AMERICAN INTERNATIONAL UNIVERSITY-BANGLADESH

408/1, Kuratoli, Khilkhet, Dhaka 1229, Bangladesh

Assignment Title: **Hotel Room Management**

Date of Submission:13/05/23

Course Title: INTRODUCTION TO DATABASE

Sec=on:O

Semester: Spring 2022-23 Course Teacher: DR. MOHAMMAD RABIUL ISLAM

Declaration and Statement of Authorship:

1. I/we hold a copy of this Assignment/Case-Study, which can be produced if the original is lost/damaged.
2. This Assignment/Case-Study is my/our original work and no part of it has been copied from any other student's work or from any other source except where due acknowledgement is made.
3. No part of this Assignment/Case-Study has been written for me/us by any other person except where such collaborationhas been authorized by the concerned teacher and is clearly acknowledged in the assignment.
4. I/we have not previously submitted or currently submitting this work for any other course/unit.
5. This work may be reproduced, communicated, compared and archived for the purpose of detecting plagiarism.
6. I/we give permission for a copy of my/our marked work to be retained by the Faculty for review and comparison, including review by external examiners.
7. I/we understand thatPlagiarism is the presentation of the work, idea or creation of another person as though it is your own. It is a formofcheatingandisaveryseriousacademicoffencethatmayleadtoexpulsionfromtheUniversity. Plagiarized material can be drawn from, and presented in, written, graphic and visual form, including electronic data, and oral presentations. Plagiarism occurs when the origin of them arterial used is not appropriately cited.
8. I/we also understand that enabling plagiarism is the act of assisting or allowing another person to plagiarize or to copy my/our work.

Group Name / No.: Blue

Group Members	ID	Details of Contributions
Najib Mahfuj	22-48248-2	Introduction, Scenario Description,Normalization, Er diagram, Schema Diagram ,Relation Algebra, Conclusion
MD. Monjurul Haque Fahat	22-47348-2	Data Insertion, Relation Algebra, Scenario Description, Conclusion.
MD. Mahmudul Haque Rahat	22-47347-2	Table Creation, Relation algebra, Scenario Description, Conclusion.
Kangkhita, Jerin Jaman	22-47794-2.	Query writing, conclusion, Scenario Description, Relation algebra.

Faculty use only		
FACULTYCOMMENTS	**Marks Obtained**	
	Total Marks	

Marking Criteria

Topic Discussion	Page Number	Marks Distribution
Content	3	
Introduction	4	
Scenario Description	4	
Er diagram	5	
Normalization	6-8	
Schema Diagram	9	
Table creation	10-23	
Data Insertion	24-34	
Query Writing	35-39	
Relation Algebra	40-44	
Conclusion	45	

Project name: Hotel management System

Tables name:

1. Guest
2. Reservation
3. Room
4. Room Type
5. Room Status
6. Check-In
7. Check-Out
8. Billing
9. Payment Method
10. Payment Status
11. Employee
12. Department
13. Position
14. Shift
15. Schedule
16. Service
17. Service Type
18. Menu
19. Orders
20. Feedback

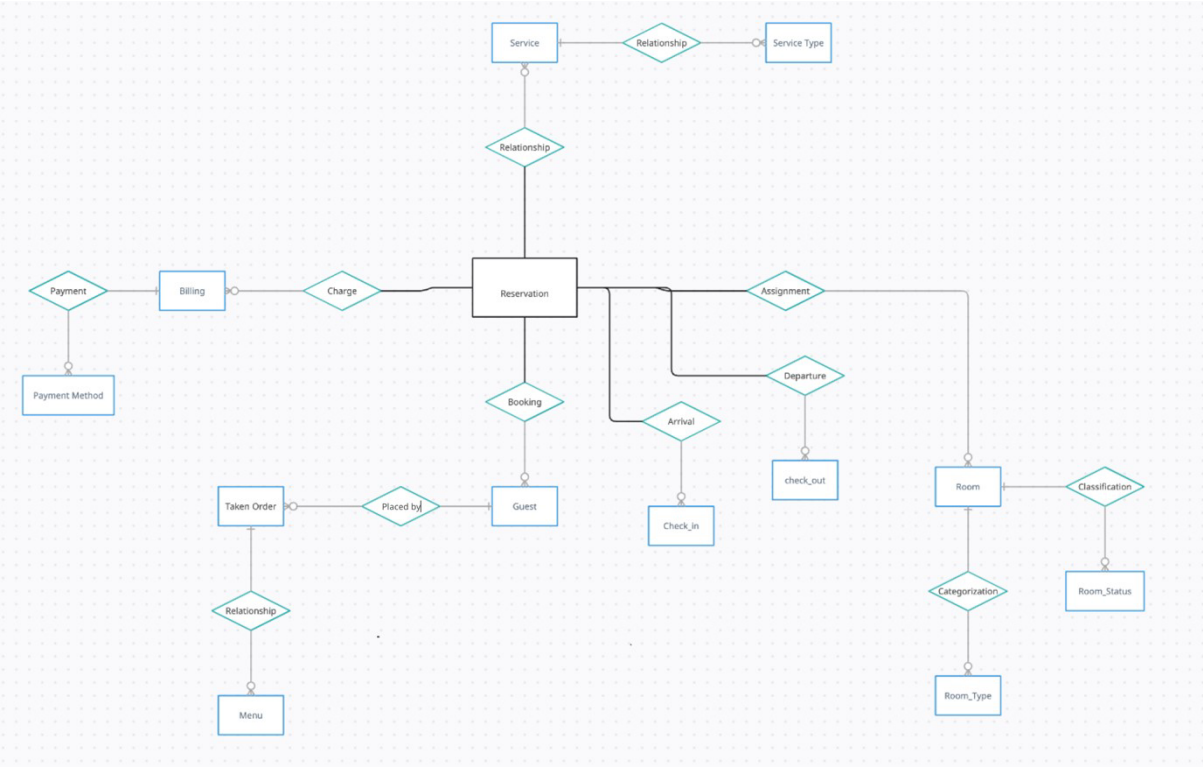
Introduction

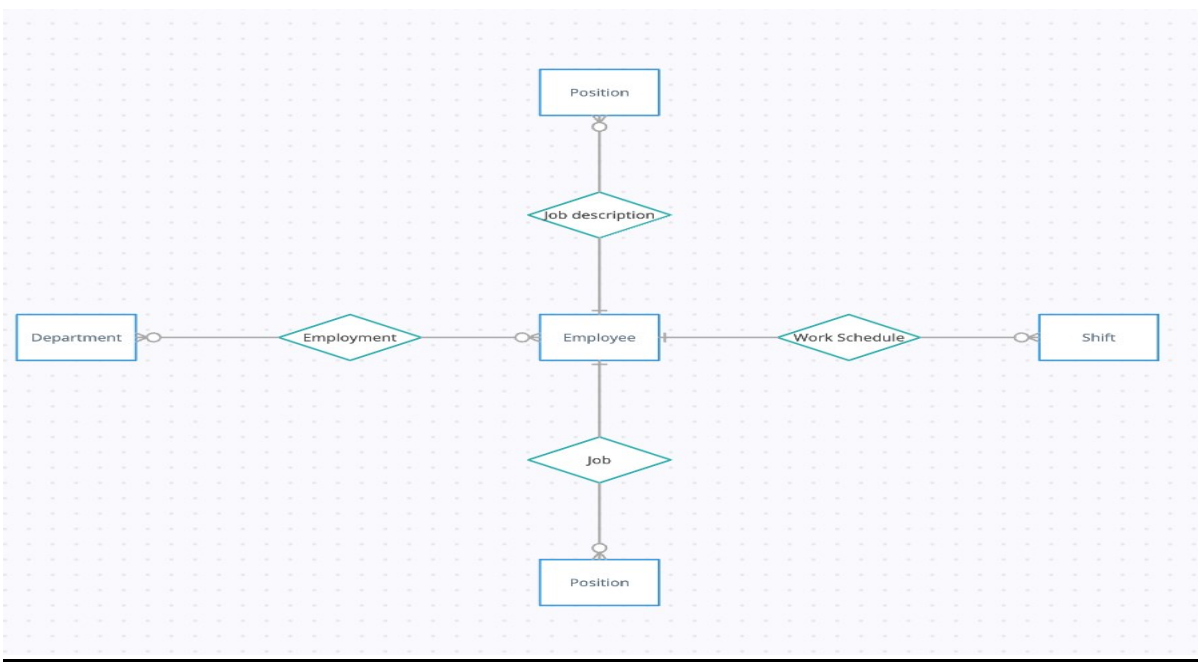
This SQL project is a hotel management system designed to streamline and automate the day-to-day operations of a hotel. The project consists of multiple SQL tables that store information about guests, reservations, rooms, employees, billing, payment methods, services, menus, orders, and feedback. The system provides an efficient way to manage guest bookings, track room availability, handle employee scheduling and payment processing, and offer additional services to guests. By using SQL to manage data, the project ensures the reliability and accuracy of the information, which is critical for providing excellent customer service and running a successful hotel.

Scenario Description:

The scenario for this hotel management system is a typical hotel operation where guests make reservations and stay in rooms, use additional services such as food and beverage, laundry, and transportation, and pay for their stay and services using different payment methods. The hotel staff manages guest bookings, assigns rooms, processes check-ins and check-outs, handles billing and payments, schedules employees, manages service orders, and collects feedback from guests. The system provides a comprehensive solution to manage all these tasks in an efficient and effective manner, ensuring a smooth and pleasant experience for both guests and staff.

Er Diagram


Service
Relationship
Service Type
Relationship
Payment
Billing
Charge
Reservation
Assignment
Payment Method
Booking
Departure
Arrival
check_out
Taken Order
Placed by
Guest
Check_in
Room
Classification
Relationship
Categorization
Room_Status
Menu
Room_Type


Position
Job description
Department
Employment
Employee
Work Schedule
Shift
Job
Position

NORMALIZATION

Relation name: Hotel Room Management

All attributes: ROOM_NO (pk), ROOM_TYPE, ROOM_PRICE, BOOKING_STATUS, CHECK_IN_DATE, CHECK_OUT_DATE.

1NF: No multivalued attributes.

2NF: CHECK_IN_DATE and CHECK_OUT_DATE are partially dependent on ROOM_NO, BOOKING_STATUS because they depend only on a specific booking of a room. So, we can create a new table to normalize.

Room: ROOM_NO (pk), ROOM_TYPE, ROOM_PRICE.

Booking: BOOKING_ID (pk), ROOM_NO (fk), BOOKING_STATUS.

Booking_Details: BOOKING_ID (pk, fk), CHECK_IN_DATE, CHECK_OUT_DATE.

3NF: No transitive dependencies here.

So, we obtain from Hotel Room Management,

Room: ROOM_NO (pk), ROOM_TYPE, ROOM_PRICE.

Booking: BOOKING_ID (pk), ROOM_NO (fk), BOOKING_STATUS.

Booking_Details: BOOKING_ID (pk, fk), CHECK_IN_DATE, CHECK_OUT_DATE.

Relation name: Hotel Employee Management

All attributes: EMPLOYEE_ID (pk), EMPLOYEE_FIRST_NAME, EMPLOYEE_LAST_NAME, EMPLOYEE_EMAIL, EMPLOYEE_PHONE, DEPARTMENT_NAME, POSITION_NAME, SHIFT_NAME

1NF: No multivalued attributes.

2NF: DEPARTMENT_NAME and POSITION_NAME are dependent only on EMPLOYEE_ID, so we can create a new table to normalize.

Employee: EMPLOYEE_ID (pk), EMPLOYEE_FIRST_NAME, EMPLOYEE_LAST_NAME, EMPLOYEE_EMAIL, EMPLOYEE_PHONE

Position: POSITION_NAME (pk), DEPARTMENT_NAME (fk)

Shift: SHIFT_NAME (pk)

Employee_Position: EMPLOYEE_ID (fk), POSITION_NAME (fk)

Employee_Shift: EMPLOYEE_ID (fk), SHIFT_NAME (fk)

3NF: No transitive dependencies here.

So, we obtain from Hotel Employee Management:

Employee: EMPLOYEE_ID (pk), EMPLOYEE_FIRST_NAME, EMPLOYEE_LAST_NAME, EMPLOYEE_EMAIL, EMPLOYEE_PHONE

Position: POSITION_NAME (pk), DEPARTMENT_NAME (fk)

Shift: SHIFT_NAME (pk)

Employee_Position: EMPLOYEE_ID (fk), POSITION_NAME (fk)

Employee_Shift: EMPLOYEE_ID (fk), SHIFT_NAME (fk)

Relation name: Order Management

All attributes: ORDER_ID (pk), ITEM_ID (pk, fk), MENU_NAME, ITEM_DESCRIPTION, ITEM_PRICE, ITEM_QUANTITY.

1NF: No multivalued attributes.

2NF: MENU_NAME, ITEM_DESCRIPTION, ITEM_PRICE are dependent only on ITEM_ID, so we can create a new table to normalize.

Menu: ITEM_ID (pk), MENU_NAME, ITEM_DESCRIPTION, ITEM_PRICE.

Order_Item: ORDER_ID (pk), ITEM_ID (pk, fk), ITEM_QUANTITY.

3NF: No transitive dependencies here.

So, we obtain from Order:

Menu: ITEM_ID (pk), MENU_NAME, ITEM_DESCRIPTION, ITEM_PRICE.

Order_Item: ORDER_ID (pk), ITEM_ID (pk, fk), ITEM_QUANTITY.

