

Endterm Project

Hotel

Belgibekov Daniyar- 210103372

Dossanov Nurbek- 210103158

Orzhanov Nauryzbek-210103169

Database Management 2

Introduction

The database project for the hotel is an important component of commerce, which provides storage of information about customer numbers, booking, payment. This allows you to manage all aspects of the hotel's activities.

An ER diagram is a graphical representation of objects and the relationships between them. In the project, the ER diagram allowed us to understand what data needs to be stored and how they relate to each other. This made it possible to determine the correct database structure and correctly distribute information across tables.

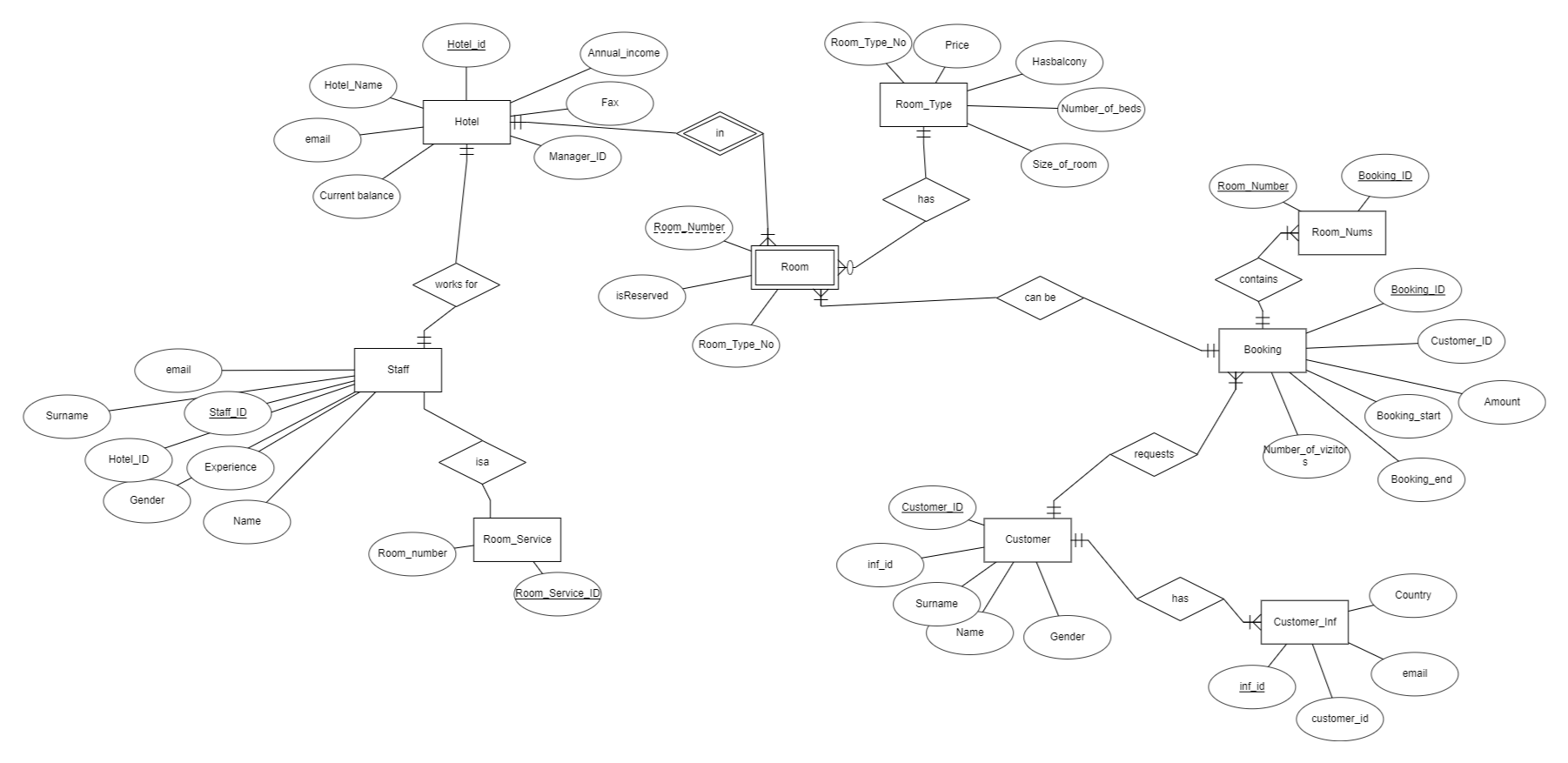
Table normalization is a database optimization process that avoids redundant and contradictory information. In the project, we used three forms of normalization, which allowed us to reduce the size of the database and improve its performance.

