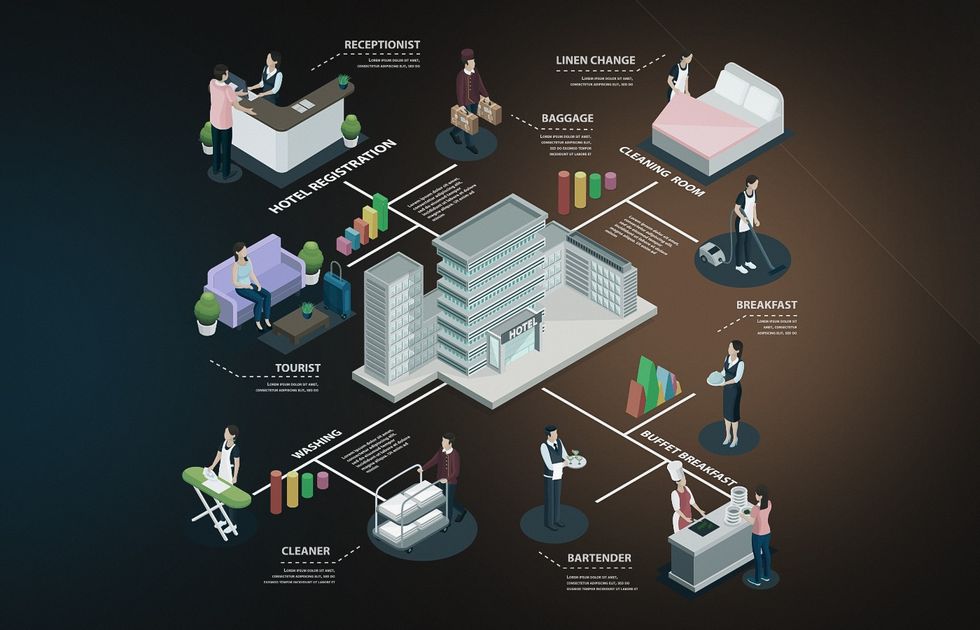
# Case study #-Hostel Mnagement System

**Introduction:**

In today's fast-paced urban environment, the demand for efficient and effective housing solutions has led to a rise in the popularity of paying guest (PG) accommodations. These establishments cater primarily to students and young professionals seeking convenient living arrangements that offer flexibility and affordability. Managing a PG facility, however, comes with its own set of challenges, from tenant management to billing and maintenance tracking.

This case study explores the development of a comprehensive management system using MySQL, aimed at streamlining operations for PG owners and enhancing the experience for residents. By leveraging a robust database management system, the proposed solution addresses key functionalities such as tenant registration, room allocation, payment processing, and maintenance requests.



The implementation of this PG management system not only simplifies administrative tasks but also provides valuable insights through data analytics, helping owners make informed decisions. Through this case study, we will outline the system's architecture, key features, and the benefits of adopting such technology in the PG sector.

**Problem Statement:**

Specifically, the PG(Paid Guest) administration struggles with:

1. **Room Allocation**:

The current process for assigning rooms is cumbersome and prone to errors, leading to double bookings and dissatisfaction among students.

1. **Billing and Payments**:

Inaccurate billing systems cause confusion and disputes regarding payment deadlines, fees, and outstanding balances.

1. **Maintenance Requests**:

The manual handling of maintenance requests results in delays in service and inadequate tracking of issues reported by residents.

1. **Data Management**:

A lack of a centralized database makes it difficult to access and manage guest records, leading to inefficient operations and reporting challenges.

**Solutions Statement:**

1. **Centralized Tenant Management**:

The system will provide a centralized database to securely store and manage tenant information, including personal details, lease agreements, and move-in/move-out records. This will streamline tenant onboarding and minimize administrative errors.

1. **Automated Room Allocation:**

An automated room allocation feature will ensure efficient management of available rooms, preventing double bookings and improving communication regarding occupancy status.

1. **Integrated Payment Processing:**

The system will offer a user-friendly payment processing module that allows tenants to make rent payments online, track payment history, and receive automated reminders for due dates. This will enhance financial transparency and reduce late payments.

1. **Maintenance Management System:**

A dedicated module for maintenance requests will enable tenants to easily report issues and track the status of their requests. PG owners will benefit from a systematic approach to managing maintenance tasks, ensuring timely resolutions.

