicle_id 1,  arrived_at '2021-01-01 12:00:00');
[2025-01-09 11:55:23] [P0001] ERROR: Arrival time cannot be before the event time.
[2025-01-09 11:55:23] Where: PL/pgSQL function check_arrival_time() line 4 at RAISE
```

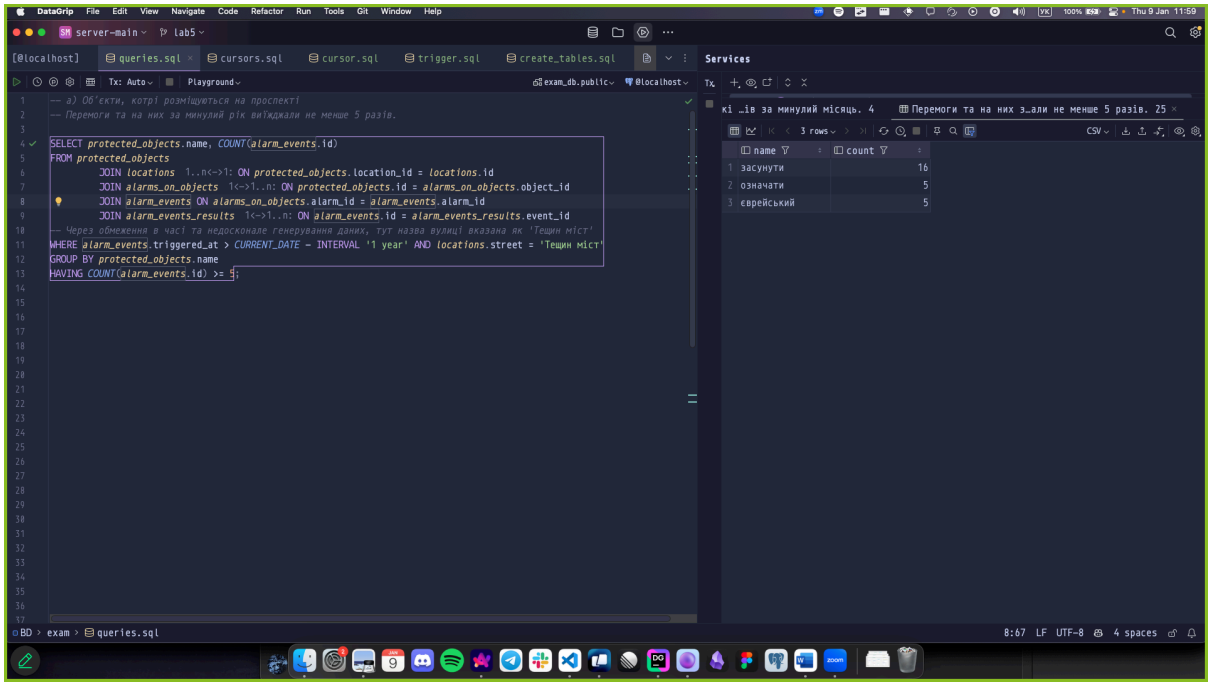
```sql
-- Швидкість транспортного засобу не може перевищувати 200 км/год.
