



**AMERICAN INTERNATIONAL UNIVERSITY-BANGLADESH**  
**FACULTY OF SCIENCE AND TECHNOLOGY**  
**DEPARTMENT OF COMPUTER SCIENCE**

**SPRING 2024-25**

**Project Name: Lawyer Appointment Management System**

**Course Name: Advance Database Management System**

**Section: C**

Group Member

Name	ID	Contribution	Topics
Md. Shohanur Rahman shohan	22-46013-1	25%	Introduction, Scenario Description, Table Creation, Data Insertion, Basic PL/SQL(ALL), Advance PL/SQL
Farjana Yesmin Opi	22-47018-1	25%	Normalization, Finalization, Advance PL/SQL
Most. Sayma Khatun	22-47035-1	25%	Normalization, Finalization, Basic PL/SQL
A. F. M. Rafiul Hassan	22-47048-1	25%	User Interface, ER Diagram, Conclusion

<b>Contents</b>	<b>Page</b>
<b>Introduction</b>	<b>3</b>
<b>User Interface Planning</b>	<b>3 - 9</b>
<b>Scenario Description</b>	<b>9 - 10</b>
<b>E-R Diagram</b>	<b>10</b>
<b>Normalization</b>	<b>11 - 30</b>
<b>Table Creation</b>	<b>31 - 51</b>
<b>Data Insertion</b>	<b>52 - 63</b>
<b>Basic PL/SQL</b>	<b>64 - 81</b>
<b>2 variables</b>	
<b>2 operators</b>	
<b>2 single-row function</b>	
<b>2 group function</b>	
<b>2 loop</b>	
<b>2 conditional statements</b>	
<b>2 subquery</b>	
<b>2 joining</b>	
<b>Advance PL/SQL</b>	
<b>2 stored function.....</b>	
<b>2 stored procedure.....</b>	
<b>2 table-based record.....</b>	
<b>2 explicit cursor.....</b>	
<b>2 cursor-based record.....</b>	
<b>2 row level trigger.....</b>	
<b>2 statement level trigger.....</b>	
<b>2 package.....</b>	
<b>Conclusion.....</b>	

## **Introduction**

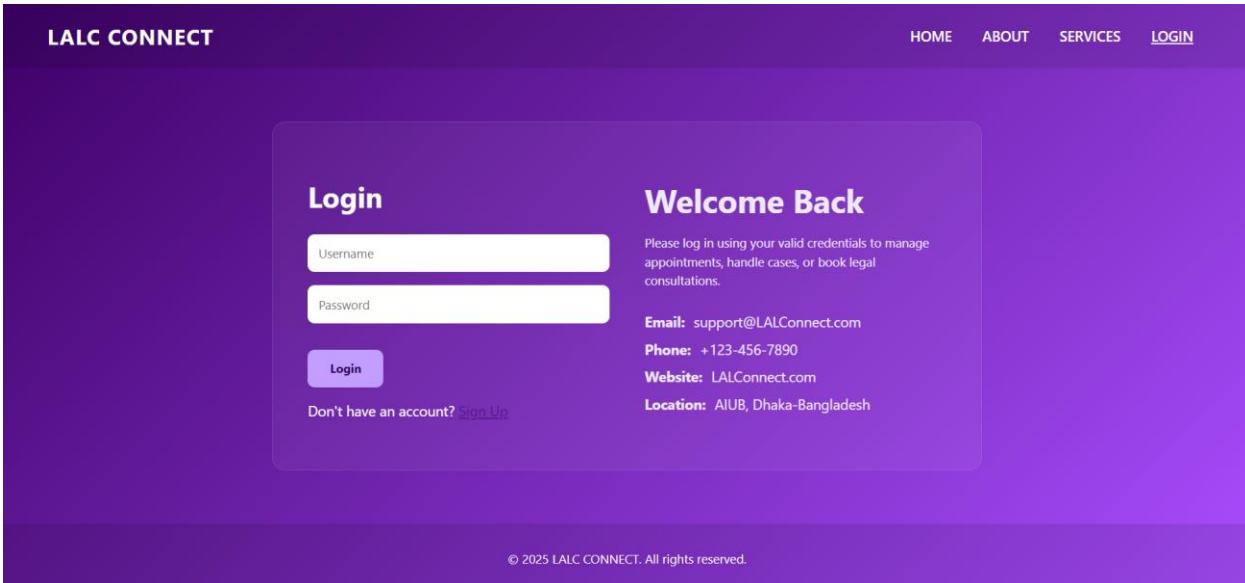
Lawyer Appointment Management System is designed to help people in Bangladesh easily connect with qualified lawyers. It acts as a central system that keeps everything well-organized for both clients and lawyers. The platform helps users find legal support based on specific needs—such as family matters, property disputes, or business issues. It stores key information about clients and lawyers, ensures appointments are scheduled properly, and keeps records of all legal cases and payments. To ensure trust and safety, the system verifies important contact details like email addresses and phone numbers. After a meeting or case, clients can leave feedback about their experience. This helps others find reliable legal support based on honest reviews. In short, this platform simplifies the process of getting legal help by keeping all information secure, organized, and easy to access—for both clients and lawyers.

## User Interface Planning

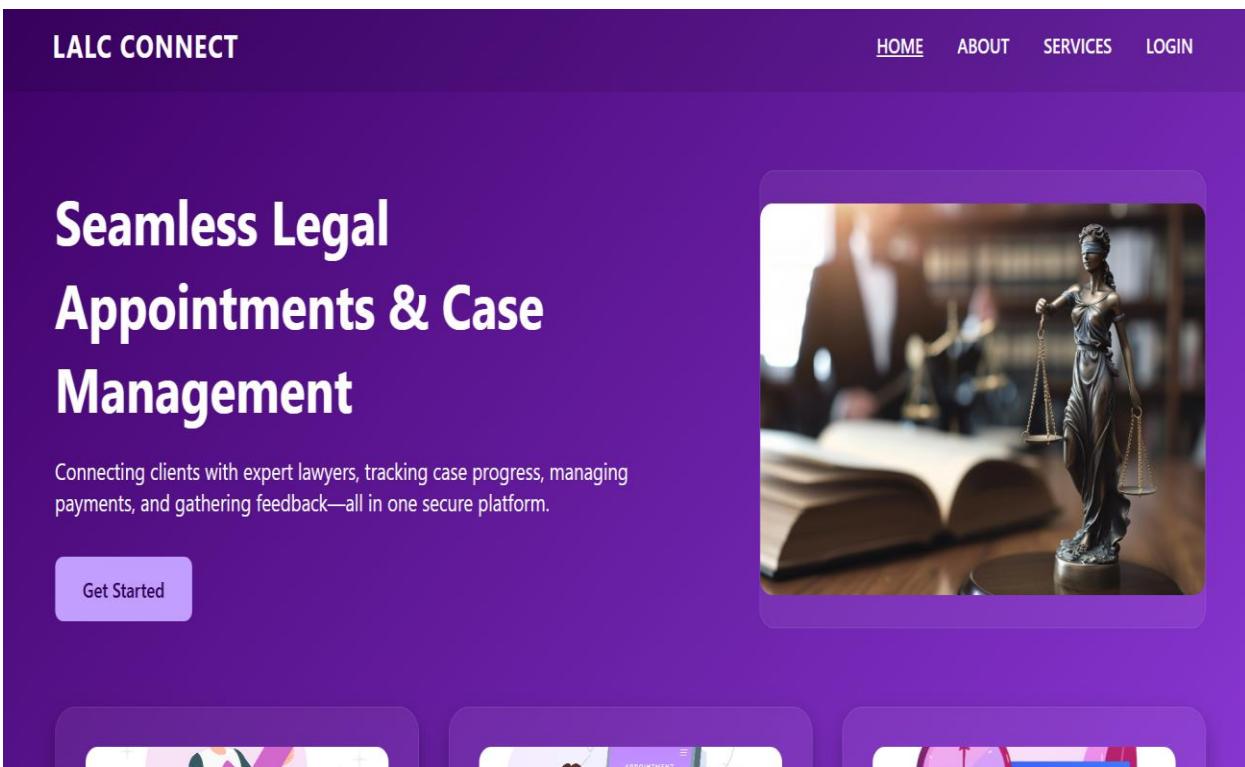
### 1. Registration Page:

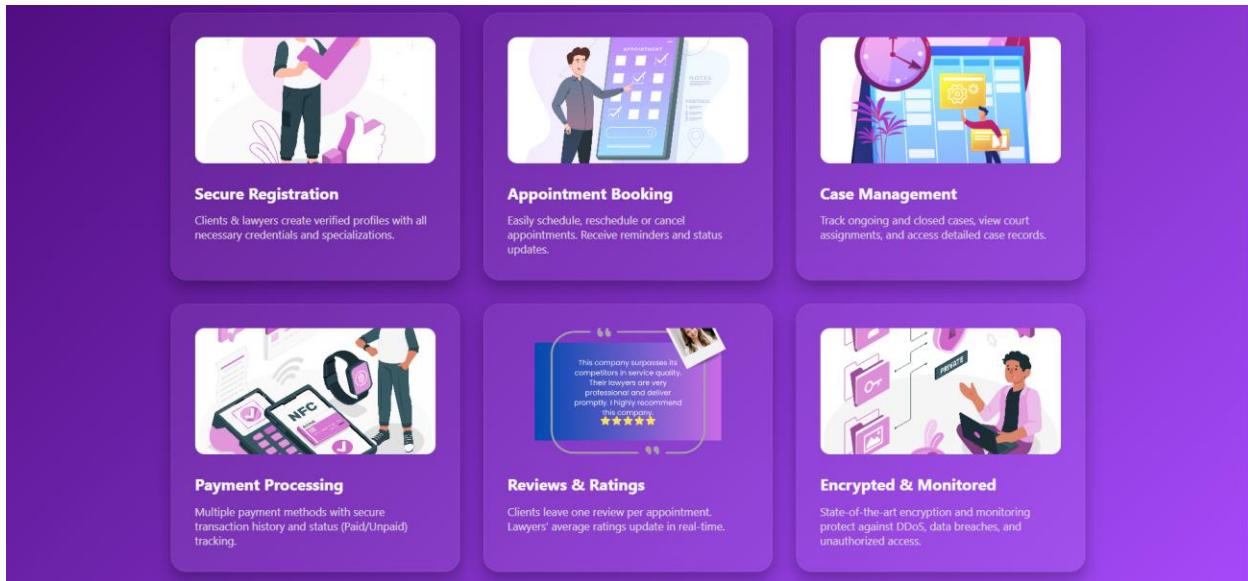
The screenshot shows the registration form for LALC CONNECT. At the top, there's a purple header bar with the LALC CONNECT logo on the left and navigation links for HOME, ABOUT, SERVICES, and LOGIN on the right. Below the header is a large, light-gray rounded rectangular form area. On the left side of this form, the heading "Fill the form" is displayed above several input fields. These include a dropdown menu for "Select User Type", two input fields for "First Name" and "Last Name", an input field for "Email Address", an input field for "Username" with a date picker for "mm/dd/yyyy", an input field for "Password", and a matching input field for "Confirm Password". A blue "Submit" button is located at the bottom left of the form. To the right of the input fields, the word "Registration" is prominently displayed in a large, bold, dark blue font. Below "Registration", a small paragraph of text reads: "Whether you have questions about our services, need support, or want to share your feedback, our dedicated team is here to assist you every step of the way." Further down, there are four lines of contact information: Email (support@LALConnect.com), Phone (+123-456-7890), Website (LALConnect.com), and Location (AIUB, Dhaka-Bangladesh). At the very bottom of the form area, a small link says "Already have an account? [Sign In](#)". At the very bottom of the entire page, a small copyright notice reads "© 2025 LALC CONNECT. All rights reserved."

### 2. Login Page:



3. Homepage:





#### 4. Lawyer's Dashboard:

ID	Client	Date	Time	Status
A010	Alice Lee	2025-05-03	11:00 AM	Confirmed
A011	Bob Khan	2025-05-03	02:00 PM	Pending

Case ID	Type	Status	Client
C001	Family Law	Ongoing	Mrs. Smith
C002	Criminal	Completed	Mr. Johnson
C003	Contract Dispute	In Progress	Mr. Doe

## 5. Client's Dashboard

The dashboard features a sidebar with links for Overview, My Appointments, My Cases, My Payments, My Reviews, and Logout. The main area displays four summary boxes: Upcoming Appointments (3), Ongoing Cases (5), Pending Payments (2), and New Reviews (4). Below these are sections for Recent Appointments (listing A001 and A002) and My Cases (with headers for Case ID, Type, Status, and Court).

## 6. Client Appointments: Clients can book appointments.

The interface shows a sidebar with Overview, My Appointments, My Cases, My Payments, My Reviews, and Logout. The main section allows booking a new appointment with fields for Date (05/05/2025), Time (11:00 AM), Lawyer (dropdown menu showing John Doe (Criminal) and Jane Smith (Civil), with Jane Smith selected), and a Book button. It also displays a list of Your Upcoming Appointments (A001 and A002) with columns for ID, Date, Time, Status, Description, and Action (Review).

## 7. Client Cases: Client can open new cases.

The screenshot shows the LALC CONNECT client interface. On the left, a sidebar menu includes 'Overview', 'My Appointments', 'My Cases' (which is selected and highlighted in blue), 'My Payments', 'My Reviews', and 'Logout'. The main content area has a header 'LALC CONNECT' with links for 'Home', 'Services', 'Register', and 'Login'. Below this is a section titled 'Open New Case' with dropdown menus for 'Type' (Family) and 'Court' (District Court). A 'Create' button is also present. The main area is titled 'My Cases' and displays a table with two rows of case information:

Case ID	Type	Status	Court
C100	Criminal	Ongoing	High Court
C101	Civil	Closed	District Court

At the bottom, a copyright notice reads '© 2025 LALC CONNECT. All rights reserved.'

## 8. Client Payment: Client can pay their appointment/cases payment through their preferred payment method.

The screenshot shows the LALC CONNECT client interface. The sidebar menu is identical to the previous screenshot. The main content area has a header 'LALC CONNECT' with links for 'Home', 'Services', 'Register', and 'Login'. Below this is a section titled 'Make a Payment' with fields for 'Appointment ID' (A002) and 'Amount' (\$1000). A dropdown menu for 'Method' shows options: 'Method', 'Cash', 'Mobile Banking' (which is selected and highlighted in blue), and 'Bank Transfer'. A 'Pay' button is located next to the dropdown. The main area is titled 'My Payments' and displays a table with two rows of payment information:

Payment ID	Amount	Date	Status
P500	\$150	2025-05-01	Paid
P501	\$200	2025-05-03	Unpaid

At the bottom, a copyright notice reads '© 2025 LALC CONNECT. All rights reserved.'

9. Reviews: Client can rate/feedback their specific cases and selected lawyers.

**LALC CONNECT**

**Leave a Review**

A002 - 2025-05-12      5      Very Professional      Submit Review

**My Reviews**

Review ID	Date	Rating	Comments
R300	2025-04-30	5	Excellent service
R301	2025-05-02	4	Very helpful

© 2025 LALC CONNECT. All rights reserved.

10. About-us Page:

**LALC CONNECT**

**ABOUT**

## About Us

Our team is committed to helping clients navigate their legal journeys smoothly and effectively. With a passion for justice and innovation, we provide a platform that connects clients to experienced lawyers, tracks case progress, and simplifies the entire legal experience.



**Olivia Wilson**  
Client Relations Manager



**Rahul Sharma**  
Lead Legal Advisor



**Priya Mehta**  
Sr. Graphic Designer



**Daniel Kim**  
Backend Developer

## 11. Our Services Page:

The screenshot shows a dark-themed website for 'LALC CONNECT'. At the top, there's a navigation bar with links for 'HOME', 'ABOUT', 'SERVICES', and 'LOGIN'. Below the navigation, the title 'Our Legal Services' is centered. Three service categories are displayed in rounded rectangular boxes: 'Client Services' (image of a hand holding a network of people icons), 'Lawyer Services' (image of two lawyers at a desk with a gavel), and 'System Features' (image of a laptop screen with a checkmark icon). Each category has a list of features below it. At the bottom of the page, a copyright notice reads '© 2025 LALC CONNECT. All rights reserved.'

- Client Services**
  - Client registration & profile
  - Book and manage appointments
  - Track legal case status
  - Make secure payments
- Lawyer Services**
  - Lawyer registration
  - Appointment handling
  - Case assignment & updates
  - Client feedback integration
- System Features**
  - Secure login & session
  - Court & case mapping
  - Rating & review tracking
  - Admin-level control features

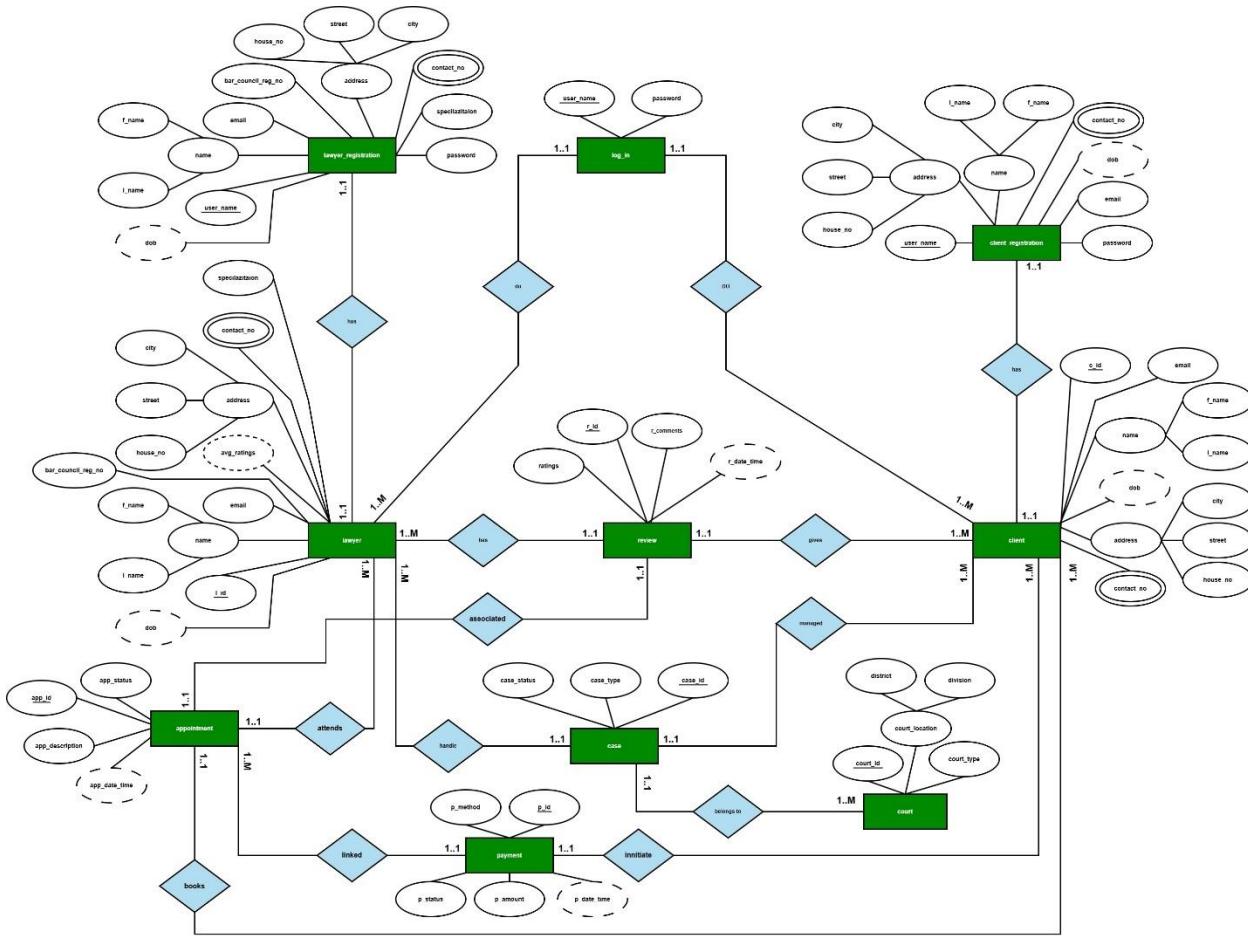
© 2025 LALC CONNECT. All rights reserved.

## Scenario Description

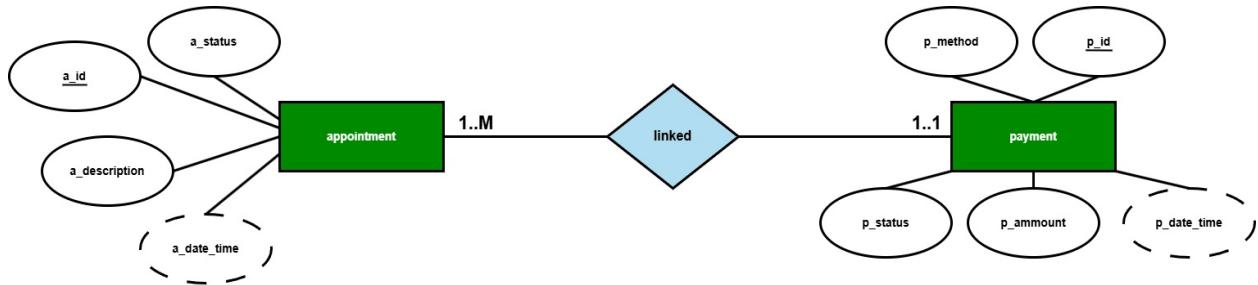
In a lawyer appointment management system, a client can register by providing a unique username, first name, last name, date of birth, contact number, email, and password. Each address consists of a city name, street number, and house number. A lawyer can also register using the same credentials, with the addition of a bar registration number and his/her specialization. Each client and lawyer can do log in using their valid username and password. After Log In each client and lawyer Identified with unique ID and other entities. A client can book multiple appointments, and each appointment stores the unique appointment ID, date, time, status, and an optional description, while a lawyer attends appointments from many clients. One lawyer can handle multiple appointments and cases, but each appointment is associated with exactly one lawyer and one client. A legal case is identified by a case ID and contains case type, case description, and case status (either Ongoing or Closed). Each case is managed by a client and handled by a lawyer, and every case must belong to exactly one court. Courts are identified by a court ID, and the system also stores the court type (Supreme, High, or District), court location, division, and district information. Payments are managed carefully in the system and are recorded with payment ID, amount, date, method (Cash, Mobile Banking, or Bank Transfer), and status (Paid or Unpaid). Each payment

must be linked to an appointment. Every payment is always initiated by a client. The review system allows clients to give feedback to lawyers. A review is uniquely identified by an ID and includes the review date, rating (between 1 and 5), and optional comments. A client can give only one review per appointment, preventing duplicate feedback. Reviews are associated with an appointment where lawyers get average ratings updated.

## E-R Diagram



# Normalization



**linked:**

**UNF**

app\_id, app\_status, app\_description, app\_date\_time, p\_id, p\_method, p\_status, p\_amount, p\_date\_time

**1NF:**

No multivalued attribute. (same as unf)

app\_id, app\_status, app\_description, app\_date\_time, p\_id, p\_method, p\_status, p\_amount, p\_date\_time

**2NF:**

1. app\_id, app\_status, app\_description, app\_date\_time
2. p\_id, p\_method, p\_status, p\_amount, p\_date\_time

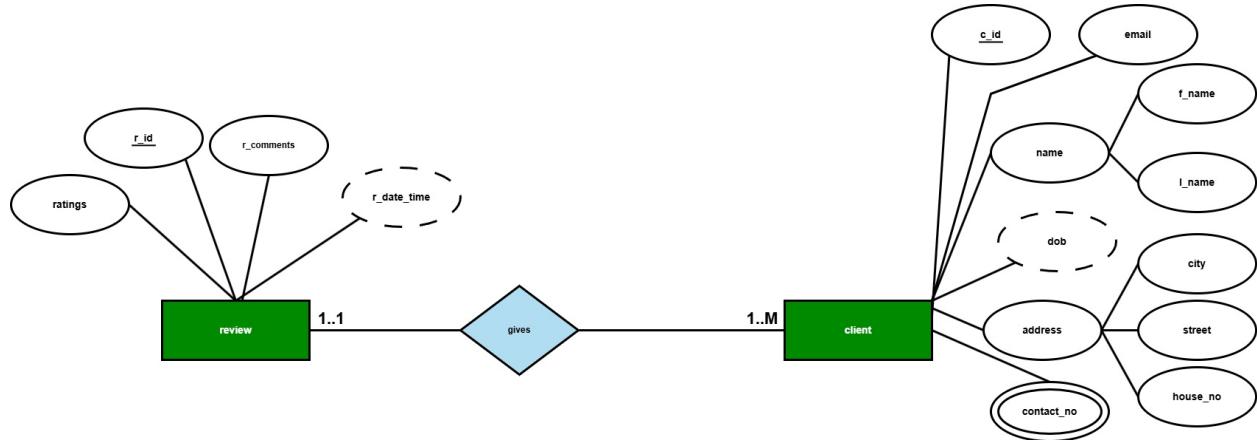
**3NF**

There is no transitive attribute. Relation is already in 3nf.