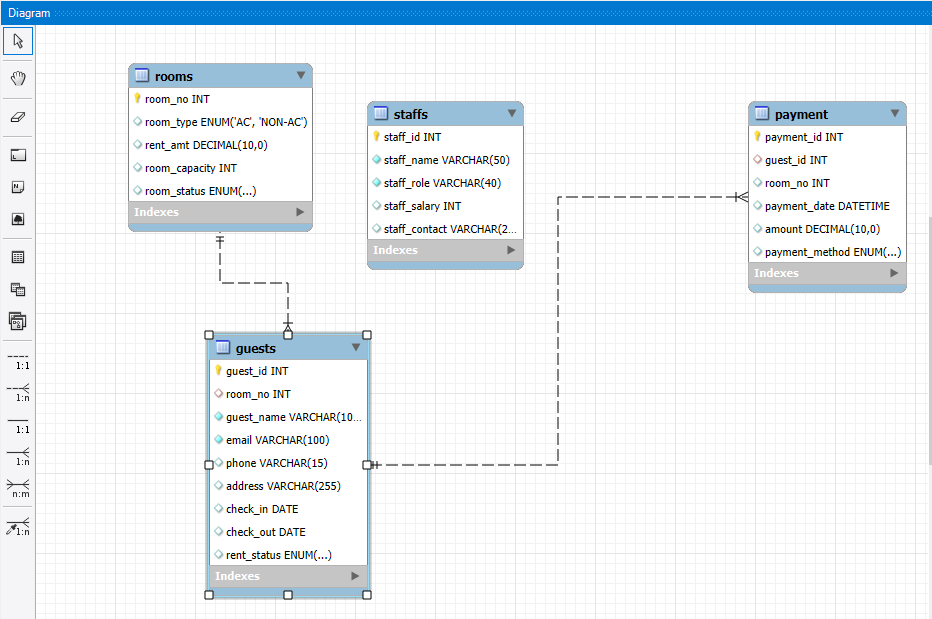
1. **Data Analytics and Reporting:**

The system will incorporate data analytics tools to provide insights into occupancy rates, payment patterns, and maintenance trends. This will empower PG owners to make informed decisions and optimize resource allocation.

1. **Comprehensive Record-Keeping:**

By maintaining accurate and accessible records of tenant history, payments, and maintenance logs, the system will help ensure compliance and improve operational efficiency.

**Entity-Relationship Diagram:**



Data Set:

CREATE DATABASE PG\_Management;

USE PG\_Management;

-----------------------------------------------------Guests Table---------------------------------------------------------

CREATE TABLE Guests (

guest\_id INT AUTO\_INCREMENT PRIMARY KEY,

room\_no int , /\*foreign key for room table\*/

guest\_name VARCHAR(100) NOT NULL,

email VARCHAR(100) NOT NULL,

phone VARCHAR(15),

address VARCHAR(255),

check\_in date,

check\_out date,

rent\_status ENUM('paid','unpaid'));

INSERT INTO Guests (room\_no, guest\_name, email, phone, address, check\_in, check\_out, rent\_status) VALUES

(500, 'Alice Johnson', 'alice@example.com', '1234567890', '123 Elm St', '2024-10-01', '2024-10-05', 'paid'),

(501, 'Bob Smith', 'bob@example.com', '1234567891', '124 Elm St', '2024-10-02', '2024-10-06', 'unpaid'),

(502, 'Charlie Brown', 'charlie@example.com', '1234567892', '125 Elm St', '2024-10-03', '2024-10-07', 'paid'),

(503, 'Daisy Ridley', 'daisy@example.com', '1234567893', '126 Elm St', '2024-10-04', '2024-10-08', 'unpaid'),

(504, 'Edward Norton', 'edward@example.com', '1234567894', '127 Elm St', '2024-10-05', '2024-10-09', 'paid'),

(505, 'Fiona Gallagher', 'fiona@example.com', '1234567895', '128 Elm St', '2024-10-06', '2024-10-10', 'unpaid'),

(506, 'George Martin', 'george@example.com', '1234567896', '129 Elm St', '2024-10-07', '2024-10-11', 'paid'),

(507, 'Hannah Montana', 'hannah@example.com', '1234567897', '130 Elm St', '2024-10-08', '2024-10-12', 'unpaid'),

