

# **Hotel Management System**

## **PROJECT REPORT**

### **Group Members**

MAHNDSE23.2F-007

J.G. Sachini Gayanika

MADHNSE23.2F-008

W.G. Ashen Wishwa Geeth Jayarathna

# **ADVANCED DATABASE MANAGEMENT SYSTEM**

**Assessment NO:02**

Higher Diploma in Software Engineering

National Institute of Business Management

Under supervision

Ms. Kaushalya Dissanayake

Consultant/Lecture

## **DECLARATION**

“I certify that this project does not incorporate without acknowledgement, any material previously submitted for a Diploma in any institution and to the best of my knowledge and belief ,it does not contain any material previously published or written by another person or myself except where due reference is made in the text. I also hereby give consent for my project report, if accepted, to be made available for photocopying and for interlibrary loans, and for the title and summary to be made available to outside organizations”

<b>Index no</b>	<b>Name</b>	<b>Signature</b>
MAHNDSE23.2F-007	J.G. Sachini Gayanika	.....
MAHNDSE23.2F-008	W.G. Ashen Wishwa Geeth Jayarathna	.....

# **Acknowledgement**

We would like to express our sincere appreciation to everyone who contributed to the successful completion of this assignment. First and foremost, we extend our gratitude to our lecturer, MS Kaushalya Dissanayake for providing us with the opportunity to explore and deepen our knowledge of Oracle database management and PL/SQL programming. We would also like to thank our fellow group members for their collaboration and shared commitment to the project. Their insights and hard work greatly enriched the final deliveries. Additionally, we are thankful for the support and resources provided by our institution, which made this assignment possible. Last but not least, we appreciate the patience and understanding of our friends and family during the intense work on this project. Their encouragement and support were invaluable throughout this journey.

## **TABLE OF CONTENT**

### **Contents**

LIST OF TABLES .....	6
LIST OF FIGURES .....	7
ER DIAGRAM .....	9
Chapter 1 .....	10
Introduction.....	10
1.1 Background .....	10
1.2 Problem Statement .....	10
Aim and Objectives.....	11
CHAPTER 2 .....	12
Design and Result .....	12

## LIST OF TABLES

Table 1:Hotel Table.....	12
Table 2:Room Table .....	12
Table 3:Guest Table.....	12
Table 4:Reservation Table.....	13
Table 5:Payment Table.....	13

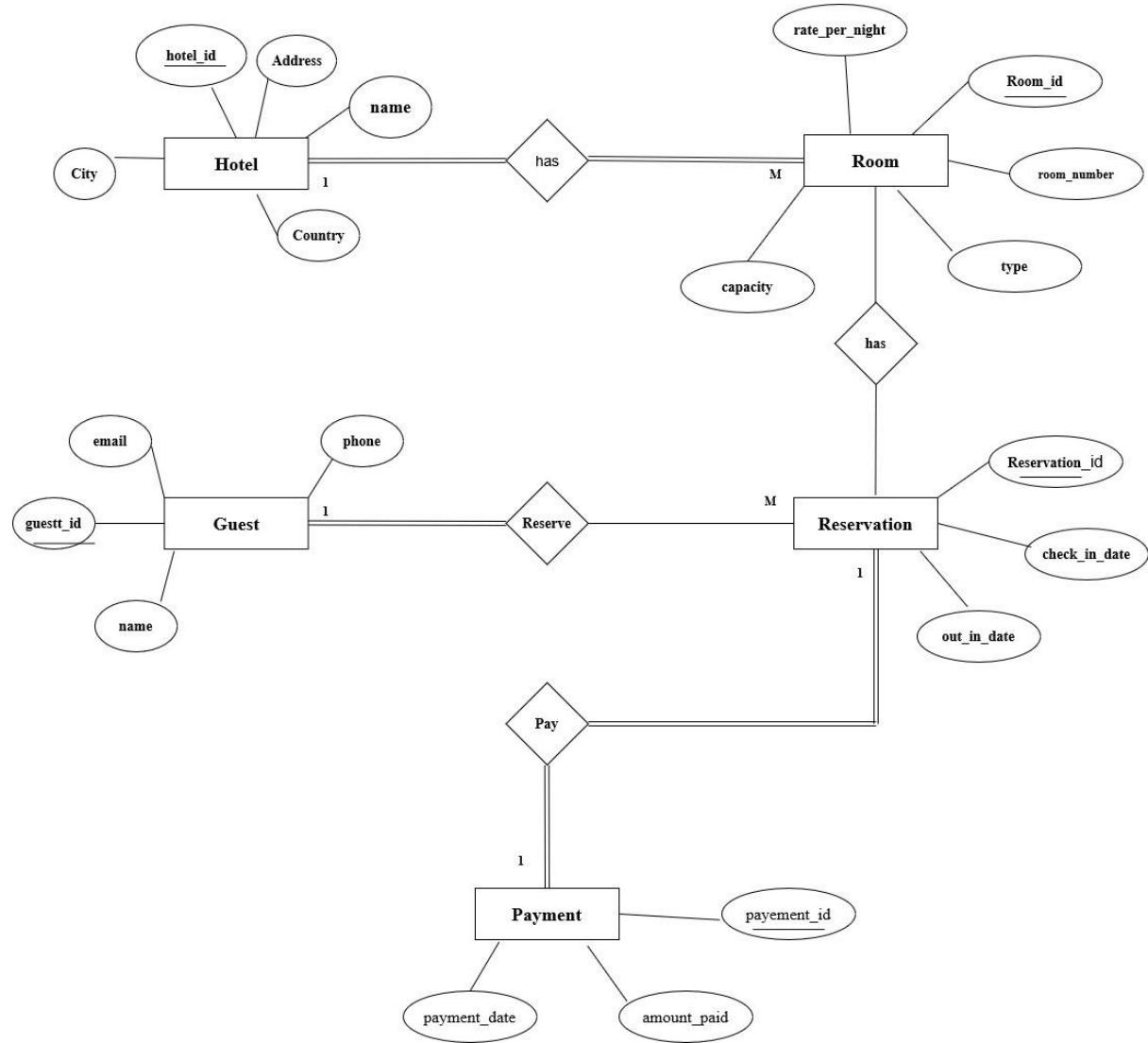
## LIST OF FIGURES

Figure 1 Create User .....	13
Figure 2Grant .....	14
Figure 3Grant All .....	14
Figure 4:Create Connection .....	15
Figure 5:Create Hotel Table .....	15
Figure 6:Create Room Table .....	16
Figure 7:Create Guest Table.....	17
Figure 8:Create Reservation Table.....	17
Figure 9:Create Payment Table.....	18
Figure 10:Insert data to Guest table .....	18
Figure 11:Insert Data to Hotel Table.....	19
Figure 12:Inset Data to Room Table .....	19
Figure 13:Insert Data to Reservation Table .....	20
Figure 14:Insert Data to Payment Table.....	20
Figure 15:DQL Query .....	21
Figure 16:DQL Query .....	22
Figure 17:DQL Query .....	22
Figure 18:DQL Query .....	23
Figure 19:DQL Query .....	23
Figure 20:DQL Query .....	24
Figure 21:DQL Query.....	24
Figure 22:DQL Query .....	25
Figure 23:DQL Query.....	25
Figure 24:DQL Query .....	26
Figure 25:DQL Query .....	26
Figure 26:DQL Query .....	27
Figure 27:DQL Query .....	27
Figure 28:DQL Query .....	28
Figure 29:DQL Query .....	28
Figure 30:DQL Query .....	29
Figure 31:DQL Query .....	29
Figure 32:DQL Query .....	30
Figure 33:DQL Query .....	30
Figure 34:User Given Inputs.....	31
Figure 35:User Given Inputs.....	31
Figure 36:User Given Inputs.....	32
Figure 37:User Given Inputs.....	32
Figure 38:User Given Inputs.....	33
Figure 39:User Given Inputs.....	33
Figure 40:User Given Inputs.....	34
Figure 41:User Given Inputs.....	34

Figure 42:User Given Inputs.....	35
Figure 43:User Given Inputs.....	35
Figure 44:User Given Inputs.....	36
Figure 45:User Given Inputs.....	36
Figure 46:User Given Inputs.....	37
Figure 47:User Given Inputs.....	37
Figure 48:User Given Inputs.....	38
Figure 49:IF ELSE .....	38
Figure 50:IF ELSE .....	39
Figure 51:Case .....	39
Figure 52:Case .....	40
Figure 53:For Loop .....	40
Figure 54:While Loop.....	41
Figure 55:While Loop.....	41
Figure 56>Create Procedure .....	42
Figure 57>Create Procedure .....	42
Figure 58>Create Procedure .....	43
Figure 59:Function.....	43
Figure 60:Function.....	44
Figure 61:Function.....	44
Figure 62:View.....	45
Figure 63:View.....	46
Figure 64:View.....	46
Figure 65:materialized view.....	47
Figure 66:materialized view.....	48
Figure 67:materialized view.....	48

## ER DIAGRAM

**ER diagram for hotel management system**



## Chapter 1

### Introduction

In the dynamic hospitality industry 1, efficient management of hotel operations is paramount for providing exceptional guest experience and maximizing profitability.

A Hotel management system facilitates the automation of numerous operational tasks, reducing manual errors and improving efficiency. This system handles guest reservations and check-in to manage room inventory and billing processes.

The primary goal is to enhance efficiency, reduce manual workload, and improve organizational effectiveness.

#### 1.1 Background

The hotel management system project aims to provide an efficient and organized solution for managers to handle hotel resources among employees and guests. In this comprehensive database design, we have created tables for hotels, rooms, guests, reservations, and payment, each meticulously structured with primary keys, foreign keys, check constraints, and other integrity features.

#### 1.2 Problem Statement

The objective of this project is to design and develop a comprehensive Hotel Management System that streamlines and automates various management processes within a hotel. The system aims to improve overall efficiency, communication, and data management while providing a user-friendly system for all employees and guests. It controls unnecessary waste of time.

- Unnecessary waste of money.
- Prevention of Data Loss.
- Weaknesses in file usage

## Aim and Objectives

This project's major goal is to provide a comprehensive and user-friendly system that makes it easier for guests and employees to manage and communicate effectively. The system tries to address issues with resource allocation, data management, and communication breakdowns the hotels.

- **In short,**
- Efficiency
- Guest Satisfaction
- Revenue Optimization
- Data-Driven Decisions
- Communication
- Security and Compliance
- Adaptability

# CHAPTER 2

## Design and Result

### Hotel table

Table 1:Hotel Table

NAME	DATA TYPE	CONSTRAINT
Hotel_id	INT	PRIMARY KEY
name	VARCHAR	NOT NULL
address	VARCHAR	
city	VARCHAR	
country	VARCHAR	

### Room table

Table 2:Room Table

NAME	DATA TYPE	CONSTRAINT
Room_id	INT	PRIMARY KEY
Hotel_id	INT	Foreign key
Room_number	INT	
type	VARCHAR	
Rate_per_night	NUMBER	
capacity	INT	

### Guest table

Table 3:Guest Table

NAME	DATA TYPE	CONSTRAINT
Guest_id	INT	PRIMARY KEY
name	VARCHAR	NOT_NULL
email	VARCHAR	
phone	VARCHAR	

## Reservation table

Table 4:Reservation Table

NAME	DATA TYPE	CONSTRAINT
Reservation_id	INT	PRIMARY KEY
Guest_id	INT	Foreign key
Room_id	INT	Foreign key
Check_in_date	DATE	
Chek_out_date	DATE	

## Payment table

Table 5:Payment Table

NAME	DATA TYPE	CONSTRAINT
Payment_id	INT	PRIMARY KEY
Reservation_id	INT	Foreign key
Amount_paid	NUMBER	
Payment_date	DATE	

- 1) Create a new user with assigning necessary privileges [Should provide explanation of user].

The screenshot shows the Oracle SQL Developer interface. In the central 'Worksheet' pane, there is a 'Query Builder' window containing the following SQL code:

```
create user HOTEL_SYSTEM IDENTIFIED BY hotel
default tablespace users temporary tablespace temp
quota 20m on users
account unlock;

grant create session to HOTEL_MANAGER;
grant all PRIVILEGES to HOTEL_MANAGER;
```

In the bottom right corner of the worksheet, the message "User HOTEL\_SYSTEM created." is displayed. The bottom status bar indicates "Task completed in 0.36 seconds".

Figure 1 Create User

The screenshot shows the Oracle SQL Developer interface. In the Connections pane, there are several database connections listed under Oracle Connections, including BOOK, DEMO, HR, USER1, USER2, and XE\_System. The XE\_System connection is selected. In the Worksheet tab of the main area, the following SQL code is entered:

```
create user hotel_system IDENTIFIED BY hotel
default tablespace users temporary tablespace temp
quota 20m on users
account unlock;

grant create session to hotel_system;

grant all PRIVILEGES to hotel_system;
```

In the Script Output pane at the bottom, the message "Grant succeeded." is displayed. The status bar at the bottom right indicates "Line 8 Column 38 | Insert | Modified | Windows: C".

Figure 2Grant

This screenshot is identical to Figure 2, except the last line of the SQL script in the Worksheet tab is changed to:

```
GRANT ALL PRIVILEGES TO hotel_system;
```

The rest of the interface, including the connections list, the grant message in the output pane, and the status bar, remains the same.

Figure 3Grant All

2) Create a new database connection for above created user.

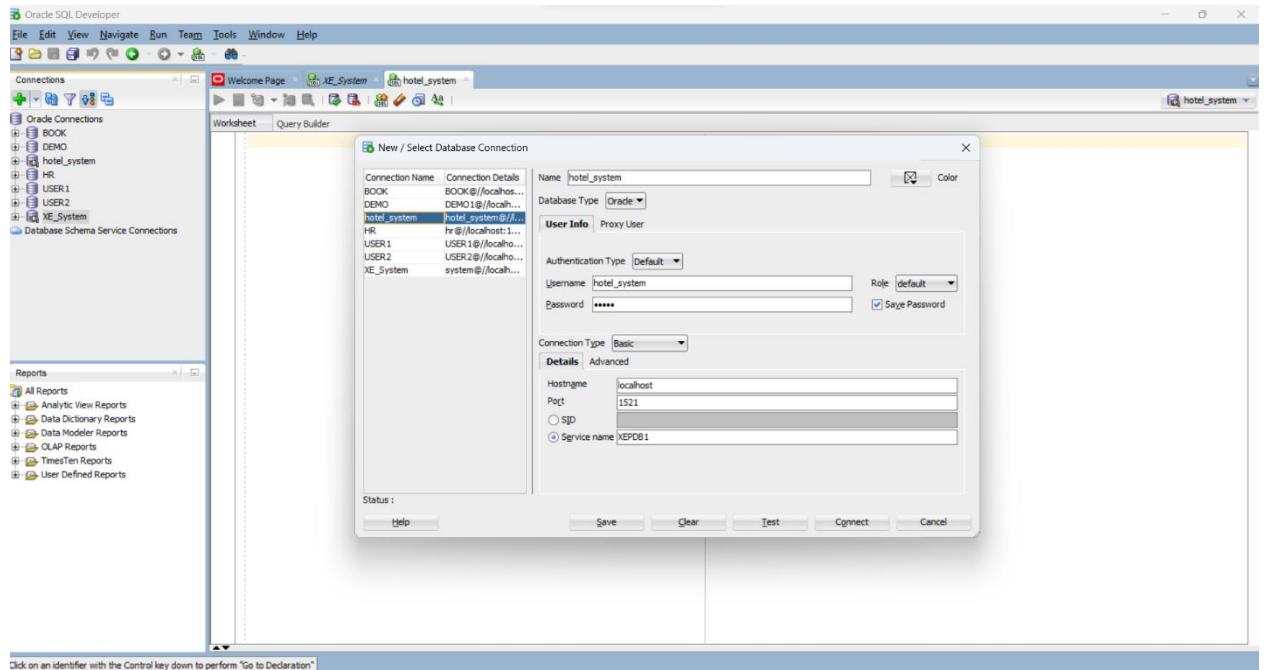


Figure 4:Create Connection

3) The tables should create with considering following features.

