

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 готельного комплексу \_\_\_\_\_

Студента (ки) 2 курсу ІП-13  
групи  
спеціальності 121 «Інженерія програмного  
забезпечення»

Лисенко Анастасії

Олегівни

(прізвище та ініціали)

Керівник Ліщук О.В.

(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 ІІІ-13 Семестр 3

**З А В Д А Н Н Я  
НА КУРСОВУ РОБОТУ СТУДЕНТУ**

Лисенко Анастасії Олегівні  
(прізвище, ім'я, по батькові)

1. Тема роботи База даних готельного комплексу

керівник роботи Ліщук О.В.  
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

2. Строк подання студентом роботи  
08.01.2023

3. Вихідні дані до роботи

створена база даних відповідно до умови, SQL скрипти

4. 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 системи управління базою даних

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

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)  
ER-діаграма, реляційна схема бази даних, приклади використання SQL скриптів

6. Дата видачі

завдання 08.11.2022

## КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання курсового проекту	Строк виконання етапів проекту	Примітка
1	Аналіз предметного середовища	12.11.2022	
2	Побудова ER-моделі	23.11.2022	
3	Побудова реляційної схеми з ER-моделі	5.12.2022	
4	Створення бази даних, у форматі обраної системи управління базою даних	5.12.2022	
5	Створення користувачів бази даних	7.12.2022	
6	Імпорт даних з використанням засобів СУБД в створену базу даних	15.12.2022	
7	Створення мовою SQL запитів	25.12.2022	
8	Оптимізація роботи запитів	02.01.2023	
9	Оформлення пояснювальної записки	10.01.2022	
10	Захист курсової роботи	10.01.2022	

**Студент**

\_\_\_\_\_ Лисенко А.О.  
(підпис ) (прізвище та ініціали)

**Керівник роботи**

\_\_\_\_\_ Ліщук О.В.  
(підпис ) (прізвище та ініціали)

## *Contents*

Вступ.....	4
1. Опис предметного середовища .....	5
2. Постановка задачі .....	6
3. ER-Діаграма.....	7
3.1.Бізнес-правила:.....	7
3.2.Вибір сутностей: .....	7
3.3.Набори атрибутів сутностей .....	8
4. Реляційна модель бази даних.....	12
5. Реалізація бази даних.....	13
6. Створення користувачів .....	18
7. SQL запити.....	21
7.1.Створення тригерів на таблиці, що будуть змінюватись користувачами. 21	
7.2.DML-запити .....	28
7.3.Процедури та функції .....	42
7.4.Приклад роботи індексів .....	49
Висновок .....	51
Посилання на джерела .....	52

## Вступ

Бази даних використовуються для зберігання та керування великими обсягами структурованих даних. Вони є важливим інструментом для багатьох компаній, оскільки дозволяють ефективно організовувати та маніпулювати даними, а також дозволяють легко отримувати певну інформацію.

Використання бази даних може значно підвищити ефективність і продуктивність бізнесу. Наприклад, компанія може використовувати базу даних для зберігання інформації про клієнтів, такої як контактні дані та історія покупок. Потім можна легко отримати доступ до цієї інформації та отримати запит для створення цільових маркетингових кампаній або надання персоналізованого обслуговування клієнтів. Крім того, централізоване зберігання всієї цієї інформації спрощує співпрацю та обмін даними всередині компанії.

Бази даних також можуть допомогти підприємствам приймати кращі рішення, надаючи доступ до точних і актуальних даних. Наприклад, компанія може використовувати базу даних для відстеження тенденцій продажів і аналізу поведінки споживачів, що може стати основою для стратегічного планування та розподілу ресурсів.

Підсумовуючи, бази даних є цінним інструментом для бізнесу, оскільки вони дозволяють організовувати, маніпулювати та отримувати великі обсяги даних, підвищуючи ефективність і обґрунтовуючи прийняття рішень. В даній курсовій роботі я виконала завдання створення бази даних готельного комплексу. Створена база даних суттєво полегшить процес роботи готельного комплексу як структури.

## 1. Опис предметного середовища

Готельний комплекс – велика багатофункціональна будівля. Метою готельного комплексу є забезпечення проживання та обслуговування мандрівників і туристів. У базі даних готельного комплексу, як правило, є таблиці для зберігання інформації про номери, гостей, працівників, бронювання та різні інші аспекти діяльності готелю. Процедури та функції можна використовувати для виконання певних завдань, таких як створення й керування бронюваннями, обчислення рівня зайнятості номерів і створення звітів. Тригери можна використовувати для забезпечення виконання бізнес-правил і забезпечення цілісності даних. Використовуючи базу даних для зберігання та керування даними про готельний комплекс, можна легше відстежувати та аналізувати інформацію про гостей, номери та інші аспекти роботи готелю, а також приймати більш обґрунтовані рішення про те, як керувати та покращувати бізнес.

## 2. Постановка задачі

Метою даної роботи є розробка бази даних для готельного комплексу. Тобто організація даних таким чином, щоб робота готельного комплексу виконувалась максимально ефективно. Отже основні задачі:

Гость готелю повинен мати змогу:

- Зробити бронювання на номер
- Отримати знижку за пільговою групою, якщо він до неї належить
- Заздалегідь дізнатися ціну на певний тип номеру в залежності від часу перебування в готелі
- Подивитися всі свої бронювання

Працівник готелю повинен мати змогу:

- Переглянути список вільних кімнат на певний термін часу
- Зробити бронювання для певного гостя
- Переглянути усі бронювання в готелі
- Змінити ціну кімнати за потребою

### 3. ER-Діаграма

Після аналізу було виділено такі сутності та зв'язки між ними:

#### 3.1. Бізнес-правила:

- Дата в'їзду до готелю не може бути меншою за дату виїзду.
- Кожен гость та працівник готелю має бути повнолітнім.
- В одну й ту саму кімнату в один і той самий час не можуть заїхати різні клієнти.
- В готелі не може бути двох клієнтів з однаковими паспортними даними.

#### 3.2. Вибір сутностей:

- Гость готелю
- Робітник готелю
- Посада робітника
- Готель
- Тип кімнати
- Кімната
- Бронювання
- В'їзд
- Виїзд
- Типи знижок
- Знижки



### 3.3. Набори атрибутів сутностей

Таблиця 3.1. – Набір сутностей та їх атрибутів

Сутність	Атрибути
Booking	booking_id room_id customer_id checkin_date checkout_date duration
Booking_discount	booking_discount_id booking_id discount_id
Check_in	check_in_id check_in_date check_out_date manager_id
Check_out	check_out_id check_out_date manager_id
Customer	customer_id first_name last_name additional_info passport_number passport_series age gender

Продовження таблиці 3.1.

Discount	discount_id reason_for percent_amount
Employee	employee_id first_name last_name passport_number passport_series age gender type_of_employee_id working_hours_month
Hotel	hotel_id hotel_name capacity location
Room	room_id room_number room_type_id hotel_id housekeeper_id
Room_type	room_type_id room_name bedroom_amount price_per_night
Type_of_employee	type_of_employee_id position_name wage

Готель може мати багато номерів, тому зв'язок між таблицями «Hotel» і «Room» є «один до багатьох».

Кімната може бути пов'язана лише з одним типом кімнати, тому зв'язок між таблицями «Room» і «Room\_type» є «один до одного».

Клієнт може зробити багато бронювань, тому зв'язок між таблицями «Customer» і «Booking» є «один до багатьох».

Бронювання пов'язане з одним номером і одним клієнтом, тому зв'язок між таблицями «Booking» і «Room», а також таблицями «Booking» і «Customer» є «один до одного».

До бронювання може бути застосовано багато знижок, тому зв'язок між таблицями «Booking» та «Discount» є «один до багатьох».

Робітник готелю може бути пов'язаний із багатьма бронюваннями (або як менеджер, відповідальний за реєстрацію клієнта, або як працівник, який обробив бронювання), тому зв'язок між таблицями «Employee» і «Booking» є «один до багатьох».

Робітник готелю може бути пов'язаний лише з одним типом співробітника, тому зв'язок між таблицями «Employee» і «Type\_of\_employee» є «один до одного».