(508, 'Ian Curtis', 'ian@example.com', '1234567898', '131 Elm St', '2024-10-09', '2024-10-13', 'paid'),

(509, 'Jessica Jones', 'jessica@example.com', '1234567899', '132 Elm St', '2024-10-10', '2024-10-14', 'unpaid'),

(510, 'Kevin Spacey', 'kevin@example.com', '1234567800', '133 Elm St', '2024-10-11', '2024-10-15', 'paid'),

(511, 'Liam Neeson', 'liam@example.com', '1234567801', '134 Elm St', '2024-10-12', '2024-10-16', 'unpaid'),

(512, 'Mona Lisa', 'mona@example.com', '1234567802', '135 Elm St', '2024-10-13', '2024-10-17', 'paid'),

(513, 'Nora Fatehi', 'nora@example.com', '1234567803', '136 Elm St', '2024-10-14', '2024-10-18', 'unpaid'),

(514, 'Oscar Wilde', 'oscar@example.com', '1234567804', '137 Elm St', '2024-10-15', '2024-10-19', 'paid'),

(515, 'Peter Parker', 'peter@example.com', '1234567805', '138 Elm St', '2024-10-16', '2024-10-20', 'unpaid'),

(516, 'Quinn Fabray', 'quinn@example.com', '1234567806', '139 Elm St', '2024-10-17', '2024-10-21', 'paid'),

(517, 'Rachel Green', 'rachel@example.com', '1234567807', '140 Elm St', '2024-10-18', '2024-10-22', 'unpaid'),

(518, 'Steve Jobs', 'steve@example.com', '1234567808', '141 Elm St', '2024-10-19', '2024-10-23', 'paid'),

(519, 'Tina Fey', 'tina@example.com', '1234567809', '142 Elm St', '2024-10-20', '2024-10-24', 'unpaid'),

(520, 'Uma Thurman', 'uma@example.com', '1234567810', '143 Elm St', '2024-10-21', '2024-10-25', 'paid'),

(521, 'Victor Hugo', 'victor@example.com', '1234567811', '144 Elm St', '2024-10-22', '2024-10-26', 'unpaid'),

(522, 'Will Smith', 'will@example.com', '1234567812', '145 Elm St', '2024-10-23', '2024-10-27', 'paid'));

--------------------------------------------------------Rooms Table----------------------------------------------------------

CREATE TABLE Rooms (

room\_no INT AUTO\_INCREMENT PRIMARY KEY,

room\_type enum('AC','NON-AC'),

rent\_amt decimal,

room\_capacity int,

room\_status enum('booked','unbooked','balance avail'));

ALTER TABLE rooms AUTO\_INCREMENT=500;

INSERT INTO Rooms (room\_type, rent\_amt, room\_capacity, room\_status) VALUES

('AC', 1500.00, 2, 'unbooked'),

('NON-AC', 1000.00, 2, 'unbooked'),

('AC', 2000.00, 3, 'booked'),

('NON-AC', 1200.00, 2, 'balance avail'),

('AC', 1800.00, 4, 'unbooked'),

('NON-AC', 800.00, 2, 'unbooked'),

('AC', 2200.00, 3, 'booked'),

('NON-AC', 900.00, 2, 'unbooked'),

('AC', 1600.00, 2, 'balance avail'),

('NON-AC', 1100.00, 2, 'unbooked'),

('AC', 1750.00, 2, 'booked'),

('NON-AC', 950.00, 2, 'balance avail'),

('AC', 2000.00, 3, 'unbooked'),

('NON-AC', 1300.00, 4, 'booked'),

('AC', 1900.00, 3, 'unbooked'),

('NON-AC', 1000.00, 2, 'unbooked'),

('AC', 2100.00, 3, 'balance avail'),

('NON-AC', 1150.00, 2, 'unbooked'),

('AC', 1650.00, 2, 'unbooked'),

('NON-AC', 1050.00, 2, 'booked'),

('AC', 1750.00, 4, 'balance avail'),

('NON-AC', 950.00, 2, 'unbooked'),

('AC', 1850.00, 2, 'unbooked'),

('NON-AC', 1200.00, 3, 'balance avail'),

('AC', 2200.00, 4, 'booked'),

('NON-AC', 970.00, 2, 'unbooked'),

('AC', 1500.00, 3, 'balance avail'),

('NON-AC', 900.00, 2, 'unbooked'),

('AC', 2000.00, 4, 'unbooked'),

('NON-AC', 1150.00, 2, 'booked'),

('AC', 1800.00, 3, 'balance avail'),

('NON-AC', 1250.00, 3, 'unbooked'),

('AC', 1700.00, 2, 'unbooked'),

('NON-AC', 1100.00, 2, 'unbooked'),

('AC', 1950.00, 3, 'booked'),

('NON-AC', 930.00, 2, 'unbooked'),

('AC', 1600.00, 2, 'balance avail'));