1. app\_id, app\_status, app\_description, app\_date\_time
2. p\_id, p\_method, p\_status, p\_amount, p\_date\_time

**Table creation:**

1. app\_id, app\_status, app\_description, app\_date\_time
2. p\_id, p\_method, p\_status, p\_ammount, p\_date\_time, **app\_id**



**gives:**

### UNF

r\_id, r\_comment, ratings, r\_date\_time, c\_id, email, f\_name, l\_name, city, street, house\_no, contact\_no, dob

### 1NF

contact\_no is a multivalued attribute.

r\_id, r\_comment, ratings, r\_date\_time, c\_id, email, f\_name, l\_name, city, street, house\_no, contact\_no, dob

### 2NF

1. r\_id, r\_comment, ratings, r\_date\_time
2. c\_id, email, contact\_no, f\_name, l\_name, city, street, house\_no, dob

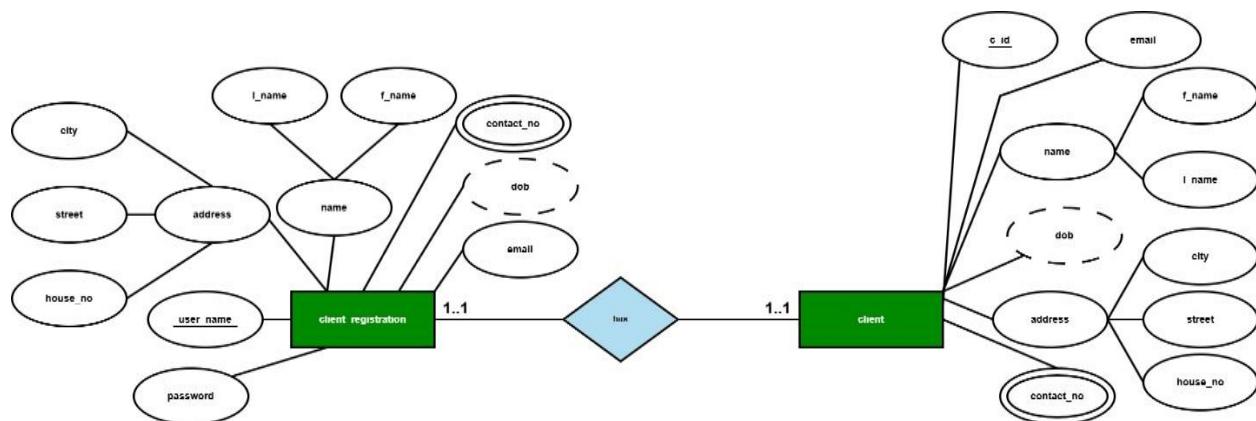
### 3NF

Name and Address are Transitive attribute.

1. r\_id, r\_comment, ratings, r\_date\_time
2. c\_id, email, contact\_no, dob
3. name\_id, f\_name, l\_name
4. a\_id, city, street, house\_no

### **Table creation:**

1. r\_id, r\_comment, ratings, r\_date\_time, **name\_id**, **a\_id**, **c\_id**
2. c\_id, email, contact\_no, dob
3. name\_id, f\_name, l\_name
4. a\_id, city, street, house\_no



**has:**

### **UNF**

user\_name, email, password, dob, contact\_no, f\_name, l\_name, city, street, house\_no, c\_id, email, f\_name, l\_name, city, street, house\_no, contact\_no, dob

### **1NF**

contact\_no is a multivalued attribute.

user\_name, email, password, dob, contact\_no, f\_name, l\_name, city, street, house\_no, c\_id, email, f\_name, l\_name, city, street, house\_no, contact\_no, dob

## 2NF

1. user\_name, email, password, dob, contact\_no, f\_name, l\_name, city, street, house\_no
2. c\_id, email, f\_name, l\_name, city, street, house\_no, contact\_no, dob

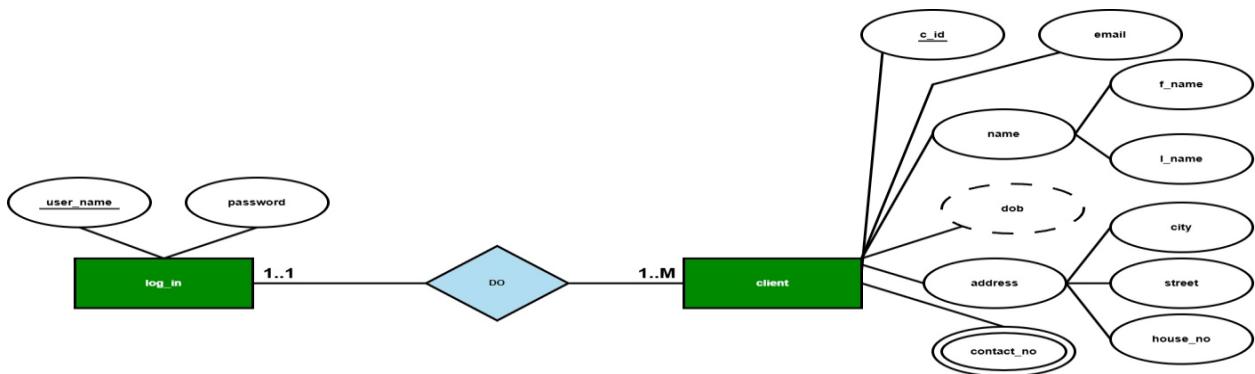
## 3NF

Name and Address are Transitive attribute.

1. user\_name, email, password, dob, contact\_no
2. name\_id, f\_name, l\_name
3. a\_id, city, street, house\_no
4. c\_id, email, contact\_no, dob

## Table creation:

1. user\_name, email, password, dob, contact\_no, name\_id, a\_id, c\_id
2. name\_id, f\_name, l\_name
3. a\_id, city, street, house\_no
4. c\_id, email, contact\_no, dob



**do:**

## UNF

user\_name, password, c\_id, email, f\_name, l\_name, city, street, house\_no, contact\_no, dob

## 1NF

contact\_no is a multivalued attribute.

user\_name, password, c\_id, email, f\_name, l\_name, city, street, house\_no, contact\_no, dob

## 2NF

1. user\_name, password
2. c\_id, email, f\_name, l\_name, city, street, house\_no, contact\_no, dob

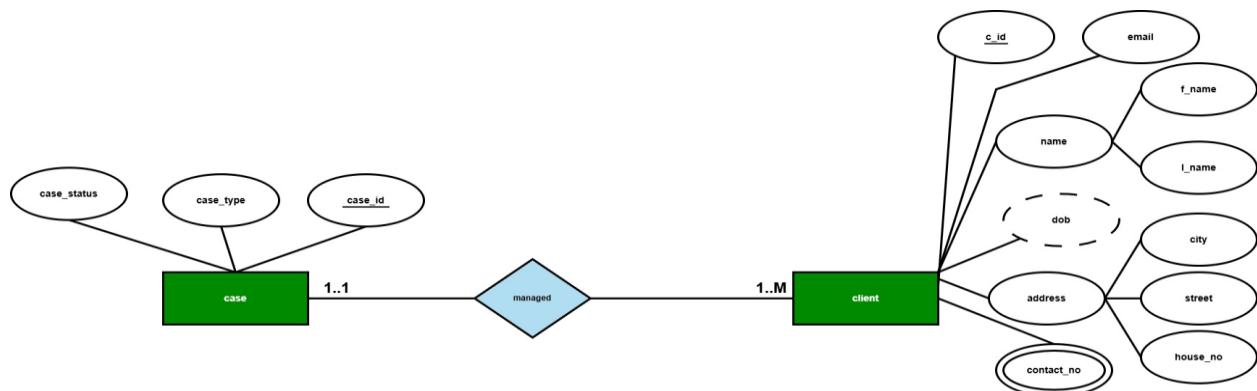
## 3NF

Name and Address are Transitive attribute.

1. user\_name, password
2. c\_id, email, contact\_no, dob
3. name\_id, f\_name, l\_name
4. a\_id, city, street, house\_no

### Table creation:

1. user\_name, password, c\_id, name\_id, a\_id
2. c\_id, email, contact\_no, dob
3. name\_id, f\_name, l\_name
4. a\_id, city, street, house\_no



**managed:**

**UNF**

case\_id, case\_status, case\_type, c\_id, email, f\_name, l\_name, city, street, house\_no, contact\_no, dob

**1NF**

contact\_no is a multivalued attribute.

case\_id, case\_status, case\_type, c\_id, email, f\_name, l\_name, city, street, house\_no, contact\_no, dob

**2NF**

1. case\_id, case\_status, case\_type
2. c\_id, email, f\_name, l\_name, city, street, house\_no, contact\_no, dob

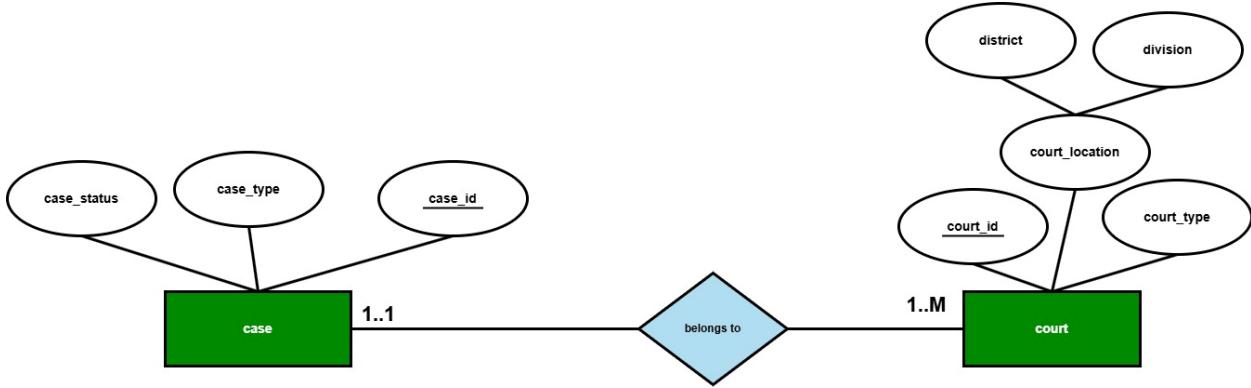
**3NF**

Name and Address are Transitive attribute

1. case\_id, case\_status, case\_type
2. c\_id, email, contact\_no, dob
3. name\_id, f\_name, l\_name
4. a\_id, city, street, house\_no

**Table creation:**

1. case\_id, case\_status, case\_type, , **c\_id, name\_id, a\_id**
2. c\_id, email, contact\_no, dob
3. name\_id, f\_name, l\_name
4. a\_id, city, street, house\_no



### **belongs to:**

#### UNF

case\_id, case\_status, case\_type, court\_id, court\_type, district, division

#### 1NF

No multivalued attribute. (same as unf)

case\_id, case\_status, case\_type, court\_id, court\_type, district, division

#### 2NF

1. case\_id, case\_status, case\_type
2. court\_id, court\_type, district, division

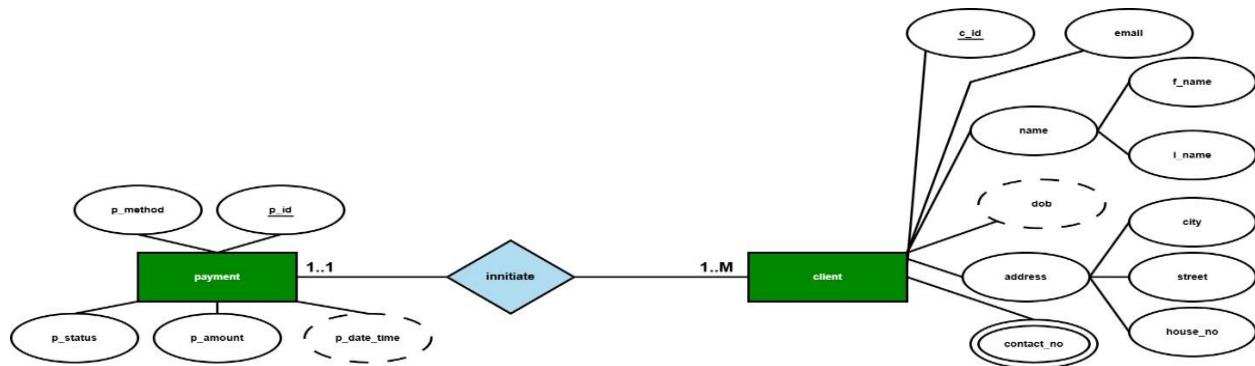
#### 3NF

Court\_Location is a Transitive attribute

1. case\_id, case\_status, case\_type
2. court\_id, court\_type
3. loc\_id, district, division

### **Table creation:**

1. case\_id, case\_status, case\_type, **court\_id**, loc\_id
2. court\_id, court\_type, case\_id
3. loc\_id, district, division



### **initiate:**

#### UNF

p\_id, p\_method, p\_status, p\_amount, p\_date\_time, c\_id, email, f\_name, l\_name, city, street, house\_no, contact\_no, dob

#### 1NF

contact\_no is a multivalued attribute.

p\_id, p\_method, p\_status, p\_amount, p\_date\_time, c\_id, email, f\_name, l\_name, city, street, house\_no, contact\_no, dob

#### 2NF

1. p\_id, p\_method, p\_status, p\_amount, p\_date\_time
2. c\_id, email, f\_name, l\_name, city, street, house\_no, contact\_no, dob

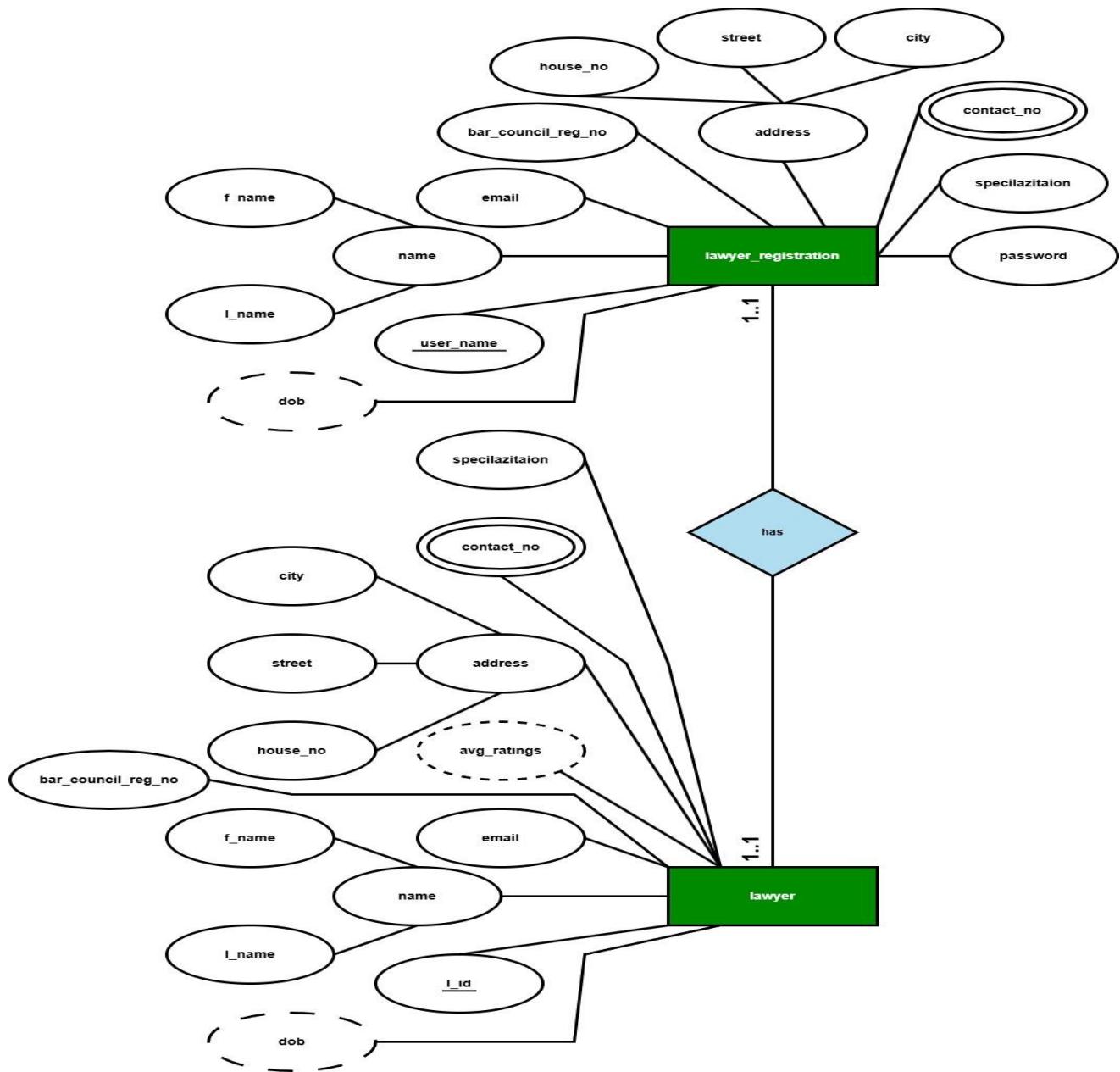
#### 3NF

Name and Address are Transitive attributes

1. p\_id, p\_method, p\_status, p\_amount, p\_date\_time
2. c\_id, email, contact\_no, dob
3. name\_id, f\_name, l\_name
4. a\_id, city, street, house\_no

#### Table creation

1. p\_id, p\_method, p\_status, p\_amount, p\_date\_time, c\_id, name\_id, a\_id
2. c\_id, email, contact\_no, dob
3. name\_id, f\_name, l\_name
4. a\_id, city, street, house\_no



**has:**

## UNF

l\_id, email, bar\_council\_reg\_no, avg\_ratings, specialization, contact\_no, dob, f\_name, l\_name, city, street, house\_no, user\_name, password, email, bar\_council\_reg\_no, specialization, contact\_no, dob, f\_name, l\_name, city, street, house\_no

## 1NF

contact\_no is a multivalued attribute.

l\_id, email, bar\_council\_reg\_no, avg\_ratings, specialization, contact\_no, dob, f\_name, l\_name, city, street, house\_no, user\_name, password, email, bar\_council\_reg\_no, specialization, contact\_no, dob, f\_name, l\_name, city, street, house\_no

## 2NF

1. l\_id, email, bar\_council\_reg\_no, avg\_ratings, specialization, contact\_no, dob, f\_name, l\_name, city, street, house\_no
2. user\_name, password, email, bar\_council\_reg\_no, specialization, contact\_no, dob, f\_name, l\_name, city, street, house\_no

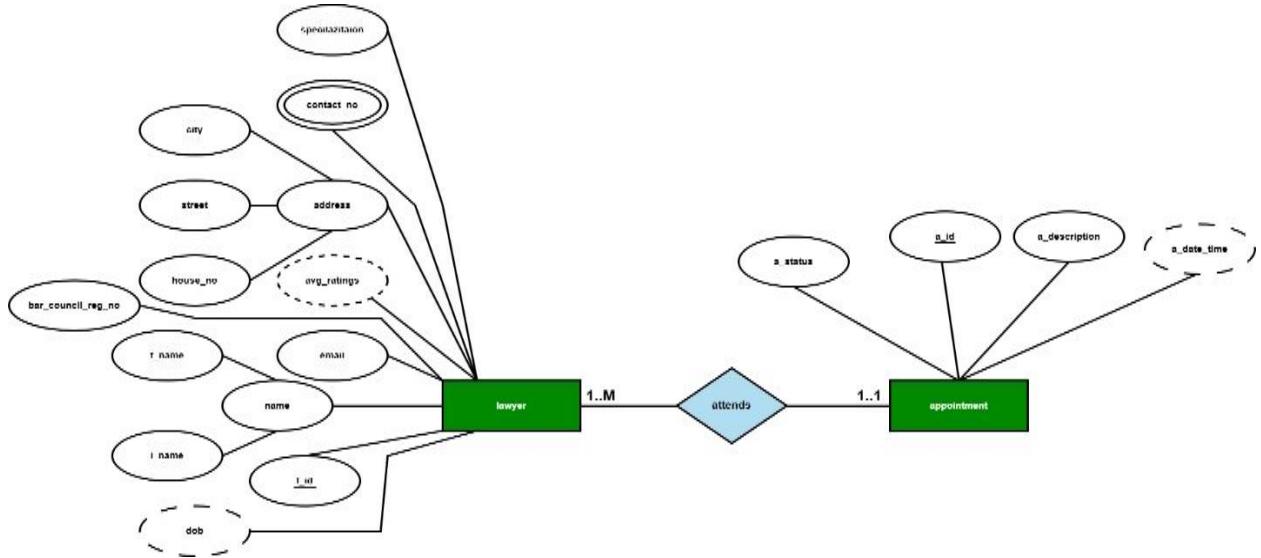
## 3NF

Name and Address are Transitive attribute

1. l\_id, email, bar\_council\_reg\_no, avg\_ratings, specialization, contact\_no, dob
2. name\_id, f\_name, l\_name
3. a\_id, city, street, house\_no
4. user\_name, password, email, bar\_council\_reg\_no, specialization, contact\_no, dob

## Table creation:

1. l\_id, email, bar\_council\_reg\_no, avg\_ratings, specialization, contact\_no, dob, **name\_id**, **a\_id**, **user\_name**
2. name\_id, f\_name, l\_name
3. a\_id, city, street, house\_no
4. user\_name, password, email, bar\_council\_reg\_no, specialization, contact\_no, dob



### attends:

#### UNF

app\_id, app\_status, app\_description, app\_date\_time, l\_id, email, bar\_council\_reg\_no, avg\_ratings, specialization, contact\_no, dob, f\_name, l\_name, city, street, house\_no

#### 1NF

contact\_no is a multivalued attribute.

app\_id, app\_status, app\_description, app\_date\_time, l\_id, email, bar\_council\_reg\_no, avg\_ratings, specialization, contact\_no, dob, f\_name, l\_name, city, street, house\_no

#### 2NF

1. app\_id, app\_status, app\_description, app\_date\_time
2. l\_id, email, bar\_council\_reg\_no, avg\_ratings, specialization, contact\_no, dob, f\_name, l\_name, city, street, house\_no

#### 3NF

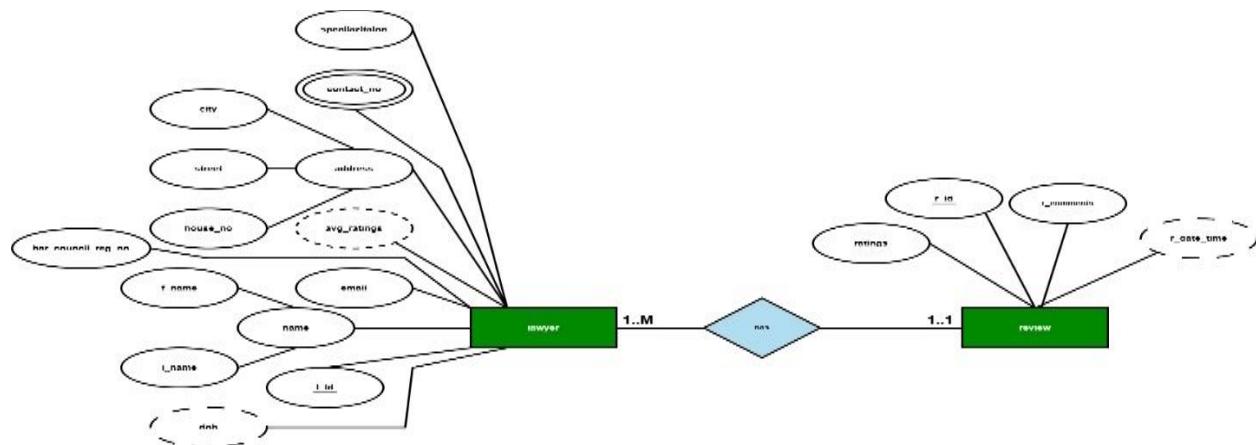
Name and Address are Transitive attribute

1. app\_id, app\_status, app\_description, app\_date\_time

2. name\_id, f\_name, l\_name
3. a\_id, city, street, house\_no
4. l\_id, email, bar\_council\_reg\_no, avg\_ratings, specialization, contact\_no, dob

### **Table creation**

1. app\_id, app\_status, app\_description, app\_date\_time, **name\_id**, **a\_id**, **l\_id**
2. name\_id, f\_name, l\_name
3. a\_id, city, street, house\_no
4. l\_id, email, bar\_council\_reg\_no, avg\_ratings, specialization, contact\_no, dob



**has:**

### **UNF**

r\_id, ratings, r\_comments, r\_date\_time, l\_id, email, bar\_council\_reg\_no, avg\_ratings, specialization, contact\_no, dob, f\_name, l\_name, city, street, house\_no

### **1NF**

contact\_no is a multivalued attribute.

r\_id, ratings, r\_comments, r\_date\_time, l\_id, email, bar\_council\_reg\_no, avg\_ratings, specialization, contact\_no, dob, f\_name, l\_name, city, street, house\_no

## 2NF

1. r\_id, ratings, r\_comments, r\_date\_time
2. l\_id, email, bar\_council\_reg\_no, avg\_ratings, specialization, contact\_no, dob, f\_name, l\_name, city, street, house\_no