Schema Diagram

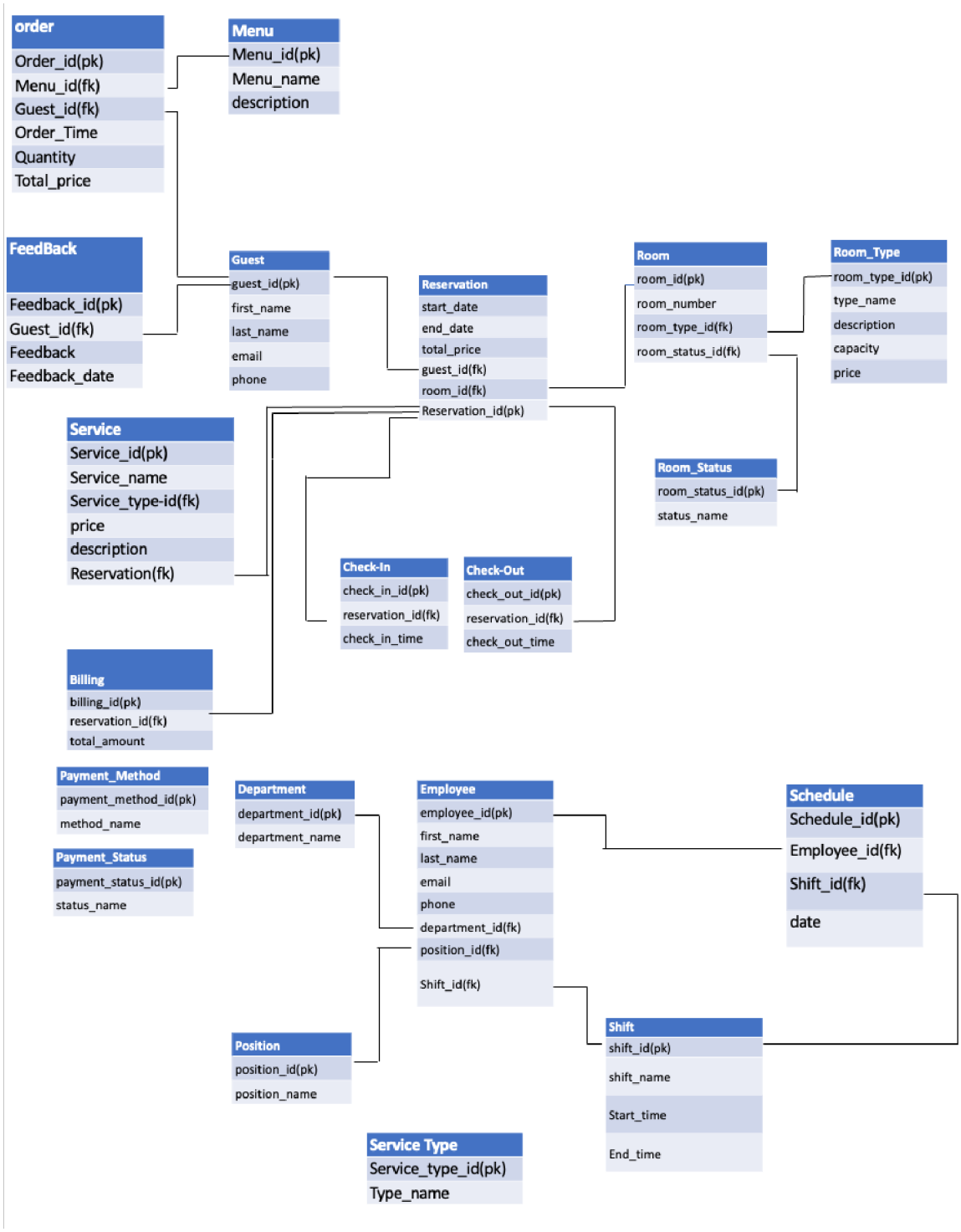


Table Creation

Guest:

```
CREATE TABLE Guest (
  guest_id INT NOT NULL ,
  first_name VARCHAR(50) NOT NULL,
  last_name VARCHAR(50) NOT NULL,
  email VARCHAR(100),
  phone VARCHAR(20),
  PRIMARY KEY (guest_id)
);
create sequence guest_id minvalue 1 start with 1 cache 1000;
DESC Guest
```

Results Explain Describe Saved SQL History

Object Type **TABLE** Object **GUEST**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
GUEST	GUEST_ID	Number	-	-	0	1	-	-	-
	FIRST_NAME	Varchar2	50	-	-	-	-	-	-
	LAST_NAME	Varchar2	50	-	-	-	-	-	-
	EMAIL	Varchar2	100	-	-	-	✓	-	-
	PHONE	Varchar2	20	-	-	-	✓	-	-

1 - 5

Reservation:

```
CREATE TABLE Reservation (
  reservation_id INT NOT NULL,
  guest_id INT NOT NULL,
  room_id INT NOT NULL,
  start_date DATE NOT NULL,
  end_date DATE NOT NULL,
  total_price DECIMAL(10,2),
  PRIMARY KEY (reservation_id),
  FOREIGN KEY (guest_id) REFERENCES Guest(guest_id),
  FOREIGN KEY (room_id) REFERENCES Room(room_id)
);

CREATE SEQUENCE   reservation_id_seq
    START WITH 1
    INCREMENT BY 1
    NOMAXVALUE
    NOCYCLE
    NOCACHE;
DESC Reservation
```

Results Explain **Describe** Saved SQL History

Object Type **TABLE** Object **RESERVATION**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
RESERVATION	RESERVATION_ID	Number	-	-	0	1	-	-	-
	GUEST_ID	Number	-	-	0	-	-	-	-
	ROOM_ID	Number	-	-	0	-	-	-	-
	START_DATE	Date	7	-	-	-	-	-	-
	END_DATE	Date	7	-	-	-	-	-	-
	TOTAL_PRICE	Number	-	10	2	-	✓	-	-

1 - 6

Room:

```
CREATE TABLE Room (
  room_id INT NOT NULL,
  room_number VARCHAR(10) NOT NULL,
  room_type_id INT NOT NULL,
  room_status_id INT NOT NULL,
  PRIMARY KEY (room_id),
  FOREIGN KEY (room_type_id) REFERENCES Room_Type(room_type_id),
  FOREIGN KEY (room_status_id) REFERENCES Room_Status(room_status_id)
);


CREATE SEQUENCE room_id_seq
    START WITH 1
    INCREMENT BY 1
    NOMAXVALUE
    NOCYCLE
    NOCACHE;


DESC Room
```

Results Explain **Describe** Saved SQL History

Object Type **TABLE** Object **ROOM**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
ROOM	ROOM_ID	Number	-	-	0	1	-	-	-
	ROOM_NUMBER	Varchar2	10	-	-	-	-	-	-
	ROOM_TYPE_ID	Number	-	-	0	-	-	-	-
	ROOM_STATUS_ID	Number	-	-	0	-	-	-	-

1 - 4

ROOM_TYPE:

```
CREATE TABLE Room_Type (
  room_type_id INT NOT NULL ,
  type_name VARCHAR(50) NOT NULL,
  description TEXT,
  capacity INT NOT NULL,
  price DECIMAL(10,2) NOT NULL,
  PRIMARY KEY (room_type_id)
);


CREATE SEQUENCE     room_type_id_seq
    START WITH 1
    INCREMENT BY 1
    NOMAXVALUE
    NOCYCLE
    NOCACHE;

DESC Room_Type;
```

Results Explain **Describe** Saved SQL History

Object Type **TABLE** Object **ROOM_TYPE**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
ROOM_TYPE	ROOM_TYPE_ID	Number	-	-	0	1	-	-	-
	TYPE_NAME	Varchar2	50	-	-	-	-	-	-
	DESCRIPTION	Varchar2	500	-	-	-	✓	-	-
	CAPACITY	Number	-	-	0	-	-	-	-
	PRICE	Number	-	10	2	-	-	-	-

1 - 5

ROOM_STATUS:

```
CREATE TABLE Room_Status (
  room_status_id INT NOT NULL,
  status_name VARCHAR(50) NOT NULL,
  PRIMARY KEY (room_status_id)
);


CREATE SEQUENCE     room_status_id_seq
    START WITH 1
    INCREMENT BY 1
    NOMAXVALUE
    NOCYCLE
    NOCACHE;

DESC Room_Status;
```

Results Explain **Describe** Saved SQL History

Object Type **TABLE** Object **ROOM_STATUS**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
ROOM_STATUS	ROOM_STATUS_ID	Number	-	-	0	1	-	-	-
	STATUS_NAME	Varchar2	50	-	-	-	-	-	-

1 - 2

CHECK_IN:

```
CREATE TABLE Check_In (
  check_in_id INT NOT NULL,
  reservation_id INT NOT NULL,
  check_in_time DATE NOT NULL,
  PRIMARY KEY (check_in_id),
  FOREIGN KEY (reservation_id) REFERENCES Reservation(reservation_id)
);

CREATE SEQUENCE    check_in_id_seq
    START WITH 1
    INCREMENT BY 1
    NOMAXVALUE
    NOCYCLE
    NOCACHE;

DESC Check_In;
```

Results Explain Describe Saved SQL History

Object Type **TABLE** Object **CHECK_IN**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
CHECK_IN	CHECK_IN_ID	Number	-	-	0	1	-	-	-
	RESERVATION_ID	Number	-	-	0	-	-	-	-
	CHECK_IN_TIME	Date	7	-	-	-	-	-	-

1 - 3

CHECK_OUT:

```
CREATE TABLE Check_Out (
  check_out_id NUMBER NOT NULL,
  reservation_id NUMBER NOT NULL,
  check_out_time DATE NOT NULL,
  PRIMARY KEY (check_out_id),
  FOREIGN KEY (reservation_id) REFERENCES Reservation(reservation_id)
);

CREATE SEQUENCE    check_out_id_seq
    START WITH 1
    INCREMENT BY 1
    NOMAXVALUE
    NOCYCLE
    NOCACHE;

DESC Check_Out;
```

Results Explain Describe Saved SQL History

Object Type **TABLE** Object **CHECK_OUT**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
CHECK_OUT	CHECK_OUT_ID	Number	-	-	-	1	-	-	-
	RESERVATION_ID	Number	-	-	-	-	-	-	-
	CHECK_OUT_TIME	Date	7	-	-	-	-	-	-

1 - 3

BILLING:

```
CREATE TABLE Billing (
  billing_id INT NOT NULL,
  reservation_id INT NOT NULL,
  total_amount DECIMAL(10,2),
  PRIMARY KEY (billing_id),
  FOREIGN KEY (reservation_id) REFERENCES Reservation(reservation_id)
);


CREATE SEQUENCE   billing_id_seq
    START WITH 1
    INCREMENT BY 1
    NOMAXVALUE
    NOCYCLE
    NOCACHE;

DESC Billing;
```

Results Explain Describe Saved SQL History

Object Type **TABLE** Object **BILLING**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
BILLING	BILLING_ID	Number	-	-	0	1	-	-	-
	RESERVATION_ID	Number	-	-	0	-	-	-	-
	TOTAL_AMOUNT	Number	-	10	2	-	✓	-	-

1 - 3

PAYMENT_METHOD:

```
CREATE TABLE Payment_Method (
  payment_method_id INT NOT NULL,
  method_name VARCHAR(50) NOT NULL,
  PRIMARY KEY (payment_method_id)
);


CREATE SEQUENCE    payment_method_id_seq
    START WITH 1
    INCREMENT BY 1
    NOMAXVALUE
    NOCYCLE
    NOCACHE;

DESC Payment_Method;
```

Results Explain Describe Saved SQL History

Object Type **TABLE** Object **PAYMENT_METHOD**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
PAYMENT_METHOD	PAYMENT_METHOD_ID	Number	-	-	0	1	-	-	-
	METHOD_NAME	Varchar2	50	-	-	-	-	-	-

1 - 2

PAYMENT STATUS:

```
CREATE TABLE Payment_Status (
  payment_status_id INT NOT NULL,
  status_name VARCHAR(50) NOT NULL,
  PRIMARY KEY (payment_status_id)
);


CREATE SEQUENCE    payment_status_id_seq
    START WITH 1
    INCREMENT BY 1
    NOMAXVALUE
    NOCYCLE
    NOCACHE;

DESC Payment_Status;
```

Results Explain **Describe** Saved SQL History

Object Type **TABLE** Object **PAYMENT_STATUS**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
PAYMENT_STATUS	PAYMENT_STATUS_ID	Number	-	-	0	1	-	-	-
	STATUS_NAME	Varchar2	50	-	-	-	-	-	-

1 - 2

EMPLOYEE:

```
CREATE TABLE Employee (
   employee_id INT NOT NULL,
   first_name VARCHAR(50) NOT NULL,
   last_name VARCHAR(50) NOT NULL,
   email VARCHAR(100),
   phone VARCHAR(20),
   department_id INT NOT NULL,
   position_id INT NOT NULL,
   shift_id INT NOT NULL,
   PRIMARY KEY (employee_id),
   FOREIGN KEY (department_id) REFERENCES Department(department_id),
   FOREIGN KEY (position_id) REFERENCES Position(position_id),
   FOREIGN KEY (shift_id) REFERENCES Shift(shift_id)
)


CREATE SEQUENCE   employee_id_seq
     START WITH 1
     INCREMENT BY 1
     NOMAXVALUE
     NOCYCLE
     NOCACHE;

DESC Employee;
```

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
EMPLOYEE	EMPLOYEE_ID	Number	-	-	0	1	-	-	-
	FIRST_NAME	Varchar2	50	-	-	-	-	-	-
	LAST_NAME	Varchar2	50	-	-	-	-	-	-
	EMAIL	Varchar2	100	-	-	-	✓	-	-
	PHONE	Varchar2	20	-	-	-	✓	-	-
	DEPARTMENT_ID	Number	-	-	0	-	-	-	-
	POSITION_ID	Number	-	-	0	-	-	-	-
	SHIFT_ID	Number	-	-	0	-	-	-	-
									1 - 8

Department:

```
CREATE TABLE Department (
  department_id INT NOT NULL,
  department_name VARCHAR(50) NOT NULL,
  PRIMARY KEY (department_id)
);

CREATE SEQUENCE    department_id_seq
    START WITH 1
    INCREMENT BY 1
    NOMAXVALUE
    NOCYCLE
    NOCACHE;

DESC Department;
```

Results Explain **Describe** Saved SQL History

Object Type **TABLE** Object **DEPARTMENT**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
DEPARTMENT	DEPARTMENT_ID	Number	-	-	0	1	-	-	-
	DEPARTMENT_NAME	Varchar2	50	-	-	-	-	-	-
									1 - 2

POSITION:

```
CREATE TABLE Position (
  position_id INT NOT NULL,
  position_name VARCHAR(50) NOT NULL,
  PRIMARY KEY (position_id)
);


CREATE SEQUENCE    position_id_seq
    START WITH 1
    INCREMENT BY 1
    NOMAXVALUE
    NOCYCLE
    NOCACHE;

DESC Position;
```

Results Explain Describe Saved SQL History

Object Type **TABLE** Object **POSITION**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
POSITION	POSITION_ID	Number	-	-	0	1	-	-	-
	POSITION_NAME	Varchar2	50	-	-	-	-	-	-

1 - 2

Application Express 2.1.0.00.39

SHIFT:

Autocommit Display 10 Save Run

```
 CREATE TABLE Shift (
  shift_id INT NOT NULL ,
  shift_name VARCHAR(50) NOT NULL,
  start_time VARCHAR(50)  NOT NULL,
  end_time VARCHAR(100)  NOT NULL,
  PRIMARY KEY (shift_id)
);


CREATE SEQUENCE shift_id _seq
START WITH 1
INCREMENT BY 1
NOMAXVALUE
NOCYCLE
NOCACHE;

DESC SHIFT
```

Results Explain Describe Saved SQL History

Object Type **TABLE** Object **SHIFT**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
SHIFT	SHIFT_ID	Number	-	-	0	1	-	-	-
	SHIFT_NAME	Varchar2	50	-	-	-	-	-	-
	START_TIME	Varchar2	50	-	-	-	✓	-	-
	END_TIME	Varchar2	100	-	-	-	✓	-	-

1 - 4

Schedule:

```
CREATE TABLE Schedule (
  schedule_id INT NOT NULL,
  employee_id INT NOT NULL,
  shift_id INT NOT NULL,
  schedule_date DATE NOT NULL,
  PRIMARY KEY (schedule_id),
  FOREIGN KEY (employee_id) REFERENCES Employee(employee_id),
  FOREIGN KEY (shift_id) REFERENCES Shift(shift_id)
);


CREATE SEQUENCE   schedule_id_seq
    START WITH 1
    INCREMENT BY 1
    NOMAXVALUE
    NOCYCLE
    NOCACHE;

DESC Schedule;
```

Results Explain **Describe** Saved SQL History

Object Type **TABLE** Object **SCHEDULE**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
SCHEDULE	SCHEDULE_ID	Number	-	-	0	1	-	-	-
	EMPLOYEE_ID	Number	-	-	0	-	-	-	-
	SHIFT_ID	Number	-	-	0	-	-	-	-
	SCHEDULE_DATE	Date	7	-	-	-	-	-	-

1 - 4

Service:

```
CREATE TABLE Service (
  service_id INT NOT NULL,
  service_name VARCHAR(50) NOT NULL,
  service_type_id INT NOT NULL,
  description VARCHAR(4000),
  price DECIMAL(10,2) NOT NULL,
  reservation_id INT,
  PRIMARY KEY (service_id),
  FOREIGN KEY (service_type_id) REFERENCES Service_Type(service_type_id),
  FOREIGN KEY (reservation_id) REFERENCES Reservation(reservation_id)
);



CREATE SEQUENCE     service_id_seq
    START WITH 1
    INCREMENT BY 1
    NOMAXVALUE
    NOCYCLE
    NOCACHE;

DESC Service;
```

Object Type **TABLE** Object **SERVICE**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
SERVICE	SERVICE_ID	Number	-	-	0	1	-	-	-
	SERVICE_NAME	Varchar2	50	-	-	-	-	-	-
	SERVICE_TYPE_ID	Number	-	-	0	-	-	-	-
	DESCRIPTION	Varchar2	4000	-	-	-	✓	-	-
	PRICE	Number	-	10	2	-	-	-	-
	RESERVATION_ID	Number	-	-	0	-	✓	-	-

1 - 6

Application Express 21.0.0.0.39

Service_type:

```
CREATE TABLE Service_Type (
  service_type_id INT NOT NULL,
  type_name VARCHAR(50) NOT NULL,
  PRIMARY KEY (service_type_id)
);


CREATE SEQUENCE   service_type_id_seq
    START WITH 1
    INCREMENT BY 1
    NOMAXVALUE
    NOCYCLE
    NOCACHE;

DESC Service_Type;
```

Results Explain **Describe** Saved SQL History

Object Type **TABLE** Object **SERVICE_TYPE**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
SERVICE_TYPE	SERVICE_TYPE_ID	Number	-	-	0	1	-	-	-
	TYPE_NAME	Varchar2	50	-	-	-	-	-	-

1 - 2

MENU:

```
CREATE TABLE Menu (
  menu_id INT NOT NULL,
  menu_name VARCHAR(100) NOT NULL,
  description VARCHAR(500) ,
  PRIMARY KEY (menu_id)
);



CREATE SEQUENCE     menu_id_seq
    START WITH 1
    INCREMENT BY 1
    NOMAXVALUE
    NOCYCLE
    NOCACHE;

DESC  Menu;
```

Results Explain **Describe** Saved SQL History

Object Type **TABLE** Object **MENU**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
MENU	MENU_ID	Number	-	-	0	1	-	-	-
	MENU_NAME	Varchar2	100	-	-	-	-	-	-
	DESCRIPTION	Varchar2	500	-	-	-	✓	-	-

1 - 3

ORDERS:

```
CREATE TABLE Orders (
  order_id INT NOT NULL ,
  guest_id INT NOT NULL,
  menu_id INT NOT NULL,
  order_time DATE NOT NULL,
  quantity INT NOT NULL,
  total_price DECIMAL(10,2),
  PRIMARY KEY (order_id),
  FOREIGN KEY (guest_id) REFERENCES Guest(guest_id),
  FOREIGN KEY (menu_id) REFERENCES Menu(menu_id)
);


CREATE SEQUENCE     order_id_seq
    START WITH 1
    INCREMENT BY 1
    NOMAXVALUE
    NOCYCLE
    NOCACHE;

DESC  Orders;
```

Results Explain **Describe** Saved SQL History

Object Type **TABLE** Object **ORDERS**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
ORDERS	ORDER_ID	Number	-	-	0	1	-	-	-
	GUEST_ID	Number	-	-	0	-	-	-	-
	MENU_ID	Number	-	-	0	-	-	-	-
	ORDER_TIME	Date	7	-	-	-	-	-	-
	QUANTITY	Number	-	-	0	-	-	-	-
	TOTAL_PRICE	Number	-	10	2	-	✓	-	-

1 - 6

FEEDBACK:

```
CREATE TABLE Feedback (
  feedback_id INT NOT NULL,
  guest_id INT NOT NULL,
  feedback VARCHAR2(500) NOT NULL,
  feedback_date DATE NOT NULL,
  PRIMARY KEY (feedback_id),
  FOREIGN KEY (guest_id) REFERENCES Guest(guest_id)
);



CREATE SEQUENCE     feedback_id_seq
    START WITH 1
    INCREMENT BY 1
    NOMAXVALUE
    NOCYCLE
    NOCACHE;

DESC  Feedback;
```

Results Explain **Describe** Saved SQL History

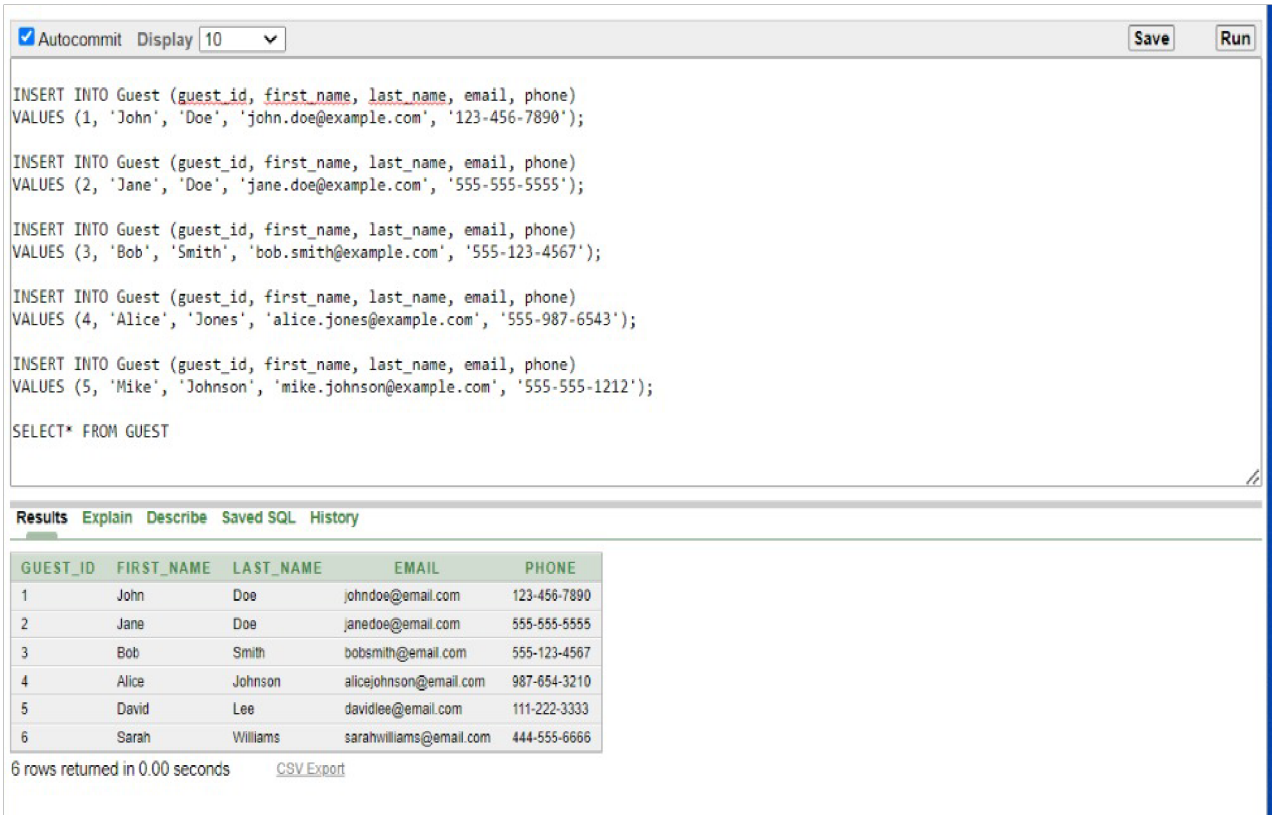
Object Type **TABLE** Object **FEEDBACK**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
FEEDBACK	FEEDBACK_ID	Number	-	-	0	1	-	-	-
	GUEST_ID	Number	-	-	0	-	-	-	-
	FEEDBACK	Varchar2	500	-	-	-	-	-	-
	FEEDBACK_DATE	Date	7	-	-	-	-	-	-

1 - 4

Data Insertion

Guest:



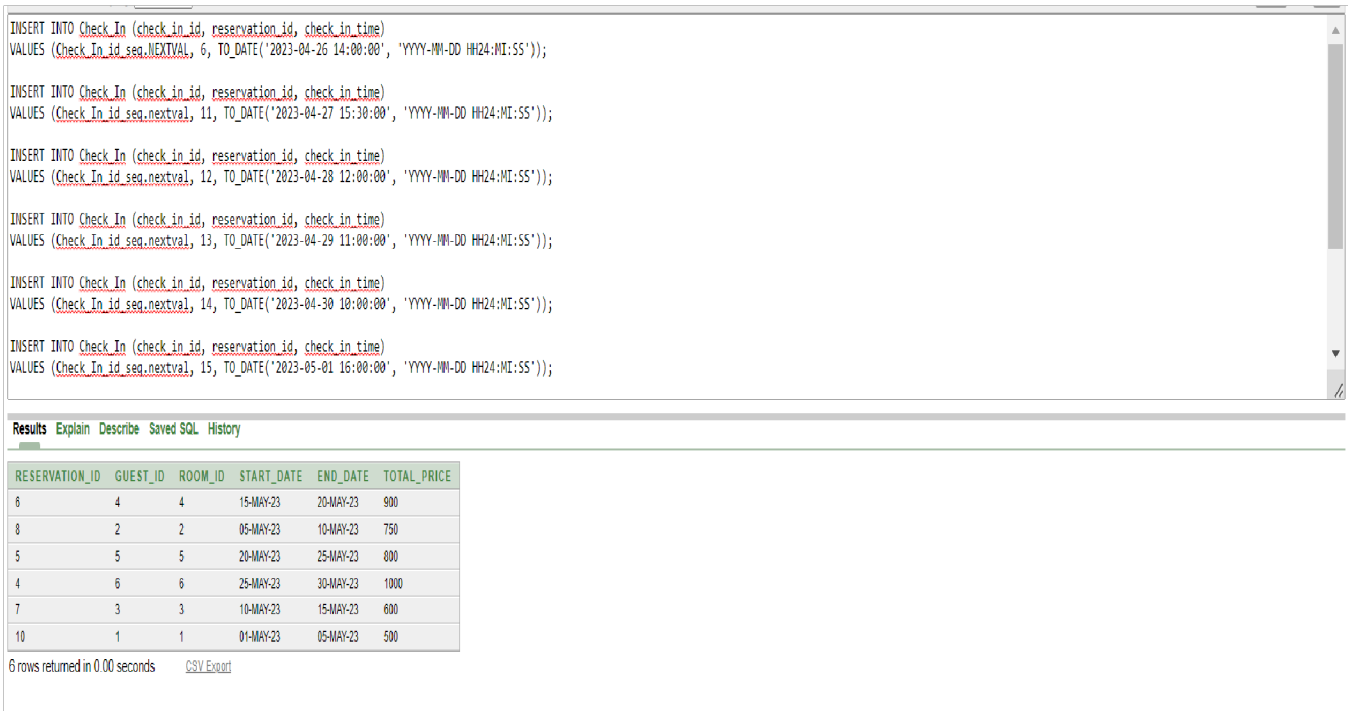
```
INSERT INTO Guest (guest_id, first_name, last_name, email, phone)
VALUES (1, 'John', 'Doe', 'john.doe@example.com', '123-456-7890');

INSERT INTO Guest (guest_id, first_name, last_name, email, phone)
VALUES (2, 'Jane', 'Doe', 'jane.doe@example.com', '555-555-5555');

INSERT INTO Guest (guest_id, first_name, last_name, email, phone)
VALUES (3, 'Bob', 'Smith', 'bob.smith@example.com', '555-123-4567');

INSERT INTO Guest (guest_id, first_name, last_name, email, phone)
VALUES (4, 'Alice', 'Jones', 'alice.jones@example.com', '555-987-6543');

INSERT INTO Guest (guest_id, first_name, last_name, email, phone)
VALUES (5, 'Mike', 'Johnson', 'mike.johnson@example.com', '555-555-1212');

SELECT* FROM GUEST
```

Results Explain Describe Saved SQL History

GUEST_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE
1	John	Doe	johndoe@email.com	123-456-7890
2	Jane	Doe	janedoe@email.com	555-555-5555
3	Bob	Smith	bobsmith@email.com	555-123-4567
4	Alice	Johnson	alicejohnson@email.com	987-654-3210
5	David	Lee	davidlee@email.com	111-222-3333
6	Sarah	Williams	sarahwilliams@email.com	444-555-6666