---------------------------------------------------------payment Table-------------------------------------------------------

CREATE TABLE payment(

payment\_ID INT PRIMARY KEY AUTO\_INCREMENT,

guest\_id int,

room\_no int,

payment\_date datetime,

amount decimal,

payment\_method ENUM('cash','GPAY','others'),

FOREIGN KEY(guest\_id) REFERENCES GUESTS(guest\_id),

FOREIGN KEY(room\_no) REFERENCES ROOMS(room\_no));

INSERT INTO Payment (guest\_id, room\_no, payment\_date, amount, payment\_method) VALUES

(1051, 500, '2024-10-01 12:00:00', 1800.00, 'cash'),

(1052, 501, '2024-10-02 12:30:00', 1200.00, 'GPAY'),

(1053, 502, '2024-10-03 13:00:00', 2200.00, 'cash'),

(1054, 503, '2024-10-04 14:00:00', 1000.00, 'others'),

(1055, 504, '2024-10-05 15:00:00', 2000.00, 'cash'),

(1056, 505, '2024-10-06 16:00:00', 1500.00, 'GPAY'),

(1057, 506, '2024-10-07 17:00:00', 2500.00, 'cash'),

(1058, 507, '2024-10-08 18:00:00', 1800.00, 'others'),

(1059, 508, '2024-10-09 19:00:00', 1200.00, 'GPAY'),

(1060, 509, '2024-10-10 20:00:00', 2200.00, 'cash'),

(1061, 510, '2024-10-11 21:00:00', 1300.00, 'others'),

(1062, 511, '2024-10-12 22:00:00', 2000.00, 'GPAY'),

(1063, 512, '2024-10-13 23:00:00', 1400.00, 'cash'),

(1064, 513, '2024-10-14 12:00:00', 2500.00, 'others'),

(1065, 514, '2024-10-15 12:30:00', 1700.00, 'GPAY'),

(1066, 515, '2024-10-16 13:00:00', 1600.00, 'cash'),

(1067, 516, '2024-10-17 14:00:00', 1800.00, 'others'),

(1068, 517, '2024-10-18 15:00:00', 1900.00, 'GPAY'),

(1069, 518, '2024-10-19 16:00:00', 2100.00, 'cash'),

(1070, 519, '2024-10-20 17:00:00', 1250.00, 'others'),

(1071, 520, '2024-10-21 18:00:00', 1750.00, 'GPAY'),

(1072, 521, '2024-10-22 19:00:00', 1300.00, 'cash'),

(1073, 522, '2024-10-23 20:00:00', 1450.00, 'others'),

(1074, 523, '2024-10-24 21:00:00', 1550.00, 'GPAY'));

------------------------------------------------------------Staffs Table------------------------------------------------------------

CREATE TABLE staffs(

staff\_id int AUTO\_INCREMENT PRIMARY KEY,

staff\_name varchar(50) NOT NULL,

staff\_role varchar(40) NOT NULL,

staff\_salary int DEFAULT 15000,

staff\_contact varchar(25));

INSERT INTO Staffs (staff\_name, staff\_role, staff\_salary, staff\_contact) VALUES