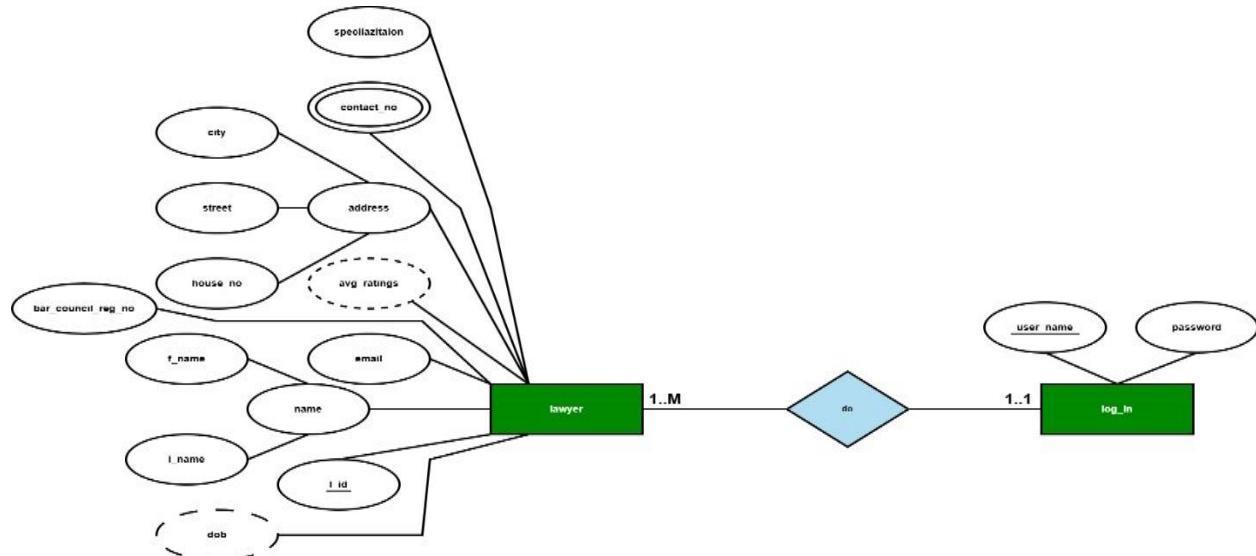
## 3NF

Name and Address are Transitive attribute

1. r\_id, ratings, r\_comments, r\_date\_time
2. name\_id, f\_name, l\_name
3. a\_id, city, street, house\_no
4. l\_id, email, bar\_council\_reg\_no, avg\_ratings, specialization, contact\_no, dob

## Table creation

1. r\_id, ratings, r\_comments, r\_date\_time, **name\_id**, **a\_id**, **l\_id**
2. name\_id, f\_name, l\_name
3. a\_id, city, street, house\_no
4. l\_id, email, bar\_council\_reg\_no, avg\_ratings, specialization, contact\_no, dob



**do:**

**UNF**

user\_name, password, l\_id, email, bar\_council\_reg\_no, avg\_ratings, specialization, contact\_no, dob, f\_name, l\_name, city, street, house\_no

**1NF**

contact\_no is a multivalued attribute.

user\_name, password, l\_id, email, bar\_council\_reg\_no, avg\_ratings, specialization, contact\_no, dob, f\_name, l\_name, city, street, house\_no

**2NF**

1. user\_name, password
2. l\_id, email, bar\_council\_reg\_no, avg\_ratings, specialization, contact\_no, dob, f\_name, l\_name, city, street, house\_no

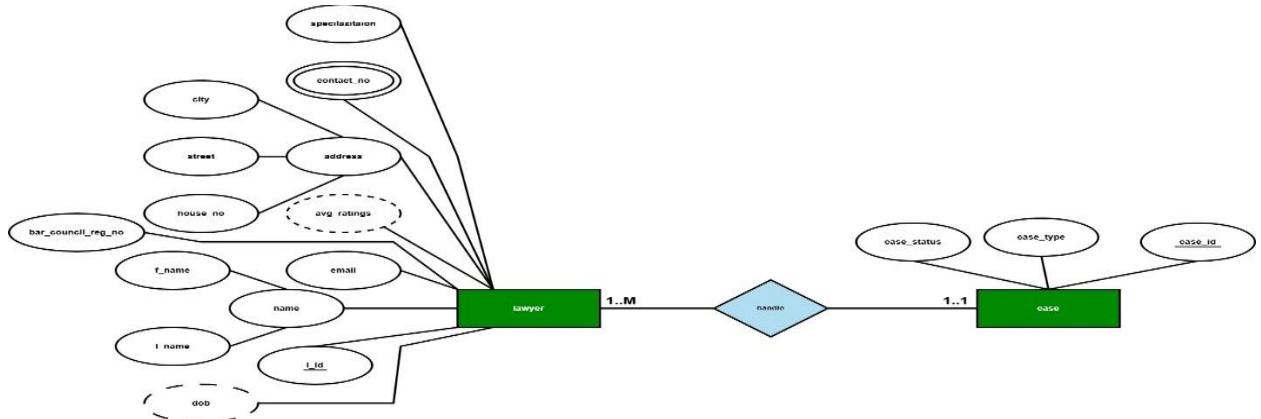
**3NF**

Name and Address are Transitive attributes

1. user\_name, password
2. name\_id, f\_name, l\_name
3. a\_id, city, street, house\_no
4. l\_id, email, bar\_council\_reg\_no, avg\_ratings, specialization, contact\_no, dob

**Table creation:**

1. user\_name, password, **name\_id**, **a\_id**, **l\_id**
2. name\_id, f\_name, l\_name
3. a\_id, city, street, house\_no
4. l\_id, email, bar\_council\_reg\_no, avg\_ratings, specialization, contact\_no, dob



**handle:**

### UNF

case\_id, case\_status, case\_type, l\_id, email, bar\_council\_reg\_no, avg\_ratings, specialization,  
contact\_no, dob, f\_name, l\_name, city, street, house\_no

### 1NF

contact\_no is a multivalued attribute.

case\_id, case\_status, case\_type, l\_id, email, bar\_council\_reg\_no, avg\_ratings, specialization,  
contact\_no, dob, f\_name, l\_name, city, street, house\_no

### 2NF

1. case\_id, case\_status, case\_type
2. l\_id, email, bar\_council\_reg\_no, avg\_ratings, specialization, contact\_no, dob, f\_name,  
l\_name, city, street, house\_no

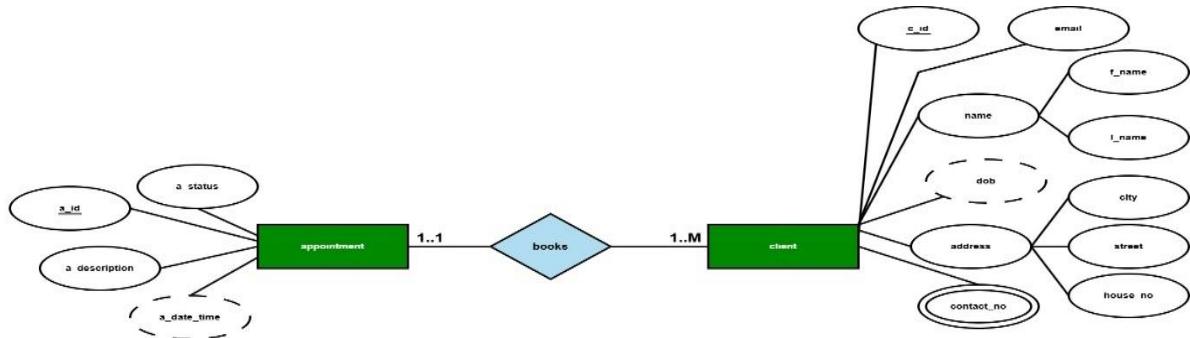
### 3NF

Name and Address are Transitive attribute

1. case\_id, case\_status, case\_type
2. name\_id, f\_name, l\_name
3. a\_id, city, street, house\_no
4. l\_id, email, bar\_council\_reg\_no, avg\_ratings, specialization, contact\_no, dob

## Table creation

1. case\_id, case\_status, case\_type, **name\_id**, **a\_id**, **l\_id**
2. name\_id, f\_name, l\_name
3. a\_id, city, street, house\_no
4. l\_id, email, bar\_council\_reg\_no, avg\_ratings, specialization, contact\_no, dob



## **books:**

### UNF

app\_id, app\_status, app\_description, app\_date\_time, c\_id, email, f\_name, l\_name, city, street, house\_no, contact\_no, dob

### 1NF

contact\_no is a multivalued attribute.

app\_id, app\_status, app\_description, app\_date\_time, c\_id, email, f\_name, l\_name, city, street, house\_no, contact\_no, dob

### 2NF

1. app\_id, app\_status, app\_description, app\_date\_time
2. c\_id, email, f\_name, l\_name, city, street, house\_no, contact\_no, dob

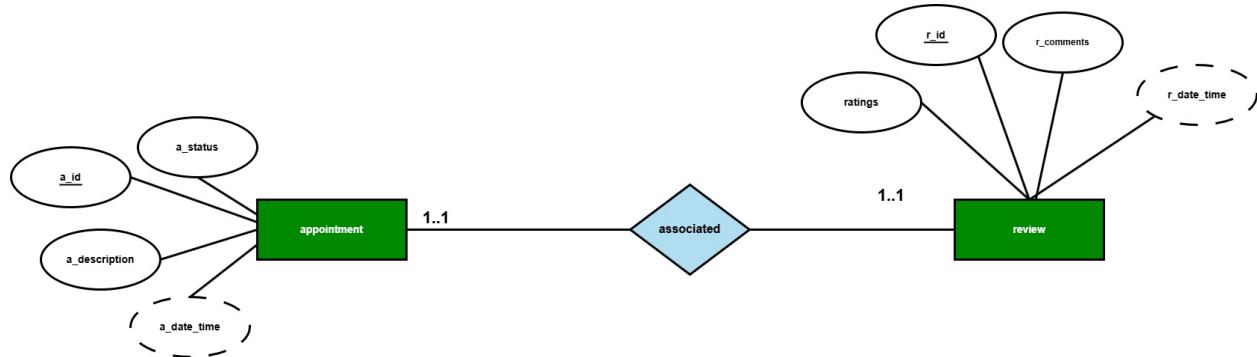
### 3NF

Name and Address are Transitive attribute

1. app\_id, app\_status, app\_description, app\_date\_time
2. name\_id, f\_name, l\_name
3. a\_id, city, street, house\_no
3. c\_id, email, contact\_no, dob

### Table creation

1. app\_id, app\_status, app\_description, app\_date\_time, **name\_id**, **a\_id**, **c\_id**
2. name\_id, f\_name, l\_name
3. a\_id, city, street, house\_no
4. c\_id, email, contact\_no, dob



**associated:**

### UNF

app\_id, app\_status, app\_description, app\_date\_time, r\_id, ratings, r\_comments, r\_date\_time

### 1NF

no multivalued attribute. (same as unf)

app\_id, app\_status, app\_description, app\_date\_time, r\_id, ratings, r\_comments, r\_date\_time

### 2NF

1. app\_id, app\_status, app\_description, app\_date\_time

2. r\_id, ratings, r\_comments, r\_date\_time

### 3NF

There is no transitive attribute. Relation is already in 3nf.

1. app\_id, app\_status, app\_description, app\_date\_time
2. r\_id, ratings, r\_comments, r\_date\_time

### Table creation

1. app\_id, app\_status, app\_description, app\_date\_time
2. r\_id, ratings, r\_comments, r\_date\_time, **app\_id**

## **Temporary Tables:**

1. app\_id, app\_status, app\_description, app\_date\_time
2. p\_id, p\_method, p\_status, p\_ammount, p\_date\_time, **app\_id**
3. r\_id, r\_comment, ratings, r\_date\_time, **name\_id**, **a\_id**, **c\_id**
4. c\_id, email, contact\_no, dob
5. name\_id, f\_name, l\_name
6. a\_id, city, street, house\_no
7. user\_name, email, password, dob, contact\_no, **name\_id**, **a\_id**, **c\_id**
8. name\_id, f\_name, l\_name
9. a\_id, city, street, house\_no
10. c\_id, email, contact\_no, dob
11. user\_name, password, **name\_id**, **c\_id**, **a\_id**
12. c\_id, email, contact\_no, dob
13. name\_id, f\_name, l\_name
14. a\_id, city, street, house\_no
15. case\_id, case\_status, case\_type, **name\_id**, **c\_id**, **a\_id**
16. c\_id, email, contact\_no, dob

17. name\_id, f\_name, l\_name
18. a\_id, city, street, house\_no
19. case\_id, case\_status, case\_type, **court\_id**, **loc\_id**
20. court\_id, court\_type, case\_id
21. loc\_id, district, division
22. p\_id, p\_method, p\_status, p\_amount, p\_date\_time, **c\_id**, **name\_id**, **a\_id**
23. c\_id, email, contact\_no, dob
24. name\_id, f\_name, l\_name
25. a\_id, city, street, house\_no
26. l\_id, email, bar\_council\_reg\_no, avg\_ratings, specialization, contact\_no, dob, **name\_id**, **a\_id**, **user\_name**
27. name\_id, f\_name, l\_name
28. a\_id, city, street, house\_no
29. user\_name, password, email, bar\_council\_reg\_no, specialization, contact\_no, dob
30. app\_id, app\_status, app\_description, app\_date\_time, **name\_id**, **a\_id**, **l\_id**
31. name\_id, f\_name, l\_name
32. a\_id, city, street, house\_no
33. l\_id, email, bar\_council\_reg\_no, avg\_ratings, specialization, contact\_no, dob
34. r\_id, ratings, r\_comments, r\_date\_time, **name\_id**, **a\_id**, **l\_id**
35. name\_id, f\_name, l\_name
36. a\_id, city, street, house\_no
37. l\_id, email, bar\_council\_reg\_no, avg\_ratings, specialization, contact\_no, dob
38. user\_name, password, **name\_id**, **a\_id**, **l\_id**
39. name\_id, f\_name, l\_name
40. a\_id, city, street, house\_no
41. l\_id, email, bar\_council\_reg\_no, avg\_ratings, specialization, contact\_no, dob
42. case\_id, case\_status, case\_type, **name\_id**, **a\_id**, **l\_id**
43. name\_id, f\_name, l\_name
44. a\_id, city, street, house\_no
45. l\_id, email, bar\_council\_reg\_no, avg\_ratings, specialization, contact\_no, dob
46. app\_id, app\_status, app\_description, app\_date\_time, **name\_id**, **a\_id**, **c\_id**

47. name\_id, f\_name, l\_name
48. a\_id, city, street, house\_no
49. c\_id, email, contact\_no, dob
50. app\_id, app\_status, app\_description, app\_date\_time
51. r\_id, ratings, r\_comments, r\_date\_time, **app\_id**

## Final Tables

1. user\_name, email, password, dob, contact\_no, **name\_id**, **a\_id**
2. name\_id, f\_name, l\_name
3. a\_id, city, street, house\_no
4. c\_id, **user\_name**
5. l\_id, bar\_council\_reg\_no, avg\_ratings, specialization, **user\_name**
6. court\_id, court\_type, loc\_id
7. loc\_id, district, division
8. case\_id, case\_status, case\_type, **c\_id**, **l\_id**, **court\_id**
9. app\_id, app\_status, app\_description, app\_date\_time, **c\_id**, **l\_id**
10. p\_id, p\_method, p\_status, p\_amount, p\_date\_time, **c\_id**, **app\_id**
11. r\_id, rating, r\_comments, r\_date\_time, **c\_id**, **l\_id**, **app\_id**

# Table Creation

Table 1: name

Sequence:

```
CREATE SEQUENCE name_seq START WITH 1001 INCREMENT BY 1 NOCACHE NOCYCLE;
```

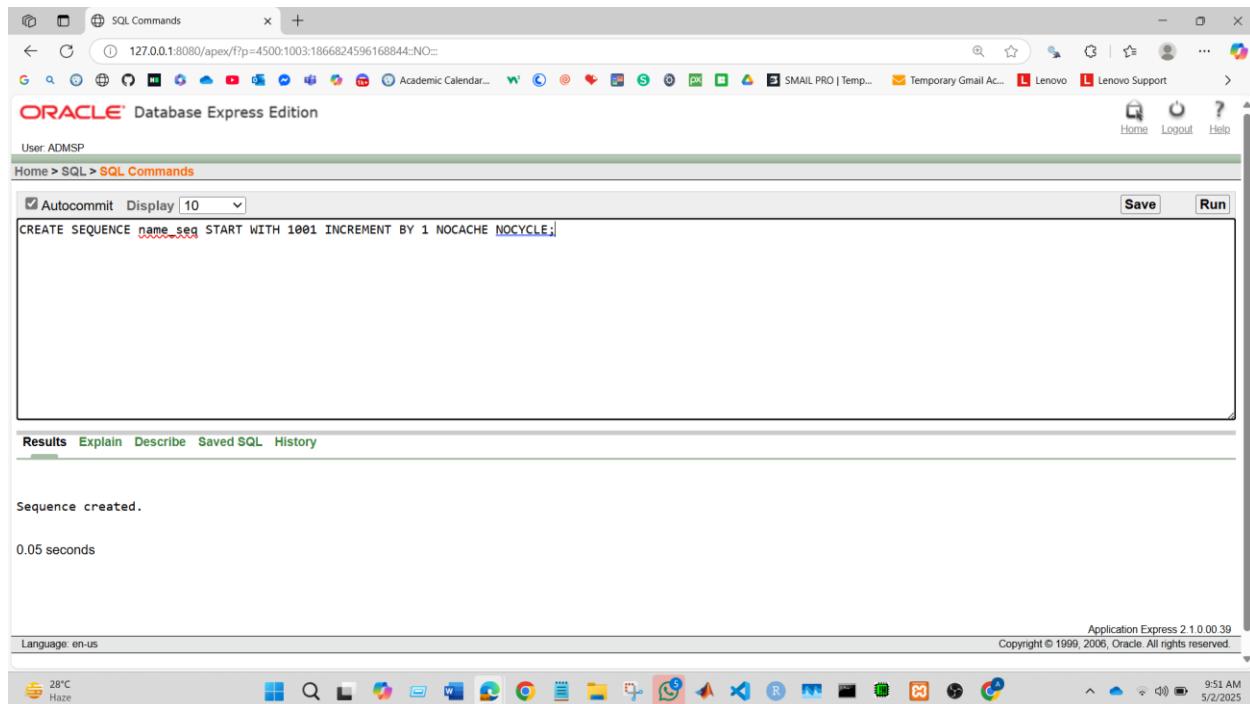


Table:

```
CREATE TABLE name (
    name_id NUMBER PRIMARY KEY,
    f_name VARCHAR2(50),
    l_name VARCHAR2(50)
);
```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. In the SQL editor, the following SQL code is entered:

```
CREATE TABLE name (
    name_id NUMBER PRIMARY KEY,
    f_name VARCHAR2(50),
    l_name VARCHAR2(50)
);
```

The 'Run' button is highlighted. Below the editor, the results show:

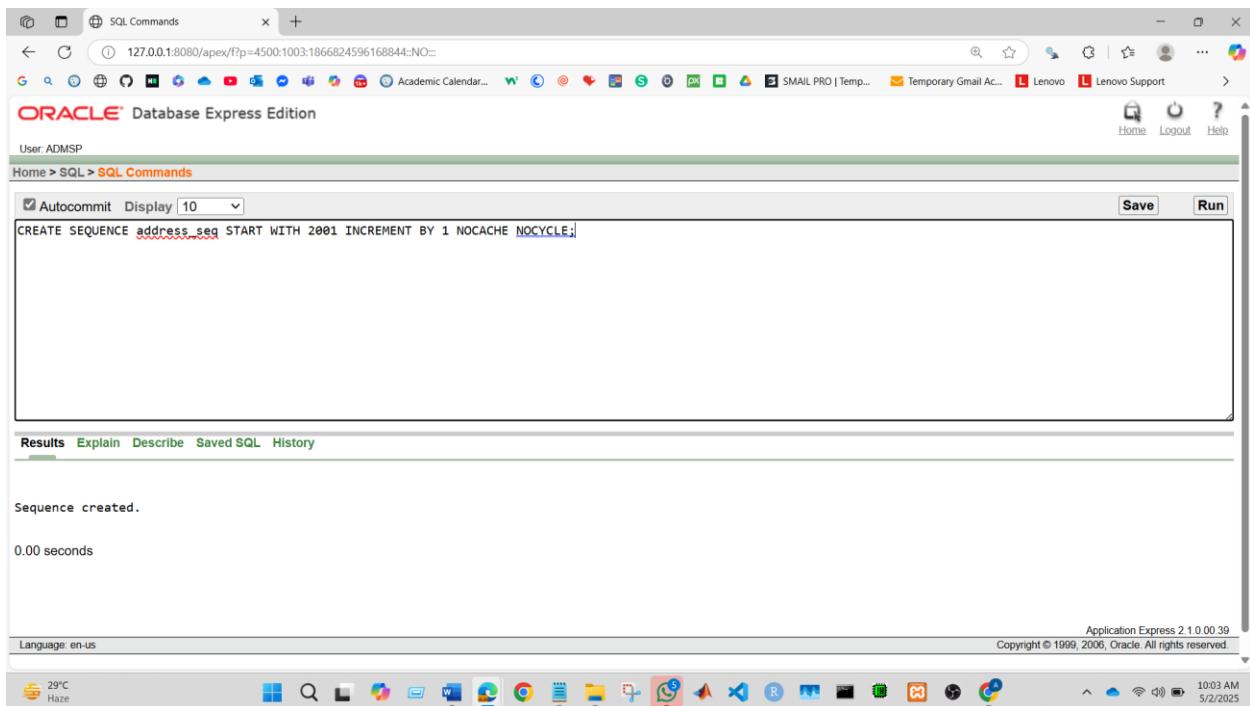
Table created.  
0.04 seconds

Language: en-us Application Express 2.1.0.00.39  
Copyright © 1999, 2006, Oracle. All rights reserved.

**Table 2: address**

**Sequence:**

```
CREATE SEQUENCE address_seq START WITH 2001 INCREMENT BY 1 NOCACHE NOCYCLE;
```



## Table:

```
CREATE TABLE address (
    a_id NUMBER PRIMARY KEY,
    city VARCHAR2(50),
    street VARCHAR2(100),
    house_no VARCHAR2(20)
);
```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL command entered is:

```
CREATE TABLE address (
    a_id NUMBER PRIMARY KEY,
    city VARCHAR2(50),
    street VARCHAR2(100),
    house_no VARCHAR2(20)
);
```

The results pane shows the message "Table created." and "0.00 seconds". The status bar at the bottom right indicates "Application Express 2.1.0.0.39 Copyright © 1999, 2006, Oracle. All rights reserved." and the date "5/2/2025".

**Table 3: user**

### Sequence:

```
CREATE SEQUENCE user_seq START WITH 1 INCREMENT BY 1 NOCACHE NOCYCLE;
```

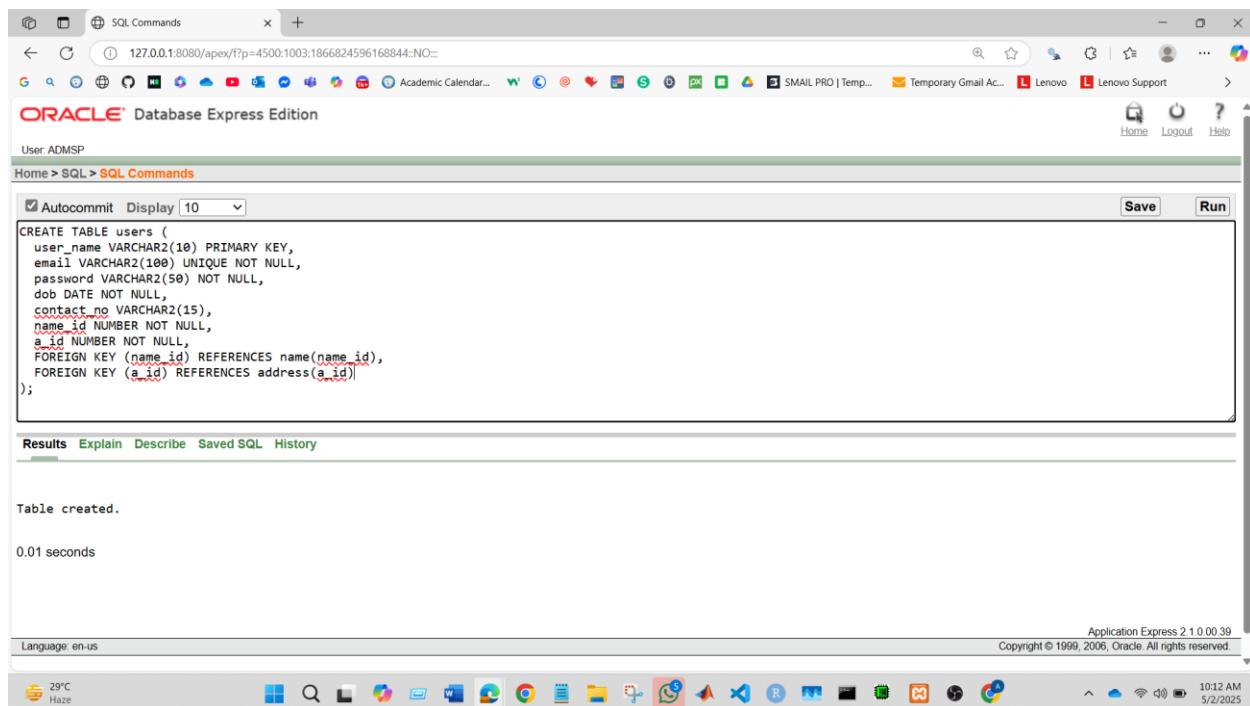
The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL command entered is:

```
CREATE SEQUENCE user_seq START WITH 1 INCREMENT BY 1 NOCACHE NOCYCLE;
```

The results pane shows the message "Sequence created." and "0.00 seconds". The status bar at the bottom right indicates "Application Express 2.1.0.0.39 Copyright © 1999, 2006, Oracle. All rights reserved." and the date "5/2/2025".