6 rows returned in 0.00 seconds CSV Export

Reservation:



```
INSERT INTO Check_In (check_in_id, reservation_id, check_in_time)
VALUES (Check_In_id_seq.NEXTVAL, 6, TO_DATE('2023-04-26 14:00:00', 'YYYY-MM-DD HH24:MI:SS'));

INSERT INTO Check_In (check_in_id, reservation_id, check_in_time)
VALUES (Check_In_id_seq.nextval, 11, TO_DATE('2023-04-27 15:30:00', 'YYYY-MM-DD HH24:MI:SS'));

INSERT INTO Check_In (check_in_id, reservation_id, check_in_time)
VALUES (Check_In_id_seq.nextval, 12, TO_DATE('2023-04-28 12:00:00', 'YYYY-MM-DD HH24:MI:SS'));

INSERT INTO Check_In (check_in_id, reservation_id, check_in_time)
VALUES (Check_In_id_seq.nextval, 13, TO_DATE('2023-04-29 11:00:00', 'YYYY-MM-DD HH24:MI:SS'));

INSERT INTO Check_In (check_in_id, reservation_id, check_in_time)
VALUES (Check_In_id_seq.nextval, 14, TO_DATE('2023-04-30 10:00:00', 'YYYY-MM-DD HH24:MI:SS'));

INSERT INTO Check_In (check_in_id, reservation_id, check_in_time)
VALUES (Check_In_id_seq.nextval, 15, TO_DATE('2023-05-01 16:00:00', 'YYYY-MM-DD HH24:MI:SS'));
```

Results Explain Describe Saved SQL History

RESERVATION_ID	GUEST_ID	ROOM_ID	START_DATE	END_DATE	TOTAL_PRICE
6	4	4	15-MAY-23	20-MAY-23	900
8	2	2	05-MAY-23	10-MAY-23	750
5	5	5	20-MAY-23	25-MAY-23	800
4	6	6	25-MAY-23	30-MAY-23	1000
7	3	3	10-MAY-23	15-MAY-23	600
10	1	1	01-MAY-23	05-MAY-23	500

6 rows returned in 0.00 seconds CSV Export

Room:

```
INSERT INTO Room (room_id, room_number, room_type_id, room_status_id)
VALUES (room_id_seq.nextval, '101', 1, 1);

INSERT INTO Room (room_id, room_number, room_type_id, room_status_id)
VALUES (room_id_seq.nextval, '102', 1, 1);

INSERT INTO Room (room_id, room_number, room_type_id, room_status_id)
VALUES (room_id_seq.nextval, '103', 2, 1);

INSERT INTO Room (room_id, room_number, room_type_id, room_status_id)
VALUES (room_id_seq.nextval, '201', 1, 2);

INSERT INTO Room (room_id, room_number, room_type_id, room_status_id)
VALUES (room_id_seq.nextval, '202', 2, 2);

INSERT INTO Room (room_id, room_number, room_type_id, room_status_id)
VALUES (room_id_seq.nextval, '203', 2, 2);

Select* from ROOM;
```

Results Explain Describe Saved SQL History

ROOM_ID	ROOM_NUMBER	ROOM_TYPE_ID	ROOM_STATUS_ID
1	101	1	1
2	102	1	1
3	103	2	1
4	201	1	2
5	202	2	2
6	203	2	2

6 rows returned in 0.00 seconds CSV Export

ROOM TYPE:

```
INSERT INTO Room_Type (room_type_id, type_name, description, capacity, price)
VALUES (room_type_id_seq.nextval, 'Single', 'A room with a single bed', 1, 100.00);

INSERT INTO Room_Type (room_type_id, type_name, description, capacity, price)
VALUES (room_type_id_seq.nextval, 'Double', 'A room with two beds', 2, 150.00);

INSERT INTO Room_Type (room_type_id, type_name, description, capacity, price)
VALUES (room_type_id_seq.nextval, 'Suite', 'A spacious room with a sitting area', 4, 300.00);

INSERT INTO Room_Type (room_type_id, type_name, description, capacity, price)
VALUES (room_type_id_seq.nextval, 'Executive Suite', 'A luxurious room with a separate living room and bedroom', 6, 500.00);

INSERT INTO Room_Type (room_type_id, type_name, description, capacity, price)
VALUES (room_type_id_seq.nextval, 'Twin', 'A room with two single beds', 2, 120.00);

INSERT INTO Room_Type (room_type_id, type_name, description, capacity, price)
VALUES (room_type_id_seq.nextval, 'Triple', 'A room with three beds', 3, 180.00);

SELECT * FROM ROOM_TYPE;
```

Results Explain Describe Saved SQL History

ROOM_TYPE_ID	TYPE_NAME	DESCRIPTION	CAPACITY	PRICE
1	Single	A room with a single bed	1	100
2	Double	A room with two beds	2	150
3	Suite	A spacious room with a sitting area	4	300
4	Executive Suite	A luxurious room with a separate living room and bedroom	6	500
5	Twin	A room with two single beds	2	120
6	Triple	A room with three beds	3	180

6 rows returned in 0.00 seconds CSV Export

ROOM_Statues:

Autocommit Display 10 Save Run

```
INSERT INTO Room_Status (room_status_id, status_name) VALUES (1, 'Available');
INSERT INTO Room_Status (room_status_id, status_name)VALUES (2, 'Occupied');
INSERT INTO Room_Status (room_status_id, status_name) VALUES (3, 'Reserved');
INSERT INTO Room_Status (room_status_id, status_name) VALUES (4, 'Under Maintenance');
INSERT INTO Room_Status (room_status_id, status_name) VALUES (5, 'Out of Order');
INSERT INTO Room_Status (room_status_id, status_name) VALUES (6, 'Cleaning in Progress');

SELECT * FROM Room_Status;
```

Results Explain Describe Saved SQL History

ROOM_STATUS_ID	STATUS_NAME
1	Available
2	Occupied
3	Reserved
4	Under Maintenance
5	Out of Order
6	Cleaning in Progress

6 rows returned in 0.00 seconds CSV Export

Check_in:

```
VALUES (Check_In_id_seq.nextval,8, TO_DATE('2023-04-27 15:30:00', 'YYYY-MM-DD HH24:MI:SS'));

INSERT INTO Check_In (check_in_id, reservation_id, check_in_time)
VALUES (Check_In_id_seq.nextval, 5, TO_DATE('2023-04-28 12:00:00', 'YYYY-MM-DD HH24:MI:SS'));

INSERT INTO Check_In (check_in_id, reservation_id, check_in_time)
VALUES (Check_In_id_seq.nextval, 4, TO_DATE('2023-04-29 11:00:00', 'YYYY-MM-DD HH24:MI:SS'));

INSERT INTO Check_In (check_in_id, reservation_id, check_in_time)
VALUES (Check_In_id_seq.nextval,7, TO_DATE('2023-04-30 10:00:00', 'YYYY-MM-DD HH24:MI:SS'));

INSERT INTO Check_In (check_in_id, reservation_id, check_in_time)
VALUES (Check_In_id_seq.nextval, 10, TO_DATE('2023-05-01 16:00:00', 'YYYY-MM-DD HH24:MI:SS'));

SELECT * FROM Check_In ;
```

Results Explain Describe Saved SQL History

CHECK_IN_ID	RESERVATION_ID	CHECK_IN_TIME
6	6	26-APR-23
7	8	27-APR-23
8	5	28-APR-23
9	4	29-APR-23
10	7	30-APR-23
11	10	01-MAY-23

6 rows returned in 0.01 seconds CSV Export

Check_out:

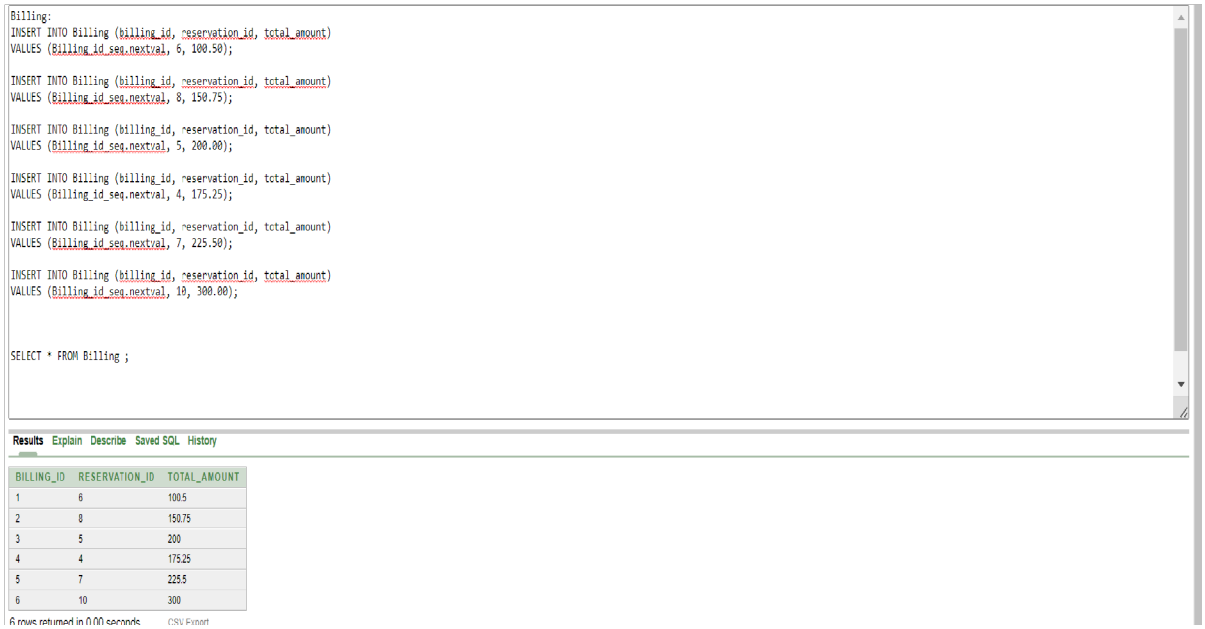
```
INSERT INTO Check_Out (check_out_id, reservation_id, check_out_time)
VALUES (Check_Out_id_seq.NEXTVAL, 6, TO_DATE('2023-05-01 12:00:00', 'YYYY-MM-DD HH24:MI:SS'));

INSERT INTO Check_Out (check_out_id, reservation_id, check_out_time)
VALUES (Check_Out_id_seq.NEXTVAL, 8, TO_DATE('2023-05-03 11:00:00', 'YYYY-MM-DD HH24:MI:SS'));

INSERT INTO Check_Out (check_out_id, reservation_id, check_out_time)
VALUES (Check_Out_id_seq.NEXTVAL, 5, TO_DATE('2023-05-06 10:30:00', 'YYYY-MM-DD HH24:MI:SS'));

INSERT INTO Check_Out (check_out_id, reservation_id, check_out_time)
VALUES (Check_Out_id_seq.NEXTVAL, 4, TO_DATE('2023-05-05 09:45:00', 'YYYY-MM-DD HH24:MI:SS'));

INSERT INTO Check_Out (check_out_id, reservation_id, check_out_time)
VALUES (Check_Out_id_seq.NEXTVAL, 7, TO_DATE('2023-05-02 08:15:00', 'YYYY-MM-DD HH24:MI:SS'));

INSERT INTO Check_Out (check_out_id, reservation_id, check_out_time)
VALUES (Check_Out_id_seq.NEXTVAL, 10, TO_DATE('2023-05-04 07:30:00', 'YYYY-MM-DD HH24:MI:SS'));


SELECT * FROM Check_Out ;
```

Results Explain Describe Saved SQL History

CHECK_OUT_ID	RESERVATION_ID	CHECK_OUT_TIME
1	6	01-MAY-23
2	8	03-MAY-23
3	5	06-MAY-23
4	4	05-MAY-23
5	7	02-MAY-23
6	10	04-MAY-23

6 rows returned in 0.00 seconds CSV Export

Billing:



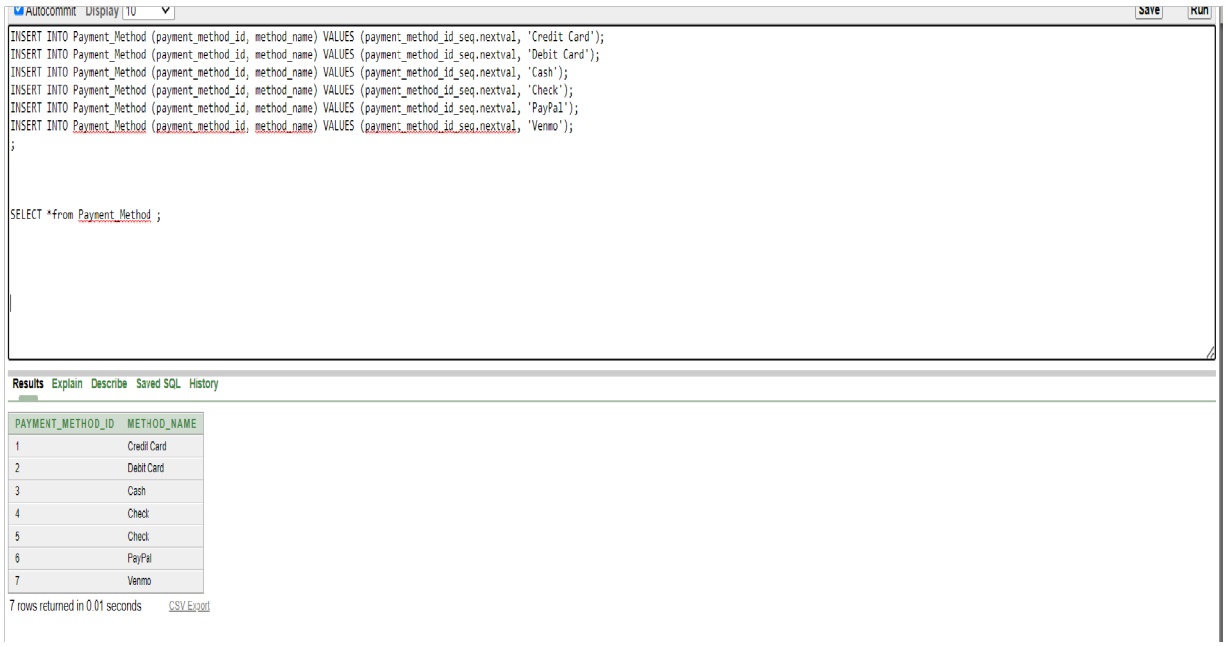
```
Billing:
INSERT INTO Billing (billing_id, reservation_id, total_amount)
VALUES (Billing_id_seq.nextval, 6, 100.50);

INSERT INTO Billing (billing_id, reservation_id, total_amount)
VALUES (Billing_id_seq.nextval, 8, 150.75);

INSERT INTO Billing (billing_id, reservation_id, total_amount)
VALUES (Billing_id_seq.nextval, 5, 200.00);

INSERT INTO Billing (billing_id, reservation_id, total_amount)
VALUES (Billing_id_seq.nextval, 4, 175.25);

INSERT INTO Billing (billing_id, reservation_id, total_amount)
VALUES (Billing_id_seq.nextval, 7, 225.50);

INSERT INTO Billing (billing_id, reservation_id, total_amount)
VALUES (Billing_id_seq.nextval, 10, 300.00);



SELECT * FROM Billing ;
```