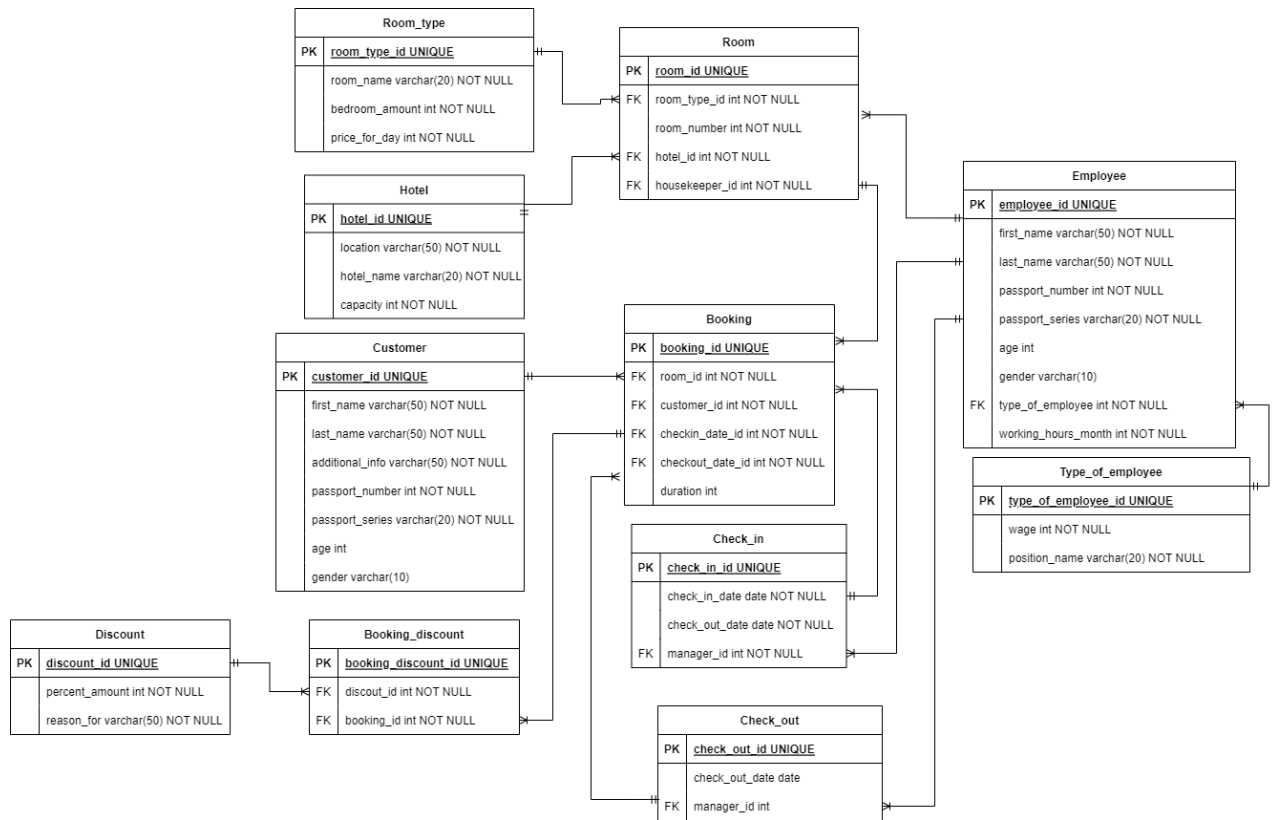


Рисунок 3.3.1 - ER-Діаграма

## 4. Реляційна модель бази даних

Побудова необхідних відношень та визначення первинних та зовнішніх ключів.

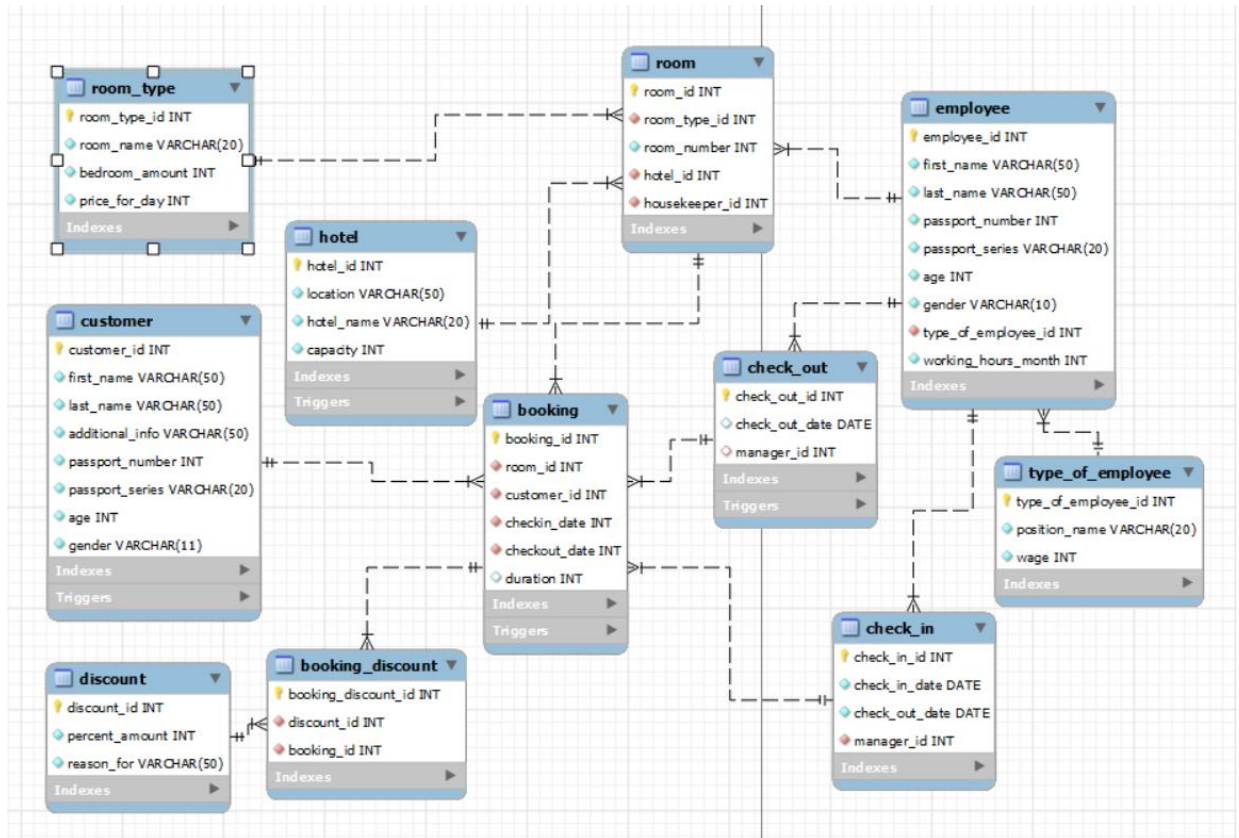


Рисунок 4.1 - Реляційна модель бази даних

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.

## 5. Реалізація бази даних

### 5.1. Створення бази даних

```
DROP DATABASE IF EXISTS hotel_complex;  
CREATE DATABASE hotel_complex;  
USE hotel_complex;
```

```
DROP TABLE IF EXISTS Hotel;  
DROP TABLE IF EXISTS Room_type;  
DROP TABLE IF EXISTS Room;  
DROP TABLE IF EXISTS Customer;  
DROP TABLE IF EXISTS Booking;  
DROP TABLE IF EXISTS Discount;  
DROP TABLE IF EXISTS Booking_discount;  
DROP TABLE IF EXISTS Check_in;  
DROP TABLE IF EXISTS Check_out;  
DROP TABLE IF EXISTS Employee;  
DROP TABLE IF EXISTS Type_of_employee;
```

```
CREATE TABLE Hotel(  
    hotel_id INT AUTO_INCREMENT PRIMARY KEY,  
    location VARCHAR(50) NOT NULL,  
    hotel_name VARCHAR(20) NOT NULL,  
    capacity INT NOT NULL  
);
```

```
CREATE TABLE Room_type(  
    room_type_id INT AUTO_INCREMENT PRIMARY KEY,  
    room_name VARCHAR(20) NOT NULL,  
    bedroom_amount INT NOT NULL,  
    price_for_day INT NOT NULL  
);
```

```
CREATE TABLE Type_of_employee(  
    type_of_employee_id INT AUTO_INCREMENT PRIMARY KEY,  
    position_name VARCHAR(20) NOT NULL,  
    wage INT NOT NULL  
);
```

```
CREATE TABLE Employee(  
    employee_id INT AUTO_INCREMENT PRIMARY KEY,  
    first_name VARCHAR(50) NOT NULL,
```

```

last_name VARCHAR(50) NOT NULL,
passport_number INT NOT NULL,
passport_series VARCHAR(20) NOT NULL,
age INT NOT NULL,
gender VARCHAR(10) NOT NULL,
type_of_employee_id INT NOT NULL,
working_hours_month INT NOT NULL,
FOREIGN          KEY(type_of_employee_id)          REFERENCES
Type_of_employee(type_of_employee_id) ON DELETE CASCADE
);

```

```

CREATE TABLE Room(
    room_id INT AUTO_INCREMENT PRIMARY KEY,
    room_type_id INT NOT NULL,
    room_number INT NOT NULL,
    hotel_id INT NOT NULL,
    housekeeper_id INT NOT NULL,
FOREIGN          KEY(housekeeper_id)          REFERENCES
Employee(employee_id) ON DELETE CASCADE,
    FOREIGN KEY(hotel_id) REFERENCES Hotel(hotel_id) ON
DELETE CASCADE,
    FOREIGN          KEY(room_type_id)          REFERENCES
Room_type(room_type_id) ON DELETE CASCADE
);

```

```

CREATE TABLE Customer(
    customer_id INT AUTO_INCREMENT PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    additional_info VARCHAR(50) NOT NULL,
    passport_number INT NOT NULL,
    passport_series VARCHAR(20) NOT NULL,
    age INT NOT NULL,
    gender VARCHAR(11) NOT NULL
);

```

```

CREATE TABLE Check_in(
    check_in_id INT AUTO_INCREMENT PRIMARY KEY,
    check_in_date DATE NOT NULL,
    check_out_date DATE NOT NULL,

```

```

        manager_id INT NOT NULL,
        FOREIGN          KEY(manager_id)          REFERENCES
Employee(employee_id) ON DELETE CASCADE
    );

```

```

CREATE TABLE Check_out(
    check_out_id INT AUTO_INCREMENT PRIMARY KEY,
    check_out_date DATE ,
    manager_id INT ,
    FOREIGN KEY(manager_id) REFERENCES Employee(employee_id)
ON DELETE CASCADE
);

```

```

CREATE TABLE Booking(
    booking_id INT AUTO_INCREMENT PRIMARY KEY,
    room_id INT NOT NULL,
    customer_id INT NOT NULL,
    checkin_date INT NOT NULL,
    checkout_date INT NOT NULL,
    duration INT,
    FOREIGN KEY(room_id) REFERENCES Room(room_id) ON
DELETE CASCADE,
    FOREIGN KEY(customer_id) REFERENCES Customer(customer_id)
ON DELETE CASCADE,
    FOREIGN KEY(checkin_date) REFERENCES Check_in(check_in_id)
ON DELETE CASCADE,
    FOREIGN          KEY(checkout_date)          REFERENCES
Check_out(check_out_id) ON DELETE CASCADE
);

```

```

CREATE TABLE Discount(
    discount_id INT AUTO_INCREMENT PRIMARY KEY,
    percent_amount INT NOT NULL,
    reason_for VARCHAR(50) NOT NULL
);