CREATE OR REPLACE FUNCTION check_max_speed()
    RETURNS TRIGGER AS
$$
BEGIN
    IF NEW.max_speed > 200 THEN
        RAISE EXCEPTION 'Speed cannot be greater than 200 km/h.';
    END IF;

    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 ( id 1,  registration_number 'AA1234AA',  model 'Model',  brand 'Brand',  color 'Color',  max_speed 201,  seating_capacity 5,  year_of_manufacture 2021);
```

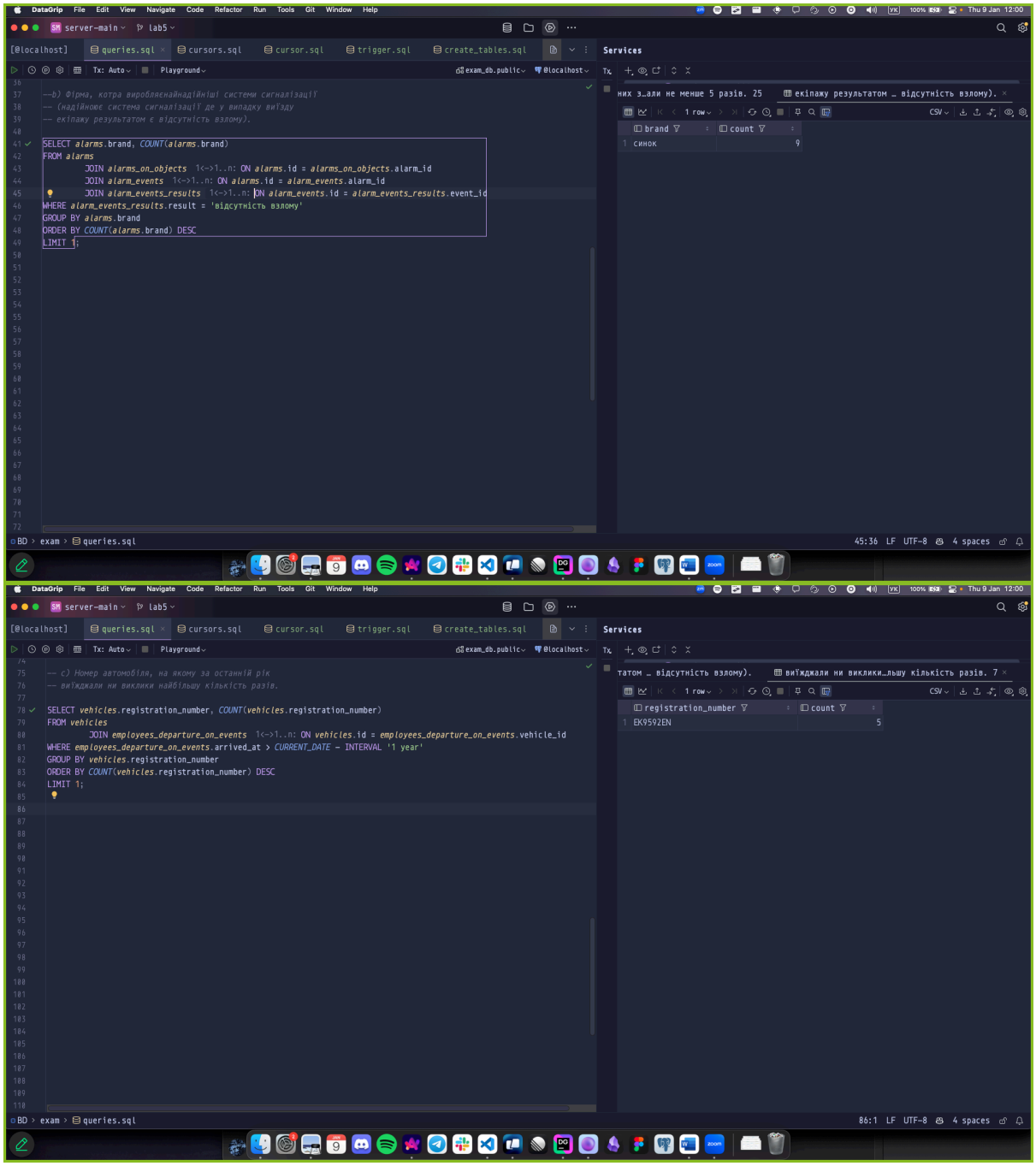
```
[P0001] ERROR: Speed cannot be greater than 200 km/h.