('John Doe', 'Manager', 30000, 9876543210),

('Jane Smith', 'Receptionist', 20000, 9876543211),

('Mike Johnson', 'Housekeeping', 15000, 9876543212),

('Emily Davis', 'Chef', 25000, 9876543213),

('Chris Lee', 'Security', 18000, 9876543214),

('Patricia Brown', 'Front Desk', 22000, 9876543215),

('Robert Wilson', 'Maintenance', 19000, 9876543216),

('Linda Taylor', 'Accountant', 30000, 9876543217),

('James Anderson', 'Bellboy', 12000, 9876543218),

('Barbara Thomas', 'Waitress', 15000, 9876543219),

('William Jackson', 'Concierge', 25000, 9876543220),

('Elizabeth White', 'Housekeeper', 14000, 9876543221),

('David Harris', 'Gardener', 13000, 9876543222),

('Jennifer Martin', 'Barista', 16000, 9876543223),

('Charles Thompson', 'Pool Attendant', 14000, 9876543224),

('Matthew Garcia', 'IT Support', 22000, 9876543225),

('Joseph Martinez', 'Marketing', 27000, 9876543226),

('Nancy Robinson', 'Events Coordinator', 23000, 9876543227),

('Karen Clark', 'Sales', 24000, 9876543228),

('Thomas Rodriguez', 'Bartender', 15000, 9876543229),

('Barbara Lewis', 'Laundry Staff', 13000, 9876543230),

('Daniel Walker', 'Spa Manager', 28000, 9876543231),

('Laura Hall', 'Reservations', 19000, 9876543232),

('Paul Allen', 'Waitstaff', 15000, 9876543233),

('Rebecca Young', 'Event Planner', 26000, 9876543234),

('Steven King', 'Bell Captain', 22000, 9876543235),

('Angela Wright', 'Food & Beverage Manager', 28000, 9876543236),

('Edward Scott', 'Executive Chef', 35000, 9876543237),

('Jessica Green', 'Dishwasher', 12000, 9876543238),

('Charles Adams', 'Room Service', 14000, 9876543239),

('Sophia Nelson', 'Waitstaff', 15000, 9876543240),

('Kevin Hill', 'Host', 16000, 9876543241),

('Alice Ramirez', 'Sous Chef', 23000, 9876543242),

('Mark Lee', 'Night Auditor', 19000, 9876543243),

('Angela Hall', 'Event Staff', 14000, 9876543244),

('Zachary Young', 'Social Media Manager', 20000, 9876543245),

('Elizabeth Hernandez', 'Sales Manager', 29000, 9876543246),

('Jason Torres', 'Guest Relations', 24000, 9876543247),

('Samantha Parker', 'Guest Services', 20000, 9876543248),

('Frank Edwards', 'Facilities Manager', 32000, 9876543249),

('Brandon Turner', 'Housekeeping Supervisor', 17000, 9876543250),

('Rachel Evans', 'Assistant Manager', 25000, 9876543251),

('Amy Turner', 'Night Manager', 27000, 9876543252),

('Paul White', 'Food Runner', 13000, 9876543253),

('Nora Roberts', 'Social Media Coordinator', 19000, 9876543254),

('Sam Taylor', 'Parking Attendant', 12000, 9876543255),

('Hannah Brooks', 'Sales Assistant', 15000, 9876543256),

('Brian Bailey', 'Houseman', 13000, 9876543257));

**View the tables:**

SELECT \* FROM Guests;

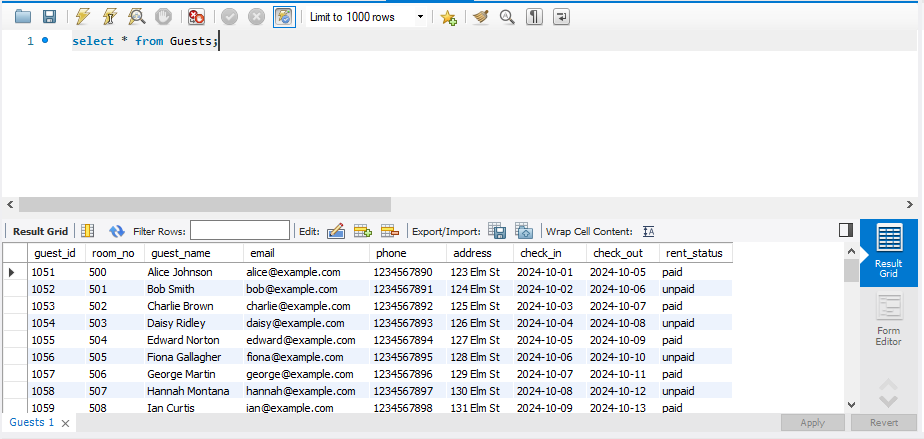
SELECT \* FROM rooms;

SELECT \* FROM staffs;

SELECT \* FROM payment;