```

```

CREATE TABLE Booking_discount(
    booking_discount_id INT AUTO_INCREMENT PRIMARY KEY,
    discount_id INT NOT NULL,
    booking_id INT NOT NULL,
    FOREIGN KEY(booking_id) REFERENCES Booking(booking_id) ON
DELETE CASCADE,

```



```

        FOREIGN KEY(discount_id) REFERENCES Discount(discount_id) ON
DELETE CASCADE
    );

```

```

ALTER TABLE Customer
ADD CONSTRAINT CheckAge CHECK(age >= 18);

```

```

ALTER TABLE Employee
ADD CONSTRAINT CheckAgeEmpl CHECK(age >= 18);

```

```

-- ALTER TABLE Customer
-- ADD CONSTRAINT CheckGenderCust CHECK(gender IN('Female',
'Male', 'Non-binary'));

```

```

ALTER TABLE Employee
ADD CONSTRAINT CheckGenderEmpl CHECK(gender IN('Female',
'Male', 'Non-binary'));

```

```

ALTER TABLE Room_type
ADD CONSTRAINT CheckRoom_type CHECK(room_name IN('Normal',
'Half-Luxe', 'Luxe'));

```

```

ALTER TABLE Room_type
ADD CONSTRAINT CheckBedroom_amount CHECK(bedroom_amount
IN(1, 2, 3, 4));

```

```

ALTER TABLE Check_in
ADD CONSTRAINT Date_check CHECK(check_in_date <
check_out_date);

```

```

ALTER TABLE Type_of_employee
ADD CONSTRAINT Wage_check CHECK(wage >= 10);

```

```

ALTER TABLE Type_of_employee
ADD CONSTRAINT Type_of_employee_check CHECK(position_name
IN('manager', 'housekeeper', 'porter', 'driver'));

```

## 5.2. Імпорт даних в базу даних

```
LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server
8.0/Uploads/employee.csv' INTO TABLE Employee
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
IGNORE 1 ROWS
(first_name,last_name,passport_number,passport_series,age,gender,type_of
_employee_id,working_hours_month);
```

Використовуючи даний скрипт завантажую дані в базу даних з .csv файлу. Маю такі результати в таблиці:

employee_id	first_name	last_name	passport_number	passport_series	age	gender	type_of_employee_id	working_hours_month
1	Norrie	Laurenceau	388998707	63spij4808	26	Female	3	25
2	Charis	Burgett	910392778	58kgsw5571	77	Female	1	40
3	Dallon	Pygott	493031705	50cuby2686	42	Non-binary	2	50
4	Hubert	Czaja	334105830	97rhfs6640	54	Male	4	38
5	Morie	Piscopo	614448470	08kqsb8155	24	Non-binary	4	21
6	Rooney	Crisell	72438128	40rcmr4040	35	Male	1	32
7	Kelsey	Marryatt	936409132	37hsjc5469	74	Male	3	15
8	Dorian	Syphus	51415028	07eulg4850	39	Male	3	45
9	Inglis	Greendale	234220241	24gosy8122	68	Non-binary	3	30
10	Giovanni	Tizard	269330016	00bev4250	25	Male	4	50
11	Evie	Milstead	744024793	94gmqj1068	49	Female	2	25
12	Nadean	Jeaffreson	709069475	03grll3774	68	Female	4	30
13	Carey	Clappson	333143121	70mbal1008	30	Male	4	24
14	Rhianon	Minghetti	731256439	73krjq0975	26	Female	4	12
15	Hanan	Trustram	924384216	17sfqu6255	70	Male	2	32
16	Meredeth	Adicot	659536071	51twlm2140	32	Non-binary	3	18
17	Hannis	Brose	123790181	68uevt2609	28	Female	4	29
18	Haleigh	Polsin	346888256	90nnhm5608	59	Female	2	39

Рисунок 5.1 - приклад заповнення таблиці бази даних

Сам файл має вигляд:

1	first_name,last_name,passport_number,passport_series,age,gender,type_of_employee_id,working_hours_month
2	Norrie,Laurenceau,388998707,63spij4808,26,Female,3,25
3	Charis,Burgett,910392778,58kgsw5571,77,Female,1,40
4	Dallon,Pygott,493031705,50cuby2686,42,Non-binary,2,50
5	Hubert,Czaja,334105830,97rhfs6640,54,Male,4,38
6	Morie,Piscopo,614448470,08kqsb8155,24,Non-binary,4,21
7	Rooney,Crisell,72438128,40rcmr4040,35,Male,1,32
8	Kelsey,Marryatt,936409132,37hsjc5469,74,Male,3,15
9	Dorian,Syphus,51415028,07eulg4850,39,Male,3,45
10	Inglis,Greendale,234220241,24gosy8122,68,Non-binary,3,30
11	Giovanni,Tizard,269330016,00bev4250,25,Male,4,50
12	Evie,Milstead,744024793,94gmqj1068,49,Female,2,25
13	Nadean,Jeaffreson,709069475,03grll3774,68,Female,4,30
14	Carey,Clappson,333143121,70mbal1008,30,Male,4,24
15	Rhianon,Minghetti,731256439,73krjq0975,26,Female,4,12
16	Hanan,Trustram,924384216,17sfqu6255,70,Male,2,32
17	Meredeth,Adicot,659536071,51twlm2140,32,Non-binary,3,18
18	Hannis,Brose,123790181,68uevt2609,28,Female,4,29
19	Haleigh,Polsin,346888256,90nnhm5608,59,Female,2,39
20	Wood,Angell,708553349,05aytf4240,27,Male,4,35
21	Gael,Paulley,982379650,87baaa4569,36,Non-binary,2,24
22	Myrlene,Castagnier,711063157,88nfl3296,75,Non-binary,3,34
23	Ade,Crown,246398070,35rciz8330,23,Male,2,43

Рисунок 5.2 - приклад файлу .csv

## 6. Створення користувачів

```
drop user anastasiia@localhost;
```

```
drop user customer@localhost;
```

```
drop user manager@localhost;
```

```
flush privileges;
```

```
CREATE USER anastasiia@localhost IDENTIFIED BY 'developer';
```

```
CREATE USER customer@localhost IDENTIFIED BY 'customer';
```

```
CREATE USER manager@localhost IDENTIFIED BY 'manager';
```

```
GRANT ALL
```

```
ON hotel_complex.*
```

```
TO anastasiia@localhost;
```

```
GRANT EXECUTE
```

```
ON PROCEDURE hotel_complex.getCustomerBookings
```

```
TO customer@localhost;
```

```
GRANT EXECUTE
```

```
ON FUNCTION hotel_complex.calculateTotalCost
```

```
TO customer@localhost;
```

```
GRANT EXECUTE  
  
ON PROCEDURE hotel_complex.createCheckIn_CheckOut  
  
TO customer@localhost;
```

```
GRANT INSERT, UPDATE, DELETE  
  
ON hotel_complex.*  
  
TO manager@localhost;
```

```
GRANT EXECUTE  
  
ON PROCEDURE hotel_complex.getFreeRooms  
  
TO manager@localhost;
```

```
GRANT EXECUTE  
  
ON PROCEDURE hotel_complex.createBooking  
  
TO manager@localhost;
```

```
GRANT EXECUTE  
  
ON PROCEDURE hotel_complex.update_room_price  
  
TO manager@localhost;
```

```
SHOW GRANTS FOR anastasiia@localhost;  
  
SHOW GRANTS FOR customer@localhost;
```

SHOW GRANTS FOR manager@localhost;

Результат виконання SHOW GRANTS FOR для кожного з користувачів:

	Grants for anastasiia@localhost
▶	GRANT USAGE ON *.* TO `anastasiia`@`localhost`
	GRANT ALL PRIVILEGES ON `hotel_complex`.* TO `anastasiia`@`localhost`

Рисунок 6.1 - привілеї адміна

	Grants for customer@localhost
▶	GRANT USAGE ON *.* TO `customer`@`localhost`
	GRANT EXECUTE ON PROCEDURE `hotel_complex`.`createcheckin_checkout` TO `customer`@`localhost`
	GRANT EXECUTE ON PROCEDURE `hotel_complex`.`getcustomerbookings` TO `customer`@`localhost`
	GRANT EXECUTE ON FUNCTION `hotel_complex`.`calculatetotalcost` TO `customer`@`localhost`

Рисунок 6.2 - привілеї гостя готелю

	Grants for manager@localhost
▶	GRANT USAGE ON *.* TO `manager`@`localhost`
	GRANT INSERT, UPDATE, DELETE ON `hotel_complex`.* TO `manager`@`localhost`
	GRANT EXECUTE ON PROCEDURE `hotel_complex`.`createbooking` TO `manager`@`localhost`
	GRANT EXECUTE ON PROCEDURE `hotel_complex`.`getfreerooms` TO `manager`@`localhost`
	GRANT EXECUTE ON PROCEDURE `hotel_complex`.`update_room_price` TO `manager`@`localhost`