- Primary key and foreign key constraints
- Check Constraint • Not Null Constraint

```

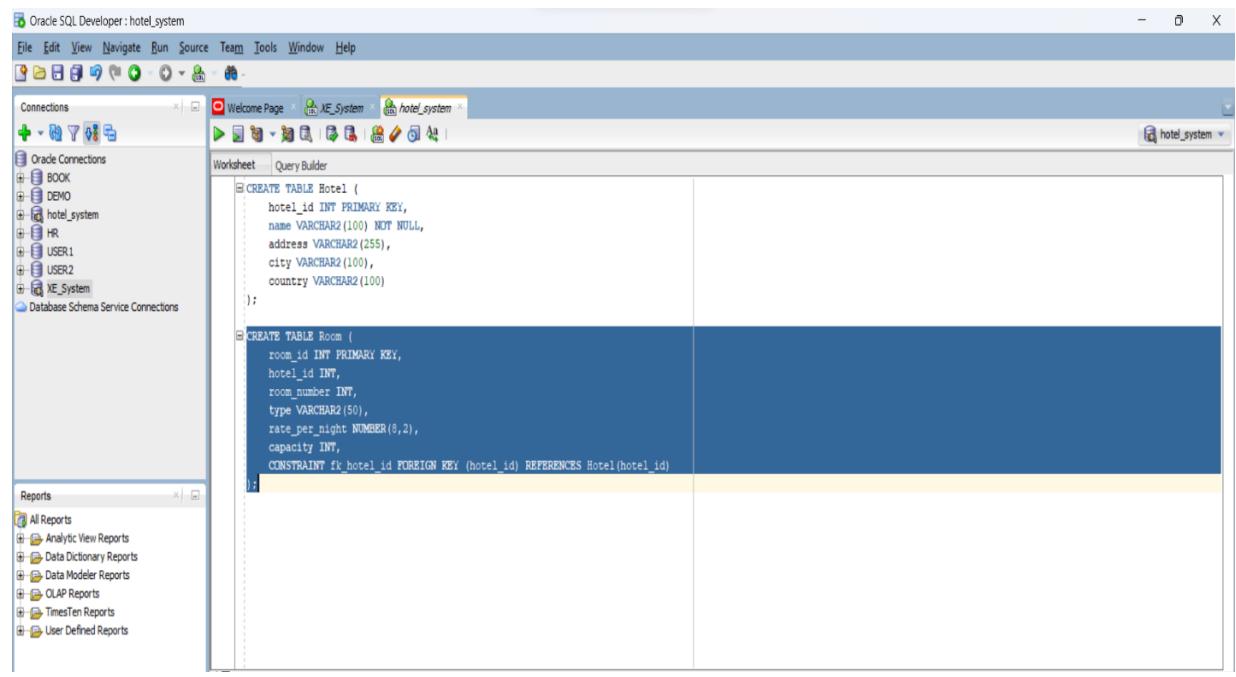
CREATE TABLE Hotel (
    hotel_id INT PRIMARY KEY,
    name VARCHAR2(100) NOT NULL,
    address VARCHAR2(255),
    city VARCHAR2(100),
    country VARCHAR2(100)
);

```

Table HOTEL created.

Figure 5:Create Hotel Table

To identify the identity of the hotel, details such as hotel\_id, name, address, city and country are in this hotel table. To identify each hotel, identity is identified by hotel\_id as its primary key



The screenshot shows the Oracle SQL Developer interface. The title bar reads "Oracle SQL Developer: hotel\_system". The menu bar includes File, Edit, View, Navigate, Run, Source, Team, Tools, Window, Help. The Connections sidebar lists Oracle Connections (BOOK, DEMO, hotel\_system, HR, USER1, USER2, XE\_System) and Database Schema Service Connections. The Reports sidebar lists All Reports, Analytic View Reports, Data Dictionary Reports, Data Model Reports, OLAP Reports, TimesTen Reports, and User Defined Reports. The central Worksheet tab contains the following SQL code:

```
CREATE TABLE Hotel (
    hotel_id INT PRIMARY KEY,
    name VARCHAR2(100) NOT NULL,
    address VARCHAR2(255),
    city VARCHAR2(100),
    country VARCHAR2(100)
);

CREATE TABLE Room (
    room_id INT PRIMARY KEY,
    hotel_id INT,
    room_number INT,
    type VARCHAR2(50),
    rate_per_night NUMBER(8,2),
    capacity INT,
    CONSTRAINT fk_hotel_id FOREIGN KEY (hotel_id) REFERENCES Hotel(hotel_id)
);
```

Figure 6:Create Room Table

Here each room acts as a single room and includes attributes such as room\_id, hotel\_id, room number, room type (single, double), rate\_per\_night and capacity. This facilitates room reservation and price management.

The screenshot shows the Oracle SQL Developer interface with the 'hotel\_system' connection selected. In the 'Worksheet' tab, the following SQL script is being run:

```

CREATE TABLE Hotel (
    hotel_id INT PRIMARY KEY,
    name VARCHAR2(100) NOT NULL,
    address VARCHAR2(255),
    city VARCHAR2(100),
    country VARCHAR2(100)
);

CREATE TABLE Room (
    room_id INT PRIMARY KEY,
    hotel_id INT,
    room_number INT,
    type VARCHAR2(50),
    rate_per_night NUMBER(8,2),
    capacity INT,
    CONSTRAINT fk_hotel_id FOREIGN KEY (hotel_id) REFERENCES Hotel(hotel_id)
);

CREATE TABLE Guest (
    guest_id INT PRIMARY KEY,
    name VARCHAR2(100) NOT NULL,
    email VARCHAR2(100),
    phone VARCHAR2(20)
);

```

The 'Script Output' pane at the bottom shows the message: "Table GUEST created.".

Figure 7:Create Guest Table

It provides information about the guests staying at the hotel. it include guest id,name,email,address and phone number.guest id is the primary key is the table.its used identified the guest separately.this table can be taken record of customers can be taken from this table.

The screenshot shows the Oracle SQL Developer interface with the 'hotel\_system' connection selected. In the 'Worksheet' tab, the following SQL script is being run:

```

CREATE TABLE Guest (
    guest_id INT PRIMARY KEY,
    name VARCHAR2(100) NOT NULL,
    email VARCHAR2(100),
    phone VARCHAR2(20)
);

CREATE TABLE Reservation (
    reservation_id INT PRIMARY KEY,
    guest_id INT,
    room_id INT,
    check_in_date DATE,
    check_out_date DATE,
    CONSTRAINT fk_guest_id FOREIGN KEY (guest_id) REFERENCES Guest(guest_id),
    CONSTRAINT fk_room_id FOREIGN KEY (room_id) REFERENCES Room(room_id),
    CONSTRAINT chk_dates CHECK (check_out_date > check_in_date)
);

```

The 'Script Output' pane at the bottom shows the message: "Table RESERVATION created.".

Figure 8:Create Reservation Table

This table shows us data details of reservations, including the reservation ID, guest ID, guest name, guest email, room ID, check-in date, and check-out date.

The screenshot shows the Oracle SQL Developer interface with the 'hotel\_system' database selected. In the 'Worksheet' tab, SQL code is being run to create the 'Payment' table. The code includes constraints for foreign keys linking to the 'Guest' and 'Room' tables, and a check constraint ensuring the check-out date is greater than the check-in date.

```

CREATE TABLE Payment
(
    payment_id NUMBER PRIMARY KEY,
    reservation_id INT,
    amount_paid NUMBER(10,2),
    payment_date DATE,
    CONSTRAINT fk_reservation_id FOREIGN KEY (reservation_id) REFERENCES Reservation(reservation_id)
);

```

The 'Script Output' tab at the bottom shows the confirmation message: "Table PAYMENT created.".

Figure 9:Create Payment Table

## Payment

The payment table is managed on the financial side.it include unique identifier payment\_id, reservation\_id,amount\_paid and payment\_date. reservation\_id is link to the reservation table. This table makes it easy to determine the payment related to guest bookings

The screenshot shows the Oracle SQL Developer interface with the 'hotel\_system' database selected. In the 'Worksheet' tab, SQL code is being run to insert five rows of data into the 'Guest' table. The code uses the 'INSERT INTO' statement with 'VALUES' clauses for each row, specifying guest\_id, name, email, and phone number.

```

-- Insert data into Guest table
INSERT INTO Guest (guest_id, name, email, phone)
VALUES (1, 'John Smith', 'john@example.com', '123-456-7890');

INSERT INTO Guest (guest_id, name, email, phone)
VALUES (2, 'Jane Doe', 'jane@example.com', '987-654-3210');

INSERT INTO Guest (guest_id, name, email, phone)
VALUES (3, 'Michael Johnson', 'michael@example.com', '456-789-0123');

INSERT INTO Guest (guest_id, name, email, phone)
VALUES (4, 'Emily Brown', 'emily@example.com', '321-654-0987');

INSERT INTO Guest (guest_id, name, email, phone)
VALUES (5, 'David Wilson', 'david@example.com', '789-012-3456');

```

The 'Script Output' tab at the bottom shows four messages: "1 row inserted.", "1 row inserted.", "1 row inserted.", and "1 row inserted.", indicating the successful insertion of all five rows.

Figure 10:Insert data to Guest table

The screenshot shows the Oracle SQL Developer interface with the 'hotel\_system' database selected. The 'Worksheet' tab is active, displaying an SQL script for inserting data into the 'Hotel' table. The script includes five INSERT statements with sample data for hotels in New York, Los Angeles, Miami, Aspen, and Chicago. Below the worksheet, the 'Script Output' window shows the results of the execution, indicating that each of the five rows was inserted successfully.

```

-- Insert data into Hotel table
INSERT INTO Hotel (hotel_id, name, address, city, country)
VALUES (1, 'Hotel ABC', '123 Main Street', 'New York', 'USA');

INSERT INTO Hotel (hotel_id, name, address, city, country)
VALUES (2, 'Grand Hotel', '456 Elm Street', 'Los Angeles', 'USA');

INSERT INTO Hotel (hotel_id, name, address, city, country)
VALUES (3, 'Beach Resort', '789 Ocean Avenue', 'Miami', 'USA');

INSERT INTO Hotel (hotel_id, name, address, city, country)
VALUES (4, 'Mountain Lodge', '101 Pine Road', 'Aspen', 'USA');

INSERT INTO Hotel (hotel_id, name, address, city, country)
VALUES (5, 'City Plaza Hotel', '555 Downtown Blvd', 'Chicago', 'USA');

```

Script Output | Task completed in 1.851 seconds

1 row inserted.  
1 row inserted.  
1 row inserted.  
1 row inserted.  
1 row inserted.

Figure 11:Insert Data to Hotel Table

The screenshot shows the Oracle SQL Developer interface with the 'hotel\_system' database selected. The 'Worksheet' tab is active, displaying an SQL script for inserting data into the 'Room' table. The script includes six INSERT statements with sample data for rooms 101 through 301, categorized by type (Single, Double, Suite) and capacity (1, 2, or 4). Below the worksheet, the 'Script Output' window shows the results of the execution, indicating that each of the six rows was inserted successfully.

```

-- Insert data into Room table
INSERT INTO Room (room_id, hotel_id, room_number, type, rate_per_night, capacity)
VALUES (1, 1, 101, 'Single', 100.00, 1);

INSERT INTO Room (room_id, hotel_id, room_number, type, rate_per_night, capacity)
VALUES (2, 1, 102, 'Double', 150.00, 2);

INSERT INTO Room (room_id, hotel_id, room_number, type, rate_per_night, capacity)
VALUES (3, 2, 201, 'Single', 120.00, 1);

INSERT INTO Room (room_id, hotel_id, room_number, type, rate_per_night, capacity)
VALUES (4, 2, 202, 'Double', 180.00, 2);

INSERT INTO Room (room_id, hotel_id, room_number, type, rate_per_night, capacity)
VALUES (5, 3, 301, 'Suite', 250.00, 4);

```

Script Output | Task completed in 0.137 seconds

1 row inserted.  
1 row inserted.  
1 row inserted.  
1 row inserted.

Figure 12:Inset Data to Room Table

The screenshot shows the Oracle SQL Developer interface with the 'hotel\_system' connection selected. The 'Worksheet' tab is active, displaying an SQL script for inserting data into the 'Reservation' table. The script contains five INSERT statements, each adding a new reservation record with guest\_id, room\_id, check\_in\_date, and check\_out\_date. The 'Script Output' pane below shows four '1 row inserted.' messages, indicating successful execution of the first four insert statements. The status bar at the bottom right shows 'Line 116 Column 92 | Insert | Modified | Windows: C'.

```
-- Insert data into Reservation table
INSERT INTO Reservation (reservation_id, guest_id, room_id, check_in_date, check_out_date)
VALUES (1, 1, 1, TO_DATE('2024-03-15', 'YYYY-MM-DD'), TO_DATE('2024-03-20', 'YYYY-MM-DD'));

INSERT INTO Reservation (reservation_id, guest_id, room_id, check_in_date, check_out_date)
VALUES (2, 2, 2, TO_DATE('2024-04-10', 'YYYY-MM-DD'), TO_DATE('2024-04-15', 'YYYY-MM-DD'));

INSERT INTO Reservation (reservation_id, guest_id, room_id, check_in_date, check_out_date)
VALUES (3, 3, 3, TO_DATE('2024-05-20', 'YYYY-MM-DD'), TO_DATE('2024-05-25', 'YYYY-MM-DD'));

INSERT INTO Reservation (reservation_id, guest_id, room_id, check_in_date, check_out_date)
VALUES (4, 4, 4, TO_DATE('2024-06-05', 'YYYY-MM-DD'), TO_DATE('2024-06-10', 'YYYY-MM-DD'));

INSERT INTO Reservation (reservation_id, guest_id, room_id, check_in_date, check_out_date)
VALUES (5, 5, 5, TO_DATE('2024-07-15', 'YYYY-MM-DD'), TO_DATE('2024-07-20', 'YYYY-MM-DD'));
```

Figure 13:Insert Data to Reservation Table

The screenshot shows the Oracle SQL Developer interface with the 'hotel\_system' connection selected. The 'Worksheet' tab is active, displaying an SQL script for inserting data into the 'Payment' table. The script contains five INSERT statements, each adding a new payment record with payment\_id, reservation\_id, amount\_paid, and payment\_date. The 'Script Output' pane below shows four '1 row inserted.' messages, indicating successful execution of the first four insert statements. The status bar at the bottom right shows 'Line 133 Column 61 | Insert | Modified | Windows: C'.

```
-- Insert data into Payment table
INSERT INTO Payment (payment_id, reservation_id, amount_paid, payment_date)
VALUES (1, 1, 500.00, TO_DATE('2024-03-18', 'YYYY-MM-DD'));

INSERT INTO Payment (payment_id, reservation_id, amount_paid, payment_date)
VALUES (2, 2, 750.00, TO_DATE('2024-04-13', 'YYYY-MM-DD'));

INSERT INTO Payment (payment_id, reservation_id, amount_paid, payment_date)
VALUES (3, 3, 1000.00, TO_DATE('2024-05-23', 'YYYY-MM-DD'));

INSERT INTO Payment (payment_id, reservation_id, amount_paid, payment_date)
VALUES (4, 4, 900.00, TO_DATE('2024-06-08', 'YYYY-MM-DD'));

INSERT INTO Payment (payment_id, reservation_id, amount_paid, payment_date)
VALUES (5, 5, 1250.00, TO_DATE('2024-07-18', 'YYYY-MM-DD'));
```

Figure 14:Insert Data to Payment Table

4) Write 20 DQL [Data Query Language] Command with covering following criteria.