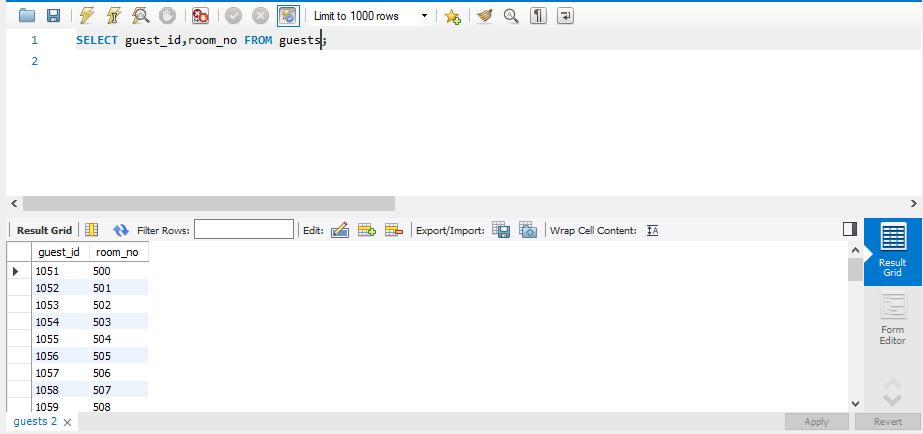
**QUESTION AND ANSWER:**

1.What SQL query would you use to retrieve all columns and all rows from a table named Table\_name?



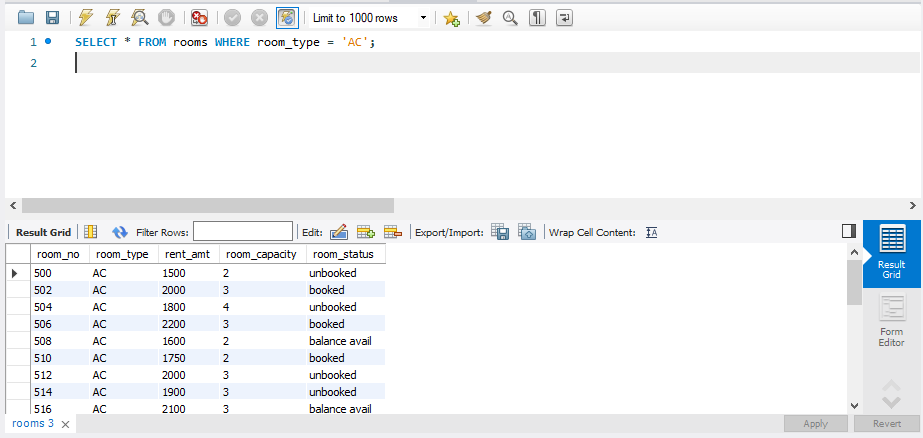
**Explanation:** This query retrieves all columns (\*) and all rows from the specified table\_name. It is the most basic form of a retrieval query.

2. Select Specific Columns!



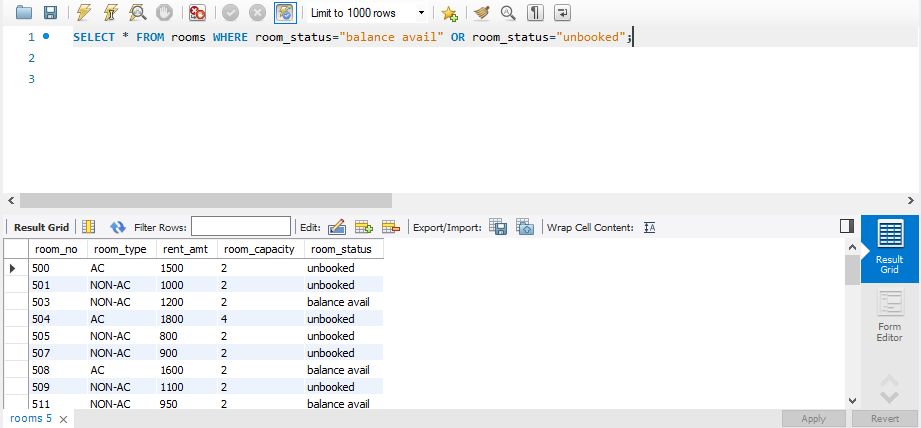
**Explanation:** Instead of retrieving all columns, you can specify which columns you want to retrieve. This is useful when you don’t need all the data from a table, improving performance and readability.