Рисунок 6.3 - привілеї менеджера

## 7. SQL запити

7.1. Створення тригерів на таблиці, що будуть змінюватись користувачами.

```
DROP TRIGGER IF EXISTS check_room_availability;
DELIMITER $$
```

```
CREATE TRIGGER check_room_availability BEFORE INSERT ON
Booking
```

```
FOR EACH ROW
```

```
BEGIN
```

```
    DECLARE room_id INT;
```

```
    DECLARE checkin_date DATE;
```

```
    DECLARE checkout_date DATE;
```

```
    SET room_id = NEW.room_id;
```

```
    SET checkin_date = (SELECT ci.check_in_date FROM Check_in ci
WHERE ci.check_in_id = NEW.checkin_date);
```

```
    SET checkout_date = (SELECT ci.check_out_date FROM Check_in ci
WHERE ci.check_in_id = NEW.checkin_date);
```

```
    -- SET checkout_date = (SELECT ci.check_out_date FROM Check_in ci
WHERE ci.check_out_id = NEW.checkout_date);
```

```
    IF EXISTS (SELECT 1 FROM Booking b
```

```
        INNER JOIN Check_in ci ON b.checkin_date = ci.check_in_id
```

```
        INNER JOIN Check_out co ON b.checkout_date = co.check_out_id
```

```
        WHERE b.room_id = room_id
```

```
        AND (ci.check_in_date BETWEEN checkin_date AND
checkout_date
```

```
OR ci.check_out_date BETWEEN checkin_date AND
checkout_date)) THEN
```

```
SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'The room is not
available for the requested dates.';
```

```
END IF;
```

```
END$$
```

```
DELIMITER ;
```

```
36 • INSERT INTO Booking(room_id, customer_id, checkin_date, checkout_date,duration)
37 VALUES(2, 8, 17, 17, 0);
38
```

booking_id	room_id	customer_id	checkin_date	checkout_date	duration
16	2	37	16	16	14
17	2	5	17	17	6
18	1	5	17	17	6
19	3	37	18	18	14
NULL	NULL	NULL	NULL	NULL	NULL

#	Time	Action	Message
801	22:04:21	INSERT INTO Booking(room_id, customer_id, checkin_date, checkout_date,duration) VALUES(2, 5, 17, 17, 0)	Error Code: 1644. The room is not available for the requested dates.
802	22:14:01	SELECT * FROM booking LIMIT 0, 1000	19 row(s) returned
803	23:30:19	INSERT INTO Booking(room_id, customer_id, checkin_date, checkout_date,duration) VALUES(2, 8, 17, 17, 0)	Error Code: 1644. The room is not available for the requested dates.

Рисунок 7.1.1 - приклад роботи трігера, який блокує додання клієнту в кімнату, яка вже зайнята

Такі значення не можна ввести, бо ми вже маємо бронювання на ці дати і відповідно кімната вже зайнята гостем готелю.

Цей тригер називається "check\_room\_availability". Він виконується перед операцією вставки в таблиці "Booking" та виконується для кожного рядка, який вставляється. Він перевіряє, чи вільний номер, указаний в операції вставки, на запитані дати на основі дат заїзду та виїзду, збережених у таблицях "Check\_in" і "Check\_out". Якщо кімната недоступна, тригер викликає помилку з відповідним повідомленням.

```
DROP TRIGGER IF EXISTS check_customer_exists;
```

```
DELIMITER $$
```

```
CREATE TRIGGER check_customer_exists BEFORE INSERT ON
Customer
```

```
FOR EACH ROW
```

```
BEGIN
```

```
DECLARE passport_series VARCHAR(20);
```

```
DECLARE passport_number INT;
```

```
SET passport_series = NEW.passport_series;
```

```
SET passport_number = NEW.passport_number;
```

```
IF EXISTS (SELECT 1 FROM Customer WHERE passport_series =
passport_series AND passport_number = passport_number) THEN
```

```
SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'A customer
with the same passport series and passport number already exists.';
```

```
END IF;
```

```
END$$
```

```
DELIMITER ;
```

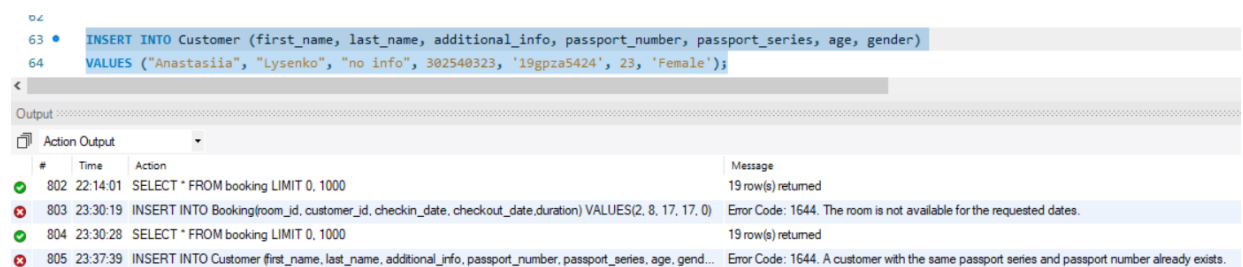


Рисунок 7.1.2 - приклад роботи трігера, який перевіряє чи вже є клієнт з серією та номером паспорту



Цей тригер викликається перед виконанням операції вставки в таблиці «Customer». Перевіряє, чи є в таблиці клієнт із такою ж серією та номером паспорта. Якщо клієнт із такою ж серією та номером паспорта вже існує, виникає помилка, і операція вставки не виконується.

```
DROP TABLE IF EXISTS Customer_Modification_Log;
CREATE TABLE Customer_Modification_Log (
    log_id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    customer_id INT NOT NULL,
    modified_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    old_first_name VARCHAR(50) NOT NULL,
    old_last_name VARCHAR(50) NOT NULL,
    old_additional_info VARCHAR(50) NOT NULL,
    old_passport_number INT NOT NULL,
    old_passport_series VARCHAR(20) NOT NULL,
    old_age INT NOT NULL,
    old_gender VARCHAR(10) NOT NULL
);
```

```
DROP TRIGGER IF EXISTS customer_modification_log;
DELIMITER //
CREATE TRIGGER customer_modification_log
AFTER UPDATE ON Customer
FOR EACH ROW
BEGIN
    INSERT INTO Customer_Modification_Log (customer_id, modified_at,
old_first_name, old_last_name, old_additional_info, old_passport_number,
old_passport_series, old_age, old_gender)
```

```
VALUES (OLD.customer_id, CURRENT_TIMESTAMP,
OLD.first_name, OLD.last_name, OLD.additional_info, OLD.passport_number,
OLD.passport_series, OLD.age, OLD.gender);
```

```
END //
```

```
DELIMITER ;
```

Початкове значення

49	Jayson	Jorissen	no info	221458180	50NKMC5432	90	Male
50	Burt	Backshall	no info	227912121	28akxr8465	89	Male
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Рисунок 7.1.3 - значення до модифікації даних

Змінене значення

49	Jayson	Jorissen	no info	221458180	50NKMC5432	90	Male
50	Anastasiia	Backshall	no info	227912121	28akxr8465	89	Male
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Рисунок 7.1.4 - після модифікації

Запис у створеній таблиці

log_id	customer_id	modified_at	old_first_name	old_last_name	old_additional_info	old_passport_number	old_passport_series	old_age	old_gender
1	50	2023-01-08 23:38:40	Burt	Backshall	no info	227912121	28akxr8465	89	Male
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Рисунок 7.1.5 - приклад роботи тригера, який записує усі зміни в таблиці гостя

Цей тригер називається "check\_checkout\_date" і він активується після оновлення в таблиці «Customer». Він вставляє рядок у таблицю «Customer\_Modification\_Log» із customer\_id, modified\_at, old\_first\_name, old\_last\_name, old\_additional\_info, old\_passport\_number, old\_passport\_series, old\_age і old\_gender старого запису клієнта. Це робиться за допомогою ключового слова OLD, яке посилається на старі дані в рядку, що оновлюється.

```
DROP TRIGGER IF EXISTS check_checkout_date;
```

```

DELIMITER $$

CREATE TRIGGER check_checkout_date BEFORE UPDATE ON
Check_out
FOR EACH ROW
BEGIN
    DECLARE check_in_id INT;
    DECLARE check_in_date DATE;

    SET check_in_id = NEW.check_out_id;

    SELECT ci.check_in_date INTO check_in_date
    FROM Check_in ci
    WHERE ci.check_in_id = check_in_id;

    IF NEW.check_out_date < check_in_date THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'The check-out
date must be later than the check-in date.';
    END IF;
END$$

DELIMITER ;

```

Цей тригер перевіряє той факт, що час виселення з готелю є більшим ніж час заселення.

Початкові дані:

	check_in_id	check_in_date	check_out_date	manager_id
	15	2022-01-18	2022-01-20	2
	16	2023-01-07	2023-01-21	2
▶	17	2022-02-07	2022-02-13	6
	18	2023-01-07	2023-01-21	45
✱	NULL	NULL	NULL	NULL

Рисунок 7.1.6 - поточні дані

При зміні дати видає помилку:

L23 • UPDATE Check\_out

L24 SET check\_out\_date = "2022-02-06"

L25 WHERE check\_out\_id = 17;

L26

Output

Action Output

#	Time	Action	Message
812	23:44:56	SELECT * FROM check_out LIMIT 0, 1000	18 row(s) returned
813	23:45:20	SELECT * FROM check_in LIMIT 0, 1000	18 row(s) returned
814	23:45:43	SELECT * FROM check_out LIMIT 0, 1000	18 row(s) returned
815	23:45:52	UPDATE Check_out SET check_out_date = "2022-02-06" WHERE check_out_id = 17	Error Code: 1644. The check-out date must be later than the check-in date.