Results Explain Describe Saved SQL History

BILLING_ID	RESERVATION_ID	TOTAL_AMOUNT
1	6	100.5
2	8	150.75
3	5	200
4	4	175.25
5	7	225.5
6	10	300

6 rows returned in 0.00 seconds CSV Export

Payement_method:



```
INSERT INTO Payment_Method (payment_method_id, method_name) VALUES (payment_method_id_seq.nextval, 'Credit Card');
INSERT INTO Payment_Method (payment_method_id, method_name) VALUES (payment_method_id_seq.nextval, 'Debit Card');
INSERT INTO Payment_Method (payment_method_id, method_name) VALUES (payment_method_id_seq.nextval, 'Cash');
INSERT INTO Payment_Method (payment_method_id, method_name) VALUES (payment_method_id_seq.nextval, 'Check');
INSERT INTO Payment_Method (payment_method_id, method_name) VALUES (payment_method_id_seq.nextval, 'PayPal');
INSERT INTO Payment_Method (payment_method_id, method_name) VALUES (payment_method_id_seq.nextval, 'Venmo');
;


SELECT *from Payment_Method ;
```

Results Explain Describe Saved SQL History

PAYMENT_METHOD_ID	METHOD_NAME
1	Credit Card
2	Debit Card
3	Cash
4	Check
5	Check
6	PayPal
7	Venmo

7 rows returned in 0.01 seconds CSV Export

Payement_statues:

```
INSERT INTO Payment_Status (payment_status_id, status_name)
VALUES (payment_status_id_seq.NEXTVAL, 'Pending');

INSERT INTO Payment_Status (payment_status_id, status_name)
VALUES (payment_status_id_seq.NEXTVAL, 'Processing');

INSERT INTO Payment_Status (payment_status_id, status_name)
VALUES (payment_status_id_seq.NEXTVAL, 'Completed');

INSERT INTO Payment_Status (payment_status_id, status_name)
VALUES (payment_status_id_seq.NEXTVAL, 'Cancelled');

INSERT INTO Payment_Status (payment_status_id, status_name)
VALUES (payment_status_id_seq.NEXTVAL, 'Refunded');

INSERT INTO Payment_Status (payment_status_id, status_name)
VALUES (payment_status_id_seq.NEXTVAL, 'Failed');


SELECT *from Payment_Status ;
```

Results Explain Describe Saved SQL History

PAYMENT_STATUS_ID	STATUS_NAME
1	Pending
2	Processing
3	Completed
4	Cancelled
5	Refunded
6	Failed

6 rows returned in 0.00 seconds CSV Export

EMPLOYEE:

Autocommit Display 10 Save Run

```
INSERT INTO Employee (employee_id, first_name, last_name, email, phone, department_id, position_id, shift_id)
VALUES (employee_id_seq.NEXTVAL, 'John', 'Doe', 'johndoe@example.com', '123-456-7890', 1, 1, 1);

INSERT INTO Employee (employee_id, first_name, last_name, email, phone, department_id, position_id, shift_id)
VALUES (employee_id_seq.NEXTVAL, 'Jane', 'Doe', 'janedoe@example.com', '987-654-3210', 1, 2, 2);

INSERT INTO Employee (employee_id, first_name, last_name, email, phone, department_id, position_id, shift_id)
VALUES (employee_id_seq.NEXTVAL, 'Bob', 'Smith', 'bobsmith@example.com', '555-555-5555', 2, 3, 3);

INSERT INTO Employee (employee_id, first_name, last_name, email, phone, department_id, position_id, shift_id)
VALUES (employee_id_seq.NEXTVAL, 'Alice', 'Johnson', 'alicejohnson@example.com', '111-111-1111', 2, 4, 2);

INSERT INTO Employee (employee_id, first_name, last_name, email, phone, department_id, position_id, shift_id)
VALUES (employee_id_seq.NEXTVAL, 'David', 'Lee', 'davidlee@example.com', '222-222-2222', 3, 5, 3);

INSERT INTO Employee (employee_id, first_name, last_name, email, phone, department_id, position_id, shift_id)
VALUES (employee_id_seq.NEXTVAL, 'Sarah', 'Kim', 'sarahkim@example.com', '333-333-3333', 3, 6, 1);

SELECT * FROM EMPLOYEE
```

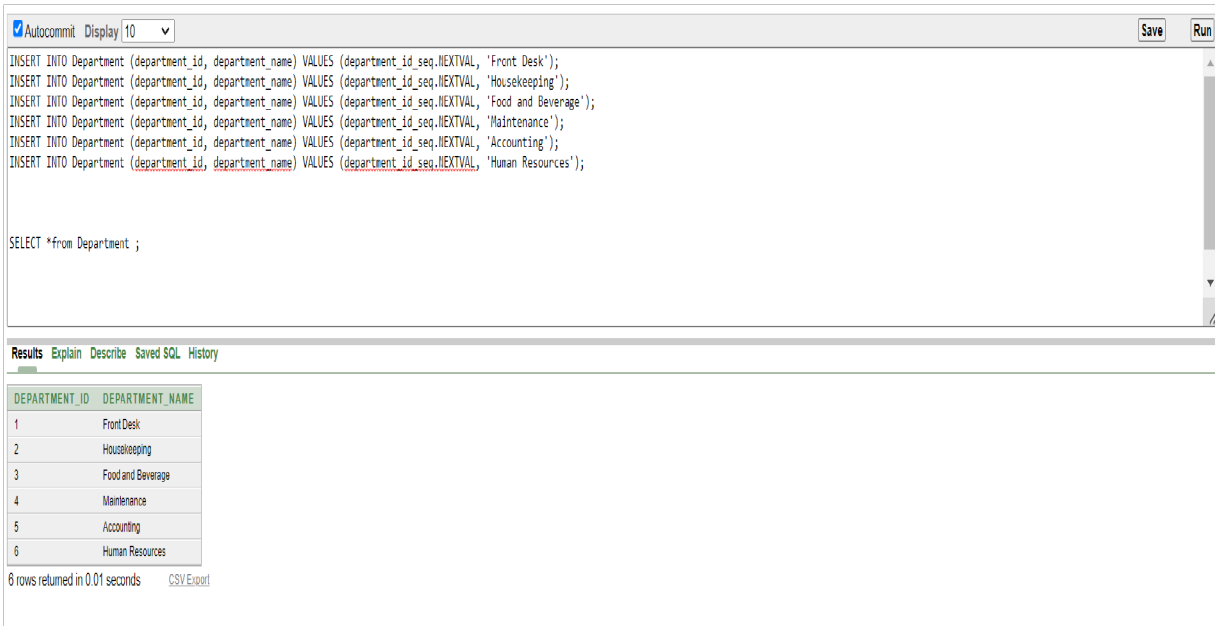
Results Explain Describe Saved SQL History

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE	DEPARTMENT_ID	POSITION_ID	SHIFT_ID
1	John	Doe	johndoe@example.com	123-456-7890	1	1	1
2	Jane	Doe	janedoe@example.com	987-654-3210	1	2	2
3	Bob	Smith	bobsmith@example.com	555-555-5555	2	3	3
4	Alice	Johnson	alicejohnson@example.com	111-111-1111	2	4	2
5	David	Lee	davidlee@example.com	222-222-2222	3	5	3
6	Sarah	Kim	sarahkim@example.com	333-333-3333	3	6	1

6 rows returned in 0.00 seconds CSV Export

Application Express 2.1.0.00.39

Department:



```
INSERT INTO Department (department_id, department_name) VALUES (department_id_seq.NEXTVAL, 'Front Desk');
INSERT INTO Department (department_id, department_name) VALUES (department_id_seq.NEXTVAL, 'Housekeeping');
INSERT INTO Department (department_id, department_name) VALUES (department_id_seq.NEXTVAL, 'Food and Beverage');
INSERT INTO Department (department_id, department_name) VALUES (department_id_seq.NEXTVAL, 'Maintenance');
INSERT INTO Department (department_id, department_name) VALUES (department_id_seq.NEXTVAL, 'Accounting');
INSERT INTO Department (department_id, department_name) VALUES (department_id_seq.NEXTVAL, 'Human Resources');



SELECT *from Department ;
```

Results Explain Describe Saved SQL History

DEPARTMENT_ID	DEPARTMENT_NAME
1	Front Desk
2	Housekeeping
3	Food and Beverage
4	Maintenance
5	Accounting
6	Human Resources

6 rows returned in 0.01 seconds CSV Export

Position:

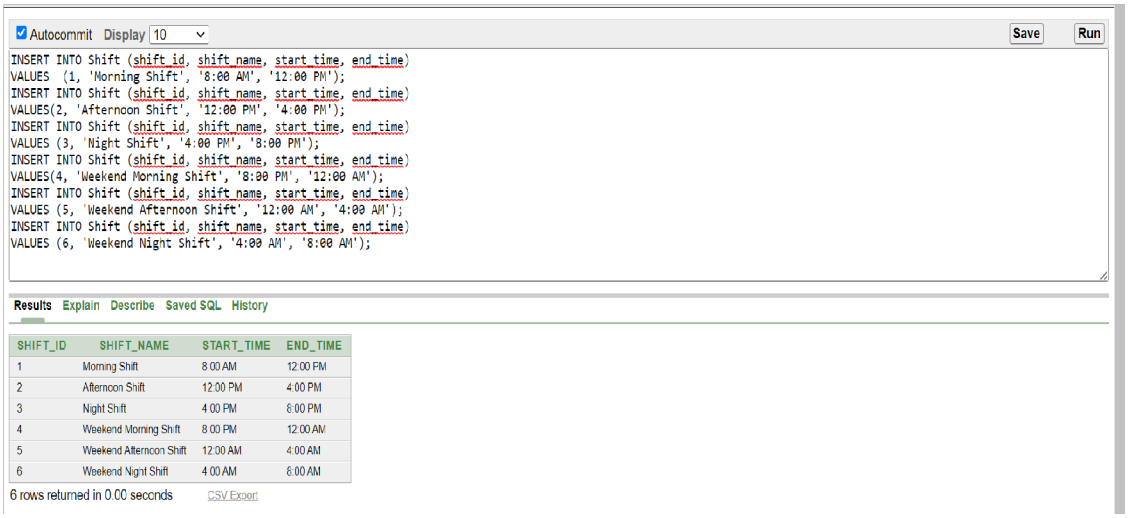
```
INSERT INTO Position (position_id, position_name)
VALUES (position_id_seq.NEXTVAL, 'Manager');

INSERT INTO Position (position_id, position_name)
VALUES (position_id_seq.NEXTVAL, 'Front Desk Clerk');

INSERT INTO Position (position_id, position_name)
VALUES (position_id_seq.NEXTVAL, 'Housekeeper');

INSERT INTO Position (position_id, position_name)
VALUES (position_id_seq.NEXTVAL, 'Maintenance Worker');

INSERT INTO Position (position_id, position_name)
VALUES (position_id_seq.NEXTVAL, 'Restaurant Server');

INSERT INTO Position (position_id, position_name)
VALUES (position_id_seq.NEXTVAL, 'Bartender');



SELECT *from Position ;
```

Results Explain Describe Saved SQL History

POSITION_ID	POSITION_NAME
1	Manager
2	Front Desk Clerk
3	Housekeeper
4	Maintenance Worker
5	Restaurant Server
6	Bartender

6 rows returned in 0.00 seconds CSV Export

SHIFT:



Autocommit Display 10 Save Run

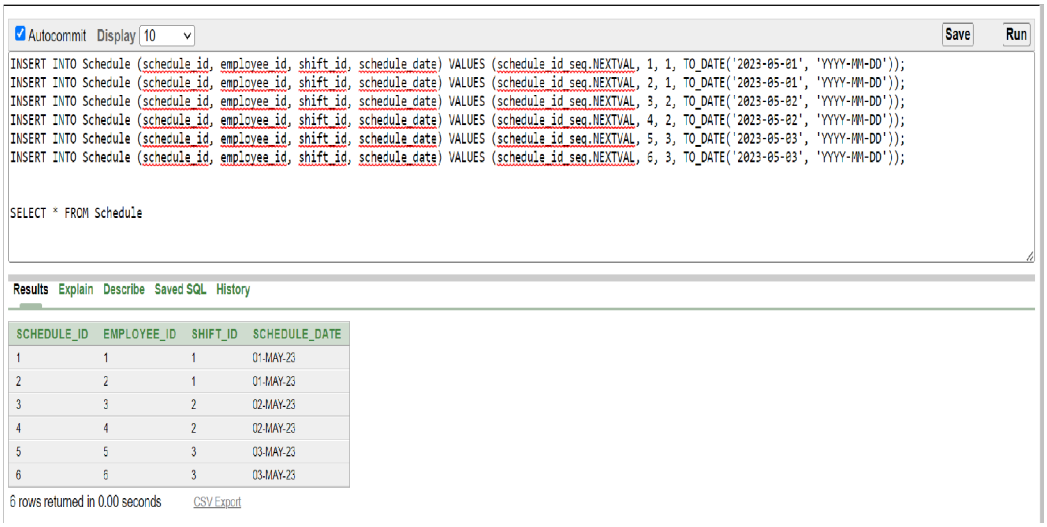
```
INSERT INTO Shift (shift_id, shift_name, start_time, end_time)
VALUES  (1, 'Morning Shift', '8:00 AM', '12:00 PM');
INSERT INTO Shift (shift_id, shift_name, start_time, end_time)
VALUES(2, 'Afternoon Shift', '12:00 PM', '4:00 PM');
INSERT INTO Shift (shift_id, shift_name, start_time, end_time)
VALUES (3, 'Night Shift', '4:00 PM', '8:00 PM');
INSERT INTO Shift (shift_id, shift_name, start_time, end_time)
VALUES(4, 'Weekend Morning Shift', '8:00 PM', '12:00 AM');
INSERT INTO Shift (shift_id, shift_name, start_time, end_time)
VALUES (5, 'Weekend Afternoon Shift', '12:00 AM', '4:00 AM');
INSERT INTO Shift (shift_id, shift_name, start_time, end_time)
VALUES (6, 'Weekend Night Shift', '4:00 AM', '8:00 AM');
```