## Table:

```
CREATE TABLE users (
    user_name VARCHAR2(10) PRIMARY KEY,
    email VARCHAR2(100) UNIQUE NOT NULL,
    password VARCHAR2(50) NOT NULL,
    dob DATE NOT NULL,
    contact_no VARCHAR2(15),
    name_id NUMBER NOT NULL,
    a_id NUMBER NOT NULL,
    FOREIGN KEY (name_id) REFERENCES name(name_id),
    FOREIGN KEY (a_id) REFERENCES address(a_id)
);
```



The screenshot shows the Oracle Database Express Edition interface. In the SQL Commands window, the 'users' table is being created with the following DDL:

```
CREATE TABLE users (
    user_name VARCHAR2(10) PRIMARY KEY,
    email VARCHAR2(100) UNIQUE NOT NULL,
    password VARCHAR2(50) NOT NULL,
    dob DATE NOT NULL,
    contact_no VARCHAR2(15),
    name_id NUMBER NOT NULL,
    a_id NUMBER NOT NULL,
    FOREIGN KEY (name_id) REFERENCES name(name_id),
    FOREIGN KEY (a_id) REFERENCES address(a_id)
);
```

The table is successfully created, as indicated by the message "Table created." in the results pane. The system status bar at the bottom shows "Language en-us" and the date/time "5/2/2025 10:12 AM".

Table 4: client

## Sequence:

```
CREATE SEQUENCE client_seq START WITH 3001 INCREMENT BY 1 NOCACHE NOCYCLE;
```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The URL in the address bar is 127.0.0.1:8080/apex/f?p=4500:1003:1866824596168844:NO-. The page title is "SQL Commands". The SQL command entered is:

```
CREATE SEQUENCE client_seq START WITH 3001 INCREMENT BY 1 NOCACHE NOCYCLE;
```

The results section shows the message "Sequence created." and a timestamp of "0.00 seconds". The bottom status bar indicates the language is "en-us" and the application version is "Application Express 2.1.0.0.39".

### Table:

```
CREATE TABLE client (
    c_id NUMBER PRIMARY KEY,
    user_name VARCHAR2(20),
    FOREIGN KEY (user_name) REFERENCES users(user_name)
);
```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. In the SQL editor, the following SQL command is entered:

```
CREATE TABLE client (
    c_id NUMBER PRIMARY KEY,
    user_name VARCHAR2(20),
    FOREIGN KEY (user_name) REFERENCES users(user_name)
);
```

The results pane shows the message "Table created." and a execution time of "0.01 seconds". The status bar at the bottom right indicates "Application Express 2.1.0.0.39 Copyright © 1999, 2006, Oracle. All rights reserved." and the date "5/2/2025".

**Table 5: lawyer**

### Sequence:

```
CREATE SEQUENCE lawyer_seq START WITH 4001 INCREMENT BY 1 NOCACHE NOCYCLE;
```

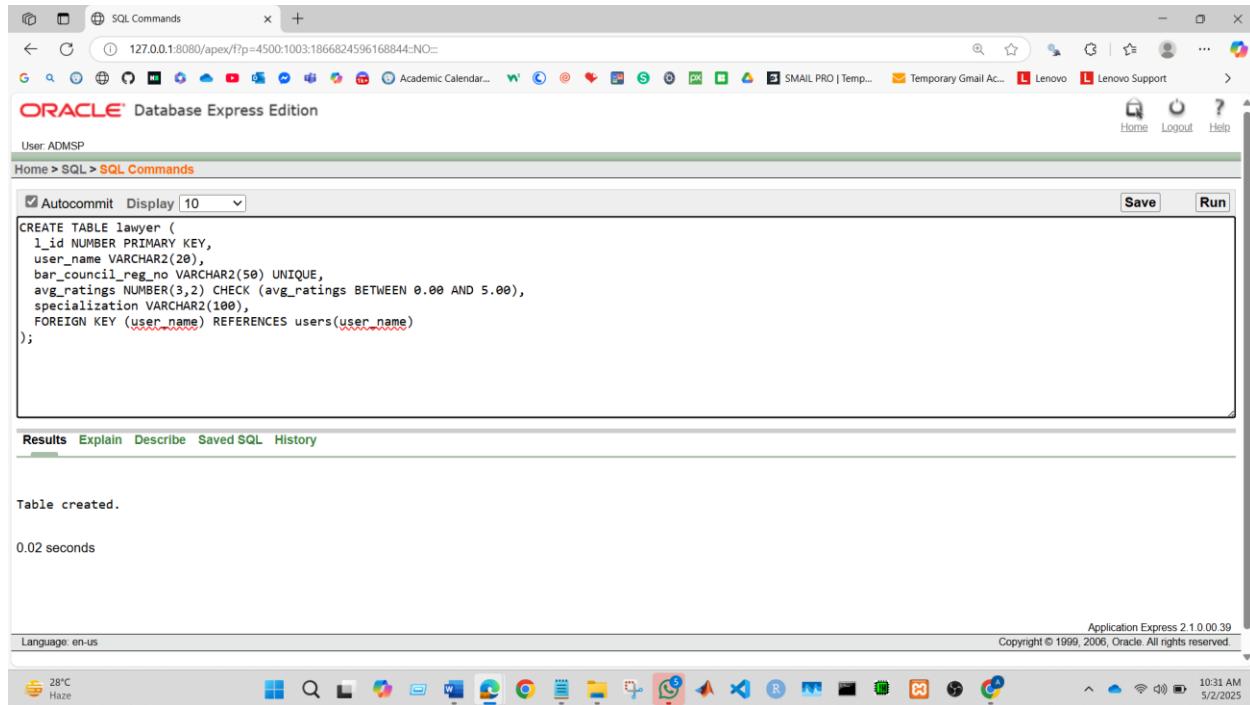
The screenshot shows the Oracle Database Express Edition SQL Commands interface. In the SQL editor, the following SQL command is entered:

```
CREATE SEQUENCE lawyer_seq START WITH 4001 INCREMENT BY 1 NOCACHE NOCYCLE;
```

The results pane shows the message "Sequence created." and a execution time of "0.00 seconds". The status bar at the bottom right indicates "Application Express 2.1.0.0.39 Copyright © 1999, 2006, Oracle. All rights reserved." and the date "5/2/2025".

## Table:

```
CREATE TABLE lawyer (
    l_id NUMBER PRIMARY KEY,
    user_name VARCHAR2(20),
    bar_council_reg_no VARCHAR2(50) UNIQUE,
    avg_ratings NUMBER(3,2) CHECK (avg_ratings BETWEEN 0.00 AND 5.00),
    specialization VARCHAR2(100),
    FOREIGN KEY (user_name) REFERENCES users(user_name)
);
```



The screenshot shows the Oracle Database Express Edition interface. In the top navigation bar, it says "User: ADMSP". Below that, the path "Home > SQL > SQL Commands" is visible. The main area contains the SQL code for creating the "lawyer" table. At the bottom of the code editor, there are "Save" and "Run" buttons. The results section below the code shows the message "Table created." and "0.02 seconds". The status bar at the bottom right indicates "Application Express 2.1.0.0.39" and "Copyright © 1999, 2006, Oracle. All rights reserved.". The system tray at the bottom shows various icons and the date/time "5/2/2025 10:31 AM".

```
CREATE TABLE lawyer (
    l_id NUMBER PRIMARY KEY,
    user_name VARCHAR2(20),
    bar_council_reg_no VARCHAR2(50) UNIQUE,
    avg_ratings NUMBER(3,2) CHECK (avg_ratings BETWEEN 0.00 AND 5.00),
    specialization VARCHAR2(100),
    FOREIGN KEY (user_name) REFERENCES users(user_name)
);
```

Table 6: location

## Sequence:

```
CREATE SEQUENCE location_seq START WITH 6001 INCREMENT BY 1 NOCACHE NOCYCLE;
```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. In the SQL editor, the following SQL command is entered:

```
CREATE SEQUENCE location_seq START WITH 6001 INCREMENT BY 1 NOCACHE NOCYCLE;
```

The results pane shows the output:

```
Sequence created.
```

Below the results, the status bar indicates:

```
Language: en-us Application Express 2.1.0.00.39  
Copyright © 1999, 2006, Oracle. All rights reserved.
```

### Table:

```
CREATE TABLE location (
    loc_id NUMBER PRIMARY KEY,
    district VARCHAR2(50),
    division VARCHAR2(50)
);
```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. In the SQL editor, the following SQL command is entered:

```
CREATE TABLE location (
    loc_id NUMBER PRIMARY KEY,
    district VARCHAR2(50),
    division VARCHAR2(50)
);
```

The results pane shows the message "Table created." and a execution time of "0.01 seconds". The status bar at the bottom right indicates "Application Express 2.1.0.0.39" and "Copyright © 1999, 2006, Oracle. All rights reserved.".

**Table 7: court**

### Sequence:

```
CREATE SEQUENCE court_seq START WITH 5001 INCREMENT BY 1 NOCACHE NOCYCLE;
```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. In the SQL editor, the following SQL command is entered:

```
CREATE SEQUENCE court_seq START WITH 5001 INCREMENT BY 1 NOCACHE NOCYCLE;
```

The results pane shows the message "Sequence created." and a execution time of "0.00 seconds". The status bar at the bottom right indicates "Application Express 2.1.0.0.39" and "Copyright © 1999, 2006, Oracle. All rights reserved.".

## Table:

```
CREATE TABLE court (
    court_id NUMBER PRIMARY KEY,
    court_type VARCHAR2(50),
    loc_id NUMBER,
    FOREIGN KEY (loc_id) REFERENCES location(loc_id)
);
```

The screenshot shows a web-based Oracle Database Express Edition interface. The URL in the address bar is 127.0.0.1:8080/apex/f?p=4500:1003:1866824596168844:NO-. The page title is "ORACLE Database Express Edition". The user is logged in as ADMSP. The main content area is titled "SQL Commands" and contains the SQL code for creating the "court" table. The code is as follows:

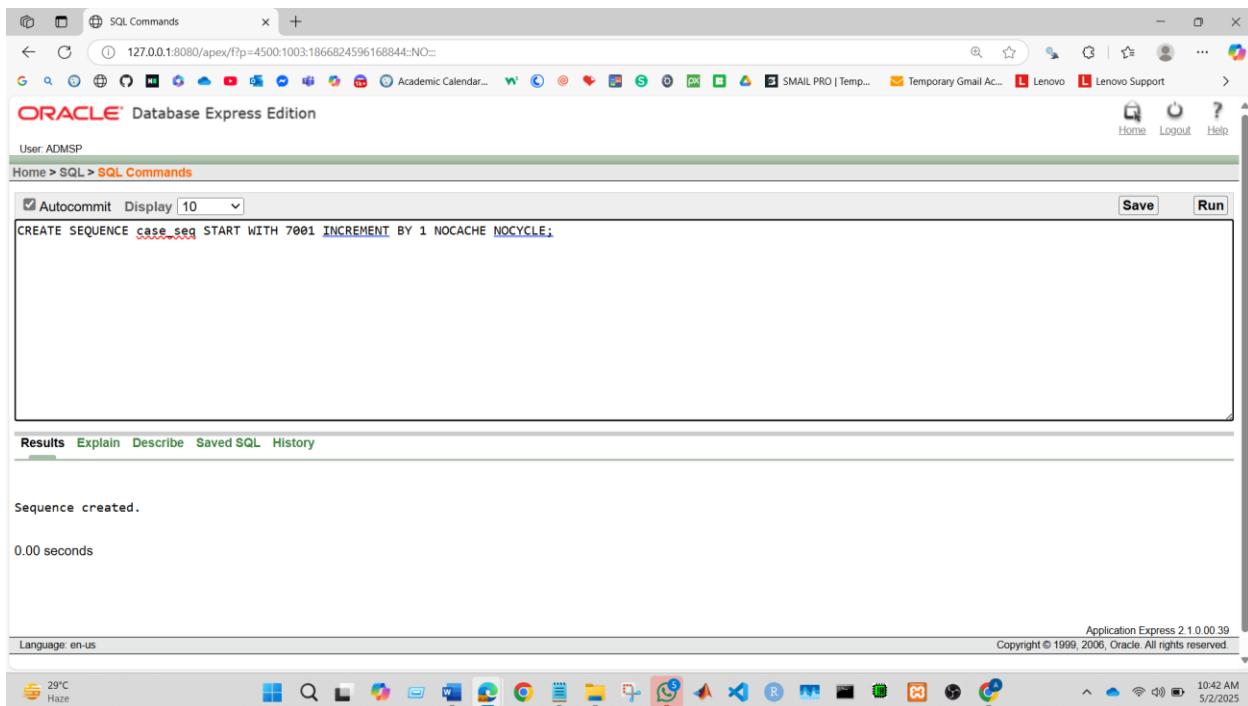
```
CREATE TABLE court (
    court_id NUMBER PRIMARY KEY,
    court_type VARCHAR2(50),
    loc_id NUMBER,
    FOREIGN KEY (loc_id) REFERENCES location(loc_id)
);
```

Below the code, there are "Save" and "Run" buttons. The results section shows the message "Table created." and "0.00 seconds". At the bottom, it indicates the language is en-us and the application version is Application Express 2.1.0.00.39. Copyright information from 1999-2006 Oracle is also present.

Table 8: case

## Sequence:

```
CREATE SEQUENCE case_seq START WITH 7001 INCREMENT BY 1 NOCACHE NOCYCLE;
```



## Table:

```
CREATE TABLE case (
    case_id NUMBER PRIMARY KEY,
    case_status VARCHAR2(50),
    case_type VARCHAR2(50),
    c_id NUMBER,
    l_id NUMBER,
    court_id NUMBER,
    FOREIGN KEY (c_id) REFERENCES client(c_id),
    FOREIGN KEY (l_id) REFERENCES lawyer(l_id),
    FOREIGN KEY (court_id) REFERENCES court(court_id)
);
```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. In the SQL editor, the following SQL code is entered:

```
CREATE TABLE case (
    case_id NUMBER PRIMARY KEY,
    case_status VARCHAR2(50),
    case_type VARCHAR2(50),
    c_id NUMBER,
    l_id NUMBER,
    court_id NUMBER,
    FOREIGN KEY (c_id) REFERENCES client(c_id),
    FOREIGN KEY (l_id) REFERENCES lawyer(l_id),
    FOREIGN KEY (court_id) REFERENCES court(court_id)
);
```

The results pane shows the message "Table created." and a duration of "0.02 seconds". The status bar at the bottom right indicates "Application Express 2.1.0.0.39 Copyright © 1999, 2006, Oracle. All rights reserved." and the date "5/2/2025".

**Table 9: appointment**

**Sequence:**

```
CREATE SEQUENCE app_seq START WITH 8001 INCREMENT BY 1 NOCACHE NOCYCLE;
```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. In the SQL editor, the following SQL code is entered:

```
CREATE SEQUENCE app_seq START WITH 8001 INCREMENT BY 1 NOCACHE NOCYCLE;
```

The results pane shows the message "Sequence created." and a duration of "0.00 seconds". The status bar at the bottom right indicates "Application Express 2.1.0.0.39 Copyright © 1999, 2006, Oracle. All rights reserved." and the date "5/2/2025".

**Table:**

```
CREATE TABLE appointment (
    app_id NUMBER PRIMARY KEY,
    app_status VARCHAR2(50),
    app_description VARCHAR2(255),
    app_date_time TIMESTAMP,
    c_id NUMBER,
    l_id NUMBER,
    FOREIGN KEY (c_id) REFERENCES client(c_id),
    FOREIGN KEY (l_id) REFERENCES lawyer(l_id)
);
```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL command for creating the 'appointment' table is entered in the main text area. The table has columns for app\_id (primary key), app\_status, app\_description, app\_date\_time, c\_id, and l\_id. It includes foreign key constraints linking c\_id to the client table and l\_id to the lawyer table. The interface includes tabs for Results, Explain, Describe, Saved SQL, and History. Below the interface, the Windows taskbar is visible with various application icons.

```
CREATE TABLE appointment (
    app_id NUMBER PRIMARY KEY,
    app_status VARCHAR2(50),
    app_description VARCHAR2(255),
    app_date_time TIMESTAMP,
    c_id NUMBER,
    l_id NUMBER,
    FOREIGN KEY (c_id) REFERENCES client(c_id),
    FOREIGN KEY (l_id) REFERENCES lawyer(l_id)
);
```

**Table 10: payment**

**Sequence:**

```
CREATE SEQUENCE payment_seq START WITH 9001 INCREMENT BY 1 NOCACHE NOCYCLE;
```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. In the top navigation bar, it says "User: ADMSP". Below that, the path "Home > SQL > SQL Commands" is visible. The main area contains the following SQL command:

```
CREATE SEQUENCE payment_seq START WITH 9001 INCREMENT BY 1 NOCACHE NOCYCLE;
```

Below the command, the output shows:

Sequence created.  
0.00 seconds

At the bottom right, there is copyright information: "Application Express 2.1.0.00.39 Copyright © 1999, 2006, Oracle. All rights reserved." The system tray at the bottom of the screen shows the date and time as "5/2/2025 10:53 AM".

### Table:

```
CREATE TABLE payment (
    p_id NUMBER PRIMARY KEY,
    p_method VARCHAR2(50) NOT NULL,
    p_status VARCHAR2(50) NOT NULL,
    p_amount NUMBER(10,2) NOT NULL,
    p_date_time TIMESTAMP,
    c_id NUMBER,
    app_id NUMBER,
    FOREIGN KEY (c_id) REFERENCES client(c_id),
    FOREIGN KEY (app_id) REFERENCES appointment(app_id)
);
```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The user is creating a table named 'payment'. The table has columns: p\_id (NUMBER PRIMARY KEY), p\_method (VARCHAR2(50) NOT NULL), p\_status (VARCHAR2(50) NOT NULL), p\_amount (NUMBER(10,2) NOT NULL), p\_date\_time (TIMESTAMP), c\_id (NUMBER), app\_id (NUMBER), FOREIGN KEY (c\_id) REFERENCES client(c\_id), and FOREIGN KEY (app\_id) REFERENCES appointment(app\_id). The command is run successfully, and the table is created.

```

CREATE TABLE payment (
    p_id NUMBER PRIMARY KEY,
    p_method VARCHAR2(50) NOT NULL,
    p_status VARCHAR2(50) NOT NULL,
    p_amount NUMBER(10,2) NOT NULL,
    p_date_time TIMESTAMP,
    c_id NUMBER,
    app_id NUMBER,
    FOREIGN KEY (c_id) REFERENCES client(c_id),
    FOREIGN KEY (app_id) REFERENCES appointment(app_id)
);

```

Table created.  
0.01 seconds

Language: en-us Application Express 2.1.0.0.39  
Copyright © 1999, 2006, Oracle. All rights reserved.

**Table 11: rating**

### Sequence:

```
CREATE SEQUENCE rating_seq START WITH 10001 INCREMENT BY 1 NOCACHE NOCYCLE;
```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The user is creating a sequence named 'rating\_seq' with a start value of 10001, increment by 1, and no cache or cycle. The sequence is created successfully.

```

CREATE SEQUENCE rating_seq START WITH 10001 INCREMENT BY 1 NOCACHE NOCYCLE;

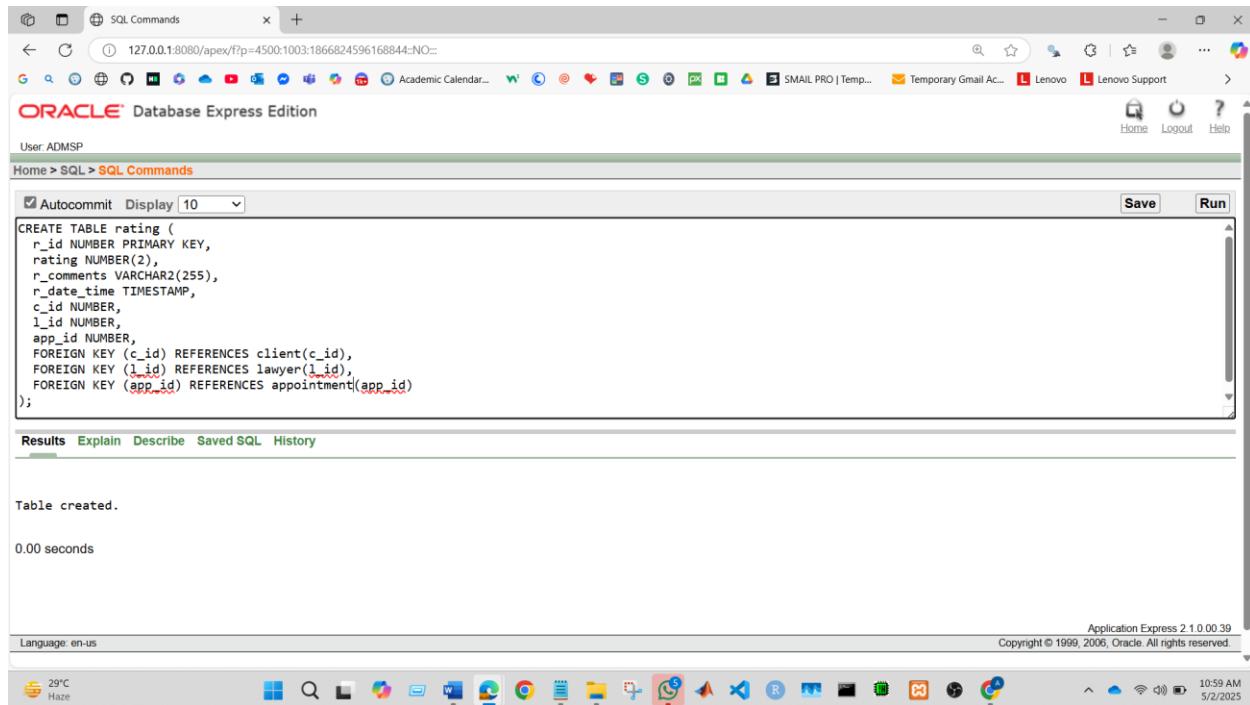
```

Sequence created.  
0.00 seconds

Language: en-us Application Express 2.1.0.0.39  
Copyright © 1999, 2006, Oracle. All rights reserved.

## Table:

```
CREATE TABLE rating (
    r_id NUMBER PRIMARY KEY,
    rating NUMBER(2),
    r_comments VARCHAR2(255),
    r_date_time TIMESTAMP,
    c_id NUMBER,
    l_id NUMBER,
    app_id NUMBER,
    FOREIGN KEY (c_id) REFERENCES client(c_id),
    FOREIGN KEY (l_id) REFERENCES lawyer(l_id),
    FOREIGN KEY (app_id) REFERENCES appointment(app_id)
);
```

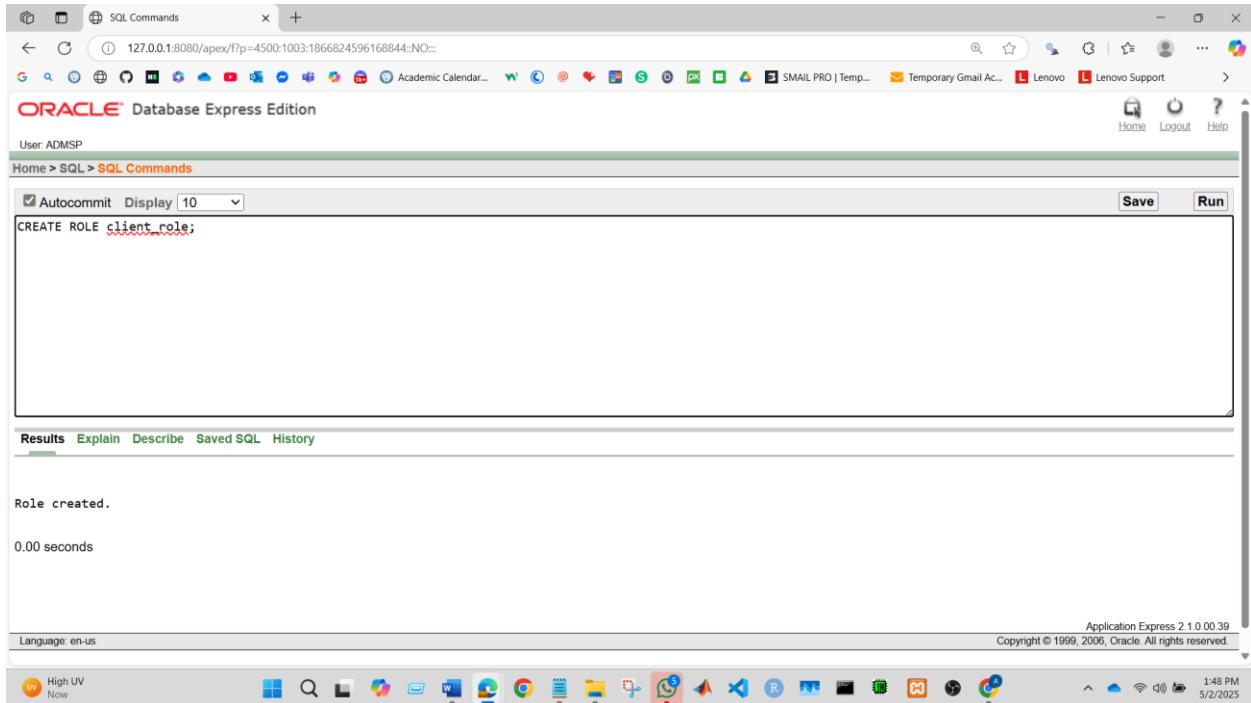


The screenshot shows a browser window with the URL `127.0.0.1:8080/apex/f?p=4500:1003:1866824596168844::NO::`. The page title is "SQL Commands". Below the address bar, the Oracle Database Express Edition logo is visible. The main content area contains the SQL code for creating the "rating" table, which includes columns for r\_id (primary key), rating (number with 2 decimal places), r\_comments (variable character string of length 255), r\_date\_time (timestamp), c\_id (number), l\_id (number), and app\_id (number). It also includes three foreign key constraints referencing the "client", "lawyer", and "appointment" tables respectively. At the bottom of the SQL editor, there are "Save" and "Run" buttons. The results section below the editor displays the message "Table created." and "0.00 seconds". The status bar at the bottom right indicates "Application Express 2.1.0.0.39 Copyright © 1999, 2006, Oracle. All rights reserved." and shows the system date and time as "5/2/2025 10:59 AM".