Рисунок 7.1.7 - приклад роботи трігера, який блокує вставку дати виселення після дати заселення

```
DROP TRIGGER IF EXISTS prevent_hotel_deletion;
DELIMITER $$
```

```
CREATE TRIGGER prevent_hotel_deletion
BEFORE DELETE ON Hotel
FOR EACH ROW
BEGIN
    DECLARE hotel_id INT;
    DECLARE room_count INT;

    SET hotel_id = OLD.hotel_id;

    SELECT COUNT(*) INTO room_count
```

```
FROM Room
WHERE Room.hotel_id = hotel_id;
```

```
IF room_count > 0 THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Deleting a hotel
with rooms is not allowed.';
```

```
END IF;
END$$
```

```
DELIMITER ;
```

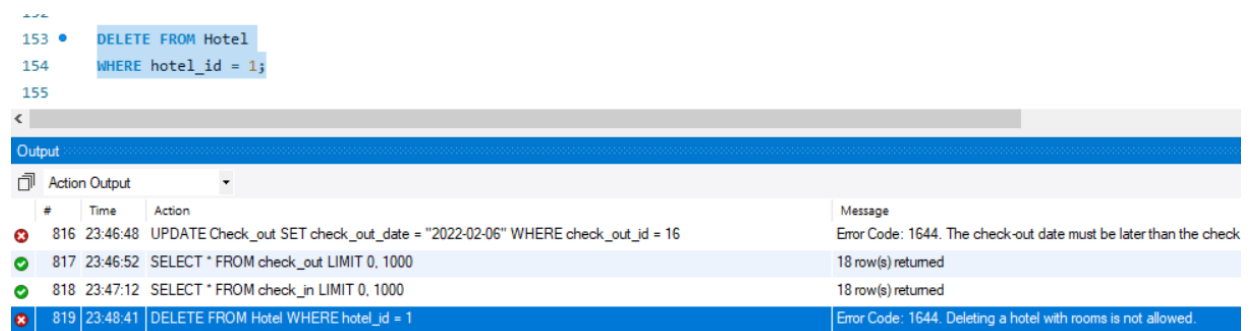


Рисунок 7.1.8 - приклад роботи трігера, який блокує видалення готелю в якому є кімнати

Цей тригер призначений для запобігання видаленню готелю з таблиці Hotel, якщо в таблиці Room є номери, пов'язані з ним. Коли робиться спроба видалити рядок із таблиці Hotel, тригер виконає та підрахує кількість номерів, пов'язаних із готелем, який видаляється в таблиці Room. Якщо кількість більше 0, тригер викличе помилку, і операцію видалення не буде дозволено продовжити.

## 7.2. DML-запити

### 1.

```
SELECT c.first_name, c.last_name, ci.check_in_date, ci.check_out_date,
b.duration, r.room_number, rt.room_name
```

```

FROM Customer c
INNER JOIN Booking b ON c.customer_id = b.customer_id
INNER JOIN Check_in ci ON b.checkin_date = ci.check_in_id
INNER JOIN Room r ON b.room_id = r.room_id
INNER JOIN Room_type rt ON r.room_type_id = rt.room_type_id
ORDER BY r.room_number;

```

Result Grid      Filter Rows: <input type="text"/>   Export:    Wrap Cell Content:							
	first_name	last_name	check_in_date	check_out_date	duration	room_number	room_name
▶	Thatcher	Cardis	2022-03-14	2022-04-04	21	1	Normal
	Parker	Gundry	2023-01-07	2023-01-21	14	1	Normal
	Iormina	Loddy	2022-10-07	2022-10-12	5	2	Half-Luxe
	Werner	Nowill	2023-01-07	2023-01-21	14	2	Half-Luxe
	Corinne	Formilli	2022-06-17	2022-08-31	75	3	Luxe
	Tobie	O'Flannery	2022-01-04	2022-01-27	23	4	Normal
	Davida	Rikard	2022-01-31	2022-02-03	3	5	Normal
	Leupold	Pittem	2022-05-15	2022-05-24	9	6	Normal
	Anastasiia	Backshall	2022-03-15	2022-04-06	22	7	Half-Luxe

Рисунок 7.2.1 - об'єднання даних про бронювання, клієнта та його номер

Цей SELECT отримує дані з таблиць Customer, Booking, Check\_in, Room і Room\_type у базі даних.

2.

```

SELECT c.first_name, c.last_name, c.passport_number, c.passport_series
FROM Customer c
INNER JOIN Booking b ON c.customer_id = b.customer_id
INNER JOIN Room r ON b.room_id = r.room_id
INNER JOIN Room_type rt ON r.room_type_id = rt.room_type_id
WHERE rt.room_name = 'Luxe' AND b.duration >= 7
ORDER BY b.duration ASC;

```




Result Grid				Filter Rows:		Export:		Wrap Cell
	first_name	last_name	passport_number	passport_series				
▶	Leopold	Watford	389069636	92slfo7649				
	Corinne	Formilli	62845215	14frhf2858				

Рисунок 7.2.2 - дані про гостя готелю, який проживав в ньому довше 7 днів в номері типу Luxe

Цей SELECT отримує ім'я, прізвище, номер паспорта та серію паспорта клієнтів, які забронювали номер типу «Люкс» і залишилися на 7 або більше днів, і впорядковує результати за тривалістю перебування в порядку зростання.

3.

```
SELECT b.room_id, c.first_name, c.last_name, d.percent_amount, d.reason_for
FROM booking_discount bd
INNER JOIN Discount d ON bd.discount_id = d.discount_id
INNER JOIN booking b ON bd.booking_id = b.booking_id
INNER JOIN customer c ON b.customer_id = c.customer_id
ORDER BY d.percent_amount;
```

room_id	first_name	last_name	percent_amount	reason_for
13	Anastasiia	Backshall	15	Large_family
15	Rani	Gerlts	15	Large_family
6	Leupold	Pittem	15	VPO
13	Anastasiia	Backshall	20	Disability
12	Leopold	Watford	20	Disability
3	Corinne	Formilli	20	ATO_veteran
11	Rani	Gerlts	30	Regular_customer

Рисунок 7.2.3 - повертає дані про гостя готелю та його знижки, якщо такі є

Цей оператор SELECT отримує дані з таблиць booking\_discount, Discount, Booking і Customer. Він повертає id номера, ім'я та прізвище клієнта, суму знижки у відсотках і причину знижки для всіх бронювань, які мають знижку. Результати впорядковані за розміром відсотка знижки.

4.

```

SELECT COUNT(c.customer_id) AS amount_discounts, d.percent_amount,
d.reason_for
FROM booking_discount bd
INNER JOIN Discount d ON bd.discount_id = d.discount_id
INNER JOIN booking b ON bd.booking_id = b.booking_id
INNER JOIN customer c ON b.customer_id = c.customer_id
GROUP BY d.reason_for;

```

	amount_discounts	percent_amount	reason_for
▶	2	20	Disability
	2	15	Large_family
	1	30	Regular_customer
	1	20	ATO_veteran
	1	15	VPO

Рисунок 7.2.4 - кількість гостей з певною знижкою

Цей запит знаходить кількість клієнтів готелю з певною причиною для знижки і групує знайдені результати по причині знижки.

5.

```

SELECT AVG(b.duration) AS average_duration, rt.room_name
FROM Booking b
INNER JOIN Check_in ci ON b.checkin_date = ci.check_in_id
INNER JOIN Room r ON b.room_id = r.room_id
INNER JOIN Room_type rt ON r.room_type_id = rt.room_type_id
GROUP BY rt.room_name;

```

Result Grid			Filter Rows:
	average_duration	room_name	
▶	10.0000	Normal	
	11.6000	Half-Luxe	
	31.3333	Luxe	

Рисунок 7.2.5 - середня довжина перебування в готелю за типами номерів



Цей оператор SELECT отримує середню тривалість усіх бронювань для кожного типу кімнати та відображає назву кімнати та середню тривалість у результатах.

6.

```
SELECT MAX(b.duration) AS maximum_duration, rt.room_name
FROM Booking b
INNER JOIN Check_in ci ON b.checkin_date = ci.check_in_id
INNER JOIN Room r ON b.room_id = r.room_id
INNER JOIN Room_type rt ON r.room_type_id = rt.room_type_id
GROUP BY rt.room_name
ORDER BY maximum_duration ASC;
```

	maximum_duration	room_name
▶	22	Half-Luxe
	23	Normal
	75	Luxe

Рисунок 7.2.6 - максимальна довжина перебування в готелю за типами номерів

Цей запит повертає максимальну тривалість бронювання для кожного типу номеру та впорядковує набір результатів у порядку зростання максимальної тривалості.

7.

```
SELECT SUM(b.duration) AS complete_duration, MONTH(ci.check_in_date) AS
Current_month
FROM Booking b
INNER JOIN Check_in ci ON b.checkin_date = ci.check_in_id
INNER JOIN Room r ON b.room_id = r.room_id
INNER JOIN Room_type rt ON r.room_type_id = rt.room_type_id
GROUP BY Current_month
```



ORDER

BY

Current\_month

asc;

Result Grid



Filter Rows:

	complete_duration	Current_month
▶	65	1
	7	2
	43	3
	15	5
	75	6
	2	7
	7	8
	5	10
	10	11
	13	12

Рисунок 7.2.7 - загальна кількість днів, які гості готелю провели в готелі за місяцями

Цей оператор SELECT отримує загальну тривалість бронювань для кожного місяця та місяць бронювання з таблиць Booking, Check\_in, Room і Room\_type, згрупованих за поточним місяцем і впорядкованих за поточним місяцем у порядку зростання.

8.

```
SELECT r.room_number, rt.room_name, rt.bedroom_amount
FROM Room r
INNER JOIN Room_type rt ON r.room_type_id = rt.room_type_id
ORDER BY r.room_number ASC;
```



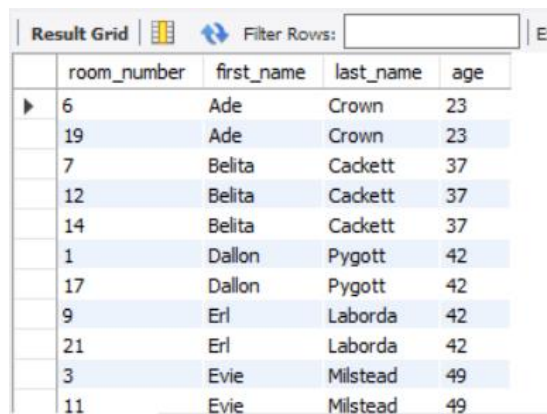
Result Grid		 Filter Rows:		Exp
	room_number	room_name	bedroom_amount	
▶	1	Normal	1	
	2	Half-Luxe	2	
	3	Luxe	3	
	4	Normal	1	
	5	Normal	1	
	6	Normal	1	
	7	Half-Luxe	2	
	8	Luxe	3	
	9	Normal	1	
	10	Half-Luxe	2	
	11	Normal	1	

Рисунок 7.2.8 - дані про кімнати

Цей оператор SELECT отримує номер кімнати, назву кімнати та кількість кімнат для кожної кімнати в таблиці Room.