3. Using WHERE Clause (Filtering Results)



**Explanation:** The WHERE clause allows you to filter rows based on a specific condition. The condition can be a comparison (=, >, <, !=), pattern matching (LIKE), or other logical expressions.

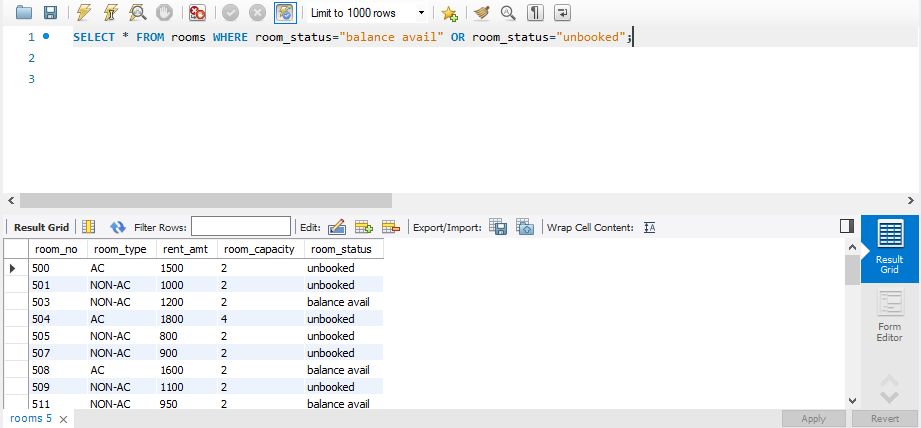
4.Using AND for Multiple Conditions



**Explanation:** You can combine multiple conditions using AND or OR.

AND returns rows where both conditions are true.

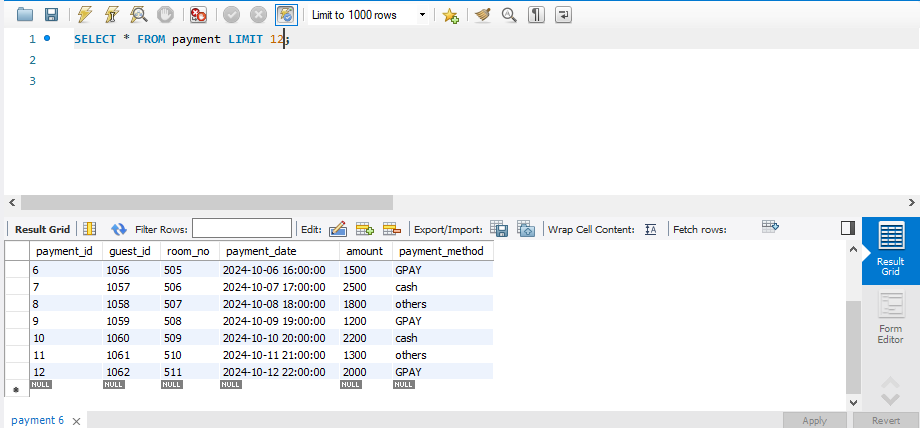
5.Using OR for Multiple Conditions



**Explanation:** You can combine multiple conditions using AND or OR.

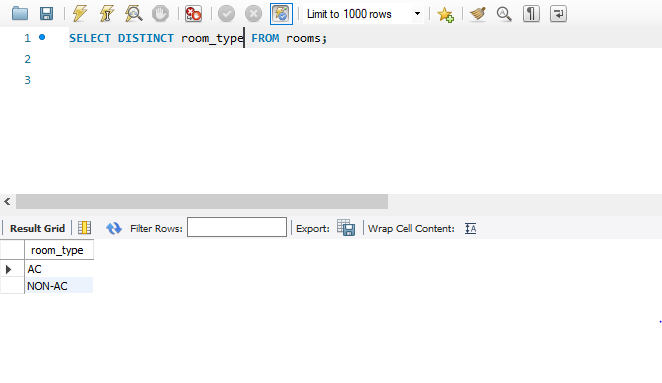
OR returns rows where at least one of the conditions is true.