Results Explain Describe Saved SQL History

SHIFT_ID	SHIFT_NAME	START_TIME	END_TIME
1	Morning Shift	8:00 AM	12:00 PM
2	Afternoon Shift	12:00 PM	4:00 PM
3	Night Shift	4:00 PM	8:00 PM
4	Weekend Morning Shift	8:00 PM	12:00 AM
5	Weekend Afternoon Shift	12:00 AM	4:00 AM
6	Weekend Night Shift	4:00 AM	8:00 AM

6 rows returned in 0.00 seconds CSV Export

SCHEDULE:



Autocommit Display 10 Save Run

```
INSERT INTO Schedule (schedule_id, employee_id, shift_id, schedule_date) VALUES (schedule_id_seq.NEXTVAL, 1, 1, TO_DATE('2023-05-01', 'YYYY-MM-DD'));
INSERT INTO Schedule (schedule_id, employee_id, shift_id, schedule_date) VALUES (schedule_id_seq.NEXTVAL, 2, 1, TO_DATE('2023-05-01', 'YYYY-MM-DD'));
INSERT INTO Schedule (schedule_id, employee_id, shift_id, schedule_date) VALUES (schedule_id_seq.NEXTVAL, 3, 2, TO_DATE('2023-05-02', 'YYYY-MM-DD'));
INSERT INTO Schedule (schedule_id, employee_id, shift_id, schedule_date) VALUES (schedule_id_seq.NEXTVAL, 4, 2, TO_DATE('2023-05-02', 'YYYY-MM-DD'));
INSERT INTO Schedule (schedule_id, employee_id, shift_id, schedule_date) VALUES (schedule_id_seq.NEXTVAL, 5, 3, TO_DATE('2023-05-03', 'YYYY-MM-DD'));
INSERT INTO Schedule (schedule_id, employee_id, shift_id, schedule_date) VALUES (schedule_id_seq.NEXTVAL, 6, 3, TO_DATE('2023-05-03', 'YYYY-MM-DD'));


SELECT * FROM Schedule
```

Results Explain Describe Saved SQL History

SCHEDULE_ID	EMPLOYEE_ID	SHIFT_ID	SCHEDULE_DATE
1	1	1	01-MAY-23
2	2	1	01-MAY-23
3	3	2	02-MAY-23
4	4	2	02-MAY-23
5	5	3	03-MAY-23
6	6	3	03-MAY-23

6 rows returned in 0.00 seconds CSV Export

SERVICE:

```
INSERT INTO Service (service_id, service_name, service_type_id, description, price, reservation_id)
VALUES (Service_id_seq.nextval, 'Massage', 6, 'Full body massage', 100.00, 6);

INSERT INTO Service (service_id, service_name, service_type_id, description, price, reservation_id)
VALUES (Service_id_seq.nextval, 'Room service', 2, 'Food and drinks', 50.00, 8);

INSERT INTO Service (service_id, service_name, service_type_id, description, price, reservation_id)
VALUES (Service_id_seq.nextval, 'Laundry', 3, 'Wash and fold', 20.00, 5);

INSERT INTO Service (service_id, service_name, service_type_id, description, price, reservation_id)
VALUES (Service_id_seq.nextval, 'Airport shuttle', 4, 'Transportation to/from airport', 50.00, 4);

INSERT INTO Service (service_id, service_name, service_type_id, description, price, reservation_id)
VALUES (Service_id_seq.nextval, 'Gym access', 7, 'Use of gym facilities', 10.00, 7);

INSERT INTO Service (service_id, service_name, service_type_id, description, price, reservation_id)
VALUES (Service_id_seq.nextval, 'Concierge', 6, 'Assistance with reservations and activities', 0.00, 10);

SELECT * FROM SERVICE;
```

Results Explain Describe Saved SQL History

SERVICE_ID	SERVICE_NAME	SERVICE_TYPE_ID	DESCRIPTION	PRICE	RESERVATION_ID
1	Massage	6	Full body massage	100	6
2	Room service	2	Food and drinks	50	8
3	Laundry	3	Wash and fold	20	5
4	Airport shuttle	4	Transportation to/from airport	50	4
5	Gym access	7	Use of gym facilities	10	7
6	Concierge	6	Assistance with reservations and activities	0	10

6 rows returned in 0.00 seconds CSV Export

SERVICE_TYPE:

Autocommit Display 10 Save Run

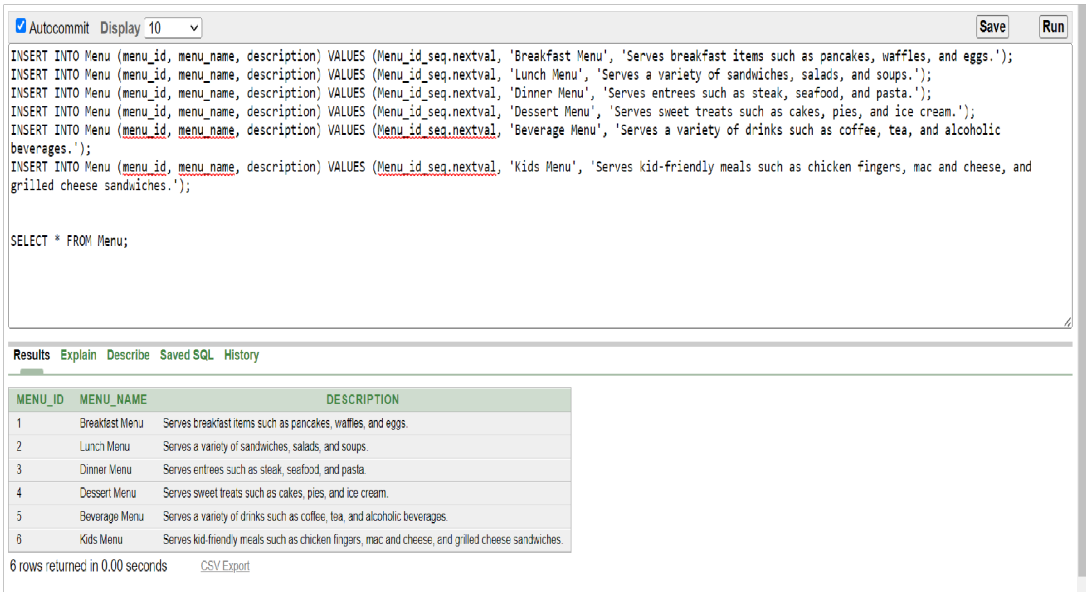
```
INSERT INTO Service_Type (service_type_id, type_name)
VALUES (Service_Type_id_seq.NEXTVAL, 'Room Service');
 INSERT INTO Service_Type (service_type_id, type_name)
VALUES  (Service_Type_id_seq.NEXTVAL, 'Housekeeping');
INSERT INTO Service_Type (service_type_id, type_name)
VALUES  (Service_Type_id_seq.NEXTVAL, 'Laundry');
INSERT INTO Service_Type (service_type_id, type_name)
VALUES  (Service_Type_id_seq.NEXTVAL, 'Restaurant');
INSERT INTO Service_Type (service_type_id, type_name)
VALUES  (Service_Type_id_seq.NEXTVAL, 'Spa');
INSERT INTO Service_Type (service_type_id, type_name)
VALUES (Service_Type_id_seq.NEXTVAL, 'Fitness Center');
SELECT * FROM SERVICE_TYPE;
```

Results Explain Describe Saved SQL History

SERVICE_TYPE_ID	TYPE_NAME
1	Room Service
2	Housekeeping
3	Laundry
4	Restaurant
5	Restaurant
6	Spa
7	Fitness Center

7 rows returned in 0.00 seconds CSV Export

Menu:



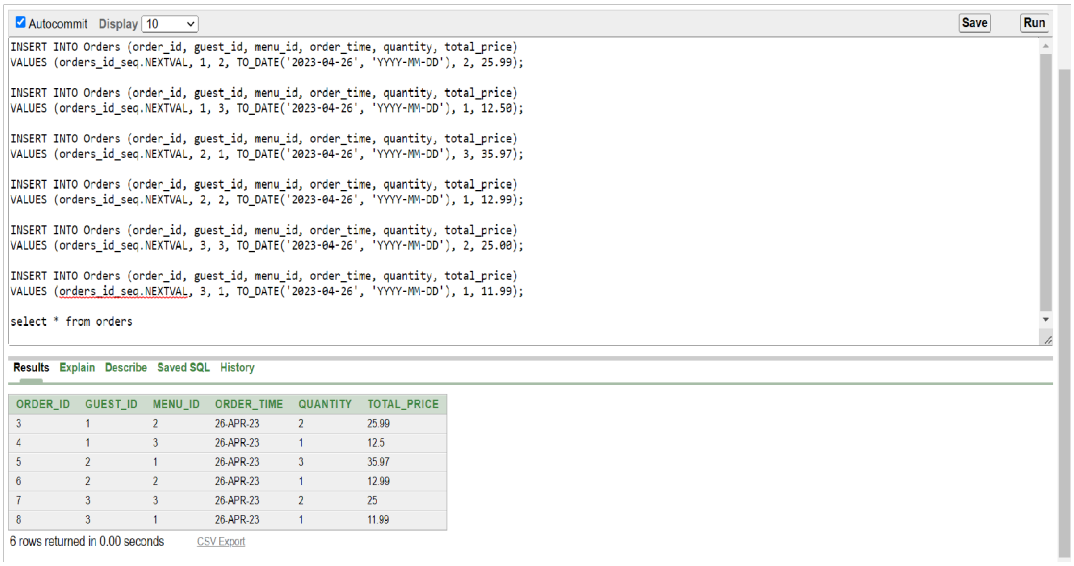
```
INSERT INTO Menu (menu_id, menu_name, description) VALUES (Menu_id_seq.nextval, 'Breakfast Menu', 'Serves breakfast items such as pancakes, waffles, and eggs.');
INSERT INTO Menu (menu_id, menu_name, description) VALUES (Menu_id_seq.nextval, 'Lunch Menu', 'Serves a variety of sandwiches, salads, and soups.');
INSERT INTO Menu (menu_id, menu_name, description) VALUES (Menu_id_seq.nextval, 'Dinner Menu', 'Serves entrees such as steak, seafood, and pasta.');
INSERT INTO Menu (menu_id, menu_name, description) VALUES (Menu_id_seq.nextval, 'Dessert Menu', 'Serves sweet treats such as cakes, pies, and ice cream.');
INSERT INTO Menu (menu_id, menu_name, description) VALUES (Menu_id_seq.nextval, 'Beverage Menu', 'Serves a variety of drinks such as coffee, tea, and alcoholic
beverages.');
INSERT INTO Menu (menu_id, menu_name, description) VALUES (Menu_id_seq.nextval, 'Kids Menu', 'Serves kid-friendly meals such as chicken fingers, mac and cheese, and
grilled cheese sandwiches.');


SELECT * FROM Menu;
```

Results Explain Describe Saved SQL History

MENU_ID	MENU_NAME	DESCRIPTION
1	Breakfast Menu	Serves breakfast items such as pancakes, waffles, and eggs.
2	Lunch Menu	Serves a variety of sandwiches, salads, and soups.
3	Dinner Menu	Serves entrees such as steak, seafood, and pasta.
4	Dessert Menu	Serves sweet treats such as cakes, pies, and ice cream.
5	Beverage Menu	Serves a variety of drinks such as coffee, tea, and alcoholic beverages.
6	Kids Menu	Serves kid-friendly meals such as chicken fingers, mac and cheese, and grilled cheese sandwiches.

6 rows returned in 0.00 seconds CSV Export

Orders:



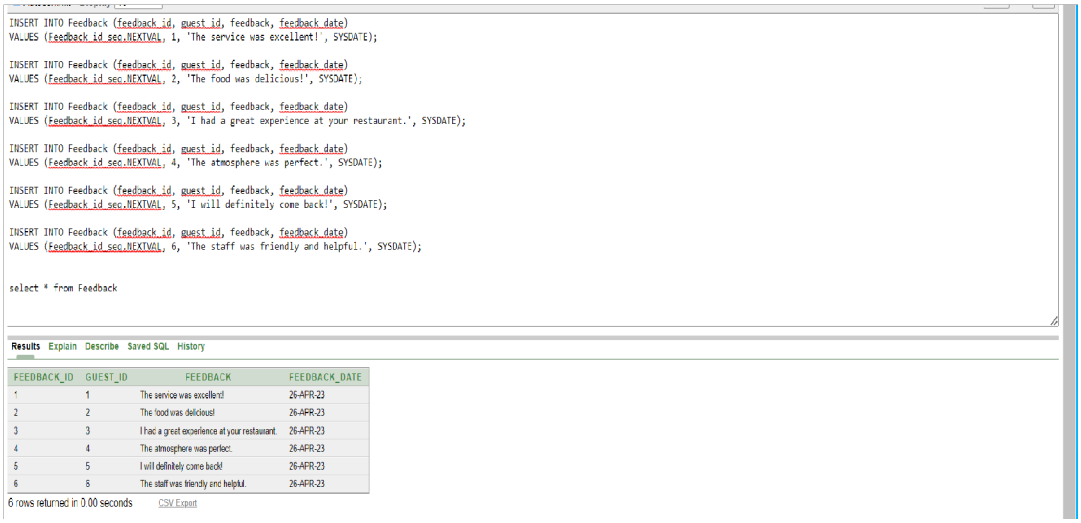
```
INSERT INTO Orders (order_id, guest_id, menu_id, order_time, quantity, total_price)
VALUES (orders_id_seq.NEXTVAL, 1, 2, TO_DATE('2023-04-26', 'YYYY-MM-DD'), 2, 25.99);

INSERT INTO Orders (order_id, guest_id, menu_id, order_time, quantity, total_price)
VALUES (orders_id_seq.NEXTVAL, 1, 3, TO_DATE('2023-04-26', 'YYYY-MM-DD'), 1, 12.50);

INSERT INTO Orders (order_id, guest_id, menu_id, order_time, quantity, total_price)
VALUES (orders_id_seq.NEXTVAL, 2, 1, TO_DATE('2023-04-26', 'YYYY-MM-DD'), 3, 35.97);

INSERT INTO Orders (order_id, guest_id, menu_id, order_time, quantity, total_price)
VALUES (orders_id_seq.NEXTVAL, 2, 2, TO_DATE('2023-04-26', 'YYYY-MM-DD'), 1, 12.99);

INSERT INTO Orders (order_id, guest_id, menu_id, order_time, quantity, total_price)
VALUES (orders_id_seq.NEXTVAL, 3, 3, TO_DATE('2023-04-26', 'YYYY-MM-DD'), 2, 25.00);

INSERT INTO Orders (order_id, guest_id, menu_id, order_time, quantity, total_price)
VALUES (orders_id_seq.NEXTVAL, 3, 1, TO_DATE('2023-04-26', 'YYYY-MM-DD'), 1, 11.99);

select * from orders
```