9.

```
SELECT r.room_number, e.first_name, e.last_name, e.age
FROM Room r
INNER JOIN Employee e ON r.housekeeper_id = e.employee_id
ORDER BY e.first_name ASC;
```



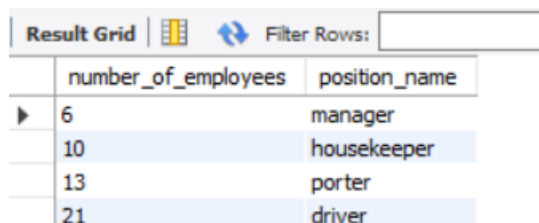
	room_number	first_name	last_name	age
▶	6	Ade	Crown	23
	19	Ade	Crown	23
	7	Belita	Cackett	37
	12	Belita	Cackett	37
	14	Belita	Cackett	37
	1	Dallon	Pygott	42
	17	Dallon	Pygott	42
	9	Erl	Laborda	42
	21	Erl	Laborda	42
	3	Evie	Milstead	49
	11	Evie	Milstead	49

Рисунок 7.2.9 - дані про прибиральниць для кожної кімнати

Цей оператор SQL вибере номер кімнати, ім'я, прізвище та вік кожної домробітниці в таблиці Room, відсортовані за зростанням імені господарки.

10.

```
SELECT COUNT(e.employee_id) AS number_of_employees, tof.position_name
FROM Employee e
INNER JOIN Type_of_employee tof ON tof.type_of_employee_id =
e.type_of_employee_id
GROUP BY tof.position_name;
```



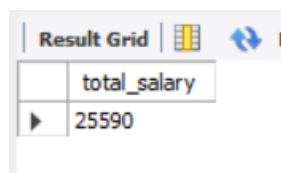
	number_of_employees	position_name
▶	6	manager
	10	housekeeper
	13	porter
	21	driver

Рисунок 7.2.10 - кількість працівників за професією

Цей оператор SQL отримує кількість працівників на кожній посаді та назву посади, згруповану за назвою посади. Результати сортуються за назвою позиції.

11.

```
SELECT SUM(e.working_hours_month * tof.wage) AS total_salary
FROM Employee e
INNER JOIN Type_of_employee tof ON tof.type_of_employee_id =
e.type_of_employee_id;
```



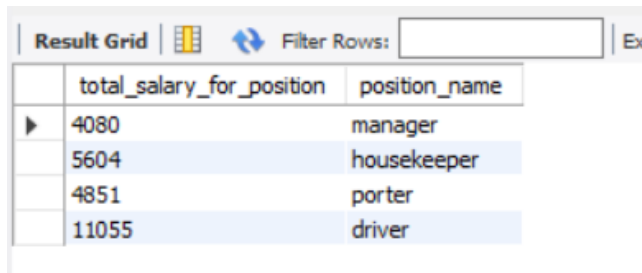
	total_salary
▶	25590

Рисунок 7.2.11 - загальна заробітня плата

Цей оператор SELECT повертає загальну зарплату всіх працівників шляхом підсумовування добутку робочих годин кожного працівника за місяць і їхньої заробітної плати, як зазначено в таблиці Type\_of\_employee.

12.

```
SELECT SUM(e.working_hours_month * tof.wage) AS total_salary_for_position,
tof.position_name
FROM Employee e
INNER JOIN Type_of_employee tof ON tof.type_of_employee_id =
e.type_of_employee_id
GROUP BY tof.position_name;
```



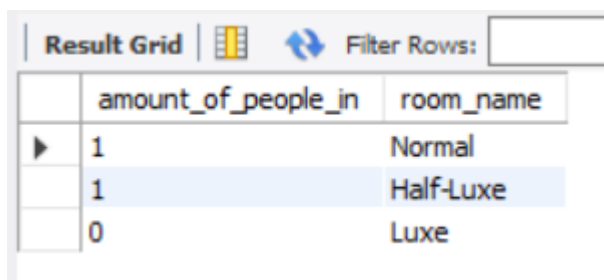
	total_salary_for_position	position_name
▶	4080	manager
	5604	housekeeper
	4851	porter
	11055	driver

Рисунок 7.2.12 - заробітня плата за професією

Цей запит обчислить загальну зарплату для кожної посади шляхом підсумовування добутку робочих годин за місяць і заробітної плати для кожного працівника на цій посаді та групування результату за назвою посади.

13.

```
SELECT SUM(ISNULL(co.check_out_date)) AS amount_of_people_in,
rt.room_name
FROM Booking b
INNER JOIN Check_out co ON b.checkout_date = co.check_out_id
INNER JOIN Room r ON b.room_id = r.room_id
INNER JOIN Room_type rt ON r.room_type_id = rt.room_type_id
GROUP BY rt.room_name;
```



	amount_of_people_in	room_name
▶	1	Normal
	1	Half-Luxe
	0	Luxe

Рисунок 7.2.13 - кількість гостей, які на даний момент знаходяться в готелі

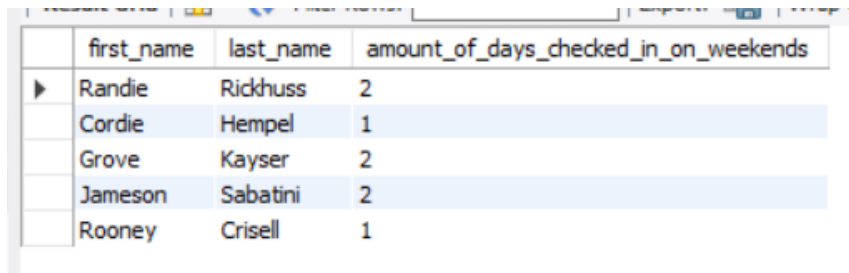
Цей запит знаходить кількість людей, які проживають в готелі та кількість згрупована за типами кімнат. Якщо гість готелю проживає в ньому, то поле check\_out\_date в таблиці Check\_out є null-ом.

14.

```

SELECT e.first_name, e.last_name, COUNT(DAYOFWEEK(ci.check_in_date))
AS amount_of_days_checked_in_on_weekends
FROM Check_in ci
INNER JOIN Employee e ON e.employee_id = ci.manager_id
WHERE DAYOFWEEK(ci.check_in_date) in (1, 7)
GROUP BY e.last_name;

```



	first_name	last_name	amount_of_days_checked_in_on_weekends
▶	Randie	Rickhuss	2
	Cordie	Hempel	1
	Grove	Kayser	2
	Jameson	Sabatini	2
	Rooney	Crisell	1

Рисунок 7.2.14 - кількість днів, кожен менеджер заселяв гостей на вихідних

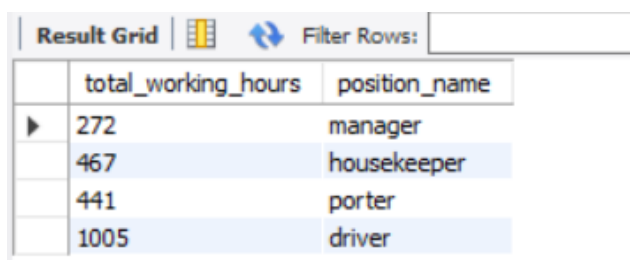
Цей запит повертає список менеджерів і кількість днів, коли вони зареєстрували клієнта у вихідний день (субота чи неділя).

15.

```

SELECT SUM(e.working_hours_month ) AS total_working_hours,
tof.position_name
FROM Employee e
INNER JOIN Type_of_employee tof ON tof.type_of_employee_id =
e.type_of_employee_id
GROUP BY tof.position_name;

```



	total_working_hours	position_name
▶	272	manager
	467	housekeeper
	441	porter
	1005	driver

Рисунок 7.2.15 - загальні робочі години за професіями

Цей оператор SELECT обчислює загальну кількість робочих годин на місяць для працівників на кожній посаді.

16.

```
SELECT b.booking_id, c.first_name, c.last_name, IF(ci.check_out_date <
co.check_out_date,          "overstayed",          "understayed")          AS
checked_out_in_the_specified_time

FROM Booking b

INNER JOIN Customer c ON b.customer_id = c.customer_id

INNER JOIN Check_in ci ON b.checkin_date = ci.check_in_id

INNER JOIN Check_out co ON b.checkout_date = co.check_out_id

WHERE          ci.check_out_date          !=          co.check_out_date;
```

	booking_id	first_name	last_name	checked_out_in_the_specified_time
▶	1	Thatcher	Cardis	overstayed
	2	Iormina	Loddy	understayed
	4	Tobie	O'Flannery	overstayed
	6	Leupold	Pittem	understayed
	9	Clifford	Brameld	overstayed
	11	Rani	Gerlts	understayed
	12	Leopold	Watford	understayed
	15	Rani	Gerlts	overstayed

Рисунок 7.2.16 - чи перевищив запланований час перебування в готелі певний клієнт

Цей запит вибирає id бронювання, ім'я клієнта, прізвище клієнта та рядок, що вказує на те, чи перевищив клієнт запланований час перебування в готелі чи навпаки.