## Create Roles

## Client Role:

```
CREATE ROLE client_role;
```



## Lawyer Role:

```
CREATE ROLE lawyer_role;
```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The URL is 127.0.0.1:8080/apex/F?p=4500:1003:18666824596168844:NO-. The user is ADMSP. The SQL command entered is:

```
CREATE ROLE lawyer_role;
```

The results show:

```
Role created.
```

0.00 seconds

At the bottom, the system status bar indicates: Application Express 2.1.0.00.39, Copyright © 1999-2006 Oracle. All rights reserved, 1:49 PM, 5/2/2025.

## Grant Privileges to Roles:

### Client Privileges:

```
GRANT INSERT, SELECT, UPDATE ON client TO client_role;

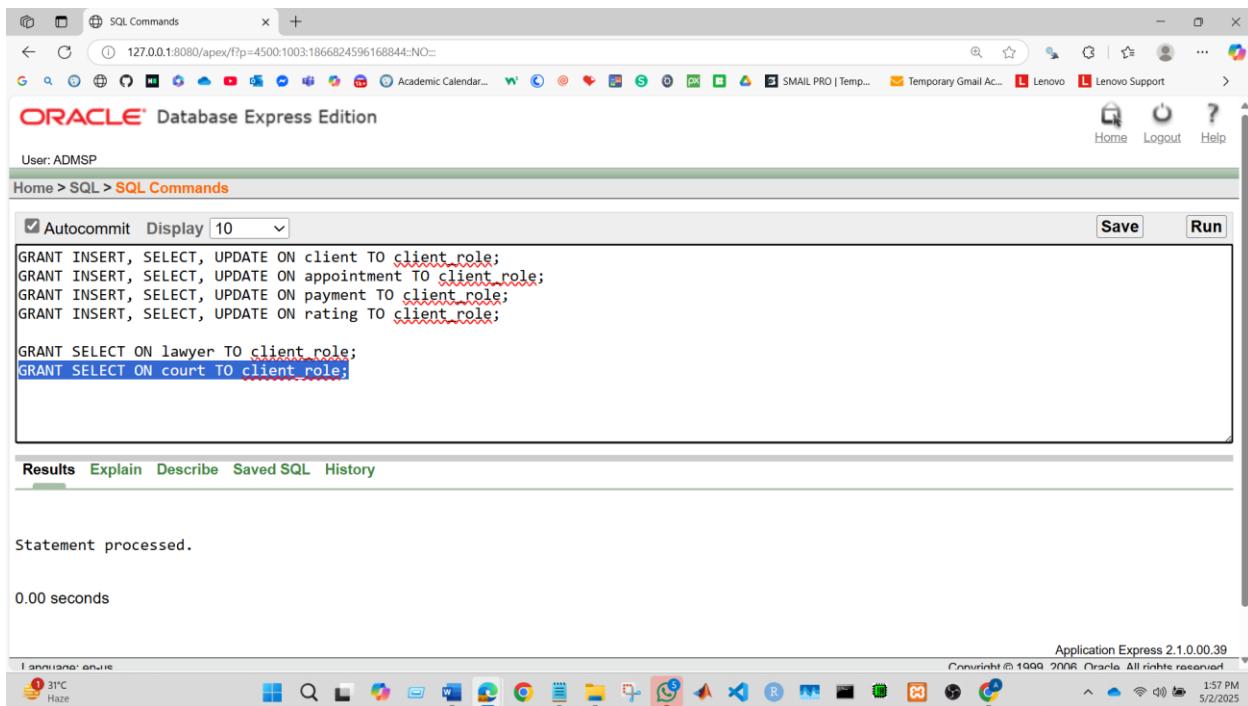
GRANT INSERT, SELECT, UPDATE ON appointment TO client_role;

GRANT INSERT, SELECT, UPDATE ON payment TO client_role;

GRANT INSERT, SELECT, UPDATE ON rating TO client_role;

GRANT SELECT ON lawyer TO client_role;

GRANT SELECT ON court TO client_role;
```



The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL editor contains the following GRANT statements:

```
GRANT INSERT, SELECT, UPDATE ON client TO client_role;
GRANT INSERT, SELECT, UPDATE ON appointment TO client_role;
GRANT INSERT, SELECT, UPDATE ON payment TO client_role;
GRANT INSERT, SELECT, UPDATE ON rating TO client_role;

GRANT SELECT ON lawyer TO client_role;
GRANT SELECT ON court TO client_role;
```

The results pane shows "Statement processed." and "0.00 seconds". The system status bar at the bottom indicates "Application Express 2.1.0.00.39", "Copyright © 1999-2006 Oracle. All rights reserved.", "31°C Haze", and the date "5/2/2025".

### Lawyer Privileges:

```
GRANT INSERT, SELECT, UPDATE ON lawyer TO lawyer_role;

GRANT INSERT, SELECT, UPDATE ON appointment TO lawyer_role;

GRANT SELECT, UPDATE ON case TO lawyer_role;

GRANT SELECT ON client TO lawyer_role;

GRANT SELECT ON court TO lawyer_role;

GRANT SELECT ON location TO lawyer_role;
```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The user is logged in as ADMSP. The SQL editor contains the following GRANT statements:

```
GRANT INSERT, SELECT, UPDATE ON lawyer TO lawyer_role;
GRANT INSERT, SELECT, UPDATE ON appointment TO lawyer_role;

GRANT SELECT, UPDATE ON case TO lawyer_role;

GRANT SELECT ON client TO lawyer_role;
GRANT SELECT ON court TO lawyer_role;
GRANT SELECT ON location TO lawyer_role;
```

The results tab shows the message "Statement processed." and a timestamp of "0.00 seconds". The system tray at the bottom indicates a temperature of 31°C and haze, and shows the application is running on Application Express 2.1.0.00.39.

# Data Insertion

Table 1: name

```
INSERT INTO name VALUES (name_seq.NEXTVAL, 'Abdul', 'Rahman');

INSERT INTO name VALUES (name_seq.NEXTVAL, 'Fatema', 'Begum');

INSERT INTO name VALUES (name_seq.NEXTVAL, 'Mohammad', 'Hossain');

INSERT INTO name VALUES (name_seq.NEXTVAL, 'Ayesha', 'Siddiqua');

INSERT INTO name VALUES (name_seq.NEXTVAL, 'Iqbal', 'Khan');

INSERT INTO name VALUES (name_seq.NEXTVAL, 'Nadia', 'Hasan');

INSERT INTO name VALUES (name_seq.NEXTVAL, 'Kamal', 'Uddin');

INSERT INTO name VALUES (name_seq.NEXTVAL, 'Sara', 'Rahim');

INSERT INTO name VALUES (name_seq.NEXTVAL, 'Zahid', 'Mahmud');

INSERT INTO name VALUES (name_seq.NEXTVAL, 'Lubna', 'Khatun');
```

```
SELECT * FROM name
```

The screenshot shows the Oracle Database Express Edition interface. In the SQL Commands section, several INSERT statements are listed, followed by a SELECT \* FROM name statement. In the Results section, a table is displayed with columns NAME\_ID, F\_NAME, and L\_NAME, containing 10 rows of data. The bottom status bar shows system information like temperature (27°C), battery level (Haze), and system time (10:35 PM 5/2/2025).

NAME_ID	F_NAME	L_NAME
1001	Abdul	Rahman
1002	Fatema	Begum
1003	Mohammad	Hossain
1004	Ayesha	Siddiqua
1005	Iqbal	Khan
1006	Nadia	Hasan
1007	Kamal	Uddin
1008	Sara	Rahim
1009	Zahid	Mahmud
1010	Lubna	Khatun

**Table 2: address**

```
INSERT INTO address VALUES (address_seq.NEXTVAL, 'Dhaka', 'Mirpur Road',  
'123/A');  
  
INSERT INTO address VALUES(address_seq.NEXTVAL, 'Chittagong', 'Agrabad C/A',  
'45-B');  
  
INSERT INTO address VALUES(address_seq.NEXTVAL, 'Sylhet', 'Zindabazar', 'Road  
7');  
  
INSERT INTO address VALUES(address_seq.NEXTVAL, 'Khulna', 'Khan Jahan Ali  
Road', '78/C');  
  
INSERT INTO address VALUES (address_seq.NEXTVAL, 'Rajshahi', 'Shaheb Bazar', '9-  
D');  
  
INSERT INTO address VALUES (address_seq.NEXTVAL, 'Barisal', 'Band Road', '12-  
F');  
  
INSERT INTO address VALUES (address_seq.NEXTVAL, 'Comilla', 'Kandirpar',  
'34/G');  
  
INSERT INTO address VALUES (address_seq.NEXTVAL, 'Rangpur', 'Central Road',  
'56-A');  
  
INSERT INTO address VALUES (address_seq.NEXTVAL, 'Mymensingh', 'Town Hall  
Road', '88-B');  
  
INSERT INTO address VALUES (address_seq.NEXTVAL, 'Jessore', 'Rail Station  
Road', '21-E');  
  
SELECT * FROM address
```

```

INSERT INTO address VALUES (address_seq.NEXTVAL, 'Dhaka', 'Mirpur Road', '123/A');
INSERT INTO address VALUES(address_seq.NEXTVAL, 'Chittagong', 'Agrabad C/A', '45-B');
INSERT INTO address VALUES(address_seq.NEXTVAL, 'Sylhet', 'Zindabazar', '23');
INSERT INTO address VALUES(address_seq.NEXTVAL, 'Khulna', 'Khan Jahan Ali Road', '78/C');
INSERT INTO address VALUES(address_seq.NEXTVAL, 'Rajshahi', 'Shaheb Bazar', '9-D');
INSERT INTO address VALUES(address_seq.NEXTVAL, 'Barisal', 'Band Road', '12-F');
INSERT INTO address VALUES (address_seq.NEXTVAL, 'Comilla', 'Kandirpar', '34/G');
INSERT INTO address VALUES (address_seq.NEXTVAL, 'Rangpur', 'Central Road', '56-A');
INSERT INTO address VALUES (address_seq.NEXTVAL, 'Mymensingh', 'Town Hall Road', '88-B');
INSERT INTO address VALUES (address_seq.NEXTVAL, 'Jessore', 'Rail Station Road', '21-E');

SELECT * FROM address;

```

A_ID	CITY	STREET	HOUSE_NO
2001	Dhaka	Mirpur Road	123/A
2002	Chittagong	Agrabad C/A	45-B
2003	Sylhet	Zindabazar	Road 7
2004	Khulna	Khan Jahan Ali Road	78/C
2005	Rajshahi	Shaheb Bazar	9-D
2006	Barisal	Band Road	12-F
2007	Comilla	Kandirpar	34/G
2008	Mymensingh	Town Hall Road	88-B
2009	Rangpur	Central Road	56-A
2010	Jessore	Rail Station Road	21-E

10 rows returned in 0.00 seconds [CSV Export](#)

Language: en-us Application Express 2.1.0.00.39 Copyright © 1999, 2006, Oracle. All rights reserved. 11:12 PM 5/2/2025

**Table 3: user**

```

INSERT INTO users VALUES ('user' || LPAD(user_seq.NEXTVAL, 3, '0'),
'abdul.rahman@bdmail.com', 'pass123', TO_DATE('1985-03-15','YYYY-MM-DD'),
'+8801712345678', 1001, 2001);

INSERT INTO users VALUES ('user' || LPAD(user_seq.NEXTVAL, 3, '0'),
'fatema.begum@example.com', 'bdbpass456', TO_DATE('1992-07-22','YYYY-MM-DD'),
'01987654321', 1002, 2002);

INSERT INTO users VALUES ('user' || LPAD(user_seq.NEXTVAL, 3, '0'),
'mhossain@gmail.com', 'mh2023', TO_DATE('1978-11-05','YYYY-MM-DD'),
'+8801911122334', 1003, 2003);

INSERT INTO users VALUES ('user' || LPAD(user_seq.NEXTVAL, 3, '0'),
'ayesha.sid@yahoo.com', 'asid789', TO_DATE('1995-05-30','YYYY-MM-DD'),
'01876543210', 1004, 2004);

INSERT INTO users VALUES ('user' || LPAD(user_seq.NEXTVAL, 3, '0'),
'iqbal.khan@hotmail.com', 'ikhan456', TO_DATE('1980-12-12','YYYY-MM-DD'),
'+8801555555555', 1005, 2005);

```

```

INSERT INTO users VALUES ('user' || LPAD(user_seq.NEXTVAL, 3, '0'), 'nadia.hasan@example.com', 'nadia123', TO_DATE('1990-02-10','YYYY-MM-DD'), '+8801700000001', 1001, 2001);

INSERT INTO users VALUES ('user' || LPAD(user_seq.NEXTVAL, 3, '0'), 'kamal.uddin@example.com', 'kamal456', TO_DATE('1982-06-25','YYYY-MM-DD'), '+8801700000002', 1002, 2002);

INSERT INTO users VALUES ('user' || LPAD(user_seq.NEXTVAL, 3, '0'), 'sara.rahim@example.com', 'sara789', TO_DATE('1995-11-18','YYYY-MM-DD'), '+8801700000003', 1003, 2003);

INSERT INTO users VALUES ('user' || LPAD(user_seq.NEXTVAL, 3, '0'), 'zahid.mahmud@example.com', 'zahid321', TO_DATE('1987-08-09','YYYY-MM-DD'), '+8801700000004', 1004, 2004);

INSERT INTO users VALUES ('user' || LPAD(user_seq.NEXTVAL, 3, '0'), 'lubna.khatun@example.com', 'lubna654', TO_DATE('1993-01-30','YYYY-MM-DD'), '+8801700000005', 1005, 2005);

```

**SELECT \* FROM users**

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The query `SELECT * FROM users` has been run, and the results are displayed in a table.

USER_NAME	EMAIL	PASSWORD	DOB	CONTACT_NO	NAME_ID	A_ID
user001	abdul.rahman@bmail.com	pass123	15-MAR-85	+8801712345678	1001	2001
user002	fatema.begum@example.com	bdpass456	22-JUL-92	01987654321	1002	2002
user003	mhossain@gmail.com	mn2023	05-NOV-78	+8801911122334	1003	2003
user004	ayesha.sid@yahoo.com	asid789	30-MAY-95	01876543210	1004	2004
user005	iqbal.khan@hotmail.com	ikhan456	12-DEC-80	+8801555555555	1005	2005
user012	sara.rahim@example.com	sara789	18-NOV-95	+8801700000003	1006	2006
user010	nadia.hasan@example.com	nadia123	10-FEB-90	+8801700000001	1006	2006
user011	kamal.uddin@example.com	kamal456	25-JUN-82	+8801700000002	1007	2007
user013	zahid.mahmud@example.com	zahid321	09-AUG-87	+8801700000004	1009	2009
user014	lubna.khatun@example.com	lubna654	30-JAN-93	+8801700000005	1010	2010

10 rows returned in 0.00 seconds [CSV Export](#)

Language: en-us Copyright © 1999, 2006, Oracle. All rights reserved. Application Express 2.1.0.0.39 11:18 PM 5/2/2025

**Table 4 client:**

```

INSERT INTO client VALUES(client_seq.NEXTVAL, 'user001');

INSERT INTO client VALUES(client_seq.NEXTVAL, 'user002');

INSERT INTO client VALUES(client_seq.NEXTVAL, 'user003');

INSERT INTO client VALUES(client_seq.NEXTVAL, 'user004');

INSERT INTO client VALUES(client_seq.NEXTVAL, 'user005');

```

```
SELECT * FROM client
```

The screenshot shows the Oracle Database Express Edition interface. In the SQL Commands window, several INSERT statements are executed, followed by a SELECT statement:

```

INSERT INTO client VALUES(client_seq.NEXTVAL, 'user001');
INSERT INTO client VALUES(client_seq.NEXTVAL, 'user002');
INSERT INTO client VALUES(client_seq.NEXTVAL, 'user003');
INSERT INTO client VALUES(client_seq.NEXTVAL, 'user004');
INSERT INTO client VALUES(client_seq.NEXTVAL, 'user005');

SELECT * FROM client;

```

Below the SQL window, the Results tab displays the data from the SELECT query:

C_ID	USER_NAME
3001	user001
3002	user002
3003	user003
3004	user004
3005	user005

5 rows returned in 0.00 seconds

**Table 5: lawyer**

```

INSERT INTO lawyer VALUES (lawyer_seq.NEXTVAL, 'user010', 'BCR-DHA-12345',
4.75, 'Corporate Law');

INSERT INTO lawyer VALUES (lawyer_seq.NEXTVAL, 'user011', 'BCR-CHT-67890',
4.20, 'Criminal Law');

INSERT INTO lawyer VALUES (lawyer_seq.NEXTVAL, 'user012', 'BCR-SYL-54321',
4.90, 'Family Law');

INSERT INTO lawyer VALUES (lawyer_seq.NEXTVAL, 'user013', 'BCR-KHU-09876',
4.50, 'Civil Law');

```

```
INSERT INTO lawyer VALUES (lawyer_seq.NEXTVAL, 'user014', 'BCR-RAJ-13579',
4.30, 'Property Law');
```

```
SELECT * FROM lawyer
```

The screenshot shows the Oracle Database Express Edition interface. In the SQL Commands tab, a query is being run:

```
SELECT * FROM lawyer;
```

The results pane displays the following data:

L_ID	USER_NAME	BAR_COUNCIL_REG_NO	AVG_RATINGS	SPECIALIZATION
4001	user010	BCR-DHA-12345	4.75	Corporate Law
4002	user011	BCR-CHT-67890	4.20	Criminal Law
4003	user012	BCR-SYL-54321	4.90	Family Law
4004	user013	BCR-KHU-09876	4.50	Civil Law
4005	user014	BCR-RAJ-13579	4.30	Property Law

5 rows returned in 0.00 seconds [CSV Export](#)

Language: en-us Application Express 2.1.0.00.39 Copyright © 1999, 2006, Oracle. All rights reserved.

**Table 6: location**

```
INSERT INTO location VALUES (location_seq.NEXTVAL, 'Dhaka', 'Dhaka');

INSERT INTO location VALUES (location_seq.NEXTVAL, 'Chittagong', 'Chittagong');

INSERT INTO location VALUES (location_seq.NEXTVAL, 'Sylhet', 'Sylhet');

INSERT INTO location VALUES (location_seq.NEXTVAL, 'Khulna', 'Khulna');

INSERT INTO location VALUES (location_seq.NEXTVAL, 'Rajshahi', 'Rajshahi');
```

```
SELECT * FROM location
```

The screenshot shows the Oracle Application Express SQL Commands interface. The code entered is:

```

 Autocommit Display 10
INSERT INTO location VALUES (location_seq.NEXTVAL, 'Dhaka', 'Dhaka');
INSERT INTO location VALUES (location_seq.NEXTVAL, 'Chittagong', 'Chittagong');
INSERT INTO location VALUES (location_seq.NEXTVAL, 'Sylhet', 'Sylhet');
INSERT INTO location VALUES (location_seq.NEXTVAL, 'Khulna', 'Khulna');
INSERT INTO location VALUES (location_seq.NEXTVAL, 'Rajshahi', 'Rajshahi');

SELECT * FROM location

```

The results section displays the following table:

LOC_ID	DISTRICT	DIVISION
6001	Dhaka	Dhaka
6002	Chittagong	Chittagong
6003	Sylhet	Sylhet
6004	Khulna	Khulna
6005	Rajshahi	Rajshahi

5 rows returned in 0.00 seconds [CSV Export](#)

Language: en-us Application Express 2.1.0.00.39 Copyright © 1999, 2006, Oracle. All rights reserved.

**Table 7: court**

```

INSERT INTO court VALUES (court_seq.NEXTVAL, 'Supreme Court of Bangladesh', 6001);

INSERT INTO court VALUES (court_seq.NEXTVAL, 'District Judge Court', 6002);

INSERT INTO court VALUES (court_seq.NEXTVAL, 'Metropolitan Sessions Court', 6003);

INSERT INTO court VALUES (court_seq.NEXTVAL, 'Family Court', 6004);

INSERT INTO court VALUES (court_seq.NEXTVAL, 'Labour Court', 6005);

SELECT * FROM court

```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL editor contains the following code:

```

INSERT INTO court VALUES (court_seq.NEXTVAL, 'Supreme Court of Bangladesh', 6001);
INSERT INTO court VALUES (court_seq.NEXTVAL, 'District Judge Court', 6002);
INSERT INTO court VALUES (court_seq.NEXTVAL, 'Metropolitan Sessions Court', 6003);
INSERT INTO court VALUES (court_seq.NEXTVAL, 'Family Court', 6004);
INSERT INTO court VALUES (court_seq.NEXTVAL, 'Labour Court', 6005);

SELECT * FROM court

```

The results section displays the following table:

COURT_ID	COURT_TYPE	LOC_ID
5001	Supreme Court of Bangladesh	6001
5002	District Judge Court	6002
5003	Metropolitan Sessions Court	6003
5004	Family Court	6004
5005	Labour Court	6005

5 rows returned in 0.00 seconds [CSV Export](#)

**Table 8: case**

```

INSERT INTO case VALUES (case_seq.NEXTVAL, 'Pending', 'Divorce', 3001, 4001,
5001);

INSERT INTO case VALUES (case_seq.NEXTVAL, 'Active', 'Land Dispute', 3002, 4002,
5002);

INSERT INTO case VALUES (case_seq.NEXTVAL, 'Closed', 'Criminal', 3003, 4003,
5003);

INSERT INTO case VALUES (case_seq.NEXTVAL, 'Appeal', 'Civil', 3004, 4004, 5004);

INSERT INTO case VALUES (case_seq.NEXTVAL, 'Dismissed', 'Contract', 3005, 4005,
5005);

SELECT * FROM case

```

```

SQL Commands
127.0.0.1:8080/apex/f?p=4500:1003:1866682459616844::NO::
Home > SQL > SQL Commands
Save Run
Autocommit Display 10
INSERT INTO case VALUES (case_seq.NEXTVAL, 'Pending', 'Divorce', 3001, 4001, 5001);
INSERT INTO case VALUES (case_seq.NEXTVAL, 'Active', 'Land Dispute', 3002, 4002, 5002);
INSERT INTO case VALUES (case_seq.NEXTVAL, 'Closed', 'Criminal', 3003, 4003, 5003);
INSERT INTO case VALUES (case_seq.NEXTVAL, 'Appeal', 'Civil', 3004, 4004, 5004);
INSERT INTO case VALUES (case_seq.NEXTVAL, 'Dismissed', 'Contract', 3005, 4005, 5005);

SELECT * FROM case

```

Results Explain Describe Saved SQL History

CASE_ID	CASE_STATUS	CASE_TYPE	C_ID	L_ID	COURT_ID
7001	Pending	Divorce	3001	4001	5001
7002	Active	Land Dispute	3002	4002	5002
7003	Closed	Criminal	3003	4003	5003
7004	Appeal	Civil	3004	4004	5004
7005	Dismissed	Contract	3005	4005	5005

5 rows returned in 0.00 seconds [CSV Export](#)

Language: en-us Application Express 2.1.0.00.39  
Copyright © 1999, 2006, Oracle. All rights reserved.