6.Limiting Results with LIMIT:



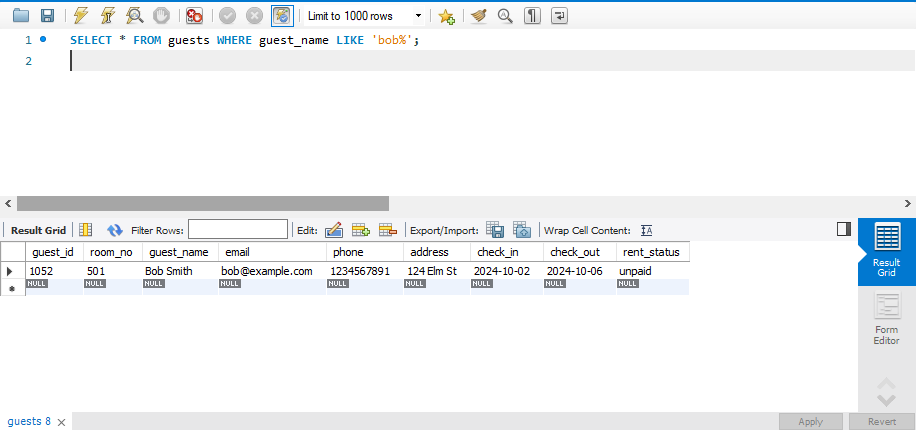
**Explanation:** The LIMIT clause is used to restrict the number of rows returned by the query.

7.Using DISTINCT to Eliminate Duplicates



**Explanation:** The DISTINCT keyword ensures that the result set contains only unique values for the specified column. It eliminates duplicate rows.

8.Pattern Matching with LIKE

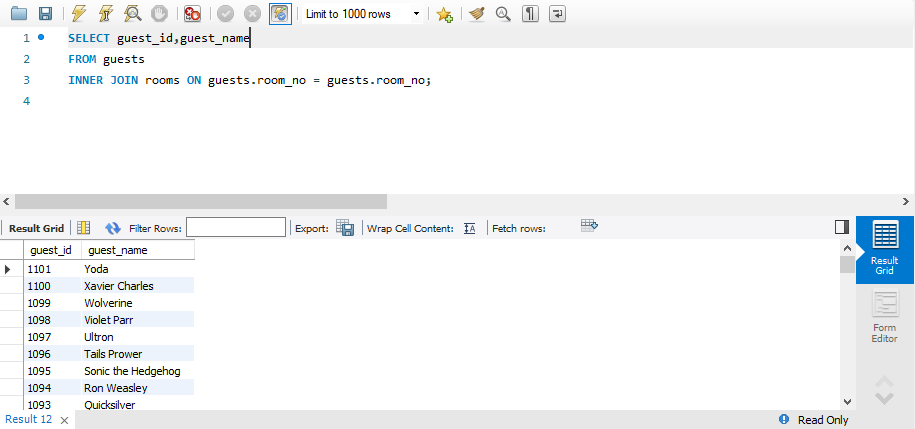


**Explanation:** The LIKE operator is used for pattern matching in string columns.

% represents zero or more characters.

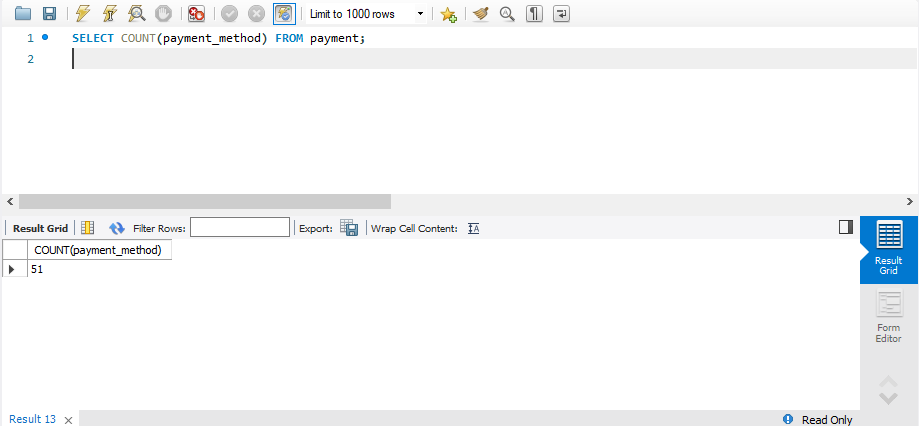
\_ represents exactly one character.

9. Using IN to Match Multiple Values



**Explanation:** The IN operator allows you to match the column values against a list of specified values.

10. Aggregate Functions (SUM, COUNT, AVG, MIN, MAX)



**Explanation:** The COUNT() function returns the number of rows in a table (or based on a condition).