- Where
- Group by
- Having
- Order by
- Like
- Count()
- Length()
- initcap()
- Max()
- Min()
- AND
- OR
- Join

The screenshot shows the Oracle SQL Developer interface. The 'Connections' sidebar lists several databases: BOOK, DEMO, hotel\_system, HR, USER1, USER2, and XE\_System. The 'Reports' sidebar shows various report types. The main 'Worksheet' tab displays a 'Query Builder' with the following SQL code:

```
VALUES (4, 4, 900.00, TO_DATE('2024-06-05', 'YYYY-MM-DD'));
INSERT INTO Payment (payment_id, reservation_id, amount_paid, payment_date)
VALUES (5, 5, 1250.00, TO_DATE('2024-07-18', 'YYYY-MM-DD'));

SELECT * FROM Room WHERE rate_per_night > 150;

SELECT hotel_id, COUNT(*) AS num_reservations
FROM Room
JOIN Reservation ON Room.room_id = Reservation.room_id
GROUP BY hotel_id;

SELECT * FROM Guest ORDER BY name;
SELECT * FROM Guest WHERE email LIKE '%example.com%';
SELECT COUNT(*) FROM Reservation WHERE guest_id = 1;
```

The 'Script Output' tab shows the results of the last query:

ROOM_ID	HOTEL_ID	ROOM_NUMBER	TYPE	RATE_PER_NIGHT	CAPACITY
1	4	2	202 Double	180	2
2	5	3	302 Suite	250	4

Figure 15:DQL Query

Retrieves all rooms where the rate per night is greater than \$150.

The screenshot shows the Oracle SQL Developer interface. The left sidebar displays connections to various databases, including 'BOOK', 'DEMO', 'hotel\_system', 'HR', 'USER1', 'USER2', and 'XE\_System'. Below this is a 'Reports' section with options like 'All Reports', 'Analytic View Reports', etc. The main workspace is titled 'Worksheet' and contains a 'Query Builder' tab. A complex DQL query is being run:

```

SELECT * FROM Room WHERE rate_per_night > 150;
SELECT hotel_id, COUNT(*) AS num_reservations
FROM Room
JOIN Reservation ON Room.room_id = Reservation.room_id
GROUP BY hotel_id;

SELECT * FROM Guest ORDER BY name;

SELECT * FROM Guest WHERE email LIKE '%example.com%';

SELECT COUNT(*) FROM Reservation WHERE guest_id = 1;

SELECT * FROM Guest WHERE LENGTH(name) > 10;

SELECT INITCAP(name) FROM Guest;

SELECT MAX(rate_per_night) FROM Room;

```

The 'Script Output' tab shows the results of the last SELECT statement:

HOTEL_ID	NUM_RESERVATIONS
1	2
2	2
3	1

Figure 16:DQL Query

Counts the number of reservations for each hotel.

This screenshot shows the same Oracle SQL Developer environment. The 'Worksheet' tab now displays a different DQL query:

```

SELECT * FROM Room WHERE rate_per_night > 150;
SELECT hotel_id, COUNT(*) AS num_reservations
FROM Room
JOIN Reservation ON Room.room_id = Reservation.room_id
GROUP BY hotel_id;

SELECT * FROM Guest ORDER BY name;

SELECT * FROM Guest WHERE email LIKE '%example.com%';

SELECT COUNT(*) FROM Reservation WHERE guest_id = 1;

SELECT * FROM Guest WHERE LENGTH(name) > 10;

SELECT INITCAP(name) FROM Guest;

SELECT MAX(rate_per_night) FROM Room;

```

The 'Script Output' tab shows the results of the last SELECT statement:

GUEST_ID	NAME	EMAIL	PHONE
1	David Wilson	david@example.com	789-012-3456
2	Emily Brown	emily@example.com	321-654-987
3	Jane Doe	jane@example.com	987-654-3210
4	John Smith	john@example.com	123-456-7890
5	Michael Johnson	michael@example.com	456-789-0123

Figure 17:DQL Query

Retrieves all guests and sorts them alphabetically by name.

The screenshot shows the Oracle SQL Developer interface. The 'Connections' sidebar lists several databases, including 'BOOK', 'DEMO', 'hotel\_system', 'HR', 'USER1', 'USER2', and 'XE\_System'. The 'Reports' sidebar includes options like 'All Reports', 'Analytic View Reports', and 'Data Dictionary Reports'. The 'Worksheet' tab displays a DQL query:

```

SELECT * FROM Room WHERE rate_per_night > 150;
SELECT hotel_id, COUNT(*) AS num_reservations
FROM Room
JOIN Reservation ON Room.room_id = Reservation.room_id
GROUP BY hotel_id;
SELECT * FROM Guest ORDER BY name;
SELECT * FROM Guest WHERE email LIKE '%example.com';
SELECT COUNT(*) FROM Reservation WHERE guest_id = 1;
SELECT * FROM Guest WHERE LENGTH(name) > 10;
SELECT INITCAP(name) FROM Guest;
SELECT MAX(rate_per_night) FROM Room;

```

The 'Script Output' and 'Query Result' panes show the results of the last query, which retrieves guest information:

GUEST_ID	NAME	EMAIL	PHONE
1	John Smith	john@example.com	123-456-7890
2	Jane Doe	jane@example.com	987-654-3210
3	Michael Johnson	michael@example.com	456-789-0123
4	Emily Brown	emily@example.com	321-654-0987
5	David Wilson	david@example.com	789-012-3456

Figure 18:DQL Query

Retrieves guests whose email addresses contain the domain 'example.com'.

The screenshot shows the Oracle SQL Developer interface. The 'Connections' sidebar lists several databases, including 'BOOK', 'DEMO', 'hotel\_system', 'HR', 'USER1', 'USER2', and 'XE\_System'. The 'Reports' sidebar includes options like 'All Reports', 'Analytic View Reports', and 'Data Dictionary Reports'. The 'Worksheet' tab displays a DQL query:

```

SELECT * FROM Guest WHERE email LIKE '%example.com';
SELECT COUNT(*) FROM Reservation WHERE guest_id = 1;
SELECT * FROM Guest WHERE LENGTH(name) > 10;
SELECT INITCAP(name) FROM Guest;
SELECT MAX(rate_per_night) FROM Room;
SELECT MIN(capacity) FROM Room;
SELECT * FROM Room WHERE rate_per_night < 200 AND capacity > 2;
SELECT * FROM Room WHERE rate_per_night < 100 OR capacity > 3;

SELECT Reservation.reservation_id, Guest.name
FROM Reservation
JOIN Guest ON Reservation.guest_id = Guest.guest_id;

```

The 'Script Output' and 'Query Result' panes show the results of the last query, which counts the number of reservations for guest\_id = 1:

COUNT(*)
1

Figure 19:DQL Query

Counts the number of reservations made by a specific guest (guest\_id = 1).

The screenshot shows the Oracle SQL Developer interface with the following details:

- Connections:** Oracle Connections, XE\_System, hotel\_system.
- Reports:** All Reports, Analytic View Reports, Data Dictionary Reports, Data Modeler Reports, OLAP Reports, TimesTen Reports, User Defined Reports.
- Worksheet:** Query Builder window containing the following DQL query:
 

```

SELECT * FROM Guest WHERE LENGTH(name) > 10;
      
```
- Script Output:** Shows the results of the query:
 

GUEST_ID	NAME	EMAIL	PHONE
1	Michael Johnson	michael@example.com	456-789-0123
2	Emily Brown	emily@example.com	321-654-0987
3	David Wilson	david@example.com	789-012-3456

Figure 20:DQL Query

Retrieves guests with names longer than 10 characters.

The screenshot shows the Oracle SQL Developer interface with the following details:

- Connections:** Oracle Connections, XE\_System, hotel\_system.
- Reports:** All Reports, Analytic View Reports, Data Dictionary Reports, Data Modeler Reports, OLAP Reports, TimesTen Reports, User Defined Reports.
- Worksheet:** Query Builder window containing the following DQL query:
 

```

SELECT INITCAP(name) FROM Guest;
      
```
- Script Output:** Shows the results of the query:
 

INITCAP(NAME)
1 John Smith
2 Jane Doe
3 Michael Johnson
4 Emily Brown
5 David Wilson

Figure 21:DQL Query

Retrieves guest names with the first letter of each word capitalized.

The screenshot shows the Oracle SQL Developer interface with the 'hotel\_system' connection selected. The 'Worksheet' tab is active, displaying the following DQL query:

```

SELECT * FROM Guest WHERE email LIKE '%example.com%';
SELECT COUNT(*) FROM Reservation WHERE guest_id = 1;
SELECT * FROM Guest WHERE LENGTH(name) > 10;
SELECT INITCAP(name) FROM Guest;

SELECT MAX(rate_per_night) FROM Room;

SELECT MIN(capacity) FROM Room;

SELECT * FROM Room WHERE rate_per_night < 200 AND capacity > 2;
SELECT * FROM Room WHERE rate_per_night < 100 OR capacity > 3;

SELECT Reservation.reservation_id, Guest.name
FROM Reservation
JOIN Guest ON Reservation.guest_id = Guest.guest_id;

```

The 'Script Output' tab shows the results of the MAX(rate\_per\_night) query:

	MAX(RATE_PER_NIGHT)
1	250

Below the interface, status messages include: 'Click on an identifier with the Control key down to perform "Go to Declaration"', 'Line 156 Column 38 | Insert | Modified | Windows:', and 'Line 156 Column 38 | Insert | Modified | Windows:'.

Figure 22:DQL Query

Retrieves the maximum rate per night across all rooms

The screenshot shows the Oracle SQL Developer interface with the 'hotel\_system' connection selected. The 'Worksheet' tab is active, displaying the same DQL query as Figure 22, but with a different result for the MIN(capacity) query:

```

SELECT * FROM Guest WHERE email LIKE '%example.com%';
SELECT COUNT(*) FROM Reservation WHERE guest_id = 1;
SELECT * FROM Guest WHERE LENGTH(name) > 10;
SELECT INITCAP(name) FROM Guest;

SELECT MAX(rate_per_night) FROM Room;

SELECT MIN(capacity) FROM Room;

SELECT * FROM Room WHERE rate_per_night < 200 AND capacity > 2;
SELECT * FROM Room WHERE rate_per_night < 100 OR capacity > 3;

SELECT Reservation.reservation_id, Guest.name
FROM Reservation
JOIN Guest ON Reservation.guest_id = Guest.guest_id;

```

The 'Script Output' tab shows the results of the MIN(capacity) query:

	MIN(CAPACITY)
1	1

Below the interface, status messages include: 'Click on an identifier with the Control key down to perform "Go to Declaration"', 'Line 158 Column 32 | Insert | Modified | Windows:', and 'Line 158 Column 32 | Insert | Modified | Windows:'.

Figure 23:DQL Query

Retrieves the minimum capacity across all rooms.

The screenshot shows the Oracle SQL Developer interface. The 'Connections' sidebar lists several databases, including 'BOOK', 'DEMO', 'hotel\_system', 'HR', 'USER1', 'USER2', and 'XE\_System'. The 'Reports' sidebar contains various report types. The 'Worksheet' tab is active, displaying the following DQL query:

```

SELECT * FROM Guest WHERE email LIKE '%example.com%';
SELECT COUNT(*) FROM Reservation WHERE guest_id = 1;
SELECT * FROM Guest WHERE LENGTH(name) > 10;
SELECT INITCAP(name) FROM Guest;
SELECT MAX(rate_per_night) FROM Room;
SELECT MIN(capacity) FROM Room;
SELECT * FROM Room WHERE rate_per_night < 200 AND capacity > 2;
SELECT * FROM Room WHERE rate_per_night < 100 OR capacity > 3;

SELECT Reservation.reservation_id, Guest.name
FROM Reservation
JOIN Guest ON Reservation.guest_id = Guest.guest_id;

```

The 'Script Output' tab shows the results of the query, which is empty. The 'Query Result' tab shows the structure of the result set:

ROOM_ID	HOTEL_ID	ROOM_NUMBER	TYPE	RATE_PER_NIGHT	CAPACITY
---------	----------	-------------	------	----------------	----------

Figure 24:DQL Query

Retrieves rooms with a rate per night less than \$200 and a capacity greater than 2.

The screenshot shows the Oracle SQL Developer interface. The 'Connections' sidebar lists several databases, including 'BOOK', 'DEMO', 'hotel\_system', 'HR', 'USER1', 'USER2', and 'XE\_System'. The 'Reports' sidebar contains various report types. The 'Worksheet' tab is active, displaying the following DQL query:

```

SELECT INITCAP(name) FROM Guest;
SELECT MAX(rate_per_night) FROM Room;
SELECT MIN(capacity) FROM Room;
SELECT * FROM Room WHERE rate_per_night < 200 AND capacity > 2;
SELECT * FROM Room WHERE rate_per_night < 100 OR capacity > 3;

SELECT Reservation.reservation_id, Guest.name
FROM Reservation
JOIN Guest ON Reservation.guest_id = Guest.guest_id;
SELECT * FROM Reservation WHERE check_in_date > TO_DATE('2024-05-01', 'YYYY-MM-DD') ORDER BY check_in_date;

SELECT Room.hotel_id, COUNT(*) AS num_reservations
FROM Reservation
JOIN Room ON Reservation.room_id = Room.room_id
GROUP BY Room.hotel_id

```

The 'Script Output' tab shows the results of the query, which is empty. The 'Query Result' tab shows the structure of the result set:

ROOM_ID	HOTEL_ID	ROOM_NUMBER	TYPE	RATE_PER_NIGHT	CAPACITY	
1	5	3	301	Suite	250	4

Figure 25:DQL Query

Retrieves rooms with a rate per night less than \$100 or a capacity greater than 3.

The screenshot shows the Oracle SQL Developer interface with the 'hotel\_system' database selected. The 'Worksheet' tab is active, displaying a DQL query:

```

SELECT * FROM Room WHERE rate_per_night < 200 AND capacity > 2;
SELECT * FROM Room WHERE rate_per_night < 100 OR capacity > 3;

SELECT Reservation.reservation_id, Guest.name
FROM Reservation
JOIN Guest ON Reservation.guest_id = Guest.guest_id;

SELECT * FROM Reservation WHERE check_in_date > TO_DATE('2024-05-01', 'YYYY-MM-DD') ORDER BY check_in_date;

SELECT Room.hotel_id, COUNT(*) AS num_reservations
FROM Reservation
JOIN Room ON Reservation.room_id = Room.room_id
GROUP BY Room.hotel_id
HAVING COUNT(*) > 2
ORDER BY num_reservations DESC;

SELECT type, COUNT(*) AS num_reservations
FROM Room

```

The 'Script Output' tab shows the execution results:

```

Script Output | Query Result | Query Result 1 |
SQL | All Rows Fetched: 3 in 0.065 seconds

RESERVATION_ID | GUEST_ID | ROOM_ID | CHECK_IN_DATE | CHECK_OUT_DATE
1 | 3 | 3 | 3 20-MAY-24 | 25-MAY-24
2 | 4 | 4 | 4 05-JUN-24 | 10-JUN-24
3 | 5 | 5 | 5 15-JUL-24 | 20-JUL-24

```

Figure 26:DQL Query

Retrieves reservation IDs along with the names of guests who made those reservations.