17.

```
SELECT COUNT(b.booking_id) AS amount_of_times, IF(ci.check_out_date <
co.check_out_date,          "overstayed",          "understayed")          AS
checked_out_in_the_specified_time

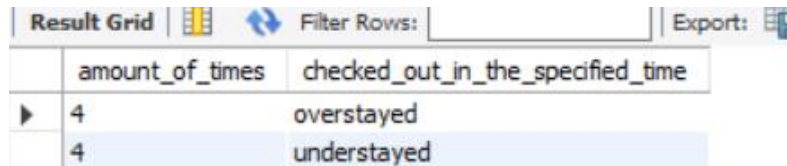
FROM Booking b

INNER JOIN Customer c ON b.customer_id = c.customer_id
```

```

INNER JOIN Check_in ci ON b.checkin_date = ci.check_in_id
INNER JOIN Check_out co ON b.checkout_date = co.check_out_id
WHERE ci.check_out_date != co.check_out_date
GROUP BY IF(ci.check_out_date < co.check_out_date, "overstayed",
"understayed");

```



	amount_of_times	checked_out_in_the_specified_time
▶	4	overstayed
	4	understayed

Рисунок 7.2.17 - кількість разів гість готелю затримувався в ньому або виїздив раніше

Цей запит підрахує кількість бронювань, у яких запланована дата виїзду та фактична дата виїзду не збігаються, а потім згрупує результат за тим, чи перевищив клієнт запланований час перебування в готелі чи навпаки.

18.

```

SELECT b.booking_id, c.first_name, c.last_name, c.passport_number,
       IF(ISNULL(co.check_out_date) = 0, (DATEDIFF(co.check_out_date,
ci.check_in_date) * rt.price_for_day), (b.duration * rt.price_for_day)) AS
predicted_price
FROM booking b
INNER JOIN customer c ON b.customer_id = c.customer_id
INNER JOIN Room r ON b.room_id = r.room_id
INNER JOIN Room_type rt ON r.room_type_id = rt.room_type_id
INNER JOIN Check_in ci ON b.checkin_date = ci.check_in_id
INNER JOIN Check_out co ON b.checkout_date = co.check_out_id
ORDER BY b.booking_id ASC;

```



	booking_id	first_name	last_name	passport_number	predicted_price
1	1	Thatcher	Cardis	70007473	23000
2	2	Iormina	Loddy	301706172	8000
3	3	Corinne	Formilli	62845215	225000
4	4	Tobie	O'Flannery	878786753	25000
5	5	Davida	Rikard	533077100	3000
6	6	Leupold	Pittem	630738534	6000
7	7	Anastasii	Backshall	227912121	44000
8	8	Weber	McAleese	42200101	18000
9	9	Clifford	Brameld	70988732	10000
10	10	Saleem	Chaudrelle	268562971	20000
11	11	Rani	Gerlts	643569274	5000

Рисунок 7.2.18 - запланована ціна за перебування

Цей оператор SELECT отримує такі дані для кожного бронювання: ідентифікатор бронювання, ім'я клієнта, прізвище клієнта, номер паспорта клієнта, прогнозована ціна бронювання.

Прогнозована ціна бронювання розраховується на основі кількості днів перебування клієнта в готелі. Якщо клієнт фізично виїхав з готелю, кількість днів розраховується як різниця між датою виїзду та датою заїзду. Якщо клієнт ще не виїхав, кількість днів береться з запланованої тривалості бронювання, яку клієнт вказує при заїзді.

19.

CREATE OR REPLACE VIEW employee\_view AS

```
SELECT      e.first_name,      e.last_name,      e.passport_number,
e.passport_series,  e.age,      e.gender,      e.working_hours_month,      tof.wage,
tof.position_name, (tof.wage * e.working_hours_month) AS salary
```

```
FROM Employee e
```

```
INNER JOIN Type_of_employee tof ON tof.type_of_employee_id =
e.type_of_employee_id
```

```
ORDER BY salary ASC;
```

```
SELECT * FROM employee_view;
```

	first_name	last_name	passport_number	passport_series	age	gender	working_hours_month	wage	position_name	salary
►	Lind	Hambelton	210799196	01mxyj1788	81	Male	10	11	porter	110
	Legra	Dulinty	727619470	29fxgn0238	32	Female	12	11	driver	132
	Rhianon	Minghetti	731256439	73krjq0975	26	Female	12	11	driver	132
	Benoite	Plitz	370858773	90xwka9006	87	Female	13	11	driver	143
	Kelsey	Marryatt	936409132	37hsjc5469	74	Male	15	11	porter	165
	Edgardo	Waldie	828205716	53wjnf0570	80	Male	17	11	porter	187
	Merill	Domke	374508750	38omjq8952	35	Male	17	11	porter	187
	Meredeth	Adicot	659536071	51twlm2140	32	Non-binary	18	11	porter	198
	Alan	Fane	211900364	44zypp0380	48	Non-binary	19	11	driver	209
	Morie	Piscopo	614448470	08kqsb8155	24	Non-binary	21	11	driver	231
	Izaak	Glusbv	162992343	37arps0445	58	Male	22	11	driver	242

Рисунок 7.2.19 - повні дані про працівника готелю

Цей запит створює представлення під назвою «employee\_view», яке відображає ім'я, прізвище, номер паспорта, серію паспорта, вік, стать, кількість робочих годин на місяць, заробітну плату, назву посади та оклад для кожного працівника в таблиці Employee.

20.

```
CREATE OR REPLACE VIEW customer_modification AS
```

```
    SELECT  CONCAT(cm.old_first_name, "/", c.first_name) AS
    OLD_NAME__NEW_NAME, CONCAT(cm.old_last_name,"/", c.last_name) AS
    OLD_SURNAME__NEW_SURNAME,
```

```
    CONCAT(cm.old_passport_number, "/", c.passport_number) AS
    OLD_NUMBER__NEW_NUMBER,  CONCAT(cm.old_passport_series, "/",
    c.passport_series) AS OLD_SERIES__NEW_SERIES,
```

```
    CONCAT(cm.old_age, "/", c.age) AS  OLD_AGE__NEW_AGE,
    CONCAT(cm.old_gender, "/", c.gender) AS OLD_GENDER__NEW_GENDER
```

```
    FROM Customer_modification_log cm
```

```
    INNER JOIN Customer c ON c.customer_id = cm.customer_id;
```

```
SELECT * FROM customer_modification;
```

	OLD_NAME__NEW_NAME	OLD_SURNAME__NEW_SURNAME	OLD_NUMBER__NEW_NUMBER	OLD_SERIES__NEW_SERIES	OLD_AGE__NEW_AGE	OLD_GENDER__NEW_GENDER
►	Burt/Anastasia	Backshall/Backshall	227912121/227912121	28akox8465/28akox8465	89/89	Male/Male

Рисунок 7.2.20 - порівняння змінених даних з поточними

Цей код створить представлення під назвою `customer_modification`, яке показує старі та нові значення для імені, прізвища, номера паспорта, серії паспорта, віку та статі кожного клієнта в таблиці `Customer`. У представленні буде показано старе значення для кожного атрибута, а потім нове значення для цього атрибута, розділене символом `/`.

### 7.3. Процедури та функції

1.

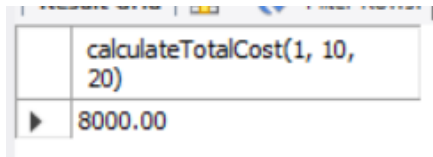
```
DROP FUNCTION IF EXISTS calculateTotalCost;

DELIMITER $$

CREATE FUNCTION calculateTotalCost(
    roomTypeId INT,
    duration INT,
    discountAmount INT
) RETURNS DECIMAL(10,2)
DETERMINISTIC
BEGIN
    DECLARE totalCost DECIMAL(10,2);
    SELECT price_for_day * duration INTO totalCost FROM Room_type
WHERE room_type_id = roomTypeId;
    SET totalCost = totalCost - (totalCost * (discountAmount / 100));
    RETURN totalCost;
END$$

DELIMITER ;

SELECT calculateTotalCost(1, 10, 20);
```



	calculateTotalCost(1, 10, 20)
▶	8000.00

Рисунок 7.3.1 - розрахунок ціни за перебування з певною знижкою

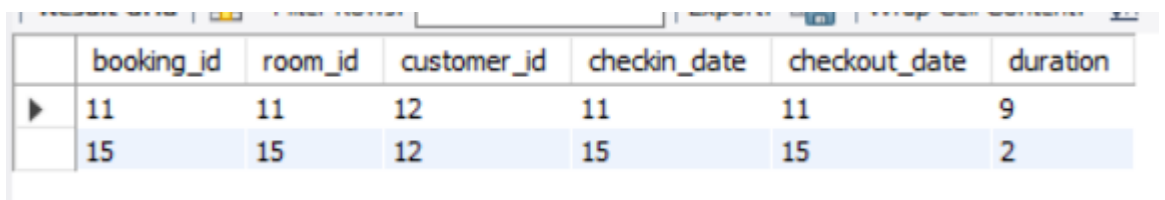
Ця функція має розрахувати загальну вартість бронювання, використовуючи такі параметри: roomTypeId: ідентифікатор типу номера, duration: тривалість перебування в днях, discountAmount: сума знижки у відсотках.

2.

```
DROP PROCEDURE IF EXISTS getCustomerBookings;
DELIMITER $$
CREATE PROCEDURE getCustomerBookings(IN customerId INT)
BEGIN
    SELECT * FROM Booking WHERE customer_id = customerId;
END$$
```

```
DELIMITER ;
```

```
CALL getCustomerBookings(12);
```



	booking_id	room_id	customer_id	checkin_date	checkout_date	duration
▶	11	11	12	11	11	9
	15	15	12	15	15	2

Рисунок 7.3.2 - вивід усіх бронювань певного клієнта

Ця процедура виводить усі бронювання певного гостя готелю за його ідентифікатором у базі даних.

3.