32°C Partly sunny 4:58 PM 5/2/2025

**Table 9: appointment**

```

INSERT INTO appointment VALUES (app_seq.NEXTVAL, 'Scheduled', 'Initial Consultation', TIMESTAMP '2023-08-15 10:00:00', 3001, 4001);

INSERT INTO appointment VALUES (app_seq.NEXTVAL, 'Completed', 'Case Review', TIMESTAMP '2023-07-20 14:30:00', 3002, 4002);

INSERT INTO appointment VALUES (app_seq.NEXTVAL, 'Cancelled', 'Document Signing', TIMESTAMP '2023-08-01 11:00:00', 3003, 4003);

INSERT INTO appointment VALUES (app_seq.NEXTVAL, 'Rescheduled', 'Hearing Preparation', TIMESTAMP '2023-08-10 09:30:00', 3004, 4004);

INSERT INTO appointment VALUES (app_seq.NEXTVAL, 'Pending', 'Payment Discussion', TIMESTAMP '2023-08-05 16:00:00', 3005, 4005);

SELECT * FROM appointment

```

The screenshot shows a browser window titled "SQL Commands" with the URL [127.0.0.1:8080/apex/f?p=4500:1003:186682459616844::NO::](http://127.0.0.1:8080/apex/f?p=4500:1003:186682459616844::NO::). The page displays a SQL command block containing five INSERT statements into the "appointment" table and a SELECT \* FROM appointment statement. Below the command block is a results grid showing five rows of appointment data with columns APP\_ID, APP\_STATUS, APP\_DESCRIPTION, APP\_DATE\_TIME, C\_ID, and L\_ID. At the bottom of the page, there is a status bar showing the weather (32°C, Partly sunny), system icons, and the date/time (5/2/2025, 5:10 PM).

```

INSERT INTO appointment VALUES (app_seq.NEXTVAL, 'Scheduled', 'Initial Consultation', TIMESTAMP '2023-08-15 10:00:00', 3001, 4001);
INSERT INTO appointment VALUES (app_seq.NEXTVAL, 'Completed', 'Case Review', TIMESTAMP '2023-07-20 14:30:00', 3002, 4002);
INSERT INTO appointment VALUES (app_seq.NEXTVAL, 'Cancelled', 'Document Signing', TIMESTAMP '2023-08-01 11:00:00', 3003, 4003);
INSERT INTO appointment VALUES (app_seq.NEXTVAL, 'Rescheduled', 'Hearing Preparation', TIMESTAMP '2023-08-10 09:30:00', 3004, 4004);
INSERT INTO appointment VALUES (app_seq.NEXTVAL, 'Pending', 'Payment Discussion', TIMESTAMP '2023-08-05 16:00:00', 3005, 4005);

SELECT * FROM appointment

```

APP_ID	APP_STATUS	APP_DESCRIPTION	APP_DATE_TIME	C_ID	L_ID
8001	Scheduled	Initial Consultation	15-AUG-23 10.00.00.000000 AM	3001	4001
8002	Completed	Case Review	20-JUL-23 02.30.00.000000 PM	3002	4002
8003	Cancelled	Document Signing	01-AUG-23 11.00.00.000000 AM	3003	4003
8004	Rescheduled	Hearing Preparation	10-AUG-23 09.30.00.000000 AM	3004	4004
8005	Pending	Payment Discussion	05-AUG-23 04.00.00.000000 PM	3005	4005

5 rows returned in 0.00 seconds [CSV Export](#)

Application Express 2.1.0.00.39  
Copyright © 1999, 2006, Oracle. All rights reserved.

Language: en-us

32°C Partly sunny 5/2/2025 5:10 PM

**Table 10: payment**

```

INSERT INTO payment VALUES (payment_seq.NEXTVAL, 'bKash', 'Paid', 5000.00,
TIMESTAMP '2023-07-21 10:15:00', 3001, 8001);

INSERT INTO payment VALUES (payment_seq.NEXTVAL, 'Nagad', 'Pending', 10000.00,
TIMESTAMP '2023-08-02 14:00:00', 3002, 8002);

INSERT INTO payment VALUES (payment_seq.NEXTVAL, 'Bank Transfer', 'Paid',
20000.00, TIMESTAMP '2023-07-25 11:30:00', 3003, 8003);

INSERT INTO payment VALUES (payment_seq.NEXTVAL, 'Cash', 'Paid', 15000.00,
TIMESTAMP '2023-08-11 09:45:00', 3004, 8004);

INSERT INTO payment VALUES (payment_seq.NEXTVAL, 'Credit Card', 'Failed',
12000.00, TIMESTAMP '2023-08-06 16:20:00', 3005, 8005);

```

```
SELECT * FROM payment
```

The screenshot shows a browser window titled "SQL Commands" at the URL [127.0.0.1:8080/apex/f?p=4500:1003:186682459616844::NO::](http://127.0.0.1:8080/apex/f?p=4500:1003:186682459616844::NO::). The user is ADMSP. The page displays a SQL script and its results.

```

INSERT INTO payment VALUES (payment_seq.NEXTVAL, 'bkash', 'Paid', 5000.00, TIMESTAMP '2023-07-21 10:15:00', 3001, 8001);
INSERT INTO payment VALUES (payment_seq.NEXTVAL, 'Nagad', 'Pending', 10000.00, TIMESTAMP '2023-08-02 14:00:00', 3002, 8002);
INSERT INTO payment VALUES (payment_seq.NEXTVAL, 'Bank Transfer', 'Paid', 20000.00, TIMESTAMP '2023-07-25 11:30:00', 3003, 8003);
INSERT INTO payment VALUES (payment_seq.NEXTVAL, 'Cash', 'Paid', 15000.00, TIMESTAMP '2023-08-11 09:45:00', 3004, 8004);
INSERT INTO payment VALUES (payment_seq.NEXTVAL, 'Credit Card', 'Failed', 12000.00, TIMESTAMP '2023-08-06 16:20:00', 3005, 8005);

SELECT * FROM payment

```

The results table shows the following data:

P_ID	P_METHOD	P_STATUS	P_AMOUNT	P_DATE_TIME	C_ID	APP_ID
9001	bkash	Paid	5000	21-JUL-23 10.15.00.000000 AM	3001	8001
9002	Nagad	Pending	10000	02-AUG-23 02.00.00.000000 PM	3002	8002
9003	Bank Transfer	Paid	20000	25-JUL-23 11.30.00.000000 AM	3003	8003
9004	Cash	Paid	15000	11-AUG-23 09.45.00.000000 AM	3004	8004
9005	Credit Card	Failed	12000	06-AUG-23 04.20.00.000000 PM	3005	8005

5 rows returned in 0.01 seconds [CSV Export](#)

Language: en-us Application Express 2.1.0.00.39 Copyright © 1999, 2006, Oracle. All rights reserved. 5:24 PM 5/2/2025

**Table 11: rating**

```

INSERT INTO rating VALUES (rating_seq.NEXTVAL, 5, 'Excellent professional service', TIMESTAMP '2023-07-22 12:00:00', 3001, 4001, 8001);

INSERT INTO rating VALUES (rating_seq.NEXTVAL, 4, 'Good communication skills', TIMESTAMP '2023-07-21 15:30:00', 3002, 4002, 8002);

INSERT INTO rating VALUES (rating_seq.NEXTVAL, 3, 'Average case handling', TIMESTAMP '2023-07-26 10:00:00', 3003, 4003, 8003);

INSERT INTO rating VALUES (rating_seq.NEXTVAL, 5, 'Highly recommended lawyer', TIMESTAMP '2023-08-12 11:00:00', 3004, 4004, 8004);

INSERT INTO rating VALUES (rating_seq.NEXTVAL, 4, 'Satisfactory outcome', TIMESTAMP '2023-08-07 17:00:00', 3005, 4005, 8005);

SELECT * FROM rating

```

SQL Commands

127.0.0.1:8080/apex/f?p=4500:1003:1866824596168844::NO::

User: ADMSP

Home > SQL > SQL Commands

Autocommit

```
INSERT INTO rating VALUES (rating_seq.NEXTVAL, 5, 'Excellent professional service', TIMESTAMP '2023-07-22 12:00:00', 3001, 4001, 8001);
INSERT INTO rating VALUES (rating_seq.NEXTVAL, 4, 'Good communication skills', TIMESTAMP '2023-07-21 15:30:00', 3002, 4002, 8002);
INSERT INTO rating VALUES (rating_seq.NEXTVAL, 3, 'Average case handling', TIMESTAMP '2023-07-26 10:00:00', 3003, 4003, 8003);
INSERT INTO rating VALUES (rating_seq.NEXTVAL, 5, 'Highly recommended lawyer', TIMESTAMP '2023-08-12 11:00:00', 3004, 4004, 8004);
INSERT INTO rating VALUES (rating_seq.NEXTVAL, 4, 'Satisfactory outcome', TIMESTAMP '2023-08-07 17:00:00', 3005, 4005, 8005);

SELECT * FROM rating
```

**Results** [Explain](#) [Describe](#) [Saved SQL](#) [History](#)

R_ID	RATING	R_COMMENTS	R_DATE_TIME	C_ID	L_ID	APP_ID
10001	5	Excellent professional service	22-JUL-23 12.00.00.000000 PM	3001	4001	8001
10002	4	Good communication skills	21-JUL-23 03.30.00.000000 PM	3002	4002	8002
10003	3	Average case handling	26-JUL-23 10.00.00.000000 AM	3003	4003	8003
10004	5	Highly recommended lawyer	12-AUG-23 11.00.00.000000 AM	3004	4004	8004
10005	4	Satisfactory outcome	07-AUG-23 05.00.00.000000 PM	3005	4005	8005

5 rows returned in 0.00 seconds [CSV Export](#)

Language: en-us Application Express 2.1.0.00.39 Copyright © 1999, 2006, Oracle. All rights reserved.

31°C Partly sunny 5:36 PM 5/2/2025

# Query Writing

## Basic PL/SQL

### Variables:

1. Display the email of a user with user\_name = 'user001' using variables.

```
-----*
-- Project: Lawyer Appointment Management System   |
-- Semester: Spring 2024-2025                      |
-- Course: Advanced Database Management System      |
-- Section: C                                         |
-----*

DECLARE
    v_user_email VARCHAR2(50);
BEGIN
    SELECT email INTO v_user_email
    FROM users
    WHERE user_name = 'user001';

    DBMS_OUTPUT.PUT_LINE('Email of user001: ' || v_user_email);
END;
```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The code executed is:

```

-- Project: Lawyer Appointment Management System
-- Semester: Spring 2024-2025
-- Course: Advanced Database Management System
-- Section: C
*
DECLARE
    v_user_email VARCHAR2(50);
BEGIN
    SELECT email INTO v_user_email
    FROM users
    WHERE user_name = 'user001';

    DBMS_OUTPUT.PUT_LINE('Email of user001: ' || v_user_email);
END;

```

The results pane shows the output:

```

Email of user001: abdul.rahman@bdmail.com
Statement processed.

0.00 seconds

```

The status bar at the bottom right indicates "Application Express 2.1.0.00.39" and "Copyright © 1999, 2006, Oracle. All rights reserved."

2. Display the payment method of a client where payment id is 9005.

```

-----*
-- Project: Lawyer Appointment Management System
-- Semester: Spring 2024-2025
-- Course: Advanced Database Management System
-- Section: C
-----
*
DECLARE
    v_p_id      NUMBER := 9005;
    v_p_method  VARCHAR2(50);
BEGIN
    SELECT p_method INTO v_p_method
    FROM payment
    WHERE p_id = v_p_id;

    DBMS_OUTPUT.PUT_LINE('Payment Method For Payment ID 9005: ' || 
    v_p_method);
END;

```

```

-- Project: Lawyer Appointment Management System
-- Semester: Spring 2024-2025
-- Course: Advanced Database Management System
-- Section: C

DECLARE
    v_p_id      NUMBER := 9005;
    v_p_method  VARCHAR2(50);
BEGIN
    SELECT p_method INTO v_p_method
    FROM payment
    WHERE p_id = v_p_id;
    DBMS_OUTPUT.PUT_LINE('Payment Method For Payment ID 9005: ' || v_p_method);
END;

```

Results Explain Describe Saved SQL History

Payment Method For Payment ID 9005: Credit Card  
Statement processed.  
0.00 seconds

Language: en-us Application Express 2.1.0.03.99  
Copyright © 1999, 2006, Oracle. All rights reserved.

## Operators:

- Calculate the total payments made by client 3001 and apply a 15% discount. Display the original and discounted amounts.

```

-----*
-- Project: Lawyer Appointment Management System
-- Semester: Spring 2024-2025
-- Course: Advanced Database Management System
-- Section: C
-----*

DECLARE
    v_client_id NUMBER:= 3001;
    v_total_pay NUMBER;
    v_discounted_pay NUMBER;
BEGIN
    SELECT p_amount INTO v_total_pay
    FROM payment
    WHERE c_id = v_client_id;

    v_discounted_pay := v_total_pay * 0.85;

```

```

DBMS_OUTPUT.PUT_LINE('Original Total: ' || v_total_pay);
DBMS_OUTPUT.PUT_LINE('Discounted Total: ' || v_discounted_pay);
END;
/

```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The code entered is a PL/SQL block. It declares variables `v_client_id`, `v_total_pay`, and `v_discounted_pay`. It then performs a SELECT statement to put the payment amount into `v_total_pay` and calculates `v_discounted_pay` as 85% of `v_total_pay`. Finally, it outputs both totals using `DBMS_OUTPUT.PUT_LINE`. The results pane shows the output: `Original Total: 5000` and `Discounted Total: 4250`.

- Display name ID where First name is Ayesha.

```

-----*
-- Project: Lawyer Appointment Management System
-- Semester: Spring 2024-2025
-- Course: Advanced Database Management System
-- Section: C
-----*

DECLARE
    v_name_id VARCHAR(10);
BEGIN
    SELECT name_id INTO v_name_id
    FROM name
    WHERE f_name LIKE 'Ayesha';

    DBMS_OUTPUT.PUT_LINE('name_id for Ayesha: ' || v_name_id);

```

```
END ;
```

```
/
```

The screenshot shows a Windows desktop environment with a taskbar at the bottom. The taskbar includes icons for various applications like File Explorer, Google Chrome, and Microsoft Word. The system tray shows the date as 5/2/2025 and the time as 9:33 PM. The main window is titled "SQL Commands" and is connected to "127.0.0.1:8080/apex/f?p=4500:1003:1866824596168844:NO--". The title bar also displays "User ADMSP". The window content shows an Oracle Database Express Edition interface with a "Home > SQL > SQL Commands" path. A code editor pane contains the following PL/SQL block:

```
-- Project: Lawyer Appointment Management System
-- Semester: Spring 2024-2025
-- Course: Advanced Database Management System
-- Section: C
*
DECLARE
    v_name_id VARCHAR(10);
BEGIN
    SELECT name_id INTO v_name_id
    FROM name
    WHERE f_name LIKE 'Ayesha';

    DBMS_OUTPUT.PUT_LINE('name_id for Ayesha: ' || v_name_id);
END;
/
```

Below the code editor, there are tabs for "Results", "Explain", "Describe", "Saved SQL", and "History". The "Results" tab is active, showing the output of the executed query:

```
name_id for Ayesha: 1004
Statement processed.
```

The results are displayed in a light gray background with black text. The entire window has a standard Windows-style border and title bar.

## Single-Row Function:

1. Show the length of the email address of user003.

```
-----*
-- Project: Lawyer Appointment Management System
-- Semester: Spring 2024-2025
-- Course: Advanced Database Management System
-- Section: C
-----
*
DECLARE
    v_email VARCHAR2(100);
    v_email_length NUMBER;
BEGIN
    SELECT email INTO v_email
    FROM users
    WHERE user_name = 'user003';

    v_email_length := LENGTH(v_email);
```

```

        DBMS_OUTPUT.PUT_LINE('Email Length for user003: ' || v_email_length);
END;
/

```

The screenshot shows the Oracle Database Express Edition interface. In the SQL Commands window, a PL/SQL block is run:

```

-- Project: Lawyer Appointment Management System
-- Semester: Spring 2024-2025
-- Course: Advanced Database Management System
-- Section: C

DECLARE
    v_email VARCHAR2(100);
    v_email_length NUMBER;
BEGIN
    SELECT email INTO v_email
    FROM users
    WHERE user_name = 'user003';

    v_email_length := LENGTH(v_email);

    DBMS_OUTPUT.PUT_LINE('Email Length for user003: ' || v_email_length);
END;
/

```

The output shows the result of the query:

```

Email Length for user003: 18
Statement processed.

0.00 seconds

```

- Display the Full name in Upper case where name\_id is 1004.

```

-----*
--      Project: Lawyer Appointment Management System
--      Semester: Spring 2024-2025
--      Course: Advanced Database Management System
--      Section: C
-----*

DECLARE
    v_f_name VARCHAR(10);
    v_l_name VARCHAR(10);
BEGIN
    SELECT f_name, l_name INTO v_f_name, v_l_name
    FROM name
    WHERE name_id = 1004;

```

```

        DBMS_OUTPUT.PUT_LINE('Full Name: ' || UPPER(v_f_name) || ' ' || v_l_name);
    END;
/

```

The screenshot shows the Oracle Application Express SQL Commands interface. The code entered is a PL/SQL block that concatenates first and last names using the DBMS\_OUTPUT.PUT\_LINE procedure. The results pane shows the output 'Full Name: AYESHA SIDDIQUA'. The interface includes a toolbar at the top, a navigation bar, and a status bar at the bottom indicating the environment.

## Group Function:

- Display the total number of appointments for client c\_id = 3001.

```

-----*
-- Project: Lawyer Appointment Management System
-- Semester: Spring 2024-2025
-- Course: Advanced Database Management System
-- Section: C
-----*

DECLARE
    v_client_id NUMBER:= 3001;
    v_total_appointments NUMBER;
BEGIN
    SELECT COUNT(app_id) INTO v_total_appointments
    FROM appointment
    WHERE c_id = v_client_id;

```

```

        DBMS_OUTPUT.PUT_LINE('Total Appointments for Client 3001: ' || 
v_total_appointments);
END;

```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The code entered is:

```

-- Project: Lawyer Appointment Management System
-- Semester: Spring 2024-2025
-- Course: Advanced Database Management System
-- Section: C

DECLARE
    v_client_id NUMBER:= 3001;
    v_total_appointments NUMBER;
BEGIN
    SELECT COUNT(app_id) INTO v_total_appointments
    FROM appointment
    WHERE c_id = v_client_id;

    DBMS_OUTPUT.PUT_LINE('Total Appointments for Client 3001: ' || v_total_appointments);
END;

```

The results pane shows the output:

```

Total Appointments for Client 3001: 1
Statement processed.

0.01 seconds

```

- Display the highest and lowest payment amounts from the payment table.

```

-----*
-- Project: Lawyer Appointment Management System |
-- Semester: Spring 2024-2025 |
-- Course: Advanced Database Management System |
-- Section: C |
-----*

DECLARE
    v_max_payment NUMBER;
    v_min_payment NUMBER;
BEGIN
    SELECT MAX(p_amount), MIN(p_amount) INTO v_max_payment, v_min_payment
    FROM payment;

    DBMS_OUTPUT.PUT_LINE('Highest Payment: ' || v_max_payment );
    DBMS_OUTPUT.PUT_LINE('Lowest Payment: ' || v_min_payment );

```

```
END ;
```

```
/
```

The screenshot shows the Oracle Database Express Edition SQL Commands window. The code in the editor is:

```
-- Project: Lawyer Appointment Management System
-- Semester: Spring 2024-2025
-- Course: Advanced Database Management System
-- Section: C
*
DECLARE
    v_max_payment NUMBER;
    v_min_payment NUMBER;
BEGIN
    SELECT MAX(p.amount), MIN(p.amount) INTO v_max_payment, v_min_payment
    FROM payment;
    DBMS_OUTPUT.PUT_LINE('Highest Payment: ' || v_max_payment);
    DBMS_OUTPUT.PUT_LINE('Lowest Payment: ' || v_min_payment);
END;
```

The results pane shows the output of the query:

```
Highest Payment: 20000
Lowest Payment: 5000
Statement processed.
0.00 seconds
```

The system tray at the bottom right indicates it's 11:38 PM on 5/2/2025.

## Loop:

1. Display the average rating of lawyer l\_id = 4014 three times using a FOR loop.

```
*
-- Project: Lawyer Appointment Management System
-- Semester: Spring 2024-2025
-- Course: Advanced Database Management System
-- Section: C
*
DECLARE
    v_avg_rating NUMBER;
BEGIN
    SELECT avg_ratings INTO v_avg_rating
    FROM lawyer
    WHERE l_id = 4005;

    FOR i IN 1..3 LOOP
```

```

DBMS_OUTPUT.PUT_LINE( 'Lawyer ID 4005 | Average Rating: ' ||

v_avg_rating);

END LOOP;

END;

/

```

The screenshot shows the Oracle Database Express Edition SQL Commands window. The code entered is a PL/SQL block that prints the average rating for lawyer ID 4005 five times using a FOR loop. The results show five rows of output, each containing 'Lawyer ID 4005 | Average Rating: 4.3'. The status bar at the bottom indicates the statement was processed in 0.00 seconds.

```

-- Project: Lawyer Appointment Management System
-- Semester: Spring 2024-2025
-- Course: Advanced Database Management System
-- Section: C

DECLARE
    v_avg_rating NUMBER;
BEGIN
    SELECT avg_ratings INTO v_avg_rating
    FROM lawyer
    WHERE l_id = 4005;

    FOR i IN 1..3 LOOP
        DBMS_OUTPUT.PUT_LINE('Lawyer ID 4005 | Average Rating: ' || v_avg_rating);
    END LOOP;
END;
/

```

Results Explain Describe Saved SQL History

Lawyer ID 4005 | Average Rating: 4.3  
Lawyer ID 4005 | Average Rating: 4.3  
Lawyer ID 4005 | Average Rating: 4.3

Statement processed.

0.00 seconds

2. Display the bar council registration number of lawyer l\_id = 5005 five times using a WHILE loop.

```

-----*
-- Project: Lawyer Appointment Management System
-- Semester: Spring 2024-2025
-- Course: Advanced Database Management System
-- Section: C
-----*

DECLARE
    v_reg_no VARCHAR2(50);
    v_counter NUMBER := 1;
BEGIN
    SELECT bar_council_reg_no INTO v_reg_no FROM lawyer WHERE l_id = 4005;

    WHILE v_counter <= 5 LOOP

```

```

        DBMS_OUTPUT.PUT_LINE('Lawyer ID: 4005 | Bar Council Reg: ' ||

v_reg_no);

        v_counter := v_counter + 1;

    END LOOP;

END;

/

```

The screenshot shows the Oracle SQL Developer interface. The code in the SQL Commands window is:

```

-- Project: Lawyer Appointment Management System
-- Semester: Spring 2024-2025
-- Course: Advanced Database Management System
-- Section: C

DECLARE
    v_reg_no VARCHAR2(50);
    v_counter NUMBER := 1;
BEGIN
    SELECT bar_council_reg_no INTO v_reg_no FROM lawyer WHERE l_id = 4005;

    WHILE v_counter <= 5 LOOP
        DBMS_OUTPUT.PUT_LINE('Lawyer ID: 4005 | Bar Council Reg: ' || v_reg_no);
        v_counter := v_counter + 1;
    END LOOP;
END;
/

```

The results tab shows the output of the query:

```

Lawyer ID: 4005 | Bar Council Reg: BCR-RAJ-13579

```

## Conditional Statements:

1. Write the query to check if a client (c\_id = 3003) has any pending appointments.

```

-----*
-- Project: Lawyer Appointment Management System
-- Semester: Spring 2024-2025
-- Course: Advanced Database Management System
-- Section: C
-----*

DECLARE
    v_pending_count NUMBER;
BEGIN
    SELECT COUNT(*)
    INTO v_pending_count

```

```

        FROM appointment
        WHERE c_id = 3003 AND app_status = 'Pending';

        IF v_pending_count > 0 THEN
            DBMS_OUTPUT.PUT_LINE('Client 3003 has ' || v_pending_count || ' pending appointment(s).');
        ELSE
            DBMS_OUTPUT.PUT_LINE('No pending appointments for Client 3003.');
        END IF;
    END;
/

```

The screenshot shows the Oracle SQL Developer interface. In the top navigation bar, it says "User ADMSP". Below that, the path "Home > SQL > SQL Commands" is visible. The main area contains the PL/SQL code provided above. The code declares a variable `v_pending_count`, selects its count from the `appointment` table for client 3003 where status is 'Pending', and then prints the count or a message based on the result. The output window below the code shows the result: "No pending appointments for Client 3003." followed by "Statement processed.".

2. Categorizes a lawyer's (4004) rating into different categories using conditional statements.

```

-----*
-- Project: Lawyer Appointment Management System |
-- Semester: Spring 2024-2025 |
-- Course: Advanced Database Management System |
-- Section: C |
-----*

DECLARE
    v_l_id NUMBER := 4004;

```

```

v_avg_rating NUMBER;
v_category VARCHAR2(20);

BEGIN
    SELECT avg_ratings INTO v_avg_rating
    FROM lawyer
    WHERE l_id = v_l_id;

    IF v_avg_rating >= 4.5 THEN
        v_category := 'Excellent (4.5-5.0)';
    ELSIF v_avg_rating >= 4.0 THEN
        v_category := 'Good (4.0-4.49)';
    ELSIF v_avg_rating >= 3.0 THEN
        v_category := 'Average (3.0-3.99)';
    ELSE
        v_category := 'Needs Improvement (<3.0)';
    END IF;

    DBMS_OUTPUT.PUT_LINE('Lawyer ID: ' || v_l_id || ' | Rating: ' ||
    TO_CHAR(v_avg_rating, '9.99') || ' | Category: ' || v_category);
END;
/

```

The screenshot shows the Oracle Database Express Edition interface. In the SQL Commands window, a PL/SQL block is being run. The code declares variables, performs a SELECT query, and uses an IF-ELSIF-ELSE-ELSIF block to determine a category based on average rating. It then outputs the lawyer ID, rating, and category to the console. The output window shows the results of the execution.

```

-- Project: Lawyer Appointment Management System
-- Semester: Spring 2024-2025
-- Course: Advanced Database Management System
-- Section: C

DECLARE
    v_l_id NUMBER := 4004;
    v_avg_rating NUMBER;
    v_category VARCHAR2(20);
BEGIN
    SELECT avg_ratings INTO v_avg_rating
    FROM lawyer
    WHERE l_id = v_l_id;

    IF v_avg_rating >= 4.5 THEN
        v_category := 'Excellent (4.5-5.0)';
    ELSIF v_avg_rating >= 4.0 THEN
        v_category := 'Good (4.0-4.49)';
    ELSIF v_avg_rating >= 3.0 THEN
        v_category := 'Average (3.0-3.99)';
    ELSE
        v_category := 'Needs Improvement (<3.0)';
    END IF;

    DBMS_OUTPUT.PUT_LINE('Lawyer ID: ' || v_l_id || ' | Rating: ' || TO_CHAR(v_avg_rating, '9.99') || ' | Category: ' || v_category);
END;
/

```

Results Explain Describe Saved SQL History

Lawyer ID: 4004 | Rating: 4.50 | Category: Excellent (4.5-5.0)

Statement processed.