The screenshot shows the Oracle SQL Developer interface with the 'hotel\_system' database selected. The 'Worksheet' tab is active, displaying a DQL query:

```

SELECT * FROM Room WHERE rate_per_night < 200 AND capacity > 2;
SELECT * FROM Room WHERE rate_per_night < 100 OR capacity > 3;

SELECT Reservation.reservation_id, Guest.name
FROM Reservation
JOIN Guest ON Reservation.guest_id = Guest.guest_id;

SELECT * FROM Reservation WHERE check_in_date > TO_DATE('2024-05-01', 'YYYY-MM-DD') ORDER BY check_in_date;

SELECT Room.hotel_id, COUNT(*) AS num_reservations
FROM Reservation
JOIN Room ON Reservation.room_id = Room.room_id
GROUP BY Room.hotel_id
HAVING COUNT(*) > 2
ORDER BY num_reservations DESC;

SELECT type, COUNT(*) AS num_reservations
FROM Room

```

The 'Script Output' tab shows the execution results:

```

Script Output | Query Result | Query Result 1 |
SQL | All Rows Fetched: 0 in 0.009 seconds

HOTEL_ID | NUM_RES...

```

Figure 27:DQL Query

Counts the number of reservations per hotel and filters for hotels with more than 2 reservations, ordering them by the number of reservations.

The screenshot shows the Oracle SQL Developer interface. The 'Connections' sidebar lists several databases, including 'BOOK', 'DEMO', 'hotel\_system', 'HR', 'USER1', 'USER2', and 'XE\_System'. The 'Reports' sidebar lists various report types. The 'Worksheet' pane contains the following DQL query:

```

SELECT type, COUNT(*) AS num_reservations
FROM Room
JOIN Reservation ON Room.room_id = Reservation.room_id
GROUP BY type
ORDER BY num_reservations DESC;

```

The 'Script Output' pane shows the results of the query:

TYPE	NUM_RESERVATIONS
1 Single	2
2 Double	2
3 Suite	1

Below the table, status information indicates: 'All Rows Parsed: 3 in 0.015 seconds'.

Figure 28:DQL Query

Counts the number of reservations for each room type, ordering them by the number of reservations.

The screenshot shows the Oracle SQL Developer interface. The 'Connections' sidebar lists databases 'BOOK', 'DEMO', 'hotel\_system', 'HR', 'USER1', 'USER2', and 'XE\_System'. The 'Reports' sidebar lists reports. The 'Worksheet' pane contains the following DQL query:

```

SELECT type, COUNT(*) AS num_reservations
FROM Room
JOIN Reservation ON Room.room_id = Reservation.room_id
GROUP BY type
ORDER BY num_reservations DESC;

SELECT * FROM Guest WHERE name LIKE 'J%';
SELECT INITCAP(name)
FROM Hotel
WHERE country = 'USA'
ORDER BY name;

SELECT hotel_id, MAX(rate_per_night) AS max_rate
FROM Room
GROUP BY hotel_id;

SELECT Room.hotel_id, MIN(rate_per_night) AS min_rate
FROM Room
JOIN Hotel ON Room.hotel_id = Hotel.hotel_id
WHERE city = 'New York';

```

The 'Script Output' pane shows the results of the query:

INITCAP(NAME)
1 Beach Resort
2 City Plaza Hotel
3 Grand Hotel
4 Hotel ABC
5 Mountain Lodge

Below the table, status information indicates: 'All Rows Parsed: 5 in 0.011 seconds'.

Figure 29:DQL Query

Retrieves hotel names in the USA with the first letter of each word capitalized, ordered by name.

```

SELECT * FROM Guest WHERE name LIKE 'J%' ORDER BY name;
SELECT INITCAP(name)
  FROM Hotel
 WHERE country = 'USA'
 ORDER BY name;

SELECT hotel_id, MAX(rate_per_night) AS max_rate
  FROM Room
 GROUP BY hotel_id;

SELECT Room.hotel_id, MIN(rate_per_night) AS min_rate
  FROM Room
 JOIN Hotel ON Room.hotel_id = Hotel.hotel_id
 WHERE city = 'New York'
 GROUP BY Room.hotel_id
 ORDER BY min_rate;

SELECT * FROM Reservation WHERE guest_id = 1;

```

Script Output x Query Result x

HOTEL_ID	MAX_RATE
1	150
2	180
3	250

Figure 30:DQL Query

Retrieves the maximum rate per night for each hotel.

```

SELECT hotel_id, MAX(rate_per_night) AS max_rate
  FROM Room
 GROUP BY hotel_id;

SELECT Room.hotel_id, MIN(rate_per_night) AS min_rate
  FROM Room
 JOIN Hotel ON Room.hotel_id = Hotel.hotel_id
 WHERE city = 'New York'
 GROUP BY Room.hotel_id
 ORDER BY min_rate;

SELECT * FROM Reservation WHERE guest_id = 1;
SELECT * FROM Room WHERE rate_per_night < 150.00;

```

Script Output x Query Result x

HOTEL_ID	MIN_RATE
1	100

Figure 31:DQL Query

Retrieves the minimum rate per night for hotels in New York, ordered by the minimum rate.

The screenshot shows the Oracle SQL Developer interface. The left sidebar contains connections to 'BOOK', 'DEMO', 'hotel\_system', 'HR', 'USER1', 'USER2', and 'XE\_System'. The 'Reports' section lists 'All Reports', 'Analytic View Reports', 'Data Dictionary Reports', 'Data Modeler Reports', 'OLAP Reports', 'TimesTen Reports', and 'User Defined Reports'. The main workspace has a title bar 'Welcome Page - XE\_System hotel\_system'. Below it is a toolbar with various icons. A code editor window titled 'Run Statement (Ctrl+Enter)' contains the following SQL code:

```

SELECT hotel_id, MAX(rate_per_night) AS max_rate
FROM Room
GROUP BY hotel_id;

SELECT Room.hotel_id, MIN(rate_per_night) AS min_rate
FROM Room
JOIN Hotel ON Room.hotel_id = Hotel.hotel_id
WHERE city = 'New York'
GROUP BY Room.hotel_id
ORDER BY min_rate;

SELECT * FROM Reservation WHERE guest_id = 1;

SELECT * FROM Room WHERE rate_per_night < 150.00;

```

Below the code editor is a 'Script Output' tab showing the execution status: 'All Rows Fetched: 1 in 0.004 seconds'. The 'Query Result' tab displays the results of the fourth query:

RESERVATION_ID	GUEST_ID	ROOM_ID	CHECK_IN_DATE	CHECK_OUT_DATE
1	1	1	15-MAR-24	20-MAR-24

At the bottom of the interface, there are status bars for 'Line 203 Column 46' and 'Insert | Modified | Windows'.

Figure 32:DQL Query

Retrieves reservations made by a specific guest (guest\_id = 1).

The screenshot shows the Oracle SQL Developer interface, identical to Figure 32 but with a different query. The code editor window now contains the following SQL code:

```

SELECT hotel_id, MAX(rate_per_night) AS max_rate
FROM Room
GROUP BY hotel_id;

SELECT Room.hotel_id, MIN(rate_per_night) AS min_rate
FROM Room
JOIN Hotel ON Room.hotel_id = Hotel.hotel_id
WHERE city = 'New York'
GROUP BY Room.hotel_id
ORDER BY min_rate;

SELECT * FROM Reservation WHERE guest_id = 1;

SELECT * FROM Room WHERE rate_per_night < 150.00;

```

The 'Query Result' tab displays the results of the last query:

ROOM_ID	HOTEL_ID	ROOM_NUMBER	TYPE	RATE_PER_NIGHT	CAPACITY
1	1	1	101 Single	100	1
2	3	2	201 Single	120	1

At the bottom of the interface, there are status bars for 'Line 205 Column 50' and 'Insert | Modified | Windows'.

Figure 33:DQL Query

Retrieves rooms with a rate per night less than \$150.00.

- 5) Write five PL/SQL programs to retrieve data based on the user given inputs.

Oracle SQL Developer: XE\_System1st sem/nd stage/ADMS/hotel\_system.sql

File Edit View Navigate Run Source Team Tools Window Help

Connections

- BOOK
- DEMO
- hotel\_system
  - Tables (Filtered)
  - Views
  - Indexes
  - Packages
  - Procedures
    - CALCULATETOTALAMOUNTPA
    - GETGUESTINFO
    - UPDATEGUESTPHONE
  - Functions
  - Operators
  - Queues
  - Queues Tables
  - Triggers

Reports

- All Reports
- Analytic View Reports
- Data Dictionary Reports
- Data Modeler Reports
- OLAP Reports
- TimesTen Reports
- User Defined Reports

SQL Worksheet History

Worksheet Query Builder

```

SET SERVEROUTPUT ON;
DECLARE
  v_hotel_id INT;
  v_hotel_name VARCHAR2(100);
  v_hotel_address VARCHAR2(255);
  v_hotel_city VARCHAR2(100);
  v_hotel_country VARCHAR2(100);
BEGIN
  v_hotel_id := :hotel_id;

  SELECT name, address, city, country
  INTO v_hotel_name, v_hotel_address, v_hotel_city,
  FROM Hotel
  WHERE hotel_id = v_hotel_id;

  DBMS_OUTPUT.PUT_LINE('Hotel Name: ' || v_hotel_name);
  DBMS_OUTPUT.PUT_LINE('Address: ' || v_hotel_address);
  DBMS_OUTPUT.PUT_LINE('City: ' || v_hotel_city);
  DBMS_OUTPUT.PUT_LINE('Country: ' || v_hotel_country);
END;
  
```

Enter Substitution Variable

Enter value for hotel\_id:

OK Cancel

Script Output

ScriptRunner Task

Click on an identifier with the Control key down to perform "Go to Declaration"

Line 233 Column 1 Insert Modified Windows

Figure 34:User Given Inputs

Oracle SQL Developer: XE\_System1st sem/nd stage/ADMS/hotel\_system.sql

File Edit View Navigate Run Source Team Tools Window Help

Connections

- BOOK
- DEMO
- hotel\_system
  - Tables (Filtered)
  - Views
  - Indexes
  - Packages
  - Procedures
    - CALCULATETOTALAMOUNTPA
    - GETGUESTINFO
    - UPDATEGUESTPHONE
  - Functions
  - Operators
  - Queues
  - Queues Tables
  - Triggers

Reports

- All Reports
- Analytic View Reports
- Data Dictionary Reports
- Data Modeler Reports
- OLAP Reports
- TimesTen Reports
- User Defined Reports

SQL Worksheet History

Worksheet Query Builder

```

SET SERVEROUTPUT ON;
DECLARE
  v_hotel_id INT;
  v_hotel_name VARCHAR2(100);
  v_hotel_address VARCHAR2(255);
  v_hotel_city VARCHAR2(100);
  v_hotel_country VARCHAR2(100);
BEGIN
  v_hotel_id := :hotel_id;

  SELECT name, address, city, country
  INTO v_hotel_name, v_hotel_address, v_hotel_city, v_hotel_country
  FROM Hotel
  WHERE hotel_id = v_hotel_id;

  DBMS_OUTPUT.PUT_LINE('Hotel Name: ' || v_hotel_name);
  DBMS_OUTPUT.PUT_LINE('Address: ' || v_hotel_address);
  DBMS_OUTPUT.PUT_LINE('City: ' || v_hotel_city);
  DBMS_OUTPUT.PUT_LINE('Country: ' || v_hotel_country);
END;
  
```

Script Output

Task completed in 20.952 seconds

Click on an identifier with the Control key down to perform "Go to Declaration"

Line 233 Column 5 Insert Modified Windows

Figure 35:User Given Inputs

The screenshot shows the Oracle SQL Developer interface. The left sidebar displays the 'Connections' tree, which includes 'BOOK', 'DEMO', and 'hotel\_system' (selected). The 'hotel\_system' node has 'Tables (Filtered)', 'Views', 'Indexes', 'Packages', and 'Procedures'. The 'Procedures' node contains three entries: 'CALCULATETOTALAMOUNTPA', 'GETGUESTINFO', and 'UPDATEGUESTPHONE'. The 'Reports' section lists 'All Reports', 'Analytic View Reports', 'Data Dictionary Reports', 'Data Modeler Reports', 'OLAP Reports', 'TimesTen Reports', and 'User Defined Reports'. The main workspace shows a PL/SQL script in the 'Worksheet' tab:

```

SET SERVEROUTPUT ON;
DECLARE
    v_hotel_id INT;
    v_hotel_name VARCHAR2(100);
    v_hotel_address VARCHAR2(255);
    v_hotel_city VARCHAR2(100);
    v_hotel_country VARCHAR2(100);
BEGIN
    v_hotel_id := :hotel_id;

    SELECT name, address, city, country
    INTO v_hotel_name, v_hotel_address, v_hotel_city, v_hotel_country
    FROM Hotel
    WHERE hotel_id = v_hotel_id;

    DBMS_OUTPUT.PUT_LINE('Hotel Name: ' || v_hotel_name);
    DBMS_OUTPUT.PUT_LINE('Address: ' || v_hotel_address);
    DBMS_OUTPUT.PUT_LINE('City: ' || v_hotel_city);
    DBMS_OUTPUT.PUT_LINE('Country: ' || v_hotel_country);
END;

```

The 'Script Output' tab at the bottom shows the results of the execution:

```

Task completed in 3.561 seconds

```

Figure 36:User Given Inputs

This will get the hotel id and the corresponding hotel name and address from the city and country database

The screenshot shows the Oracle SQL Developer interface. The left sidebar displays the 'Connections' tree, which includes 'BOOK', 'DEMO', and 'hotel\_system' (selected). The 'hotel\_system' node has 'Tables (Filtered)', 'Views', 'Indexes', 'Packages', and 'Procedures'. The 'Procedures' node contains three entries: 'CALCULATETOTALAMOUNTPA', 'GETGUESTINFO', and 'UPDATEGUESTPHONE'. The 'Reports' section lists 'All Reports', 'Analytic View Reports', 'Data Dictionary Reports', 'Data Modeler Reports', 'OLAP Reports', 'TimesTen Reports', and 'User Defined Reports'. The main workspace shows a PL/SQL script in the 'Worksheet' tab:

```

SELECT room_number, type, rate_per_night, capacity
INTO v_room_number, v_room_type, v_rate_per_night, v_capacity
FROM Room
WHERE room_id = v_room_id;

-- Displaying retrieved information
DBMS_OUTPUT.PUT_LINE('Room Number: ' || v_room_number);
DBMS_OUTPUT.PUT_LINE('Type: ' || v_room_type);
DBMS_OUTPUT.PUT_LINE('Rate per Night: ' || v_rate_per_night);
DBMS_OUTPUT.PUT_LINE('Capacity: ' || v_capacity);

END;
/