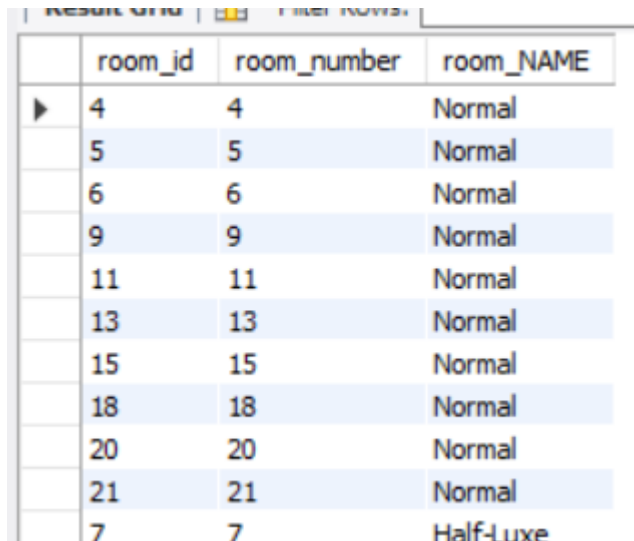
```
DROP PROCEDURE IF EXISTS getFreeRooms;

DELIMITER $$

CREATE PROCEDURE getFreeRooms(hotelId INT, startDate DATE,
endDate DATE)
BEGIN
    SELECT r.room_id, r.room_number, rt.room_NAME
    FROM Room r
    INNER JOIN Room_type rt ON r.room_type_id = rt.room_type_id
    WHERE r.hotel_id = hotelId AND r.room_id NOT IN (
        SELECT b.room_id FROM Booking b
        INNER JOIN Check_in ci ON b.checkin_date = ci.check_in_id
        INNER JOIN Check_out co ON b.checkout_date = co.check_out_id
        WHERE ci.check_in_date BETWEEN startDate AND endDate AND
        IF(ISNULL(co.check_out_date) = 0, co.check_out_date, ci.check_out_date)
        BETWEEN startDate AND endDate
    );
END$$

DELIMITER ;

CALL getFreeRooms(1, '2023-01-07', '2023-01-21');
```



	room_id	room_number	room_NAME
►	4	4	Normal
	5	5	Normal
	6	6	Normal
	9	9	Normal
	11	11	Normal
	13	13	Normal
	15	15	Normal
	18	18	Normal
	20	20	Normal
	21	21	Normal
	7	7	Half-Luxe

Рисунок 7.3.3 - список вільних номерів на певну дату

Ця функція отримує список вільних номерів у вказаному готелі між датою початку та закінчення. Ідентифікатор кімнати, номер кімнати та тип кімнати повертаються в наборі результатів.

4.

```

DROP PROCEDURE IF EXISTS createBooking;
DELIMITER $$
CREATE PROCEDURE createBooking(
    roomId INT,
    customerId INT,
    checkinDate INT,
    checkoutDate INT
)
BEGIN
    INSERT INTO Booking (room_id, customer_id, checkin_date,
        checkout_date, duration) VALUES (roomId, customerId, checkinDate,
        checkoutDate, 0);

```

```

UPDATE booking b
INNER JOIN check_in c on b.checkin_date = c.check_in_id
SET duration = DATEDIFF(c.check_out_date, c.check_in_date) WHERE
(b.booking_id > 0);
END$$

```

DELIMITER ;

```

83 • CALL createCheckIn_CheckOut("2023-01-07", "2023-01-21");
84 • SELECT @checkinId, @checkoutId;
85 • CALL createBooking(3, 37, @checkinId, @checkoutId);

```

booking_id	room_id	customer_id	checkin_date	checkout_date	duration
16	2	37	16	16	14
17	2	5	17	17	6
18	1	5	17	17	6
19	3	37	18	18	14
NULL	NULL	NULL	NULL	NULL	NULL

Рисунок 7.3.4 - створення бронювання

5.

```

DROP PROCEDURE IF EXISTS createCheckIn_CheckOut;
DELIMITER $$
CREATE PROCEDURE createCheckIn_CheckOut(
    checkinDate DATE,
    checkoutDate DATE
)
BEGIN
    INSERT INTO Check_in (check_in_date, check_out_date,
manager_id) VALUES (checkinDate, checkoutDate, (SELECT t1.employee_id
FROM employee AS t1 JOIN (SELECT employee_id FROM employee where
type_of_employee_id = 1 ORDER BY RAND() LIMIT 1) as t2 ON
t1.employee_id=t2.employee_id));

```

```

SET @checkinId = LAST_INSERT_ID();
INSERT INTO Check_out (check_out_date, manager_id) VALUES (null,
null);
SET @checkoutId = LAST_INSERT_ID();
END$$
DELIMITER ;

```

	check_in_id	check_in_date	check_out_date	manager_id
	15	2022-01-18	2022-01-20	2
	16	2023-01-07	2023-01-21	2
	17	2022-02-07	2022-02-13	6
▶	18	2023-01-07	2023-01-21	45
*	NULL	NULL	NULL	NULL

Рисунок 7.3.5 - створення запису про заселення

	check_out_id	check_out_date	manager_id
	15	2022-01-24	33
	16	NULL	NULL
	17	2022-02-13	6
▶	18	NULL	NULL
*	NULL	NULL	NULL

Рисунок 7.3.6 - створення запису про виселення

Перша процедура, createBooking, створює новий запис у таблиці Booking. Вона також розраховує тривалість бронювання та оновлює тривалість у тому самому записі. Друга процедура, createCheckIn\_CheckOut, створює новий запис у таблицях Check\_in і Check\_out. У таблиці Check\_in він також встановлює для стовпця manager\_id значення випадкового Employee\_ID із type\_of\_employee\_id, рівним 1(manager). У таблиці Check\_out для стовпців check\_out\_date та manager\_id встановлюється значення null. Він також зберігає check\_in\_id і check\_out\_id вставлених записів у змінних @checkinId і @checkoutId відповідно. Ці дві процедури працюють в тандемі. Тобто спочатку ми створюємо певні записи в таблицях Check\_in, Check\_out і далі користуємося ними для створення бронювання. Розбиття на дві



процедури було створене, щоб кожен раз коли тригер на додання бронювання спрацьовував ми не додавали повторювані записи в Check\_in та Check\_out.

6.

```

DROP PROCEDURE IF EXISTS update_room_price;

DELIMITER //

CREATE PROCEDURE update_room_price(IN p_room_type_id INT, IN
p_new_price DECIMAL(10,2))
BEGIN
    DECLARE room_type_name VARCHAR(20);

    SELECT room_name INTO room_type_name
    FROM Room_type
    WHERE room_type_id = p_room_type_id;

    UPDATE Room_type
    SET price_for_day = p_new_price
    WHERE room_type_id = p_room_type_id;
END;

//

DELIMITER ;

```

	room_type_id	room_name	bedroom_amount	price_for_day
	1	Normal	1	1000
	2	Half-Luxe	2	2000
▶	3	Luxe	3	3000
✱	NULL	NULL	NULL	NULL

Рисунок 7.3.7 - поточна ціна за номер

```

111 • call update_room_price(3, 3500);
112 • select * from room_type;
113

```

	room_type_id	room_name	bedroom_amount	price_for_day
▶	1	Normal	1	1000
	2	Half-Luxe	2	2000
	3	Luxe	3	3500

Рисунок 7.3.8 - змінена ціна

#### 7.4. Приклад роботи індексів

Індекси використовуються для підвищення швидкості операцій пошуку даних у базі даних. Вони роблять це, надаючи альтернативний спосіб пошуку даних у таблиці замість сканування всієї таблиці. Це особливо корисно для таблиць із великою кількістю рядків, оскільки це може значно скоротити кількість часу, необхідного для пошуку в таблиці. Індекси також можуть допомогти забезпечити унікальність даних у таблиці та використовувати їх для встановлення обмежень на дані. Загалом, індекси можуть значно підвищити продуктивність бази даних і зробити її ефективнішою.

Давайте перевіримо на практиці як працюють індекси. Виконаємо пошук запису в таблиці без та з використанням індексу.

Початковий запит:

```

SELECT DISTINCT customer_id, last_name
FROM customer
WHERE first_name = 'Clifford';

```

Створення індексу на поле first\_name:

```

CREATE INDEX first_name ON customer(first_name);

```

За допомогою EXPLAIN ANALYZE бачимо таку статистику:

Перед створенням індексу:

EXPLAIN
-> Filter: (customer.first_name = 'Clifford') (cost=5.25 rows=5) (actual time=0.464..0.485 rows=1 loops=1) -> Table scan on customer (cost=5.25 rows=50) (actual time=0.447..0.471 rows=50 loops=1)

Рисунок 7.4.1 - результат EXPLAIN на запиті до використання індексів

Після:

EXPLAIN
-> Index lookup on customer using first_name (first_name='Clifford') (cost=0.35 rows=1) (actual time=0.037..0.040 rows=1 loops=1)

Рисунок 7.4.2 - результат EXPLAIN на запиті після використання індексів

Бачимо, що запит виконався в рази швидше і було переглянуто менше рядків записів.

З даних результатів очевидно, що використання індексів – це невід’ємна частина роботи з базою даних.

## Висновок

Для створення бази даних для готельного комплексу було зроблено декілька кроків. По-перше, були створені необхідні таблиці для зберігання інформації про готель, такої як номери, співробітники, клієнти та бронювання. Первинний і зовнішній ключі були визначені для забезпечення цілісності даних і встановлення зв'язків між таблицями.

Далі були створені представлення даних, щоб представити дані певним чином і полегшити запити. Також було створено понад 20 DML-запитів за допомогою яких можна дивитися деякі статистичні дані пов'язані з розробленою базою даних. Нарешті, збережені процедури та функції були реалізовані для виконання конкретних завдань, таких як обчислення загальної вартості бронювання або повернення списку вільних номерів протягом певного періоду часу. Загалом, створення бази даних готельного комплексу вимагало ретельного планування та використання команд SQL для структурування та обробки даних.

### Посилання на джерела

- 1) Довідник базових функцій MySQL [Електронний ресурс] – Режим доступу до ресурсу [W3Schools](#)
- 2) Документація MySQL [Електронний ресурс] – Режим доступу до ресурсу [Documentation](#)
- 3) MySQL Workbench [Електронний ресурс] – Режим доступу до ресурсу [Workbench](#)
- 4) Mockroo. Генерація даних [Електронний ресурс] – Режим доступу до ресурсу [Mockaroo](#)