Triggers are software modules that are automatically triggered by certain user actions. In the project, we used triggers to automatically update information in the database with certain actions of the client or administrator, which increased the usability of the system.

As a result of our work, we have created a database that allows us to efficiently store, process and manage hotel data. This allows you to reduce the time and cost of processing reservations, as well as improve the quality of customer service.

Overview

ER diagram



Relationships:

1) Hotel has room  
One to many

2) Staff works for hotel

One to one

3) Staff have room\_service  
One to many

4) Room have room\_type  
Many to one

5) Room can be Booked

Many to one

6) customer requests booking

One to many

7) Customer have customer\_inf

One to many

Normalization

Table Hotel:

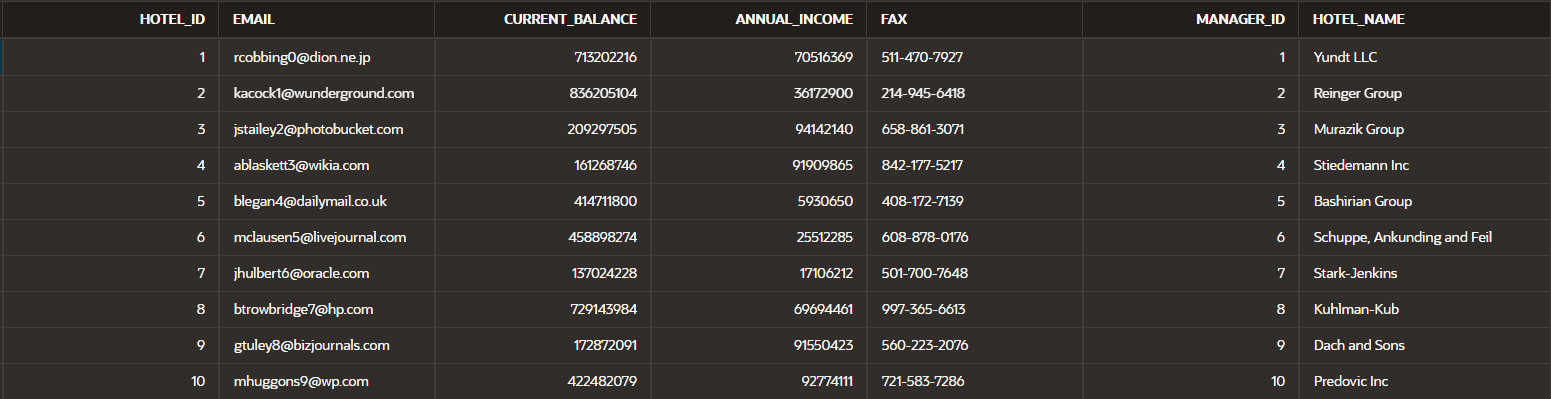


Table Staff:

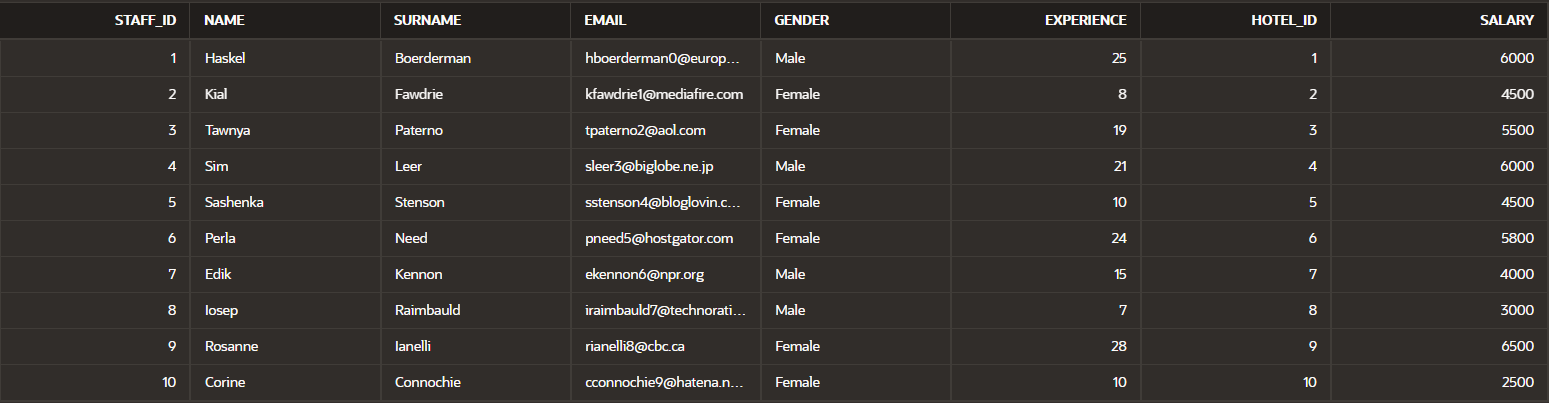


Table Room:

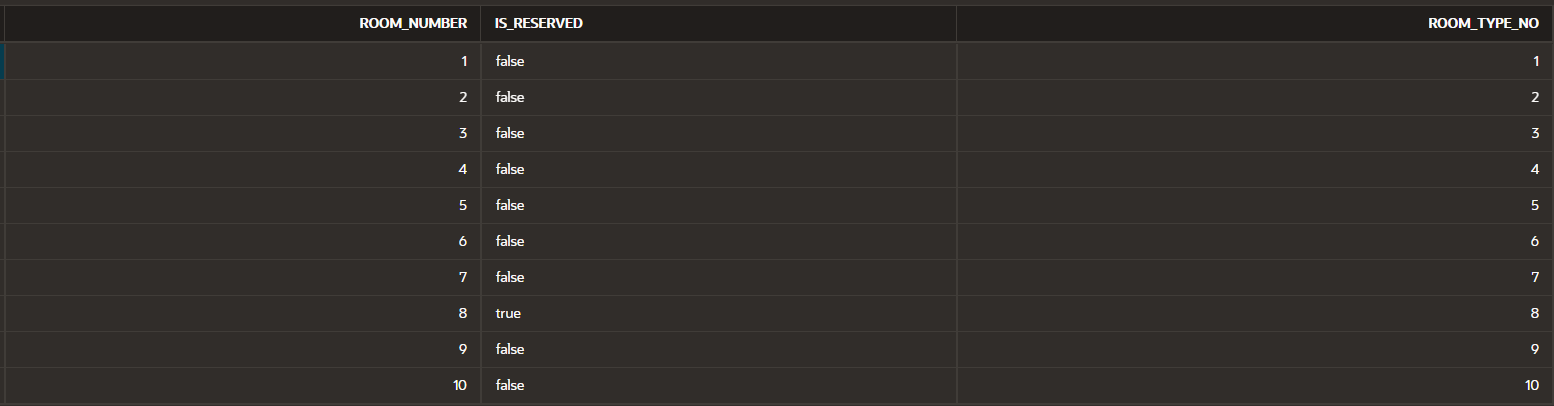


Table Room\_type:

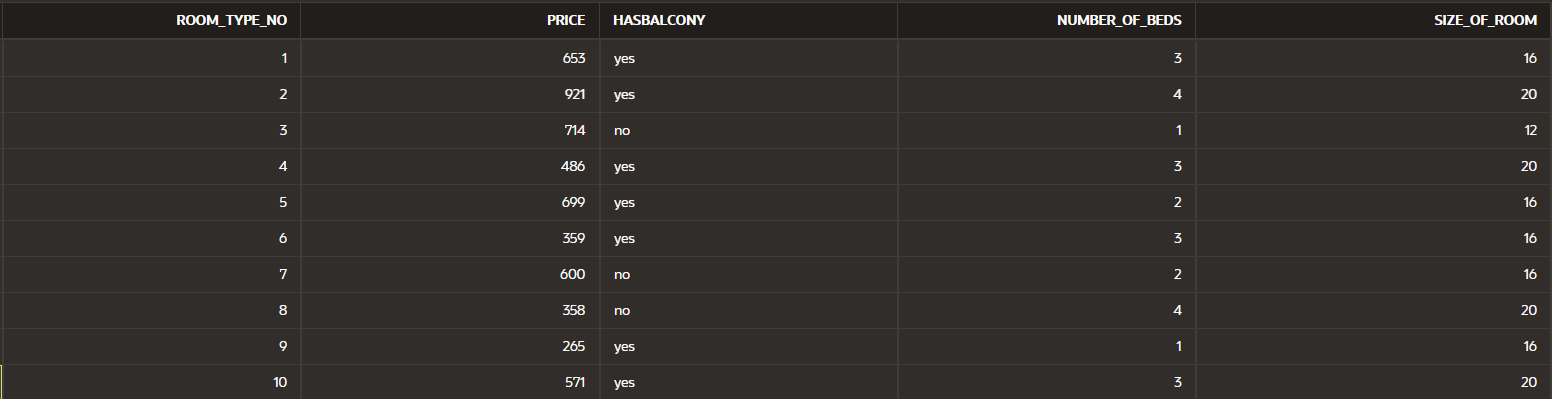


Table Customer:

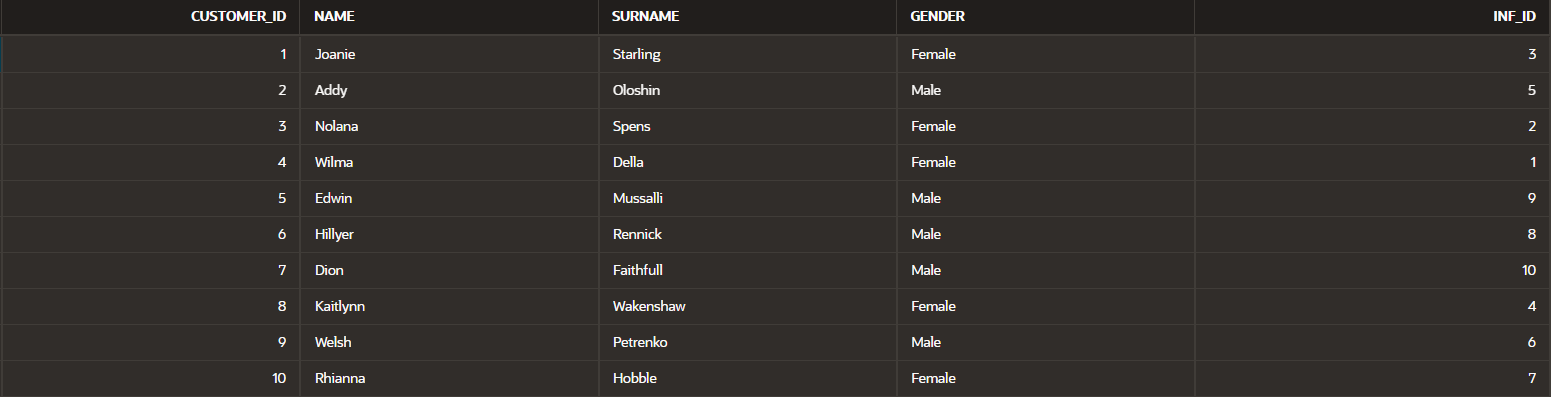


Table Booking:

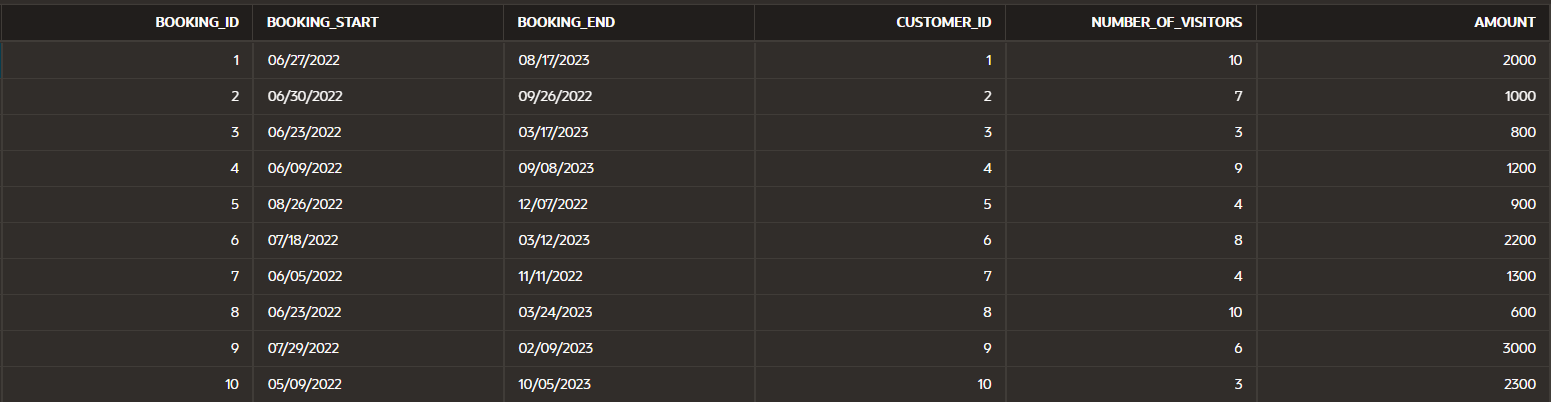


Table Room\_nums:

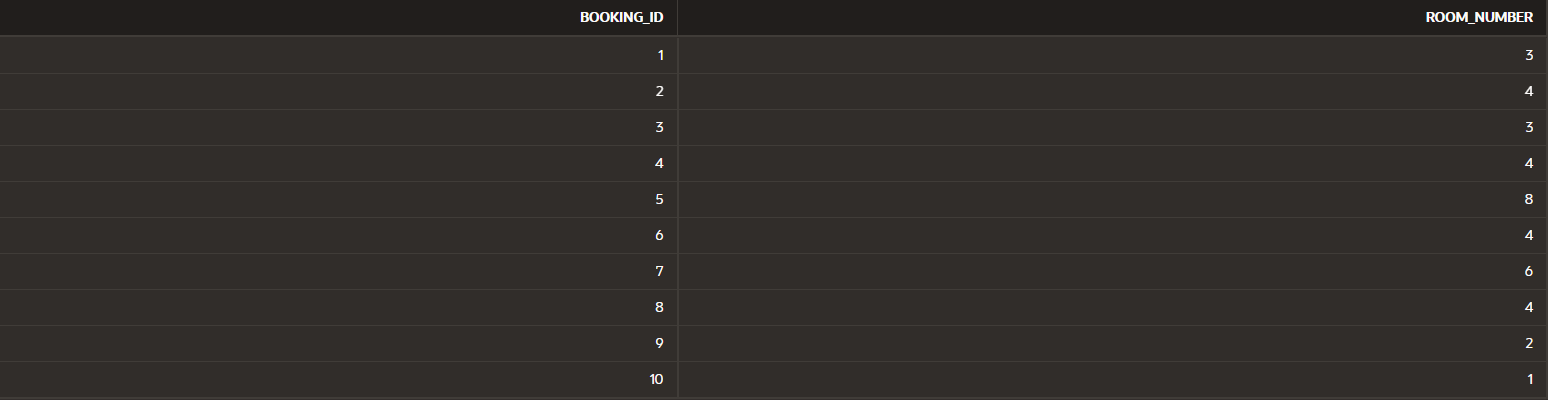


Table Customer\_inf:



Table Hotel.

**Hotel\_id** (primary key) -> {email,current\_balance,annual\_income, fax, manager\_id, hotel\_name}

Table Staff.

**Staff\_id** (primary key) -> {name,surname,emil,gender,experience,hotel\_id,salary}

Table Room**.**

**Room\_number**(primary key) -> {is\_reseved, room\_type\_no}

Table Room\_type.