```

The 'Script Output' tab at the bottom shows the results of the execution:

```

Task completed in 15.957 seconds
ID#:
Room Number: 101
Type: Single
Rate per Night: 100
Capacity: 1

```

Figure 37:User Given Inputs

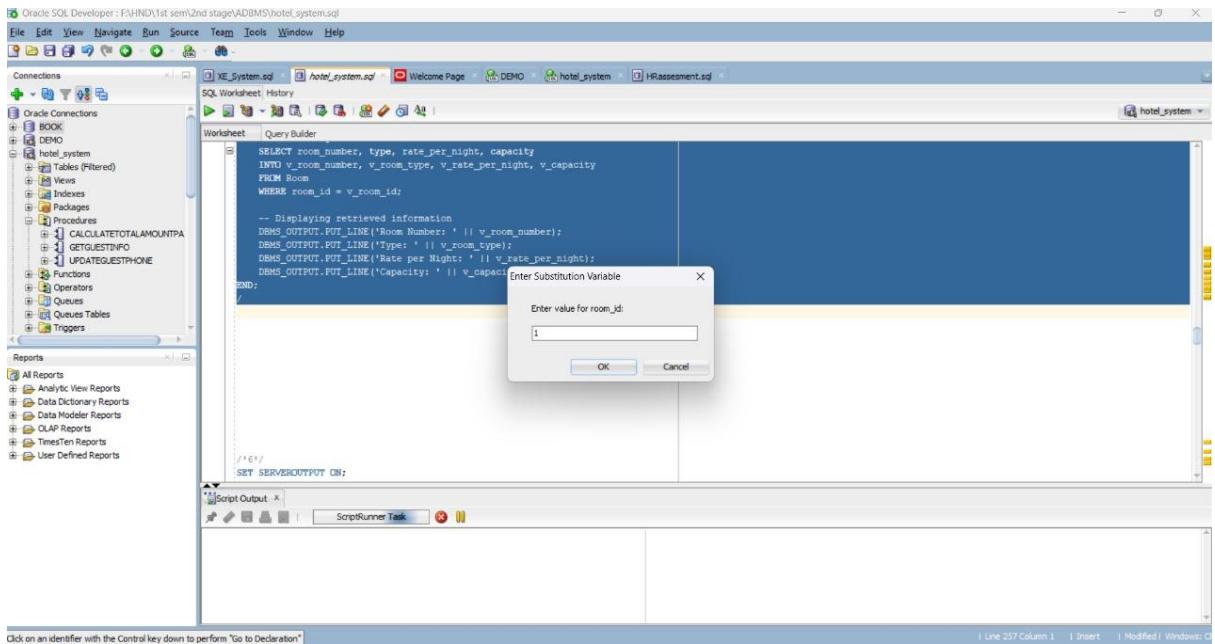


Figure 38:User Given Inputs

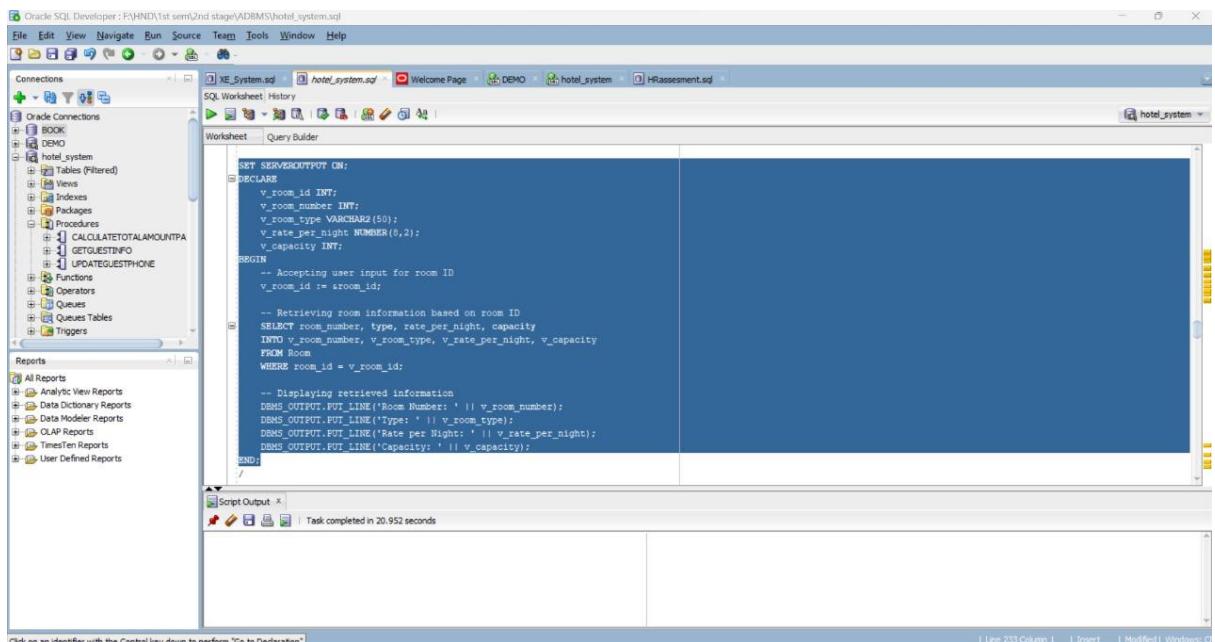


Figure 39:User Given Inputs

Here the guest input the room number and gets the details of the room related to that room number. The database displays the room number type rate per night and capacity.

.....

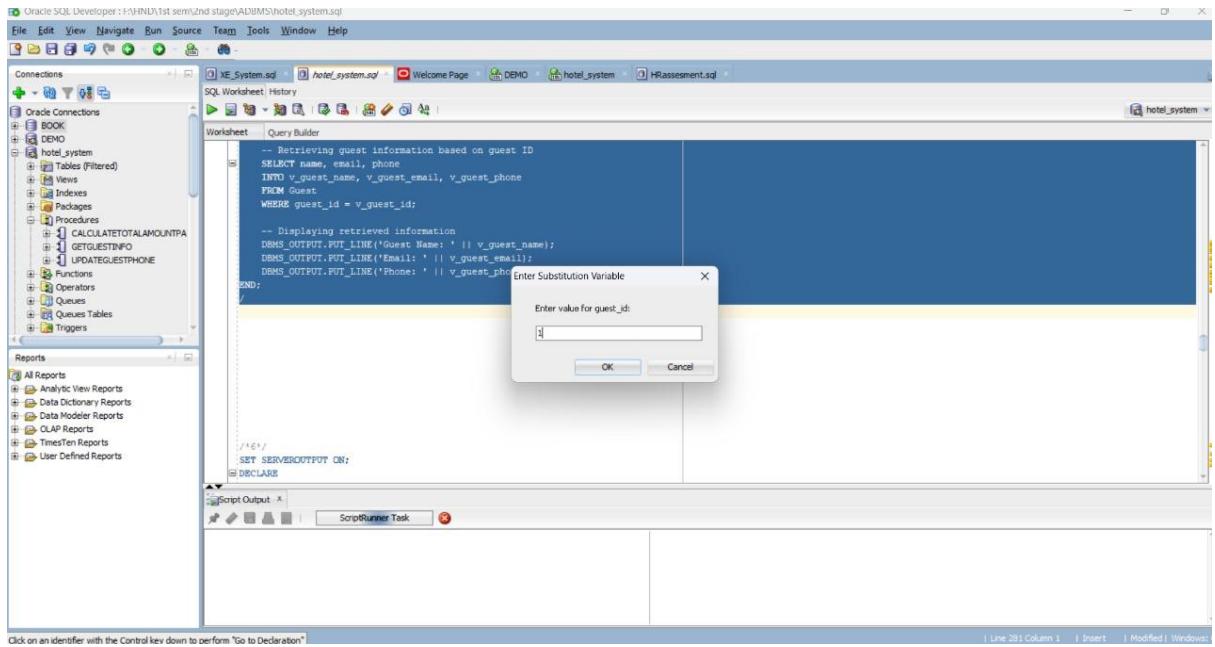


Figure 40:User Given Inputs

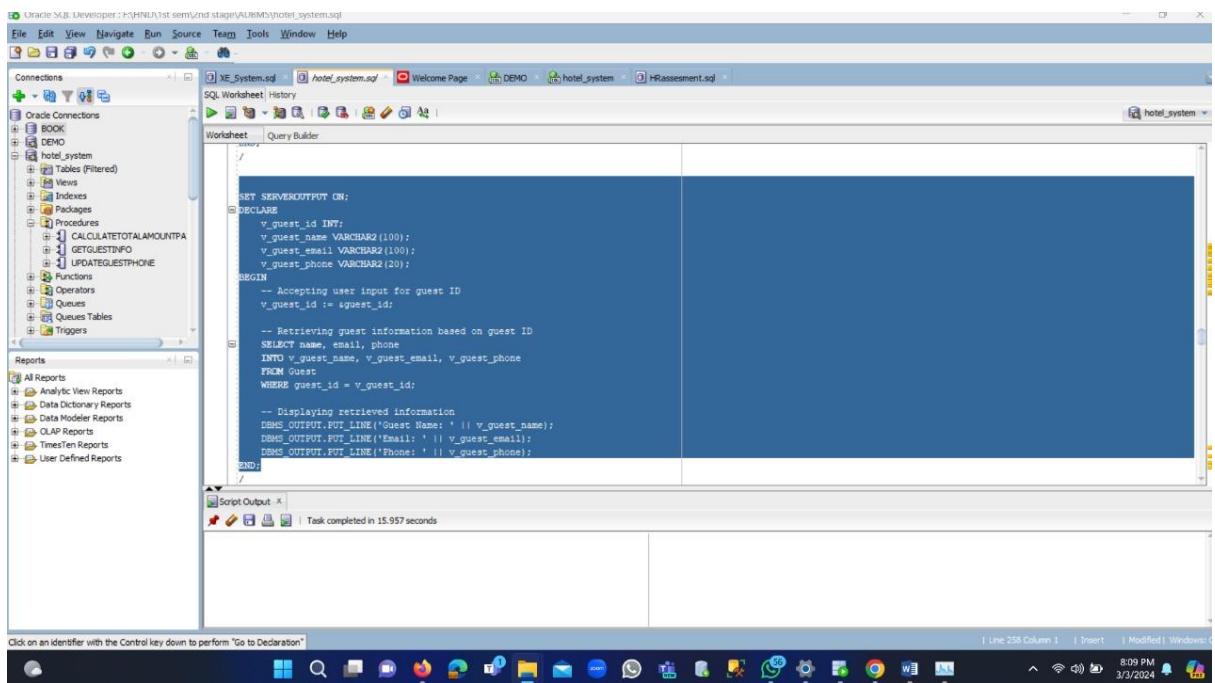


Figure 41:User Given Inputs

```

-- Retrieving guest information based on guest ID
SELECT name, email, phone
INTO v_guest_name, v_guest_email, v_guest_phone
FROM Guest
WHERE guest_id = v_guest_id;

-- Displaying retrieved information
DBMS_OUTPUT.PUT_LINE('Guest Name: ' || v_guest_name);
DBMS_OUTPUT.PUT_LINE('Email: ' || v_guest_email);
DBMS_OUTPUT.PUT_LINE('Phone: ' || v_guest_phone);

END;
/
/* */
SET SERVEROUTPUT ON;
DECLARE

```

Script Output:

```

Task completed in 12.193 seconds
DBMS_OUTPUT.PUT_LINE('Phone: ' || v_guest_phone);
END;
Guest Name: John Smith
Email: john@example.com
Phone: 123-456-7890

```

Figure 42:User Given Inputs

This one first input the guest id and information of the guest related to that id is given.it includes guest name,email and phone number.

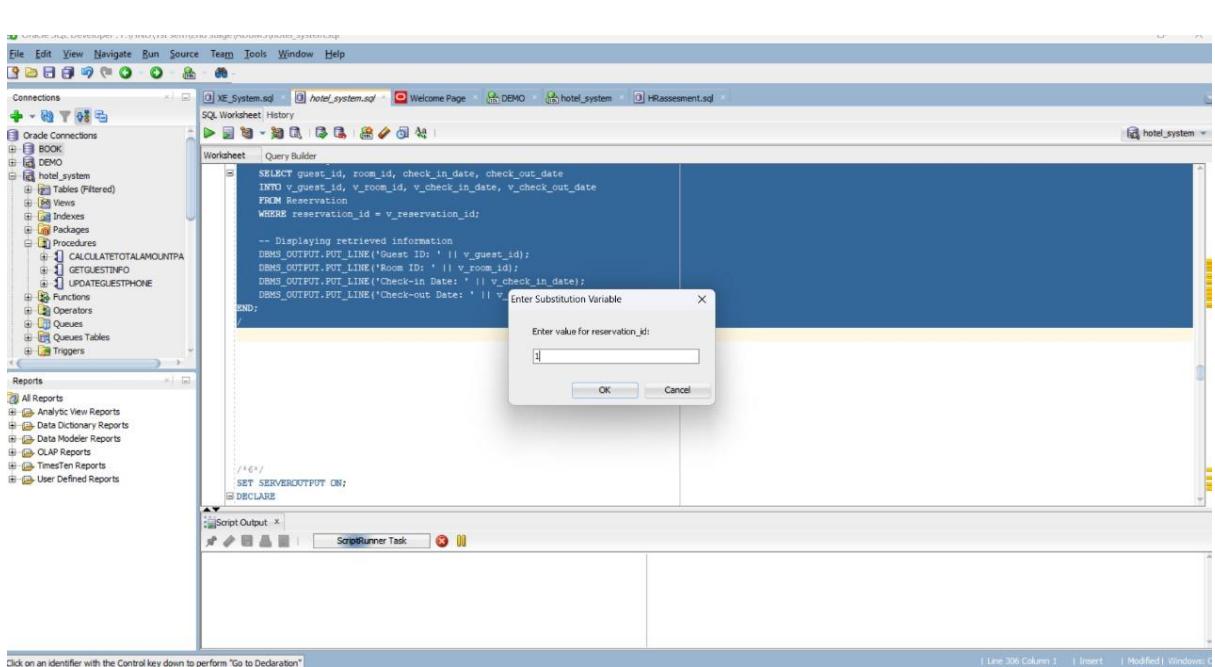


Figure 43:User Given Inputs

The screenshot shows the Oracle SQL Developer interface. The left sidebar displays the database schema with connections to XE\_System, hotel\_system, and HRassessment. The central area is the 'Worksheet' tab where a PL/SQL script is being run. The script retrieves guest information based on a reservation ID and prints it to the DBMS\_OUTPUT. The output window at the bottom shows the results.

```

SET SERVEROUTPUT ON;
DECLARE
    v_reservation_id INT;
    v_guest_id INT;
    v_room_id INT;
    v_check_in_date DATE;
    v_check_out_date DATE;
BEGIN
    -- Accepting user input for reservation ID
    v_reservation_id := &reservation_id;

    -- Retrieving reservation information based on reservation ID
    SELECT guest_id, room_id, check_in_date, check_out_date
    INTO v_guest_id, v_room_id, v_check_in_date, v_check_out_date
    FROM Reservation
    WHERE reservation_id = v_reservation_id;

    -- Displaying retrieved information
    DBMS_OUTPUT.PUT_LINE('Guest ID: ' || v_guest_id);
    DBMS_OUTPUT.PUT_LINE('Room ID: ' || v_room_id);
    DBMS_OUTPUT.PUT_LINE('Check-in Date: ' || v_check_in_date);
    DBMS_OUTPUT.PUT_LINE('Check-out Date: ' || v_check_out_date);
END;