Where: PL/pgSQL function check_max_speed() line 4 at RAISE
```

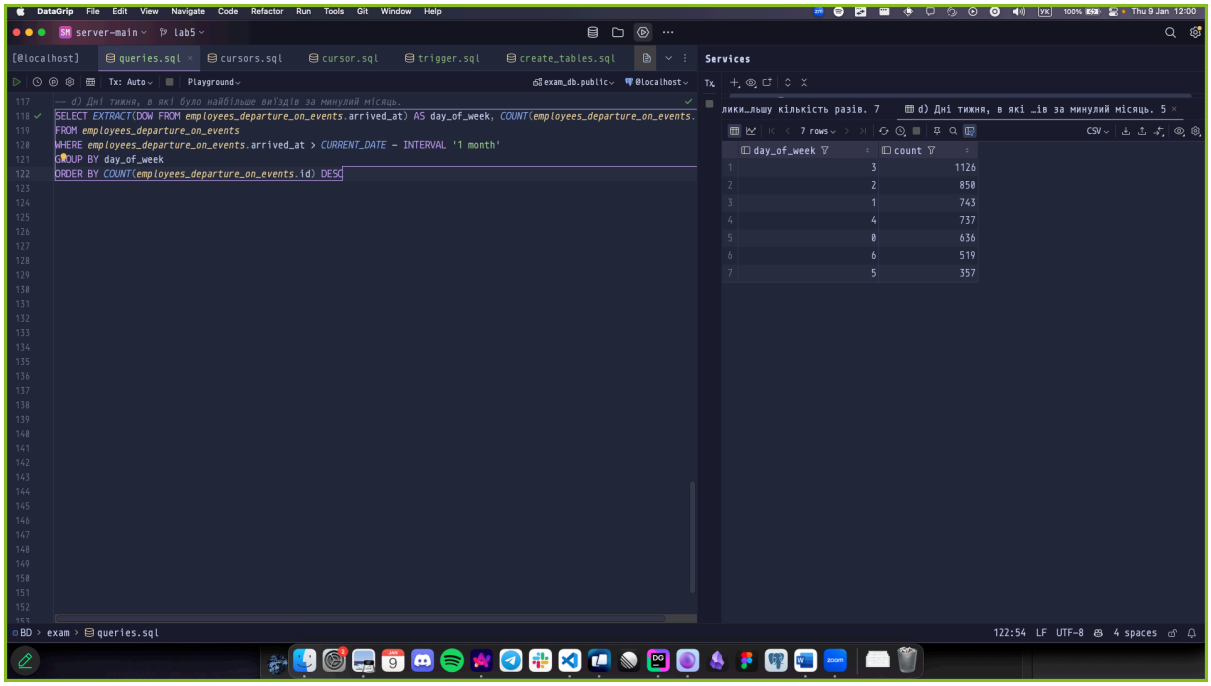
3) Звіт за допомогою курсорів



4) Запити


DataGrip File Edit View Navigate Code Refactor Run Tools Git Window Help
server-main lab5
[@localhost] queries.sql cursors.sql cursor.sql trigger.sql create_tables.sql
Tx: Auto Playground exam_db.public @localhost
-- a) Об'єкти, котрі розміщуються на проспекті
-- Перемоги та на них за минулий рік витріджали не менше 5 разів.
SELECT protected_objects.name, COUNT(alarm_events.id)
FROM protected_objects
JOIN locations 1..n<->1: ON protected_objects.location_id = locations.id
JOIN alarms_on_objects 1<->1..n: 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 1<->1..n: 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;
Services
хі ...ів за минулий місяць. 4
Перемоги та на них з...али не менше 5 разів. 25
3 rows
CSV
name
count
1 засунути 16
2 означати 5
3 єврейський 5
BD > exam > queries.sql
8:67 LF UTF-8 4 spaces


server-main
lab5
queries.sql
cursors.sql
cursor.sql
trigger.sql
create_tables.sql
Services
-- b) Фірма, котра виробляєнайдійнішІ 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 1<->1..n: ON alarms.id = alarms_on_objects.alarm_id
JOIN alarm_events 1<->1..n: ON alarms.id = alarm_events.alarm_id
JOIN alarm_events_results 1<->1..n: 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;
brand
count
синок
9
BD > exam > queries.sql
45:36 LF UTF-8 4 spaces
-- c) 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 1<->1..n: 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;
registration_number
count
EK9592EN
5
BD > exam > queries.sql
86:1 LF UTF-8 4 spaces



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 розв’язання 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();

-- a) Об’єкти, 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;



-- c) 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.cs
v", count)
```