0.00 seconds

## Subquery:

1. Show the highest rated lawyer id.

```
-----*
-- Project: Lawyer Appointment Management System |
-- Semester: Spring 2024-2025 |
-- Course: Advanced Database Management System |
-- Section: C |
-----*

DECLARE
    v_l_id NUMBER;
BEGIN
    SELECT l_id INTO v_l_id
    FROM ( SELECT l_id FROM lawyer ORDER BY avg_ratings DESC)
    WHERE ROWNUM = 1;

    DBMS_OUTPUT.PUT_LINE('Highest rated lawyer ID: ' || v_l_id);
END;
/
```

The screenshot shows a browser window for Oracle Database Express Edition. The URL is 127.0.0.1:8080/apex/f?p=4500:1003:410121169368569::NO:::. The page content displays the PL/SQL code from the previous block. The results section shows the output: "Highest rated lawyer ID: 4003" and "Statement processed." Below the results, it says "0.00 seconds". At the bottom right, it shows "Application Express 2.1.0.0.39" and the date "5/3/2025".

```
-- Project: Lawyer Appointment Management System |
-- Semester: Spring 2024-2025 |
-- Course: Advanced Database Management System |
-- Section: C |
-----*
DECLARE
    v_l_id NUMBER;
BEGIN
    SELECT l_id INTO v_l_id
    FROM ( SELECT l_id FROM lawyer ORDER BY avg_ratings DESC)
    WHERE ROWNUM = 1;

    DBMS_OUTPUT.PUT_LINE('Highest rated lawyer ID: ' || v_l_id);
END;
/
```

Highest rated lawyer ID: 4003  
Statement processed.  
0.00 seconds

Application Express 2.1.0.0.39  
Copyright © 1999-2006 Oracle. All rights reserved.  
5/3/2025

2. Display the client ID with the lowest payment.

```

-----*
--      Project: Lawyer Appointment Management System      |
--      Semester: Spring 2024-2025                          |
--      Course: Advanced Database Management System        |
--      Section: C                                         |
-----*

DECLARE
    v_client_id NUMBER;
    v_min_payment NUMBER;

BEGIN
    SELECT c_id, p_amount INTO v_client_id, v_min_payment
    FROM ( SELECT c_id, p_amount FROM payment ORDER BY p_amount ASC)
    WHERE ROWNUM = 1;

    DBMS_OUTPUT.PUT_LINE('Client ' || v_client_id || ' has the lowest
payment: ' || v_min_payment);
END;
/

```

The screenshot shows the Oracle Database Express Edition interface. In the SQL Commands window, a PL/SQL block is run. The output pane displays the result: "Client 3001 has the lowest payment: 5000". The status bar at the bottom right indicates the session is running.

```

-----*
--      Project: Lawyer Appointment Management System      |
--      Semester: Spring 2024-2025                          |
--      Course: Advanced Database Management System        |
--      Section: C                                         |
-----*

DECLARE
    v_client_id NUMBER;
    v_min_payment NUMBER;

BEGIN
    SELECT c_id, p_amount INTO v_client_id, v_min_payment
    FROM ( SELECT c_id, p_amount FROM payment ORDER BY p_amount ASC)
    WHERE ROWNUM = 1;

    DBMS_OUTPUT.PUT_LINE('Client ' || v_client_id || ' has the lowest
payment: ' || v_min_payment);
END;
/

```

Results Explain Describe Saved SQL History

Client 3001 has the lowest payment: 5000  
Statement processed.  
0.02 seconds

30°C Haze 11:00 AM 5/3/2025 Application Express 21.0.0.0.39

## Joining:

1. Display detailed information for a specific lawyer (l\_id = 4003) using joins.

```
-----*
--      Project: Lawyer Appointment Management System      |
--      Semester: Spring 2024-2025                      |
--      Course: Advanced Database Management System       |
--      Section: C                                         |
-----*

DECLARE
    v_l_id NUMBER:= 4003;
    v_full_name VARCHAR2(100);
    v_specialization VARCHAR2(100);
    v_rating NUMBER;
    v_reg_no VARCHAR(50);
    v_email VARCHAR2(50);
    v_address VARCHAR2(200);

BEGIN
    SELECT n.f_name || ' ' || n.l_name, l.specialization, l.avg_ratings,
    l.bar_council_reg_no, u.email, a.city || ', ' || a.street || ', ' ||
    a.house_no
    INTO v_full_name, v_specialization, v_rating, v_reg_no, v_email,
    v_address
    FROM lawyer l
    JOIN users u ON l.user_name = u.user_name
    JOIN name n ON u.name_id = n.name_id
    JOIN address a ON u.a_id = a.a_id
    WHERE l.l_id = v_l_id;

    DBMS_OUTPUT.PUT_LINE('Name: ' || v_full_name);
    DBMS_OUTPUT.PUT_LINE('Specialization: ' || v_specialization);
    DBMS_OUTPUT.PUT_LINE('Rating: ' || v_rating);
    DBMS_OUTPUT.PUT_LINE('Registration: ' || v_reg_no);
    DBMS_OUTPUT.PUT_LINE('Email: ' || v_email);
    DBMS_OUTPUT.PUT_LINE('Address: ' || v_address);

END;
/
```

The screenshot shows the Oracle Database Express Edition SQL Commands window. The query displayed is:

```

-- Project: Lawyer Appointment Management System
-- Semester: Spring 2024-2025
-- Course: Advanced Database Management System
-- Section: C
-----*
DECLARE
  v_l_id NUMBER:= 4093;
  v_full_name VARCHAR2(100);
  v_specialization VARCHAR2(100);
  v_rating NUMBER;
  v_reg_no VARCHAR2(50);
  v_email VARCHAR2(50);
  v_address VARCHAR2(200);
BEGIN
  SELECT n.f_name || ' ' || n.l_name, l.specialization, l.avg_ratings, l.bar_council.reg_no, u.email, a.city || ', ' || a.street || ', ' || a.house_no
  INTO v_full_name, v_specialization, v_rating, v_reg_no, v_email, v_address
  FROM lawyer l
  JOIN users u ON l.user_name = u.user_name
  JOIN name n ON u.name_id = n.name_id
  JOIN address a ON u.id = a.id
  WHERE l.l_id = v_l_id;
  DBMS_OUTPUT.PUT_LINE('Name: ' || v_full_name);
  DBMS_OUTPUT.PUT_LINE('Specialization: ' || v_specialization);
  DBMS_OUTPUT.PUT_LINE('Rating: ' || v_rating);
END;

```

The results pane shows the output for a specific lawyer:

```

Name: Sara Rahim
Specialization: Family Law
Rating: 4.9
Registration: BCR-SYL-54321
Email: sara.rahim@example.com
Address: Mymensingh, Town Hall Road, 88-B
Statement processed.

```

2. Display detailed information for a client (e.g., c\_id = 3001) using joins

```

-----*
-- Project: Lawyer Appointment Management System
-- Semester: Spring 2024-2025
-- Course: Advanced Database Management System
-- Section: C
-----*
-----*
DECLARE
  v_c_id NUMBER := 3001;
  v_full_name VARCHAR2(100);
  v_email VARCHAR2(100);
  v_contact VARCHAR2(15);
  v_address VARCHAR2(200);
BEGIN
  SELECT n.f_name || ' ' || n.l_name, u.email, u.contact_no, a.city || ', '
  || a.street || ', ' || a.house_no
  INTO v_full_name, v_email, v_contact, v_address

```

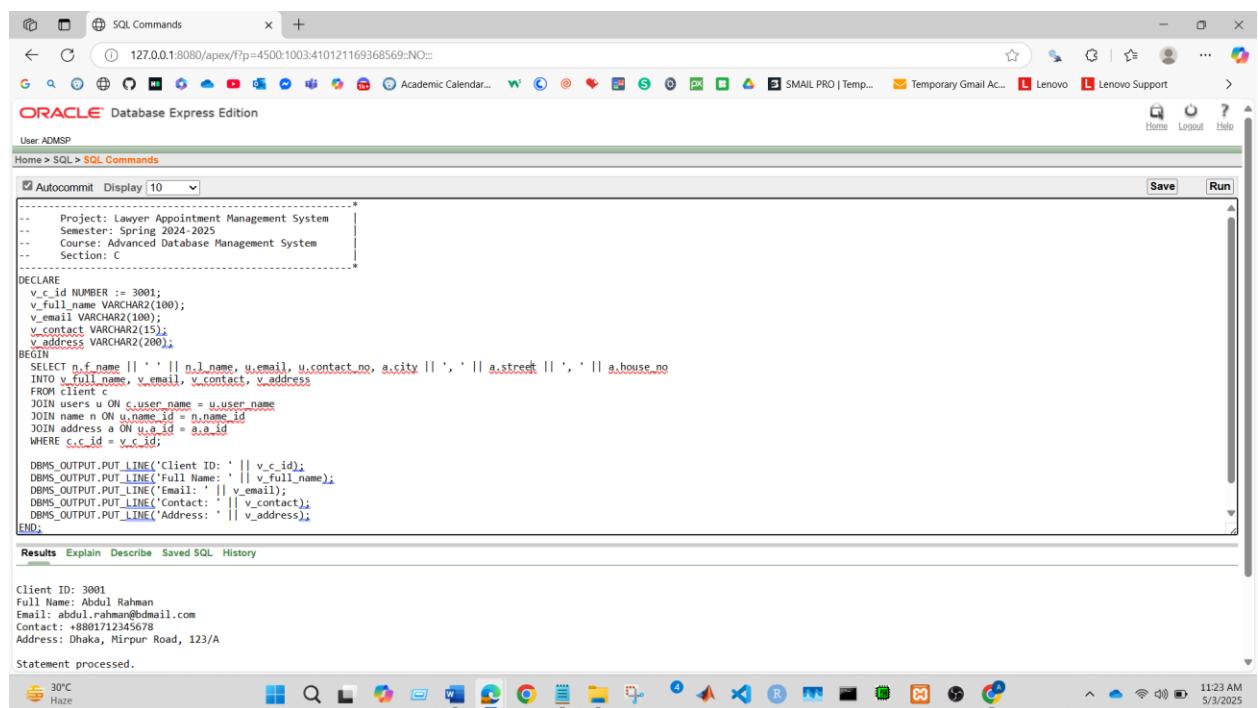
```

FROM client c
JOIN users u ON c.user_name = u.user_name
JOIN name n ON u.name_id = n.name_id
JOIN address a ON u.a_id = a.a_id
WHERE c.c_id = v_c_id;

DBMS_OUTPUT.PUT_LINE('Client ID: ' || v_c_id);
DBMS_OUTPUT.PUT_LINE('Full Name: ' || v_full_name);
DBMS_OUTPUT.PUT_LINE('Email: ' || v_email);
DBMS_OUTPUT.PUT_LINE('Contact: ' || v_contact);
DBMS_OUTPUT.PUT_LINE('Address: ' || v_address);

END;
/

```



```

-- Project: Lawyer Appointment Management System
-- Semester: Spring 2024-2025
-- Course: Advanced Database Management System
-- Section: C

DECLARE
    v_c_id NUMBER := 3001;
    v_full_name VARCHAR2(100);
    v_email VARCHAR2(100);
    v_contact VARCHAR2(15);
    v_address VARCHAR2(200);
BEGIN
    SELECT n.full_name || ' ' || u.l_name, u.email, u.contact_no, a.city || ', ' || a.street || ', ' || a.house_no
    INTO v_full_name, v_email, v_contact, v_address
    FROM client c
    JOIN users u ON c.user_name = u.user_name
    JOIN name n ON u.name_id = n.name_id
    JOIN address a ON u.a_id = a.a_id
    WHERE c.c_id = v_c_id;

    DBMS_OUTPUT.PUT_LINE('Client ID: ' || v_c_id);
    DBMS_OUTPUT.PUT_LINE('Full Name: ' || v_full_name);
    DBMS_OUTPUT.PUT_LINE('Email: ' || v_email);
    DBMS_OUTPUT.PUT_LINE('Contact: ' || v_contact);
    DBMS_OUTPUT.PUT_LINE('Address: ' || v_address);
END;

```

Results Explain Describe Saved SQL History

Client ID: 3001  
Full Name: Abdul Rahman  
Email: abdul.rahanan@gmail.com  
Contact: +8801712345678  
Address: Dhaka, Mirpur Road, 123/A

Statement processed.

# Advance PL/SQL

## Stored Function:

1. Write a function that takes lawyer ID and shows that lawyer's average rating from Lawyer table.

```
-----*
--      Project: Lawyer Appointment Management System      |
--      Semester: Spring 2024-2025                         |
--      Course: Advanced Database Management System          |
--      Section: C                                         |
-----*

CREATE OR REPLACE FUNCTION lawyer_avg_rating(lawyer_id IN NUMBER)
RETURN NUMBER
IS
    avg_rating NUMBER(3,2);
BEGIN
    SELECT avg_ratings
    INTO avg_rating
    FROM lawyer
    WHERE l_id = lawyer_id;
    RETURN avg_rating;
END;
/
SELECT lawyer_avg_rating(4001) FROM dual;
```

The screenshot shows a Microsoft Edge browser window with the URL `127.0.0.1:8080/apex/f?p=4500:1003:3284775352285948:NO::`. The page title is "What's new in Microsoft Edge" and the tab title is "SQL Commands". The content area displays Oracle Database Express Edition. A SQL command is being run:

```

IS
avg_rating NUMBER(3,2);
BEGIN
  SELECT avg_ratings
  INTO avg_rating
  FROM lawyer
  WHERE l_id = lawyer_id;
  RETURN avg_rating;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    RETURN NULL;
  WHEN OTHERS THEN
    RETURN NULL;
END;
SELECT lawyer_avg_rating(4001) FROM dual;

```

The results window shows a single row of data:

LAWYER_AVG_RATING(4001)
4.75

1 rows returned in 0.02 seconds

At the bottom right of the application window, it says "Application Express 2.1.0.05.39 Copyright © 1999, 2005, Oracle. All rights reserved."

2. Create a function that gives total payment made by one client using client ID from Payment table.

---

```

-- Project: Lawyer Appointment Management System |
-- Semester: Spring 2024-2025 |
-- Course: Advanced Database Management System |
-- Section: C |


---


CREATE OR REPLACE FUNCTION get_total_payment(client_id IN NUMBER)
RETURN NUMBER
IS
  total_payment NUMBER(10, 2);
BEGIN
  SELECT SUM(p_amount)
  INTO total_payment
  FROM payment
  WHERE c_id = client_id;

  RETURN total_payment;
END;
/

```

```
SELECT get_total_payment(3001) FROM dual;
```

The screenshot shows a Microsoft Edge browser window with the URL `127.0.0.1:8080/apex/f?p=4500:1003:3284775352285948::NO::`. The page title is "SQL Commands". The content area displays an Oracle SQL script and its execution results.

```
IS
total_payment NUMBER(10, 2);
BEGIN
  SELECT SUM(p_amount)
  INTO total_payment
  FROM payment
  WHERE c_id = client_id;
  RETURN total_payment;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    RETURN 0;
  WHEN OTHERS THEN
    RETURN NULL;
END;
/
SELECT get_total_payment(3001) FROM dual;
```

Results:

GET_TOTAL_PAYMENT(3001)
5000

1 rows returned in 0.01 seconds

CSV Export

Language: en-es Application Express 2.1.0.00.39 Copyright © 1999, 2006, Oracle. All rights reserved.

## Stored Procedure:

1. Write a procedure to change appointment status using appointment ID from appointment table.

```
-----*
--      Project: Lawyer Appointment Management System      |
--      Semester: Spring 2024-2025                         |
--      Course: Advanced Database Management System          |
--      Section: C                                         |
-----*
CREATE OR REPLACE PROCEDURE change_appointment_status(
    p_app_id IN NUMBER,
    p_new_status IN VARCHAR2
)
IS
BEGIN
    UPDATE appointment
    SET app_status = p_new_status
    WHERE app_id = p_app_id;
    COMMIT;

```

```

DBMS_OUTPUT.PUT_LINE('Appointment status updated successfully.');

END;
/
BEGIN
    change_appointment_status(8001, 'Completed');
END;
/

```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The code entered is a PL/SQL block that updates an appointment status and handles exceptions. The results pane shows the output of the executed code, indicating the status was updated successfully.

```

Autocommit: Display: 10
WHEN app_id = p_app_id;
COMMIT;
DBMS_OUTPUT.PUT_LINE('Appointment status updated successfully.');
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Appointment ID not found.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END;
/
BEGIN
    change_appointment_status(8001, 'Completed');
END;
/

```

Results:

Appointment status updated successfully.  
Statement processed.  
0.00 seconds

2. Write a procedure to add a new client with name and user info from client table.

```

-----*
-- Project: Lawyer Appointment Management System |
-- Semester: Spring 2024-2025 |
-- Course: Advanced Database Management System |
-- Section: C |
-----*

CREATE OR REPLACE PROCEDURE add_new_client (
    p_first_name IN VARCHAR2,
    p_last_name IN VARCHAR2,
    p_email IN VARCHAR2,
    p_password IN VARCHAR2,
    p_dob IN DATE,
    p_phone IN VARCHAR2,
```

```

    p_city IN VARCHAR2,
    p_street IN VARCHAR2,
    p_house IN VARCHAR2
)
IS
    v_name_id NUMBER;
    v_address_id NUMBER;
    v_user_id VARCHAR2(10);
BEGIN
    SELECT name_seq.NEXTVAL INTO v_name_id FROM dual;
    INSERT INTO name VALUES (v_name_id, p_first_name, p_last_name);

    SELECT address_seq.NEXTVAL INTO v_address_id FROM dual;
    INSERT INTO address VALUES (v_address_id, p_city, p_street, p_house);

    SELECT 'user' || LPAD(user_seq.NEXTVAL, 3, '0') INTO v_user_id FROM
dual;
    INSERT INTO users VALUES (v_user_id, p_email, p_password, p_dob,
p_phone, v_name_id, v_address_id);

    INSERT INTO client VALUES (client_seq.NEXTVAL, v_user_id);

    COMMIT;
END;
/

BEGIN
    add_new_client(
        'Mariam', 'Sultana', 'mariam.sultana@yahoo.com', 'pass12345',
        TO_DATE('1990-09-17','YYYY-MM-DD'),
        '+8801912345678', 'Sylhet', 'Zindabazar', 'Block A, 15'
    );
END;
/

```

The screenshot shows a Microsoft Edge browser window with the URL `127.0.0.1:8080/apex/f?p=4500:1003:3284775352285948::NO::`. The title bar says "What's new in Microsoft Edge" and "SQL Commands". The page header indicates "User LAWYER MANAGEMENT SYSTEM" and "Home > SQL > SQL Commands". The main content area contains the following PL/SQL code:

```

SELECT 'user' || LPAD(user_seq.NEXTVAL, 3, '0') INTO v_user_id FROM dual;
INSERT INTO users VALUES (v_user_id, p_email, p_password, p_dob, p_phone, v_name_id, v_address_id);
INSERT INTO client VALUES (client_seq.NEXTVAL, v_user_id);
COMMIT;
END;
/
BEGIN
add_new_client(
    p_name := 'Sultana', p_email := 'mariam.sultana@yahoo.com', p_pass12345,
    p_dob := TO_DATE('1990-09-17', 'YYYY-MM-DD'), p_contact_no := '+8801912345678',
    p_address := 'Zindabazar', p_block := 'Block A, 15'
);
END;
/

```

Below the code, the results show "Statement processed." and "0.02 seconds". At the bottom right, it says "Application Express 2.1.0.85.39 Copyright © 1999, 2005, Oracle. All rights reserved." The taskbar at the bottom shows various application icons.

## Table-Based Record:

1. Use a row-type variable to get one user's full details by username from user table.

```
-----*
-- Project: Lawyer Appointment Management System |
-- Semester: Spring 2024-2025 |
-- Course: Advanced Database Management System |
-- Section: C |
-----*
```

**DECLARE**

```

v_user users%ROWTYPE;
```

**BEGIN**

```

SELECT * INTO v_user
FROM users
WHERE user_name = 'user002';

DBMS_OUTPUT.PUT_LINE('User Name: ' || v_user.user_name);
DBMS_OUTPUT.PUT_LINE('Email: ' || v_user.email);
DBMS_OUTPUT.PUT_LINE('Password: ' || v_user.password);
DBMS_OUTPUT.PUT_LINE('DOB: ' || TO_CHAR(v_user.dob, 'YYYY-MM-DD'));
DBMS_OUTPUT.PUT_LINE('Contact No: ' || v_user.contact_no);
```

```

DBMS_OUTPUT.PUT_LINE('Name ID: ' || v_user.name_id);
DBMS_OUTPUT.PUT_LINE('Address ID: ' || v_user.a_id);
END;

```

The screenshot shows the Oracle Database Express Edition interface. In the SQL Commands tab, the following PL/SQL code is executed:

```

DECLARE
  v_user users%ROWTYPE;
BEGIN
  SELECT * INTO v_user
  FROM users
  WHERE user_name = 'user002';
  DBMS_OUTPUT.PUT_LINE('User Name: ' || v_user.user_name);
  DBMS_OUTPUT.PUT_LINE('Email: ' || v_user.email);
  DBMS_OUTPUT.PUT_LINE('Password: ' || v_user.password);
  DBMS_OUTPUT.PUT_LINE('DOB: ' || TO_CHAR(v_user.dob, 'YYYY-MM-DD'));
  DBMS_OUTPUT.PUT_LINE('Contact No: ' || v_user.contact_no);
  DBMS_OUTPUT.PUT_LINE('Name ID: ' || v_user.name_id);
  DBMS_OUTPUT.PUT_LINE('Address ID: ' || v_user.a_id);
END;
/

```

The results pane displays the output of the PL/SQL code:

```

User Name: user002
Email: fatema.begum@example.com
Password: bdpass456
DOB: 1992-01-01
Contact No: 01987654321
Name ID: 1002
Address ID: 2002

Statement processed.

0.05 seconds

```

At the bottom right of the interface, it says "Application Express 2.1 0.0.39 Copyright © 1999, 2006, Oracle. All rights reserved."

2. Use a row-type variable to update the city name in the address table from address table.

```

-----*
--      Project: Lawyer Appointment Management System   |
--      Semester: Spring 2024-2025                      |
--      Course: Advanced Database Management System       |
--      Section: C                                       |
-----*

DECLARE
  v_address address%ROWTYPE;
BEGIN
  SELECT * INTO v_address
  FROM address
  WHERE a_id = 2002;

  v_address.city := 'Chattogram';

  UPDATE address
  SET city = v_address.city

```

```

        WHERE a_id = v_address.a_id;

    COMMIT;

END;
/

```

```

DECLARE
    v_address address%ROWTYPE;
BEGIN
    SELECT * INTO v_address
    FROM address
    WHERE a_id = 2002;
    v_address.city := 'Chattogram';
    UPDATE address
    SET city = v_address.city
    WHERE a_id = v_address.a_id;
    COMMIT;
END;
/

```

Results Explain Describe Saved SQL History

1 row(s) updated.

0.00 seconds

Language: en-US Application Express 21.1.0.0.0 Copyright © 1999, 2005, Oracle. All rights reserved.

## Explicit Cursor:

1. Create a cursor to list all appointments where the status is 'Pending' from appointment table.

```

-----*
--      Project: Lawyer Appointment Management System   |
--      Semester: Spring 2024-2025                      |
--      Course: Advanced Database Management System       |
--      Section: C                                       |
-----*

DECLARE
    CURSOR pending_cursor IS
        SELECT app_id, app_status
        FROM appointment
        WHERE app_status = 'Pending';

    v_id appointment.app_id%TYPE;

```

```

v_status appointment.app_status%TYPE;
BEGIN
OPEN pending_cursor;
LOOP
FETCH pending_cursor INTO v_id, v_status;
EXIT WHEN pending_cursor%NOTFOUND;

DBMS_OUTPUT.PUT_LINE('Appointment ID: ' || v_id || ', Status: ' || 
v_status);
END LOOP;
CLOSE pending_cursor;
END;
/

```

The screenshot shows the Oracle Database Express Edition interface. In the top window, a SQL command is being run:

```

DECLARE
CURSOR pending_cursor IS
SELECT app_id, app_status
FROM appointment
WHERE app_status = 'Pending';
v_id appointment.app_id%TYPE;
v_status appointment.app_status%TYPE;
BEGIN
OPEN pending_cursor;
LOOP
FETCH pending_cursor INTO v_id, v_status;
EXIT WHEN pending_cursor%NOTFOUND;
DBMS_OUTPUT.PUT_LINE('Appointment ID: ' || v_id || ', Status: ' || v_status);
END LOOP;
CLOSE pending_cursor;
END;

```

In the bottom window, the results of the query are displayed:

```

Appointment ID: 8005, Status: Pending
Statement processed.
0.00 seconds

```

At the bottom right of the interface, it says "Application Express 2.1.0.0.39 Copyright © 1999, 2006, Oracle. All rights reserved."