```

Figure 44:User Given Inputs

This screenshot is similar to Figure 44, showing the Oracle SQL Developer interface with the same PL/SQL script. The output window now displays the actual results from the database, including the guest ID, room ID, check-in date (15-MAR-24), and check-out date (20-MAR-24).

```

SELECT guest_id, room_id, check_in_date, check_out_date
INTO v_guest_id, v_room_id, v_check_in_date, v_check_out_date
FROM Reservation
WHERE reservation_id = v_reservation_id;

-- Displaying retrieved information
DBMS_OUTPUT.PUT_LINE('Guest ID: ' || v_guest_id);
DBMS_OUTPUT.PUT_LINE('Room ID: ' || v_room_id);
DBMS_OUTPUT.PUT_LINE('Check-in Date: ' || v_check_in_date);
DBMS_OUTPUT.PUT_LINE('Check-out Date: ' || v_check_out_date);

```

Output:

```

Guest ID: 1
Room ID: 1
Check-in Date: 15-MAR-24
Check-out Date: 20-MAR-24

```

Figure 45:User Given Inputs

Here the booking is done using the relevant reservation id and the details related to the reservation are obtained. From the database, guest id, room id, check in date, check out date are obtained and displayed

The screenshot shows the Oracle SQL Developer interface. The title bar indicates the connection is to 'F:\HND\1st sem\2nd stage\ADBMS\hotel\_system.sql'. The left sidebar shows the 'Connections' tree, which includes 'BOOK', 'DEMO', and 'hotel\_system' (selected). Under 'hotel\_system', there are 'Tables (Filtered)', 'Views', 'Indexes', 'Triggers', and 'Procedures'. The 'Procedures' node is expanded, showing 'CALCULATETOTALAMOUNTPA', 'GETGUESTINFO', and 'UPDATEGUESTPHONE'. The 'Reports' section lists 'All Reports', 'Analytic View Reports', 'Data Dictionary Reports', 'Data Modeler Reports', 'OLAP Reports', 'TimesTen Reports', and 'User Defined Reports'. The main workspace is the 'Worksheet' tab, containing the following PL/SQL code:

```

SET SERVEROUTPUT ON;
DECLARE
  v_payment_id INT;
  v_reservation_id INT;
  v_amount_paid NUMBER(10,2);
  v_payment_date DATE;
BEGIN
  -- Accepting user input for payment ID
  v_payment_id := spayment_id;

  -- Retrieving payment information based on payment ID
  SELECT reservation_id, amount_paid, payment_date
  INTO v_reservation_id, v_amount_paid, v_payment_date
  FROM Payment
  WHERE payment_id = v_payment_id;

  -- Displaying retrieved information
  DBMS_OUTPUT.PUT_LINE('Reservation Id: ' || v_reservation_id);
  DBMS_OUTPUT.PUT_LINE('Amount Paid: ' || v_amount_paid);
  DBMS_OUTPUT.PUT_LINE('Payment Date: ' || v_payment_date);
END;

```

The 'Script Output' pane at the bottom shows the results of the execution:

```

Task completed in 13.17 seconds

```

Figure 46:User Given Inputs

This screenshot is identical to Figure 46, showing the Oracle SQL Developer interface with the same PL/SQL script in the Worksheet pane. The 'Script Output' pane now displays the results of the executed code:

```

Task completed in 16.63 seconds

```

The output details the retrieved payment information:

```

ID: 1
Reservation ID: 1
Amount Paid: 500
Payment Date: 18-MAR-24

```

Figure 47:User Given Inputs

The screenshot shows the Oracle SQL Developer interface. In the central workspace, there is a PL/SQL procedure named `getPaymentInfo`. A modal dialog box titled "Enter Substitution Variable" is open, prompting for a value to be entered into the `v_payment_id` placeholder. The code in the procedure retrieves payment information based on the provided payment ID and displays it using DBMS\_OUTPUT.PUT\_LINE.

```

SET SERVEROUTPUT ON;
DECLARE
    v_payment_id INT;
    v_reservation_id INT;
    v_amount_paid NUMBER(10,2);
    v_payment_date DATE;
BEGIN
    -- Accepting user input for payment ID
    v_payment_id := spayment_id;

    -- Retrieving payment information based on payment ID
    SELECT reservation_id, amount_paid, payment_date
    INTO v_reservation_id, v_amount_paid, v_payment_date
    FROM Payment
    WHERE payment_id = v_payment_id;

    -- Displaying retrieved information
    DBMS_OUTPUT.PUT_LINE('Reservation ID: ' || v_reservation_id);
    DBMS_OUTPUT.PUT_LINE('Amount Paid: ' || v_amount_paid);
    DBMS_OUTPUT.PUT_LINE('Payment Date: ' || v_payment_date);
END;

```

Figure 48:User Given Inputs

This code purpose is payment details based on the provided payment id.it include reservation id,amount paid and payment date from the database and display them.

- .....
- 6) Write PL/SQL programs to cover following control structures. • If/Else • Case • For Loop • While Loop

- If/Else

The screenshot shows the Oracle SQL Developer interface. In the central workspace, there is a PL/SQL procedure named `welcomeToHotel`. It checks if the number of guests is greater than zero and prints a welcome message or a message indicating no guests have arrived yet. The script output window shows the execution completed successfully.

```

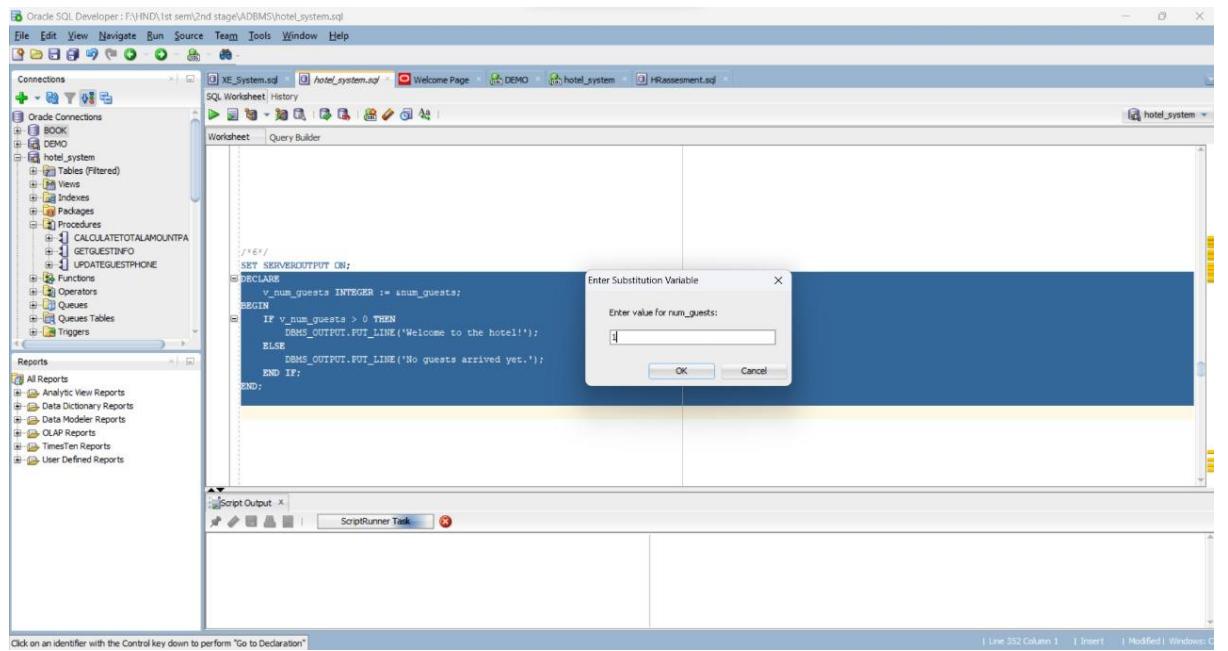
SET SERVEROUTPUT ON;
DECLARE
    v_num_guests INTEGER := &num_guests;
BEGIN
    IF v_num_guests > 0 THEN
        DBMS_OUTPUT.PUT_LINE('Welcome to the hotel!');
    ELSE
        DBMS_OUTPUT.PUT_LINE('No guests arrived yet.');
    END IF;
END;

```

Script Output X | Task completed in 24.164 seconds  
END;  
Welcome to the hotel!  
PL/SQL procedure successfully completed.

Figure 49:IF ELSE

Check the number of visitors and if it is more than 0 then show a welcome message and otherwise show another message.



```

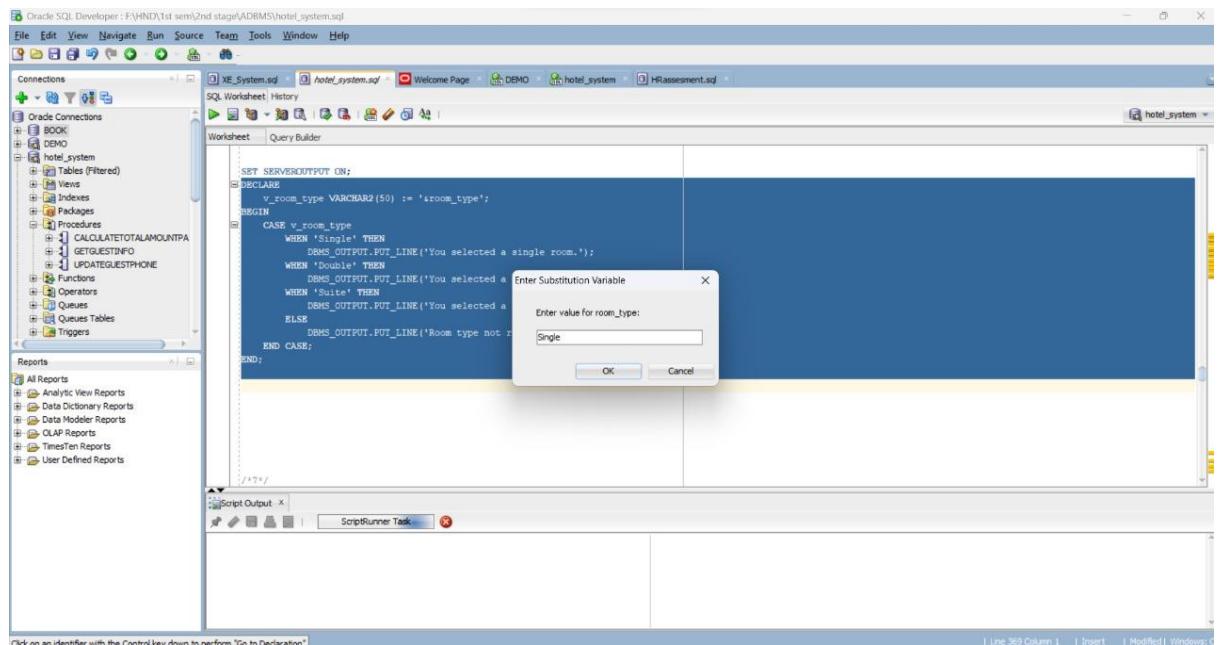
-->/
SET SERVEROUTPUT ON;
DECLARE
    v_num_guests INTEGER := inum_guests;
BEGIN
    IF v_num_guests > 0 THEN
        DBMS_OUTPUT.PUT_LINE('Welcome to the hotel!!');
    ELSE
        DBMS_OUTPUT.PUT_LINE('No guests arrived yet.');
    END IF;
END;

```

The screenshot shows the Oracle SQL Developer interface. On the left, the Connections pane shows a connection to 'hotel\_system'. The central workspace contains a PL/SQL block with an IF-ELSE statement. A modal dialog box titled 'Enter Substitution Variable' is open, prompting for the value of 'num\_guests'. The 'Script Output' pane at the bottom is empty.

Figure 50:IF ELSE

- Case



```

SET SERVEROUTPUT ON;
DECLARE
    v_room_type VARCHAR2(50) := 'room_type';
BEGIN
    CASE v_room_type
        WHEN 'Single' THEN
            DBMS_OUTPUT.PUT_LINE('You selected a single room.');
        WHEN 'Double' THEN
            DBMS_OUTPUT.PUT_LINE('You selected a double room.');
        WHEN 'Suite' THEN
            DBMS_OUTPUT.PUT_LINE('You selected a suite room.');
        ELSE
            DBMS_OUTPUT.PUT_LINE('Room type not specified.');
    END CASE;
END;

```

The screenshot shows the Oracle SQL Developer interface. On the left, the Connections pane shows a connection to 'hotel\_system'. The central workspace contains a PL/SQL block with a CASE statement. A modal dialog box titled 'Enter Substitution Variable' is open, prompting for the value of 'room\_type'. The 'Script Output' pane at the bottom is empty.

Figure 51:Case

```

SET SERVEROUTPUT ON;
DECLARE
    v_room_type VARCHAR2(50) := 'vroom_type';
BEGIN
    CASE v_room_type
        WHEN 'Single' THEN
            DBMS_OUTPUT.PUT_LINE('You selected a single room.');
        WHEN 'Double' THEN
            DBMS_OUTPUT.PUT_LINE('You selected a double room.');
        WHEN 'Suite' THEN
            DBMS_OUTPUT.PUT_LINE('You selected a suite.');
        ELSE
            DBMS_OUTPUT.PUT_LINE('Room type not recognized.');
    END CASE;
END;

```

The screenshot shows the Oracle SQL Developer interface. The left sidebar displays the database schema with the 'hotel\_system' database selected. The central workspace contains a PL/SQL script in the 'Worksheet' tab. The code implements a simple case statement to output a message based on the value of 'v\_room\_type'. The 'Script Output' tab at the bottom shows the execution results: 'Task completed in 26.475 seconds', followed by the message 'You selected a single room.', and a success message 'PL/SQL procedure successfully completed.'