**Room\_type\_no** (primary key) -> {price,hasbalcony, number\_of\_beds, size\_of\_room}

Table Customer.

**Customer\_id** (primary key) -> { inf\_id, name, gender, surname}

Table Booking.

**Booking\_id**(primary\_key) -> {customer\_id,amount,booking\_start,booking\_end, number\_of\_visitors}

Table Customer\_inf.

**Inf\_id**(primary\_key) -> {country, email, customer\_id}

**Functions & Triggers**

**1) Procedure which does group by information**

**CREATE OR REPLACE PROCEDURE group\_inform AS**

**CURSOR select\_sal IS**

**SELECT salary, COUNT(\*) AS emp\_count**

**FROM staff**

**GROUP BY salary;**

**BEGIN**

**FOR i IN select\_sal LOOP**

**DBMS\_OUTPUT.PUT\_LINE(i.salary i.emp\_count );**

**END LOOP;**

**END;**

**BEGIN**

**group\_inform;**

**END;**

**2) Function which counts the number of records**

**CREATE OR REPLACE FUNCTION count\_records(table\_name VARCHAR2) RETURN NUMBER IS**

**count\_num NUMBER;**

**BEGIN**

**EXECUTE IMMEDIATE 'SELECT COUNT(\*) FROM ' table\_name INTO count\_num;**

**RETURN count\_num;**

**END;**

**DECLARE**

**counter NUMBER;**

**BEGIN**

**counter := count\_records('hotel');**

**DBMS\_OUTPUT.PUT\_LINE(counter);**

**END;**

**3) Procedure which uses SQL%ROWCOUNT to determine the number of rows affected**

**CREATE OR REPLACE PROCEDURE update\_employee\_salary IS**

**BEGIN**

**UPDATE hotel**

**SET annual\_income = annual\_income + 5000**

**WHERE hotel\_id > 3;**

**IF SQL%ROWCOUNT > 0 THEN**

**DBMS\_OUTPUT.PUT\_LINE(SQL%ROWCOUNT);**

**END IF;**

**END;**

**select \* from hotel;**

**begin**

**update\_employee\_salary;**

**end;**

**4) Add user-defined exception which disallows to enter title of item (e.g. book) to be less than 5 characters**

**DECLARE**

**hotel\_name VARCHAR2(50) := 'Rixos';**

**error\_length EXCEPTION;**

**BEGIN**

**IF LENGTH(hotel\_name) < 5 THEN**

**RAISE error\_length;**

**ELSE**

**DBMS\_OUTPUT.PUT\_LINE('Вот енды норм');**

**END IF;**

**EXCEPTION**

**WHEN error\_length THEN**

**DBMS\_OUTPUT.PUT\_LINE('Брат бестен аз болып калды');**

**END;**

**5) Create a trigger before insert on any entity which will show the current number of rows in the table**

**CREATE OR REPLACE TRIGGER count\_r BEFORE INSERT ON hotel**

**DECLARE row\_count NUMBER;**

**BEGIN**

**SELECT COUNT(\*) INTO row\_count FROM hotel; DBMS\_OUTPUT.PUT\_LINE('Саны ' || row\_count);**

**END;**

**insert into hotel(hotel\_id, email, current\_balance, annual\_income, fax, manager\_id, hotel\_name) values(12, 'asfasf@gmaol.com', 2345,13245,222-111-2244, 12, 'dadae');**

Conclusion

As a result of the implementation of the database project for the hotel, our team has developed a reliable and efficient data management system that allows you to store, process and manage information about booking and rooms.

The ER diagram allowed us to understand what data needs to be stored and how they relate to each other. This helped us to determine the correct database structure and correctly distribute the information across the tables.

Normalization of tables helped to optimize the database and improve its performance. Triggers provide automatic updating of information in the database at certain actions of the client or administrator, which increases the usability of the system.

The database developed by our team allows you to effectively manage hotel data, reducing the time and cost of booking processing and improving the quality of customer service. We hope that our work will be useful and successful for the development of our client's business.

GitHub(https://github.com/Nurbek004/DBMS\_project)