Results Explain Describe Saved SQL History

ORDER_ID	GUEST_ID	MENU_ID	ORDER_TIME	QUANTITY	TOTAL_PRICE
3	1	2	26-APR-23	2	25.99
4	1	3	26-APR-23	1	12.5
5	2	1	26-APR-23	3	35.97
6	2	2	26-APR-23	1	12.99
7	3	3	26-APR-23	2	25
8	3	1	26-APR-23	1	11.99

6 rows returned in 0.00 seconds CSV Export

Feedback:



```
INSERT INTO Feedback (feedback_id, guest_id, feedback, feedback_date)
VALUES (feedback_id_seq.NEXTVAL, 1, 'The service was excellent!', SYSDATE);

INSERT INTO Feedback (feedback_id, guest_id, feedback, feedback_date)
VALUES (feedback_id_seq.NEXTVAL, 2, 'The food was delicious!', SYSDATE);

INSERT INTO Feedback (feedback_id, guest_id, feedback, feedback_date)
VALUES (feedback_id_seq.NEXTVAL, 3, 'I had a great experience at your restaurant.', SYSDATE);

INSERT INTO Feedback (feedback_id, guest_id, feedback, feedback_date)
VALUES (feedback_id_seq.NEXTVAL, 4, 'The atmosphere was perfect.', SYSDATE);

INSERT INTO Feedback (feedback_id, guest_id, feedback, feedback_date)
VALUES (feedback_id_seq.NEXTVAL, 5, 'I will definitely come back!', SYSDATE);

INSERT INTO Feedback (feedback_id, guest_id, feedback, feedback_date)
VALUES (feedback_id_seq.NEXTVAL, 6, 'The staff was friendly and helpful.', SYSDATE);


select * from Feedback
```

Results Explain Describe Saved SQL History

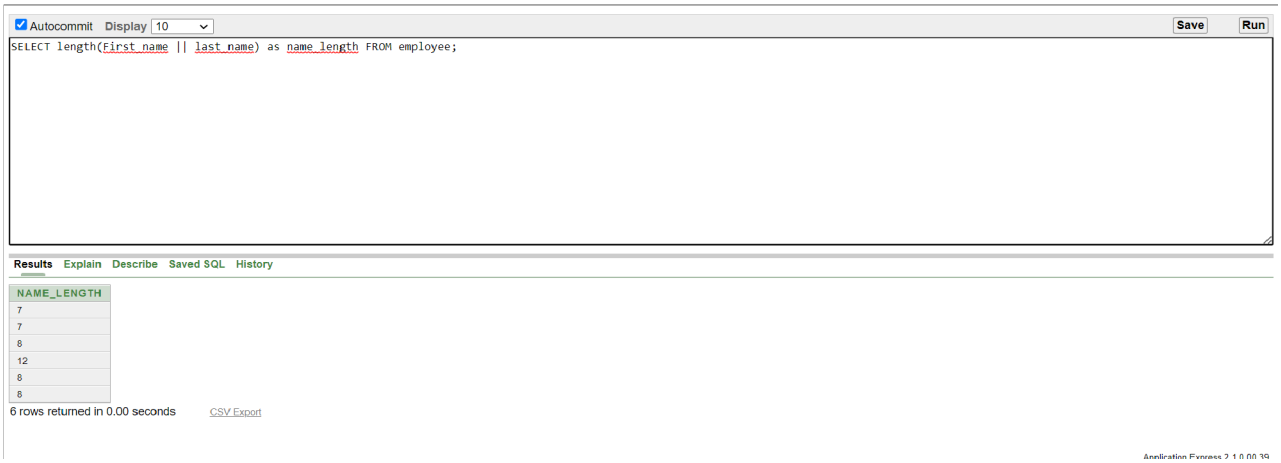
FEEDBACK_ID	GUEST_ID	FEEDBACK	FEEDBACK_DATE
1	1	The service was excellent!	26-APR-23
2	2	The food was delicious!	26-APR-23
3	3	I had a great experience at your restaurant.	26-APR-23
4	4	The atmosphere was perfect.	26-APR-23
5	5	I will definitely come back!	26-APR-23
6	6	The staff was friendly and helpful.	26-APR-23

6 rows returned in 0.00 seconds CSV Export

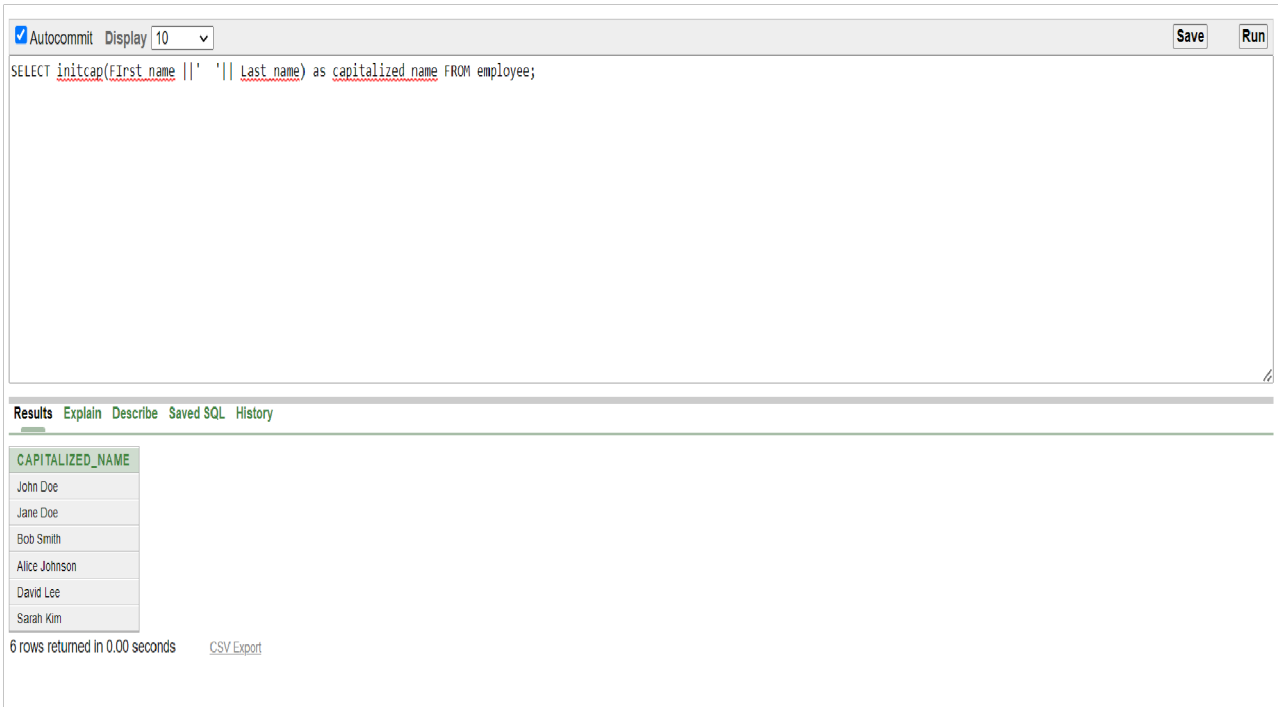
Query Writing

Single-Row-Function:

1.How can I select the length of the string for each employee's name from the employee table?



2. How can I capitalize the first letter of each employee's name from the employee table?



Group Function:

1.Find the average total price of all orders.

```
SELECT AVG(total price) AS average total price
FROM Orders;
```

Results Explain Describe Saved SQL History

AVERAGE_TOTAL_PRICE
20.74

1 rows returned in 0.02 seconds CSV Export

2.Find the maximum price of all services.

```
SELECT MAX(price) AS max service price
FROM Service;
```

Results Explain Describe Saved SQL History

MAX_SERVICE_PRICE
100

1 rows returned in 0.02 seconds CSV Export

Subquery:

1.Find all the reservations made by the guest with email 'example@example.com

```
SELECT *
FROM Reservation
WHERE guest_id = (
    SELECT guest_id
    FROM Guest
    WHERE email = 'davidlee@email.com'
);
```

Results Explain Describe Saved SQL History

RESERVATION_ID	GUEST_ID	ROOM_ID	START_DATE	END_DATE	TOTAL_PRICE
5	5	5	20-MAY-23	25-MAY-23	800

1 rows returned in 0.00 seconds CSV Export

2.Find all the rooms of type 'Single' that are currently available for reservation

```
SELECT *
FROM Room
WHERE room_type_id = (
    SELECT room_type_id
    FROM Room Type
    WHERE type_name = 'Single'
)
AND room_status_id = 1;
```

Results Explain Describe Saved SQL History

ROOM_ID	ROOM_NUMBER	ROOM_TYPE_ID	ROOM_STATUS_ID
1	101	1	1
2	102	1	1

2 rows returned in 0.00 seconds CSV Export

Joining:

1.Find the feedback given by the guest with guest_id = 1.

```
SELECT f.Feedback
FROM Feedback f
INNER JOIN Guest g ON f.guest_id = g.guest_id
WHERE g.guest_id = 1;
```

Results Explain Describe Saved SQL History

FEEDBACK
The service was excellent!

1 rows returned in 0.02 seconds CSV Export

2.Find all the orders made by the guest with email 'example@example.com'.

```
SELECT o.*
FROM Orders o
INNER JOIN Guest g ON o.guest_id = g.guest_id
WHERE g.email = 'johndoe@email.com';
```

Results Explain Describe Saved SQL History

ORDER_ID	GUEST_ID	MENU_ID	ORDER_TIME	QUANTITY	TOTAL_PRICE
3	1	2	26-APR-23	2	25.99
4	1	3	26-APR-23	1	12.5

2 rows returned in 0.00 seconds CSV Export

View:

1.Create a view of all the rooms of type 'Double'.

```
CREATE VIEW Double_Rooms AS
SELECT *
FROM Room
WHERE room_type_id = (
    SELECT room_type_id
    FROM Room_Type
    WHERE type_name = 'Double'
);

DESCRIBE Double_Rooms;
```

Results Explain Describe Saved SQL History

Object: Type **VIEW** Object **DOUBLE_ROOMS**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
DOUBLE_ROOMS	ROOM_ID	Number	-	-	0	-	-	-	-
	ROOM_NUMBER	Varchar2	10	-	-	-	-	-	-
	ROOM_TYPE_ID	Number	-	-	0	-	-	-	-
	ROOM_STATUS_ID	Number	-	-	0	-	-	-	-

1 - 4

2.Create a view of all the services with a price greater than 50.

```
CREATE VIEW Expensive_Services AS
SELECT *
FROM Service
WHERE price > 50;

SELECT *
FROM Expensive_Services;
```

Results Explain Describe Saved SQL History

SERVICE_ID	SERVICE_NAME	SERVICE_TYPE_ID	DESCRIPTION	PRICE	RESERVATION_ID
1	Massage	6	Full body massage	100	6

1 rows returned in 0.00 seconds CSV Export

Relation Algebra

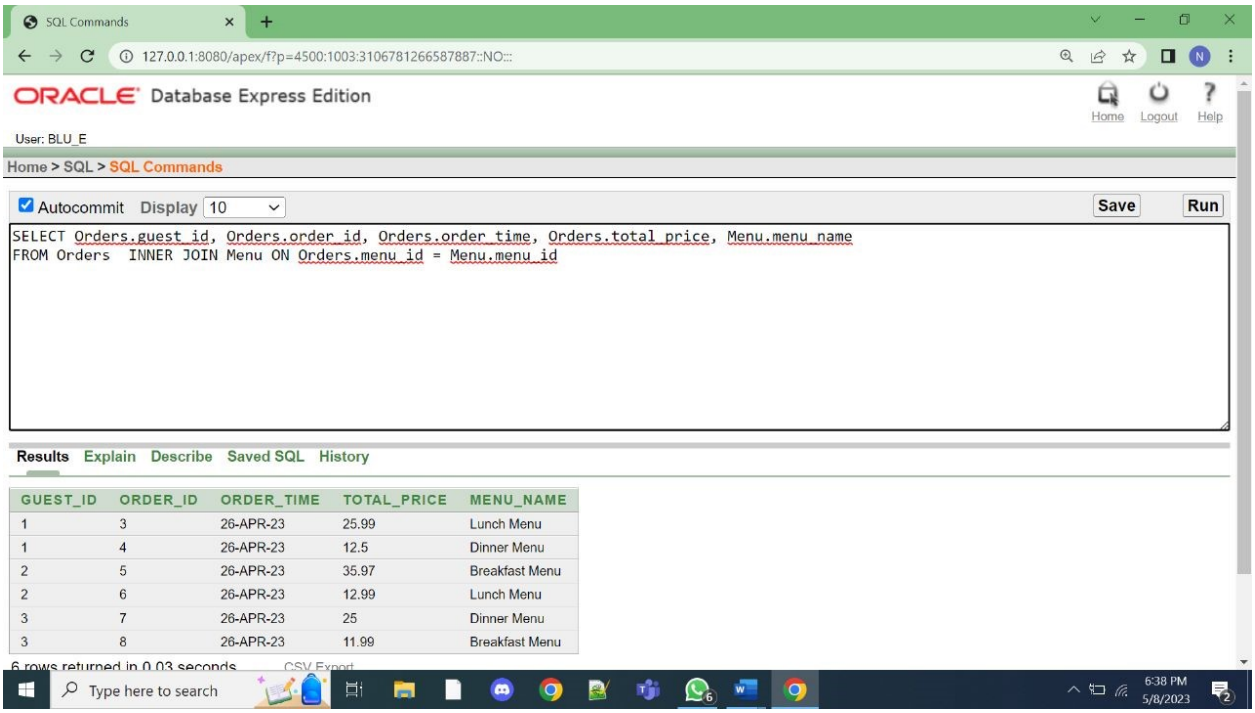
Inner-join:

Π(guest_id, order_id, order_time, total_price, menu_name)(Orders ⋈_{Orders.menu_id = Menu.menu_id} Menu)

“ SELECT Orders.guest_id, Orders.order_id, Orders.order_time, Orders.total_price, Menu.menu_name

FROM Orders INNER JOIN Menu ON Orders.menu_id = Menu.menu_id ”

It combines the Orders table and the Menu table using the menu_id column as a join condition, and then selects the specified columns (guest_id, order_id, order_time, total_price, and menu_name) from the resulting table.