Figure 52:Case

This code purpose is a message indicating the room type is displayed related to what the guest input

- For Loop

```

SET SERVEROUTPUT ON;
DECLARE
    v_max_rooms CONSTANT INTEGER := 5;
BEGIN
    FOR i IN 1..v_max_rooms LOOP
        DBMS_OUTPUT.PUT_LINE('Room ' || i || ': Available');
    END LOOP;
END;

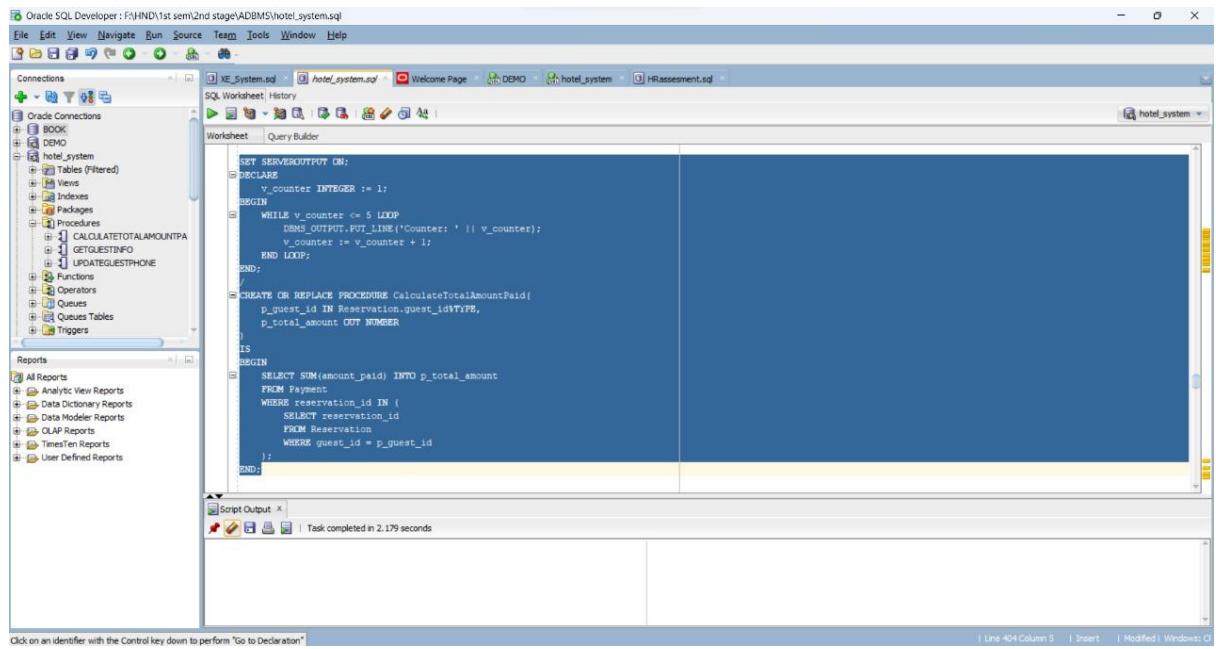
```

The screenshot shows the Oracle SQL Developer interface. The left sidebar displays the database schema with the 'hotel\_system' database selected. The central workspace contains a PL/SQL script in the 'Worksheet' tab. The code uses a for loop to iterate from 1 to 5, printing the message 'Room ' || i || ': Available' for each iteration. The 'Script Output' tab at the bottom shows the execution results: 'Task completed in 0.066 seconds', followed by five lines of output: 'Room 1: Available', 'Room 2: Available', 'Room 3: Available', 'Room 4: Available', and 'Room 5: Available'. A success message 'PL/SQL procedure successfully completed.' is also present.

Figure 53:For Loop

This code purpose the room number indicates their availability. Here a for loop is used and the maximum number of rooms is assumed to be 5.

## While Loop



The screenshot shows the Oracle SQL Developer interface. The left sidebar displays the database schema for the 'hotel\_system' database, including tables like 'BOOK', 'DEMO', and 'RESERVATION'. The central workspace contains a PL/SQL script:

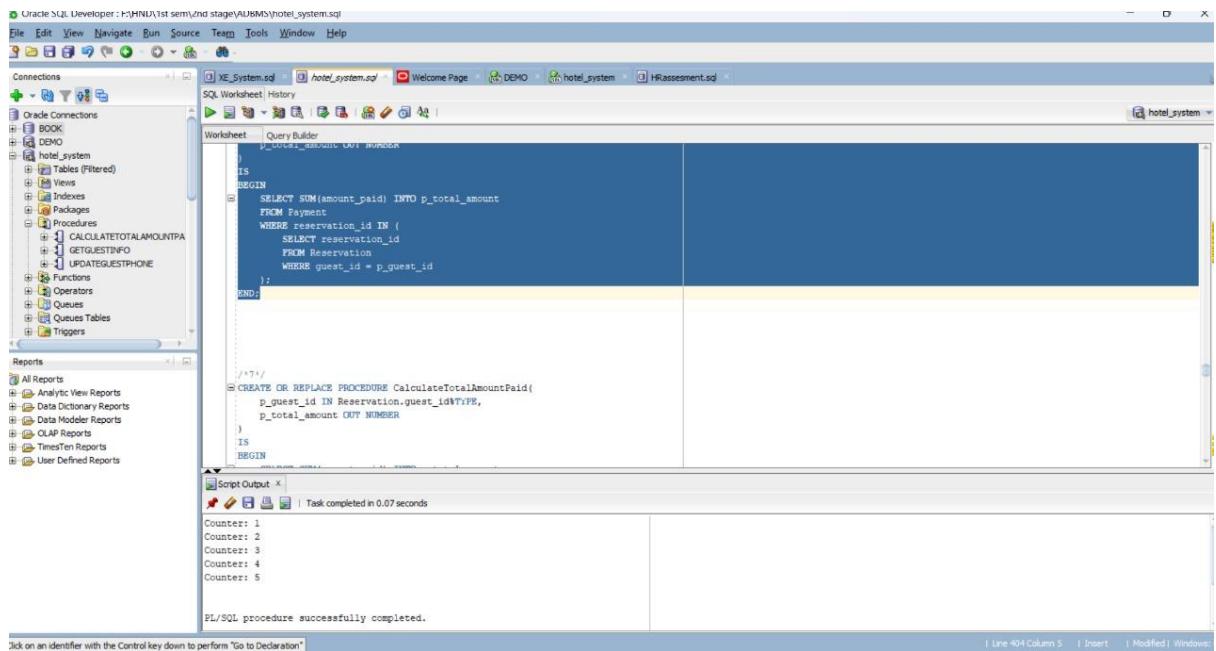
```
SET SERVEROUTPUT ON;
DECLARE
    v_counter INTEGER := 1;
BEGIN
    WHILE v_counter <= 5 LOOP
        DBMS_OUTPUT.PUT_LINE('Counter: ' || v_counter);
        v_counter := v_counter + 1;
    END LOOP;
END;

CREATE OR REPLACE PROCEDURE CalculateTotalAmountPaid(
    p_guest_id IN Reservation.guest_id%TYPE,
    p_total_amount OUT NUMBER
)
IS
BEGIN
    SELECT SUM(amount_paid) INTO p_total_amount
    FROM Payment
    WHERE reservation_id IN (
        SELECT reservation_id
        FROM Reservation
        WHERE guest_id = p_guest_id
    );
END;
```

The 'Script Output' pane at the bottom shows the execution results:

```
Task completed in 2.179 seconds
```

Figure 54:While Loop



This screenshot shows the same Oracle SQL Developer environment after running the procedure. The 'Script Output' pane displays the output of the DBMS\_OUTPUT.PUT\_LINE statements and the final value of the counter:

```
Counter: 1
Counter: 2
Counter: 3
Counter: 4
Counter: 5
PL/SQL procedure successfully completed.
```

Figure 55:While Loop

This code shows a while loop that increments from 1 to 5 and displays the current value of the counter at each iteration.

7) Create three PL/SQL procedures with in and out parameters.

The screenshot shows the Oracle SQL Developer interface with the 'hotel\_system' connection selected. The 'Procedures' node in the left sidebar is expanded, showing several procedures. In the central 'Worksheet' pane, a PL/SQL code editor displays the creation of a stored procedure:

```

CREATE OR REPLACE PROCEDURE GetGuestInfo(
    p_guest_id IN Guest.guest_id%TYPE,
    p_guest_name OUT Guest.name%TYPE,
    p_guest_email OUT Guest.email%TYPE
)
IS
BEGIN
    SELECT name, email INTO p_guest_name, p_guest_email
    FROM Guest
    WHERE guest_id = p_guest_id;
END;

CREATE OR REPLACE FUNCTION NewCalculateTotalAmountPaid(
    p_guest_id IN Reservation.guest_id%TYPE
) RETURN NUMBER

```

The bottom status bar indicates "Procedure GETGUESTINFO compiled".

Figure 56:Create Procedure

### GetGuestInfo

This procedure retrieves the guest name and guest email based on their guest id.get guest id its inout and return the guest name and email as output parameters.

The screenshot shows the Oracle SQL Developer interface with the 'hotel\_system' connection selected. The 'Procedures' node in the left sidebar is expanded, showing several procedures. In the central 'Worksheet' pane, a PL/SQL code editor displays the creation of a stored procedure:

```

CREATE OR REPLACE PROCEDURE UpdateGuestPhone(
    p_guest_id IN Guest.guest_id%TYPE,
    p_new_phone_number IN Guest.phone%TYPE
)
IS
BEGIN
    UPDATE Guest
    SET phone = p_new_phone_number
    WHERE guest_id = p_guest_id;
END;

```

The bottom status bar indicates "Procedure UPDATEGUESTPHONE compiled".

Figure 57:Create Procedure

### UpdateGuestPhone

This procedure update the guest phone number based on their guest id.it take guest id and set the new phone number.new phone number is input parameter and update phone number in guest table

```

CREATE OR REPLACE PROCEDURE CalculateTotalAmountPaid(
    p_guest_id IN Reservation.guest_id%TYPE,
    p_total_amount OUT NUMBER
)
IS
BEGIN
    SELECT SUM(amount_paid) INTO p_total_amount
    FROM Payment
    WHERE reservation_id IN (
        SELECT reservation_id
        FROM Reservation
        WHERE guest_id = p_guest_id
    );
END;

```

Procedure CALCULATETOTALAMOUNTPAID compiled

Figure 58:Create Procedure

### CalculateTotalAmountPaid

This procedure calculate amount paid by a guest is calculated based on his guest id. it takes the guest id as input and res thetital amount paid as an output parameter.

## 7) Create three PL/SQL Function.

```

CREATE OR REPLACE FUNCTION GetGuestName(
    p_guest_id IN Guest.guest_id%TYPE
) RETURN VARCHAR2
IS
    v_guest_name Guest.name%TYPE;
BEGIN
    SELECT name INTO v_guest_name
    FROM Guest
    WHERE guest_id = p_guest_id;
    RETURN v_guest_name;
END;

CREATE OR REPLACE VIEW ReservationDetails AS
SELECT r.reservation_id, r.guest_id, g.name AS guest_name, g.email AS guest_email, r.room_id, r.check_in_date, r.check_out_date
FROM Reservation r
JOIN Guest g ON r.guest_id = g.guest_id;

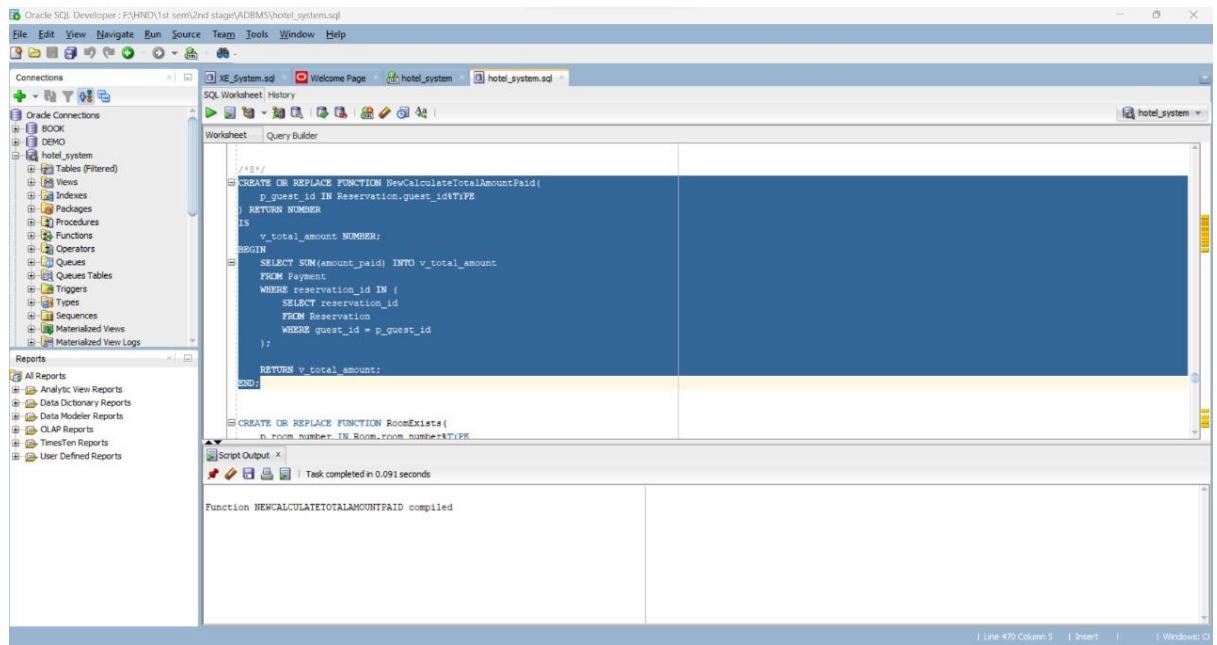
```

Function GETGUESTNAME compiled

Figure 59:Function

### GetGuestName

This function retrieve the guest name is retrieved based on that guest's ID



The screenshot shows the Oracle SQL Developer interface. The left sidebar displays the 'Connections' tree, which includes 'BOOK', 'DEMO', and 'hotel\_system'. The 'hotel\_system' node is expanded, showing 'Tables (Filtered)', 'Views', 'Indexes', 'Packages', 'Procedures', 'Functions', 'Operators', 'Queues', 'Triggers', 'Types', 'Sequences', 'Materialized Views', and 'Materialized View Logs'. The 'Reports' section lists 'All Reports', 'Analytic View Reports', 'Data Dictionary Reports', 'Data Modeler Reports', 'OLAP Reports', 'TimesTen Reports', and 'User Defined Reports'. The central workspace is titled 'XE\_System.sql' and contains the following PL/SQL code:

```
/* */
CREATE OR REPLACE FUNCTION NewCalculateTotalAmountPaid(
    p_guest_id IN Reservation.guest_id%TYPE
) RETURN NUMBER
IS
    v_total_amount NUMBER;
BEGIN
    SELECT SUM(amount_paid) INTO v_total_amount
    FROM Reservation
    WHERE reservation_id IN (
        SELECT reservation_id
        FROM Reservation
        WHERE guest_id = p_guest_id
    );
    RETURN v_total_amount;
END;

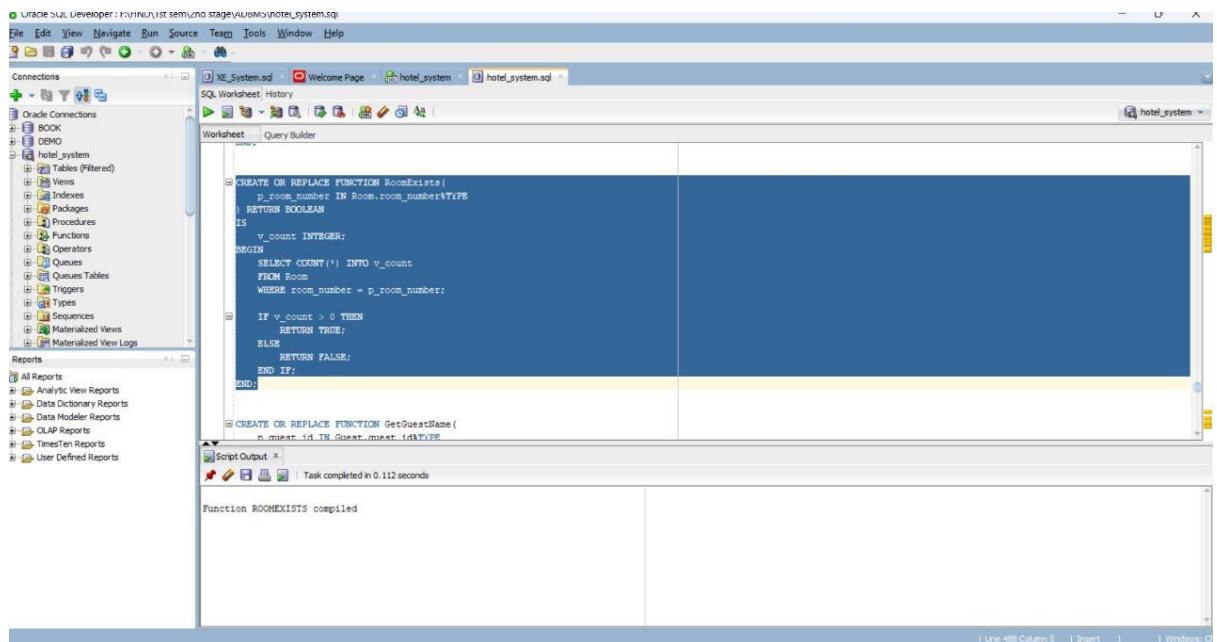
CREATE OR REPLACE FUNCTION RoomExists(
    p_room_number IN Room.room_number%TYPE
) RETURN BOOLEAN
IS
    v_count INTEGER;
BEGIN
    SELECT COUNT(*) INTO v_count
    FROM Room
    WHERE room_number = p_room_number;
    IF v_count > 0 THEN
        RETURN TRUE;
    ELSE
        RETURN FALSE;
    END IF;
END;
```

The 'Script Output' pane at the bottom shows the message: 'Function NEWCALCULATETOTALAMOUNTPAID compiled'.