2. Use a cursor to list all cases where case type is 'Criminal' from case table.

---

```

-- Project: Lawyer Appointment Management System |
-- Semester: Spring 2024-2025 |
-- Course: Advanced Database Management System |
-- Section: C |
-----*

```

**DECLARE**

```

CURSOR criminal_case_cur IS
    SELECT case_id, case_status, case_type
    FROM case
    WHERE case_type = 'Criminal';

v_case criminal_case_cur%ROWTYPE;
BEGIN
    OPEN criminal_case_cur;
    LOOP
        FETCH criminal_case_cur INTO v_case;
        EXIT WHEN criminal_case_cur%NOTFOUND;

        DBMS_OUTPUT.PUT_LINE('Case ID: ' || v_case.case_id ||
                            ', Title: ' || v_case.case_status ||
                            ', Type: ' || v_case.case_type);
    END LOOP;
    CLOSE criminal_case_cur;
END;
/

```

The screenshot shows the Oracle Database Express Edition interface. In the SQL Commands tab, a PL/SQL block is being run. The code defines a cursor for criminal cases, loops through them, and prints their ID, status, and type to the screen using DBMS\_OUTPUT.PUT\_LINE. The output pane shows the result for a single record: Case ID: 7003, Title: Closed, Type: Criminal. The bottom status bar indicates the environment and system information.

## Cursor-Based Record:

1. Use a cursor with record to show first name, last name, and contact number of all clients from client table.

```
-----*
--      Project: Lawyer Appointment Management System      |
--      Semester: Spring 2024-2025                      |
--      Course: Advanced Database Management System       |
--      Section: C                                       |
-----*

DECLARE

CURSOR client_cur IS
    SELECT n.f_name, n.l_name, u.contact_no
    FROM client c
    JOIN users u ON c.user_name = u.user_name
    JOIN name n ON u.name_id = n.name_id;

    client_rec client_cur%ROWTYPE;

BEGIN
    OPEN client_cur;
    LOOP
        FETCH client_cur INTO client_rec;
        EXIT WHEN client_cur%NOTFOUND;

        DBMS_OUTPUT.PUT_LINE(
            'Client: ' || client_rec.f_name || ' ' || client_rec.l_name ||
            ' | Contact: ' || client_rec.contact_no
        );
    END LOOP;
    CLOSE client_cur;
END;
/
```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The code entered is a PL/SQL block that loops through a cursor, concatenating client names and contact numbers, and then prints them. The output shows the concatenated data for several clients.

```

LOOP
    FETCH client_cur INTO client_rec;
    EXIT WHEN client_cur%NOTFOUND;

    DBMS_OUTPUT.PUT_LINE(
        'Client: ' || client_rec.f_name || ' ' || client_rec.l_name ||
        ' | Contact: ' || client_rec.contact_no
    );
END LOOP;
CLOSE client_cur;
END;
/

```

Results:

```

Client: Abdul Rahman | Contact: +8801712345678
Client: Fatema Begum | Contact: 01987654321
Client: Mohammad Hossain | Contact: +8801911122334
Client: Ayesha Siddiqua | Contact: 01876543210
Client: Iqbal Khan | Contact: +8801555555555
Client: Marian Sultana | Contact: +8801912345678

Statement processed.

0.00 seconds

```

2. Use a cursor with record to list lawyers who have rating 4.50 from lawyer table.

```

-----*
--      Project: Lawyer Appointment Management System      |
--      Semester: Spring 2024-2025                      |
--      Course: Advanced Database Management System       |
--      Section: C                                       |
-----*

DECLARE

    CURSOR lawyer_cur IS
        SELECT n.f_name, n.l_name, l.specialization, l.avg_ratings
        FROM lawyer l
        JOIN users u ON l.user_name = u.user_name
        JOIN name n ON u.name_id = n.name_id
        WHERE l.avg_ratings = 4.50;

        lawyer_rec lawyer_cur%ROWTYPE;

BEGIN
    OPEN lawyer_cur;
    LOOP
        FETCH lawyer_cur INTO lawyer_rec;
        EXIT WHEN lawyer_cur%NOTFOUND;
    END LOOP;
END;

```

```

DBMS_OUTPUT.PUT_LINE(
    'Lawyer: ' || lawyer_rec.f_name || ' ' || lawyer_rec.l_name ||
    ' | Specialization: ' || lawyer_rec.specialization ||
    ' | Rating: ' || lawyer_rec.avg_ratings
);
END LOOP;
CLOSE lawyer_cur;
END;
/

```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The code entered is:

```

-- Project: Lawyer Appointment Management System
-- Semester: Spring 2024-2025
-- Course: Advanced Database Management System
-- Section: C

DECLARE
  CURSOR lawyer_cur IS
    SELECT n.f_name, n.l_name, l.specialization, l.avg_ratings
    FROM lawyer l
    JOIN users u ON l.user_name = u.user_name
    JOIN name n ON u.name_id = n.name_id
    WHERE l.avg_ratings = 4.5;

```

The results pane displays the output of the query:

```

Lawyer: Zahid Mahmud | Specialization: Civil Law | Rating: 4.5
Statement processed.

0.00 seconds

```

At the bottom, the status bar indicates "Language: en-us" and "Application Express 2.1.0.0.39 Copyright © 1999, 2006, Oracle. All rights reserved."

## Row-Level Trigger:

1. Make a trigger to stop adding a payment if the amount is less than 10000.00 from payment table.

```

-----*
-- Project: Lawyer Appointment Management System
-- Semester: Spring 2024-2025
-- Course: Advanced Database Management System
-- Section: C
-----
CREATE OR REPLACE TRIGGER payment_amount_check

```

```

BEFORE INSERT ON payment
FOR EACH ROW
BEGIN
    IF :NEW.p_amount < 10000.00 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Payment amount must be at least
10,000.00 BDT.');
    END IF;
END;
/

```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL editor contains the following code:

```

-- Project: Lawyer Appointment Management System
-- Semester: Spring 2024-2025
-- Course: Advanced Database Management System
-- Section: C
CREATE OR REPLACE TRIGGER payment_amount_check
BEFORE INSERT ON payment
FOR EACH ROW
BEGIN
    IF :NEW.p_amount < 10000.00 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Payment amount must be at least 10,000.00 BDT.');
    END IF;
END;
/

```

The results pane shows the message "Trigger created." and a execution time of "0.00 seconds". The status bar at the bottom right indicates "Application Express 2.1.0.00.39" and "Copyright © 1999, 2006, Oracle. All rights reserved."

2. Make a trigger to show a message when a new rating is inserted from rating table

```

-----*
-- Project: Lawyer Appointment Management System
-- Semester: Spring 2024-2025
-- Course: Advanced Database Management System
-- Section: C
-----
CREATE OR REPLACE TRIGGER rating_insert_message
AFTER INSERT ON rating
FOR EACH ROW
BEGIN

```

```

DBMS_OUTPUT.PUT_LINE('New rating added with ID: ' || :NEW.r_id || ','
Rating: ' || :NEW.rating);
END;
/

```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL editor contains the following code:

```

-- Project: Lawyer Appointment Management System
-- Semester: Spring 2024-2025
-- Course: Advanced Database Management System
-- Section: C
-----*
CREATE OR REPLACE TRIGGER rating_insert_message
AFTER INSERT ON rating
FOR EACH ROW
BEGIN
    DBMS_OUTPUT.PUT_LINE('New rating added with ID: ' || :NEW.r_id || ', Rating: ' || :NEW.rating);
END;
/

```

The results pane shows the output of the trigger creation command:

```

Trigger created.

0.00 seconds

```

The status bar at the bottom right indicates "Application Express 2.1.0.00.39" and "Copyright © 1999, 2006, Oracle. All rights reserved."

### Statement-Level Trigger:

1. Make a trigger that logs the date and time when any update happens in the appointment table.

```

-----*
-- Project: Lawyer Appointment Management System
-- Semester: Spring 2024-2025
-- Course: Advanced Database Management System
-- Section: C
-----*
CREATE TABLE appointment_audit (
    log_id      NUMBER PRIMARY KEY,
    action      VARCHAR2(20),
    action_date DATE
);

```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL code entered is:

```
-- Project: Lawyer Appointment Management System
-- Semester: Spring 2024-2025
-- Course: Advanced Database Management System
-- Section: C

CREATE TABLE appointment_audit (
    log_id NUMBER PRIMARY KEY,
    action VARCHAR2(20),
    action_date DATE
);

CREATE OR REPLACE TRIGGER trg_appointment_audit_id
BEFORE INSERT ON appointment_audit
```

The results pane shows the output of the SQL commands:

```
Table created.
```

0.00 seconds

Language: en-us Application Express 2.1.0.00.39 Copyright © 1999, 2006, Oracle. All rights reserved.

```
CREATE OR REPLACE TRIGGER trg_appointment_audit_id
BEFORE INSERT ON appointment_audit
FOR EACH ROW
DECLARE
    v_max_id NUMBER;
BEGIN
    SELECT NVL(MAX(log_id), 0) INTO v_max_id FROM appointment_audit;
    :NEW.log_id := v_max_id + 1;
END;
/
```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL editor contains the following PL/SQL code:

```
action_date DATE
);
CREATE OR REPLACE TRIGGER trg_appointment_audit_id
BEFORE INSERT ON appointment_audit
FOR EACH ROW
DECLARE
    v_max_id NUMBER;
BEGIN
    SELECT NVL(MAX(log_id), 0) INTO v_max_id FROM appointment_audit;
    :NEW.log_id := v_max_id + 1;
END;
/
```

The results pane shows the output of the trigger creation command:

```
Trigger created.
```

0.00 seconds

Language: en-us Application Express 2.1.0.0.39  
Copyright © 1999, 2006, Oracle. All rights reserved.

```
CREATE OR REPLACE TRIGGER trg_log_appointment_update
AFTER UPDATE ON appointment
BEGIN
    INSERT INTO appointment_audit (action, action_date)
    VALUES ('UPDATE', SYSDATE);
END;
/
```

SQL Commands

ORACLE Database Express Edition

User ADMSP

Home > SQL > SQL Commands

```

 Autocommit  Display 10  Save  Run
BEGIN
    SELECT NVL(MAX(log_id), 0) INTO v_max_id FROM appointment_audit;
    :NEW.log_id := v_max_id + 1;
END;
/
CREATE OR REPLACE TRIGGER trg_log_appointment_update
AFTER UPDATE ON appointment
BEGIN
    INSERT INTO appointment_audit (action, action_date)
    VALUES ('UPDATE', SYSDATE);
END;
/

```

Results Explain Describe Saved SQL History

Trigger created.

0.02 seconds

Language: en-us Application Express 2.1.0.0.39 Copyright © 1999, 2006, Oracle. All rights reserved.

32°C Haze 3:32 PM 5/3/2025

**UPDATE appointment SET app\_status = 'Completed' WHERE app\_id = 8001;**

SQL Commands

ORACLE Database Express Edition

User ADMSP

Home > SQL > SQL Commands

```

 Autocommit  Display 10  Save  Run
CREATE OR REPLACE TRIGGER trg_log_appointment_update
AFTER UPDATE ON appointment
BEGIN
    INSERT INTO appointment_audit (action, action_date)
    VALUES ('UPDATE', SYSDATE);
END;
/
UPDATE appointment SET app_status = 'Completed' WHERE app_id = 8001;
SELECT * FROM appointment_audit;

```

Results Explain Describe Saved SQL History

Appointment ID: 8001 | Old Status: Completed | New Status: Completed | Update Time: 2025-05-03 15:33:00  
1 row(s) updated.

0.00 seconds

Language: en-us Application Express 2.1.0.0.39 Copyright © 1999, 2006, Oracle. All rights reserved.

32°C Haze 3:33 PM 5/3/2025

**SELECT \* FROM appointment\_audit;**

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL command window contains the following code:

```

CREATE OR REPLACE TRIGGER trg_log_appointment_update
AFTER UPDATE ON appointment
BEGIN
    INSERT INTO appointment_audit (action, action_date)
    VALUES ('UPDATE', SYSDATE);
END;
/
UPDATE appointment SET app_status = 'Completed' WHERE app_id = 8001;
SELECT * FROM appointment_audit;

```

The results pane shows a single row from the appointment\_audit table:

LOG_ID	ACTION	ACTION_DATE
1	UPDATE	03-MAY-25

1 rows returned in 0.00 seconds

Language: en-us Application Express 2.1.0.00.39 Copyright © 1999, 2006, Oracle. All rights reserved.

2. Make a trigger to count how many users are added after an insert from users table.

```

CREATE TABLE user_statistics (
    total_users NUMBER
);

```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL command window contains the following code:

```

-- Project: Lawyer Appointment Management System
-- Semester: Spring 2024-2025
-- Course: Advanced Database Management System
-- Section: C

CREATE TABLE user_statistics (
    total_users NUMBER
);

CREATE OR REPLACE TRIGGER trg_count_users
AFTER INSERT ON users
BEGIN
    UPDATE user_statistics

```

The results pane shows the message "Table created." and "0.00 seconds".

Language: en-us Application Express 2.1.0.00.39 Copyright © 1999, 2006, Oracle. All rights reserved.

```
CREATE OR REPLACE TRIGGER trg_log_appointment_update
```

```

AFTER UPDATE ON appointment
DECLARE
    PRAGMA AUTONOMOUS_TRANSACTION; -- Ensure logging works independently
BEGIN
    -- Auto-generate log_id using MAX + 1
    INSERT INTO appointment_audit (log_id, action, action_date)
    VALUES (
        (SELECT NVL(MAX(log_id), 0) + 1 FROM appointment_audit),
        'UPDATE',
        SYSDATE
    );
    COMMIT;
END;
/

```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL editor contains the code for creating an AFTER UPDATE trigger on the 'appointment' table. The trigger inserts a new row into the 'appointment\_audit' table with the current log\_id, 'UPDATE' as the action, and the current system date. The code is highlighted in blue, indicating it is valid SQL.

```

AFTER UPDATE ON appointment
DECLARE
    PRAGMA AUTONOMOUS_TRANSACTION; -- Ensure logging works independently
BEGIN
    -- Auto-generate log_id using MAX + 1
    INSERT INTO appointment_audit (log_id, action, action_date)
    VALUES (
        (SELECT NVL(MAX(log_id), 0) + 1 FROM appointment_audit),
        'UPDATE',
        SYSDATE
    );
    COMMIT;
END;
/

```

The results pane shows the message "Trigger created." and a execution time of "0.00 seconds". The bottom status bar indicates the language is "en-us" and the application version is "Application Express 2.1.0.0.39".

## Package:

1. Create a package that one function to return user name and create one procedure to update contact number from users table.

---

```

-- Project: Lawyer Appointment Management System
-- Semester: Spring 2024-2025

```

```

-- Course: Advanced Database Management System
-- Section: C
-----
CREATE OR REPLACE PACKAGE USER_MGMT AS
    FUNCTION GET_USER_NAME(p_user_name VARCHAR2) RETURN VARCHAR2;
    PROCEDURE UPDATE_CONTACT(
        p_user_name    VARCHAR2,
        p_new_contact VARCHAR2
    );
END USER_MGMT;
/

```

The screenshot shows the Oracle Database Express Edition interface. In the top navigation bar, it says "ORACLE Database Express Edition". Below that, it shows "User ADMSP" and "Home > SQL > SQL Commands". The main area contains the SQL code for creating the package. At the bottom, the results show "Package created." and "0.00 seconds". The status bar at the bottom right indicates "Application Express 2.1.0.00.39" and "Copyright © 1999, 2006, Oracle. All rights reserved."

```

-- Project: Lawyer Appointment Management System
-- Semester: Spring 2024-2025
-- Course: Advanced Database Management System
-- Section: C
CREATE OR REPLACE PACKAGE USER_MGMT AS
    FUNCTION GET_USER_NAME(p_user_name VARCHAR2) RETURN VARCHAR2;
    PROCEDURE UPDATE_CONTACT(
        p_user_name    VARCHAR2,
        p_new_contact VARCHAR2
    );
END USER_MGMT;
/

```

```

CREATE OR REPLACE PACKAGE BODY USER_MGMT AS
    FUNCTION GET_USER_NAME(p_user_name VARCHAR2) RETURN VARCHAR2 IS
        v_full_name VARCHAR2(100);
    BEGIN
        SELECT n.f_name || ' ' || n.l_name
        INTO v_full_name
        FROM users u
        JOIN name n ON u.name_id = n.name_id
        WHERE u.user_name = p_user_name;

        RETURN v_full_name;
    END;

```

```

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN 'User not found.';
END GET_USER_NAME;

PROCEDURE UPDATE_CONTACT(
    p_user_name    VARCHAR2,
    p_new_contact VARCHAR2
) IS
BEGIN
    UPDATE users
    SET contact_no = p_new_contact
    WHERE user_name = p_user_name;

    IF SQL%ROWCOUNT = 0 THEN
        DBMS_OUTPUT.PUT_LINE('No user found with username: ' || p_user_name);
    ELSE
        DBMS_OUTPUT.PUT_LINE('Contact updated for user: ' || p_user_name);
    END IF;
END UPDATE_CONTACT;
END USER_MGMT;
/

```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The code entered is:

```
    ) IS
    BEGIN
        UPDATE users
        SET contact_no = p_new_contact
        WHERE user_name = p_user_name;
        IF SQL%ROWCOUNT = 0 THEN
            DBMS_OUTPUT.PUT_LINE('No user found with username: ' || p_user_name);
        ELSE
            DBMS_OUTPUT.PUT_LINE('Contact updated for user: ' || p_user_name);
        END IF;
        END UPDATE_CONTACT;
    END USER_MGMT;
/
```

The results section shows:

```
Package Body created.
```

0.00 seconds

Language: en-us Application Express 2.1.0.0.39 Copyright © 1999, 2006, Oracle. All rights reserved.

```
DECLARE
    v_name VARCHAR2(100);
BEGIN
    v_name := USER_MGMT.GET_USER_NAME('user001');
    DBMS_OUTPUT.PUT_LINE('User Name: ' || v_name);
END;
/
```

SQL Commands

User ADMSP

ORACLE Database Express Edition

Home > SQL > SQL Commands

```

 Autocommit  Display 10  Save  Run
END USER_MGMT;
/
DECLARE
    v_name VARCHAR2(100);
BEGIN
    v_name := USER_MGMT.GET_USER_NAME('user001');
    DBMS_OUTPUT.PUT_LINE('User Name: ' || v_name);
END;
/
BEGIN
    USER_MGMT.UPDATE_CONTACT('user001', '+8801712345678');
END;
/

```

Results Explain Describe Saved SQL History

User Name: Abdul Rahman  
Statement processed.  
0.00 seconds

Language en-us 127.0.0.1:8080/apex/f?p=4500:1003:268477585942013 Application Express 2.1.0.0.39 Copyright © 1999, 2006, Oracle. All rights reserved.

```

BEGIN
    USER_MGMT.UPDATE_CONTACT('user001', '+8801712345678');
END;
/

```

SQL Commands

User ADMSP

ORACLE Database Express Edition

Home > SQL > SQL Commands

```

 Autocommit  Display 10  Save  Run
/
DECLARE
    v_name VARCHAR2(100);
BEGIN
    v_name := USER_MGMT.GET_USER_NAME('user001');
    DBMS_OUTPUT.PUT_LINE('User Name: ' || v_name);
END;
/
BEGIN
    USER_MGMT.UPDATE_CONTACT('user001', '+8801712345678');
END;
/

```

Results Explain Describe Saved SQL History

Contact updated for user: user001  
Statement processed.  
0.00 seconds

Language en-us 127.0.0.1:8080/apex/f?p=4500:1003:268477585942013 Application Express 2.1.0.0.39 Copyright © 1999, 2006, Oracle. All rights reserved.

2. Create another package that one procedure to show all payments of one client, and create one function to count total p\_amount from payment table.

```
-----*
-- Project: Lawyer Appointment Management System
-- Semester: Spring 2024-2025
-- Course: Advanced Database Management System
-- Section: C
-----*
```

```
CREATE OR REPLACE PACKAGE PAYMENT_MGMT AS
  PROCEDURE SHOW_CLIENT_PAYMENTS(p_c_id NUMBER);
  FUNCTION GET_TOTAL_AMOUNT(p_c_id NUMBER) RETURN NUMBER;
END PAYMENT_MGMT;
```

The screenshot shows the Oracle Database Express Edition interface. In the top navigation bar, it says "SQL Commands". Below that, the URL is "127.0.0.1:8080/apex/f?p=4500:1003:268477585942013::NO::". The main area is titled "ORACLE Database Express Edition" and shows the SQL command window. The command entered is:

```
-- Project: Lawyer Appointment Management System
-- Semester: Spring 2024-2025
-- Course: Advanced Database Management System
-- Section: C
CREATE OR REPLACE PACKAGE PAYMENT_MGMT AS
  PROCEDURE SHOW_CLIENT_PAYMENTS(p_c_id NUMBER);
  FUNCTION GET_TOTAL_AMOUNT(p_c_id NUMBER) RETURN NUMBER;
END PAYMENT_MGMT;
```

Below the command window, the results pane shows the message "Package created." and "0.00 seconds". At the bottom, the status bar indicates "Language: en-us" and "Application Express 2.1.0.0.39 Copyright © 1999, 2006, Oracle. All rights reserved." The system tray at the bottom right shows the date and time as "5/3/2023 3:08 PM".

```
CREATE OR REPLACE PACKAGE BODY PAYMENT_MGMT AS
  PROCEDURE SHOW_CLIENT_PAYMENTS(p_c_id NUMBER) IS
  BEGIN
    DBMS_OUTPUT.PUT_LINE('Payments for Client ID ' || p_c_id || ':');
    DBMS_OUTPUT.PUT_LINE('-----');

    FOR payment_rec IN (
      SELECT p_id, p_method, p_amount, p_date_time
```

```

        FROM payment
        WHERE c_id = p_c_id
    ) LOOP
    DBMS_OUTPUT.PUT_LINE(
        'Payment ID: ' || payment_rec.p_id ||
        ', Method: ' || payment_rec.p_method ||
        ', Amount: ' || payment_rec.p_amount ||
        ', Date: ' || TO_CHAR(payment_rec.p_date_time, 'DD-MON-YYYY
HH24:MI')
    );
END LOOP;
END SHOW_CLIENT_PAYMENTS;

FUNCTION GET_TOTAL_AMOUNT(p_c_id NUMBER) RETURN NUMBER IS
    v_total NUMBER := 0;
BEGIN
    SELECT SUM(p_amount)
    INTO v_total
    FROM payment
    WHERE c_id = p_c_id;

    RETURN v_total;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN 0;
END GET_TOTAL_AMOUNT;

```

**END**

**PAYMENT\_MGMT**

```

v_total NUMBER := 0;
BEGIN
  SELECT SUM(p.amount)
  INTO v_total
  FROM payment
  WHERE c_id = p_c_id;
  RETURN v_total;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    RETURN 0;
END GET_TOTAL_AMOUNT;
END PAYMENT_MGMT;

```

Results Explain Describe Saved SQL History

Package Body created.

0.00 seconds

Language: en-us Application Express 2.1.0.0.39 Copyright © 1999, 2006, Oracle. All rights reserved.

**BEGIN**

```

PAYMENT_MGMT.SHOW_CLIENT_PAYMENTS(3001);
END;
/

```

```

BEGIN
  PAYMENT_MGMT.SHOW_CLIENT_PAYMENTS(3001);
END;
/

DECLARE
  v_total NUMBER;
BEGIN
  v_total := PAYMENT_MGMT.GET_TOTAL_AMOUNT(3001);
  DBMS_OUTPUT.PUT_LINE('Total Payment: ' || v_total);
END;
/

```

Results Explain Describe Saved SQL History

Payments for Client ID 3001:  
-----  
Payment ID: 9001, Method: bKash, Amount: 5000, Date: 21-JUL-2023 10:15  
Statement processed.

0.02 seconds

Language: en-us Application Express 2.1.0.0.39 Copyright © 1999, 2006, Oracle. All rights reserved.

**DECLARE**

```

v_total NUMBER;

BEGIN
    v_total := PAYMENT_MGMT.GET_TOTAL_AMOUNT(3001);
    DBMS_OUTPUT.PUT_LINE('Total Payment: ' || v_total);
END;
/

```

The screenshot shows a browser window titled "SQL Commands" connected to "127.0.0.1:8080/apex/f?p=4500:1003:268477585942013::NO::". The page title is "ORACLE Database Express Edition". The SQL command entered is:

```

BEGIN
    PAYMENT_MGMT.SHOW_CLIENT_PAYMENTS(3001);
END;
/

```

Below this, another block of code is shown:

```

DECLARE
    v_total NUMBER;
BEGIN
    v_total := PAYMENT_MGMT.GET_TOTAL_AMOUNT(3001);
    DBMS_OUTPUT.PUT_LINE('Total Payment: ' || v_total);
END;
/

```

The results section shows the output:

```

Total Payment: 5000
Statement processed.

0.00 seconds

```

At the bottom, the system status bar indicates "Language: en-us", "Application Express 2.1.0.00.39", and "Copyright © 1999, 2006, Oracle. All rights reserved." The taskbar at the bottom shows various application icons.

## Conclusion:

This system connects clients and lawyers in Bangladesh through a secure, easy-to-use platform. It includes features like appointment booking, case and payment tracking, and verified profiles. Built with a well-structured database and PL/SQL automation (via stored procedures, functions, and triggers), it ensures smooth operation and data accuracy. This system makes legal services more accessible, organized, and trustworthy.