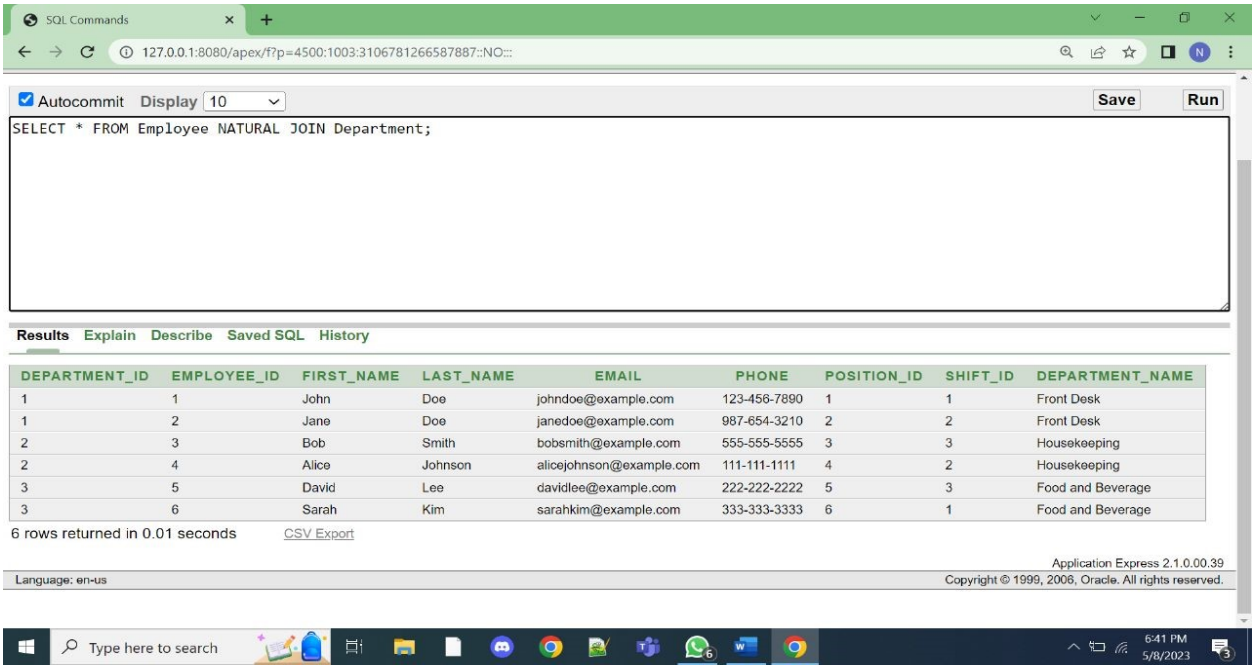


natural join :

Employee ⋈ Department

SELECT * FROM Employee NATURAL JOIN Department;

A natural join returns all the rows from both tables where the values in the common attribute(s) match. It eliminates duplicate columns and retains only one copy of each matched attribute.So, the resulting table will have all the attributes from both tables, with duplicate columns removed, and only the rows where the values in the common attribute(s) match.

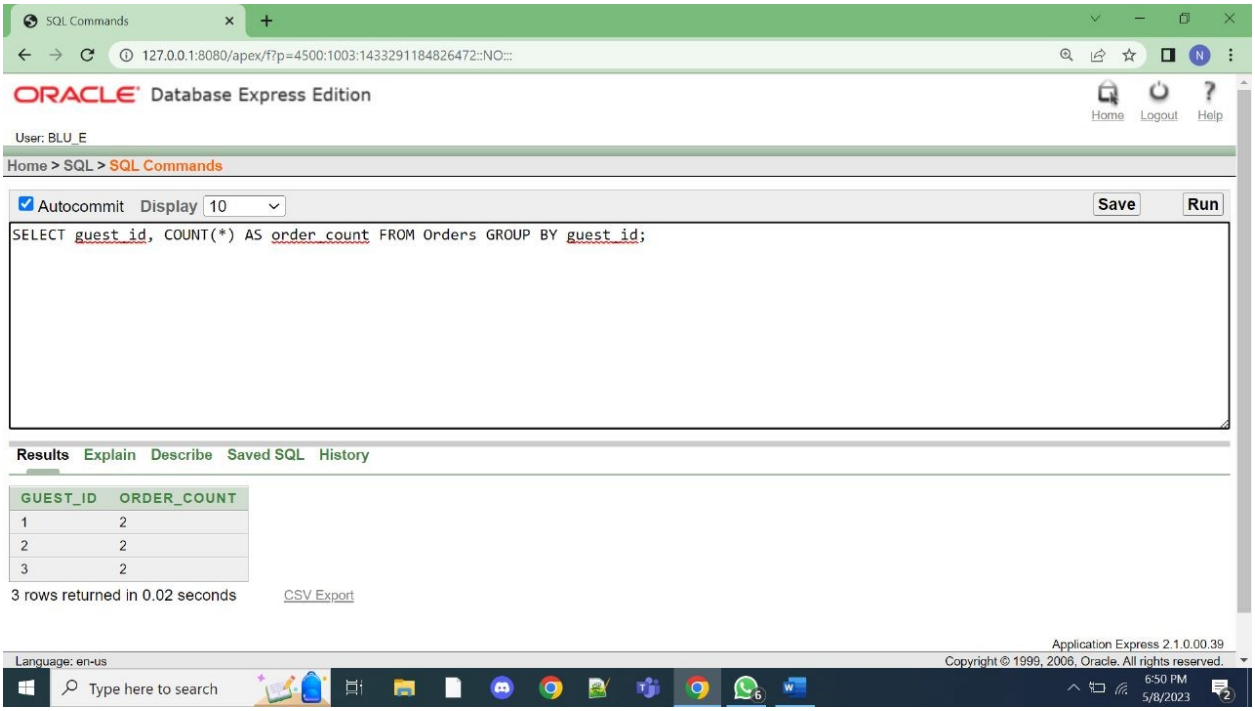


Groping:

Γ(guest_id, COUNT(*):order_count)(Orders)

SELECT guest_id, COUNT(*) AS order_count FROM Orders GROUP BY guest_id;

This operation groups the Orders table by guest_id and counts the number of rows (orders) associated with each guest_id. The result is a new table with two columns: guest_id and order_count.

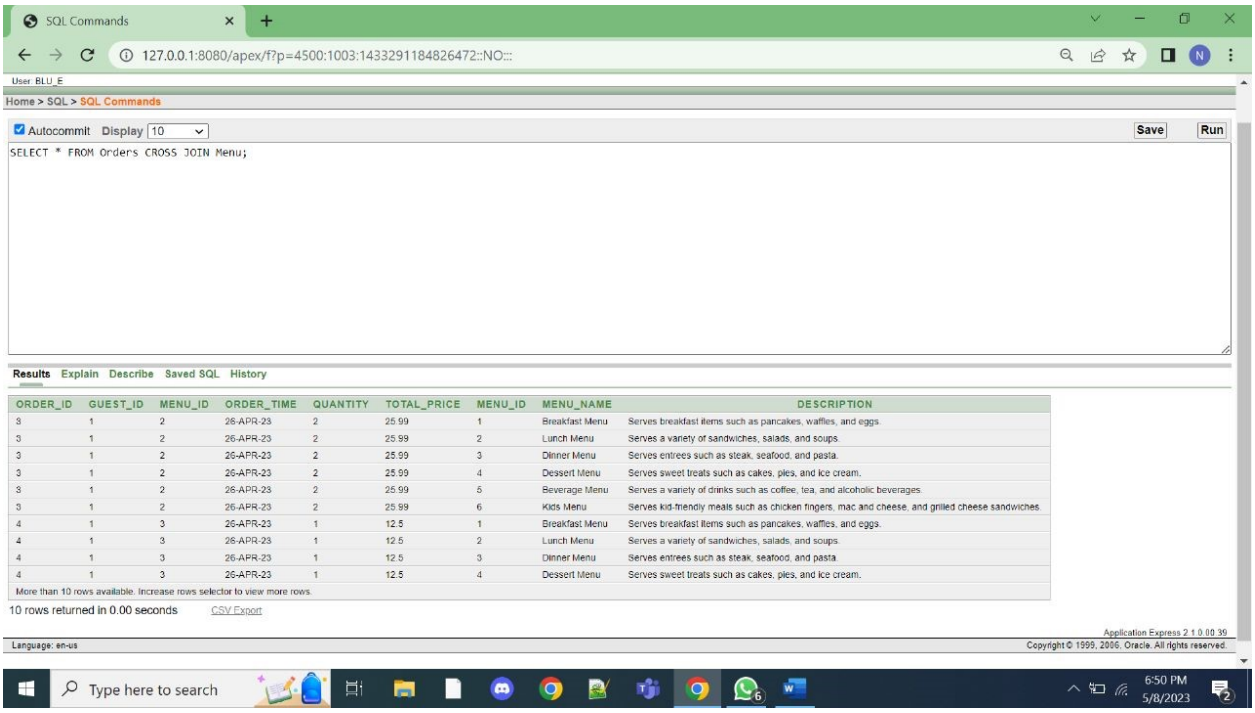


Cartesian product:

Orders × Menu

SELECT * FROM Orders CROSS JOIN Menu;

The Cartesian product of the Orders and Menu tables would result in a new table with the combination of all rows from the Orders table with all rows from the Menu table. The resulting table would have the columns: order_id, guest_id, menu_id, order_time, quantity, total_price, menu_name, and description.

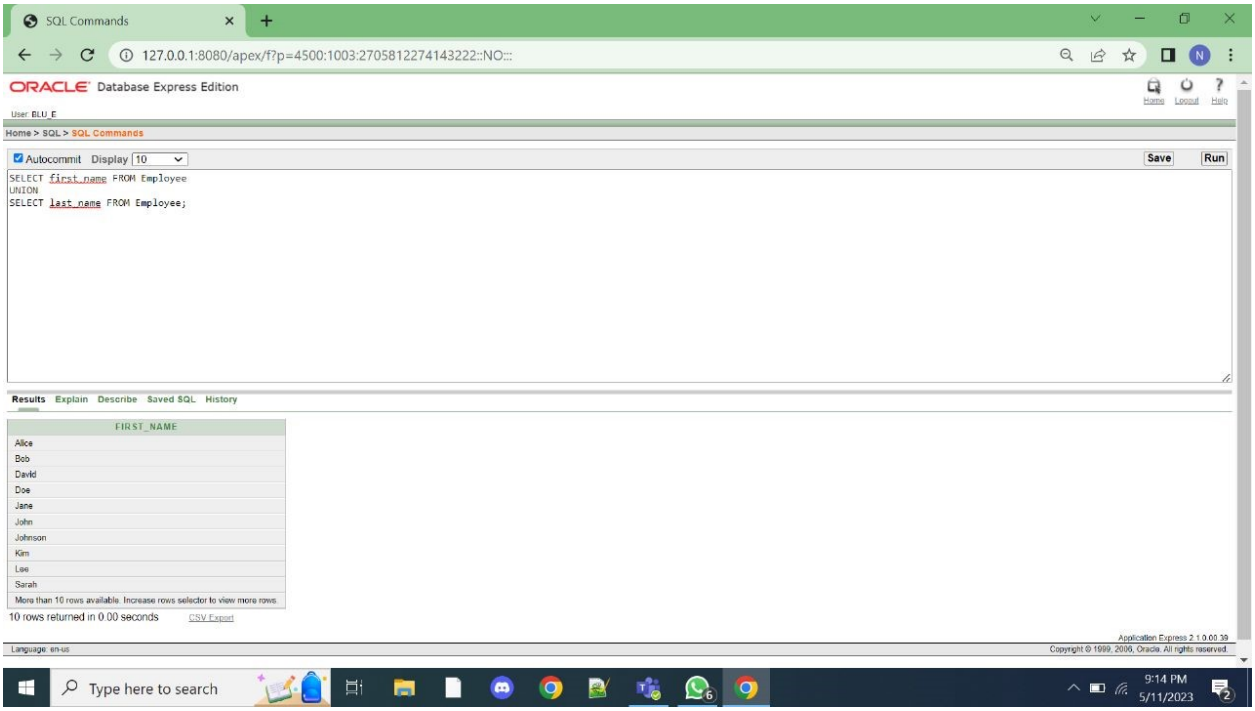


Union oparetion:

π(first_name)(Employee) ∪ π(last_name)(Employee)

SELECT first_name FROM Employee UNION SELECT last_name FROM Employee;

Here, the π symbol represents the projection operator which selects only the specified columns from the Employee table. The ∪ symbol represents the union operator which combines the results of the two projections, ensuring that duplicates are removed. So, this expression first selects the first_name column from the Employee table, then selects the last_name column from the same table, and finally combines the results and removes duplicates to produce a list of unique first and last names.



Conclusion

In conclusion, the hotel management system designed using SQL provides an efficient and reliable solution for managing hotel operations. The project successfully addressed the key requirements for managing guest bookings, room availability, employee scheduling and payment processing, service orders, and feedback collection. By utilizing SQL, the system ensures data accuracy and consistency, which is essential for providing excellent customer service and running a successful hotel.

Looking ahead, future work could include expanding the system's functionality by integrating with other software and platforms, such as online booking portals and payment gateways. Additionally, the system could incorporate machine learning algorithms to predict room occupancy and optimize room rates based on demand. The inclusion of mobile applications for both guests and staff could also improve the overall user experience.

Overall, by continuously improving and updating the hotel management system, hotel owners and staff can ensure the smooth and efficient operation of their business, leading to increased revenue and guest satisfaction.