Figure 60:Function

### NewCalculateTotalAmountPaid

This function calculate total amount paid by a guest is calculated based on his guest id.



The screenshot shows the Oracle SQL Developer interface. The left sidebar displays the 'Connections' tree, which includes 'BOOK', 'DEMO', and 'hotel\_system'. The 'hotel\_system' node is expanded, showing 'Tables (Filtered)', 'Views', 'Indexes', 'Packages', 'Procedures', 'Functions', 'Operators', 'Queues', 'Triggers', 'Types', 'Sequences', 'Materialized Views', and 'Materialized View Logs'. The 'Reports' section lists 'All Reports', 'Analytic View Reports', 'Data Dictionary Reports', 'Data Modeler Reports', 'OLAP Reports', 'TimesTen Reports', and 'User Defined Reports'. The central workspace is titled 'XE\_System.sql' and contains the following PL/SQL code:

```
CREATE OR REPLACE FUNCTION RoomExists(
    p_room_number IN Room.room_number%TYPE
) RETURN BOOLEAN
IS
    v_count INTEGER;
BEGIN
    SELECT COUNT(*) INTO v_count
    FROM Room
    WHERE room_number = p_room_number;
    IF v_count > 0 THEN
        RETURN TRUE;
    ELSE
        RETURN FALSE;
    END IF;
END;

CREATE OR REPLACE FUNCTION GetGuestName(
    p_guest_id IN Guest.guest_id%TYPE
) RETURN VARCHAR2
IS
    v_name VARCHAR2(100);
BEGIN
    SELECT first_name || ' ' || last_name INTO v_name
    FROM Guest
    WHERE guest_id = p_guest_id;
    RETURN v_name;
END;
```

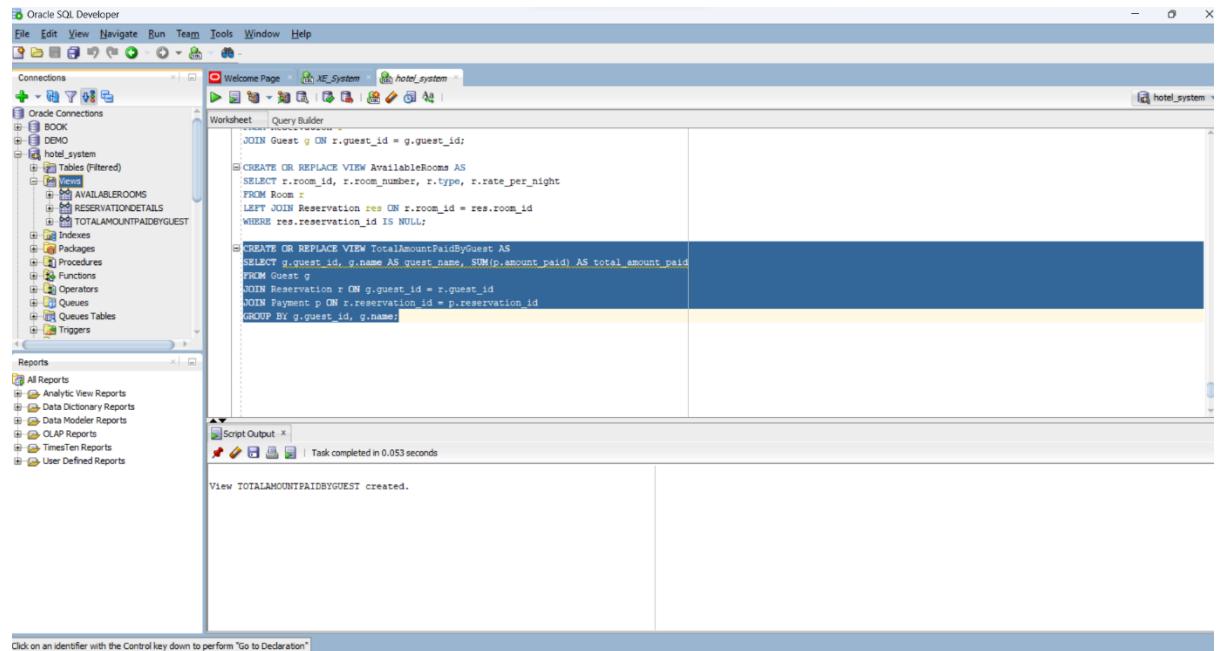
The 'Script Output' pane at the bottom shows the message: 'Function ROOMEXISTS compiled'.

Figure 61:Function

## RoomExists

This function Checks for a room with a unique room number and Returns true if present and false otherwise

### 8) Create three PL/SQL view.



The screenshot shows the Oracle SQL Developer interface. The left sidebar displays the database schema with connections, tables, and other objects. The central workspace contains a query builder window with the following PL/SQL code:

```
JOIN Guest g ON r.guest_id = g.guest_id;

CREATE OR REPLACE VIEW AvailableRooms AS
SELECT r.room_id, r.room_number, r.type, r.rate_per_night
FROM Room r
LEFT JOIN Reservation res ON r.room_id = res.room_id
WHERE res.reservation_id IS NULL;

CREATE OR REPLACE VIEW TotalAmountPaidByGuest AS
SELECT g.guest_id, g.name AS guest_name, SUM(p.amount_paid) AS total_amount_paid
FROM Guest g
LEFT JOIN Reservation r ON g.guest_id = r.guest_id
JOIN Payment p ON r.reservation_id = p.reservation_id
GROUP BY g.guest_id, g.name;
```

The bottom status bar indicates "View TOTALAMOUNTPAIDBYGUEST created." and "Task completed in 0.053 seconds".

Figure 62:View

This view shows us summary of the amount paid by each visitor is displayed . it includes guest id,guest name and the total amount they have paid across all their reservation

The screenshot shows the Oracle SQL Developer interface with the 'hotel\_system' connection selected. The 'Worksheet' tab is active, displaying the SQL code for creating the 'AvailableRooms' view. The code uses a LEFT JOIN between the 'Room' and 'Reservation' tables to filter out rooms that have no associated reservations. The 'Script Output' pane at the bottom shows the message 'View AVAILABLEROOMS created.' and a completion time of 0.049 seconds.

```

CREATE OR REPLACE VIEW AvailableRooms AS
SELECT r.room_id, r.room_number, r.type, r.rate_per_night
FROM Room r
LEFT JOIN Reservation res ON r.room_id = res.room_id
WHERE res.reservation_id IS NULL;

```

Figure 63:View

This view retrieves information about rooms that are currently available for reservation. It selects columns such as room ID, room number, room type, and rate per night from the "Room" table. It utilizes a LEFT JOIN operation with the "Reservation" table on the room ID, filtering out rooms that have no associated reservations.

The screenshot shows the Oracle SQL Developer interface with the 'hotel\_system' connection selected. The 'Worksheet' tab is active, displaying the SQL code for creating the 'ReservationDetails' view. This view joins the 'Reservation' and 'Guest' tables based on guest ID to retrieve guest details along with reservation information. The 'Script Output' pane at the bottom shows the message 'View RESERVATIONDETAILS created.' and a completion time of 0.059 seconds.

```

CREATE OR REPLACE VIEW ReservationDetails AS
SELECT r.reservation_id, r.guest_id, g.name AS guest_name, g.email AS guest_email, r.room_id, r.check_in_date, r.check_out_date
FROM Reservation r
JOIN Guest g ON r.guest_id = g.guest_id;

CREATE OR REPLACE VIEW AvailableRooms AS
SELECT r.room_id, r.room_number, r.type, r.rate_per_night
FROM Room r
LEFT JOIN Reservation res ON r.room_id = res.room_id
WHERE res.reservation_id IS NULL;

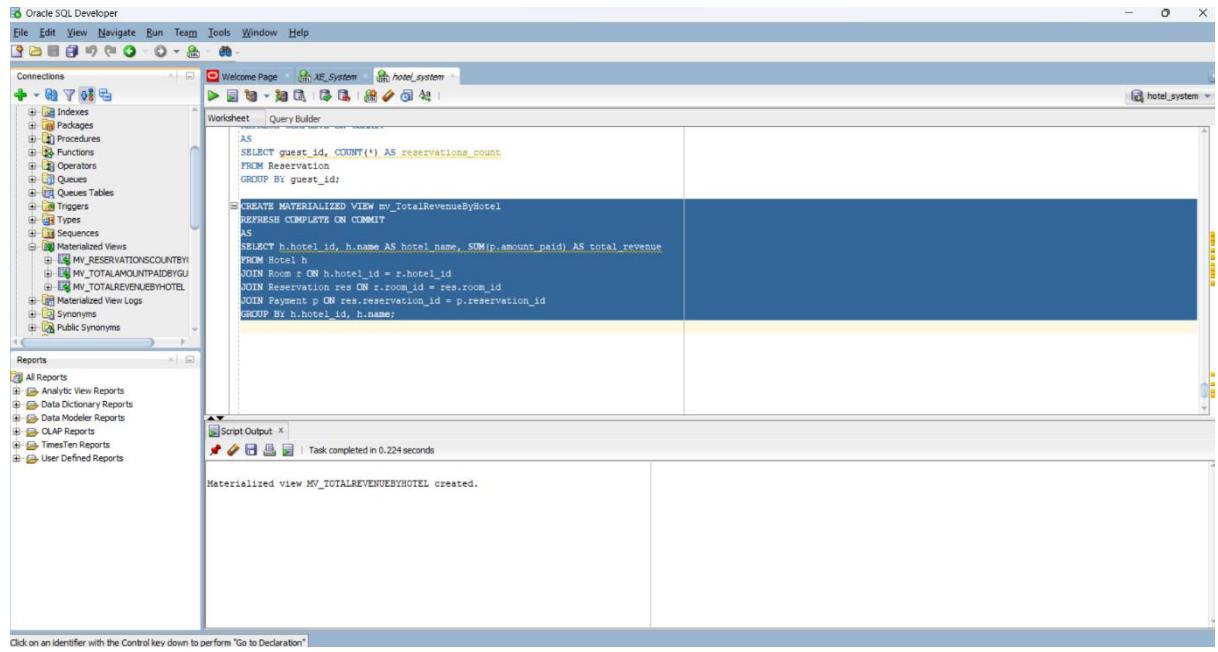
CREATE OR REPLACE VIEW TotalAmountPaidByGuest AS
SELECT g.guest_id, g.name AS guest_name, SUM(p.amount_paid) AS total_amount_paid
FROM Guest g
JOIN Reservation r ON g.guest_id = r.guest_id
JOIN Payment p ON r.reservation_id = p.reservation_id
GROUP BY g.guest_id, g.name;

```

Figure 64:View

This Oracle SQL code creates or replaces a view named "ReservationDetails". This view retrieves reservation details along with guest information. It includes columns such as reservation ID, guest ID, guest name, guest email, room ID, check-in date, and check-out date. The view is created by joining the "Reservation" table with the "Guest" table based on the guest ID.

9) Create three PL/SQL materialized view.



The screenshot shows the Oracle SQL Developer interface. The left sidebar displays the database schema with various objects like Indexes, Procedures, Triggers, and Materialized Views. The central workspace contains a query editor window titled 'Worksheet' with the following PL/SQL code:

```
AS
SELECT guest_id, COUNT(*) AS reservations_count
FROM Reservation
GROUP BY guest_id;

CREATE MATERIALIZED VIEW mv_TotalRevenueByHotel
REFRESH COMPLETE ON COMMIT
AS
SELECT h.hotel_id, h.name AS hotel_name, SUM(p.amount_paid) AS total_revenue
FROM Hotel h
JOIN Room r ON h.hotel_id = r.hotel_id
JOIN Reservation res ON r.room_id = res.room_id
JOIN Payment p ON res.reservation_id = p.reservation_id
GROUP BY h.hotel_id, h.name;
```

The bottom status bar indicates: 'Materialized view MV\_TOTALREVENUEBYHOTEL created.' and 'Task completed in 0.224 seconds.'

Figure 65:materialized view

**mv\_TotalRevenueByHotel**

Stores the total revenue earned by each hotel by summing up payments for hotel rooms

```

CREATE MATERIALIZED VIEW MV_RESERVATIONSCOUNTBYGUEST
REFRESH COMPLETE ON COMMIT
AS
SELECT g.guest_id, COUNT(*) AS reservations_count
FROM Reservation r
JOIN Payment p ON r.reservation_id = p.reservation_id
GROUP BY g.guest_id;

CREATE MATERIALIZED VIEW MV_TotalRevenueByHotel
REFRESH COMPLETE ON COMMIT
AS
SELECT h.hotel_id, h.name AS hotel_name, SUM(p.amount_paid) AS total_revenue
FROM Hotel h
JOIN Room r ON h.hotel_id = r.hotel_id
JOIN Reservation res ON r.room_id = res.room_id
JOIN Payment p ON res.reservation_id = p.reservation_id
GROUP BY h.hotel_id, h.name;

```

Materialized view MV\_RESERVATIONSCOUNTBYGUEST created.

Figure 66:materialized view

### mv\_ReservationCounitByGuest

Guest store their respective reservation count here

```

CREATE MATERIALIZED VIEW MV_TOTALAMOUNTPAIDBYGUEST
REFRESH COMPLETE ON COMMIT
AS
SELECT g.guest_id, g.name AS guest_name, SUM(p.amount_paid) AS total_amount_paid
FROM Guest g
JOIN Reservation r ON g.guest_id = r.guest_id
JOIN Payment p ON r.reservation_id = p.reservation_id
GROUP BY g.guest_id, g.name;

CREATE MATERIALIZED VIEW MV_ReservationsCountByGuest
REFRESH COMPLETE ON COMMIT
AS
SELECT guest_id, COUNT(*) AS reservations_count
FROM Reservation
GROUP BY guest_id;

CREATE MATERIALIZED VIEW MV_TotalRevenueByHotel
REFRESH COMPLETE ON COMMIT
AS

```

Materialized view MV\_TOTALAMOUNTPAIDBYGUEST created.

Figure 67:materialized view

### mv\_TotalAmountPaidByGuest

The purpose of This shows the total amount paid for each guest by summing up the payments related to their reservation