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

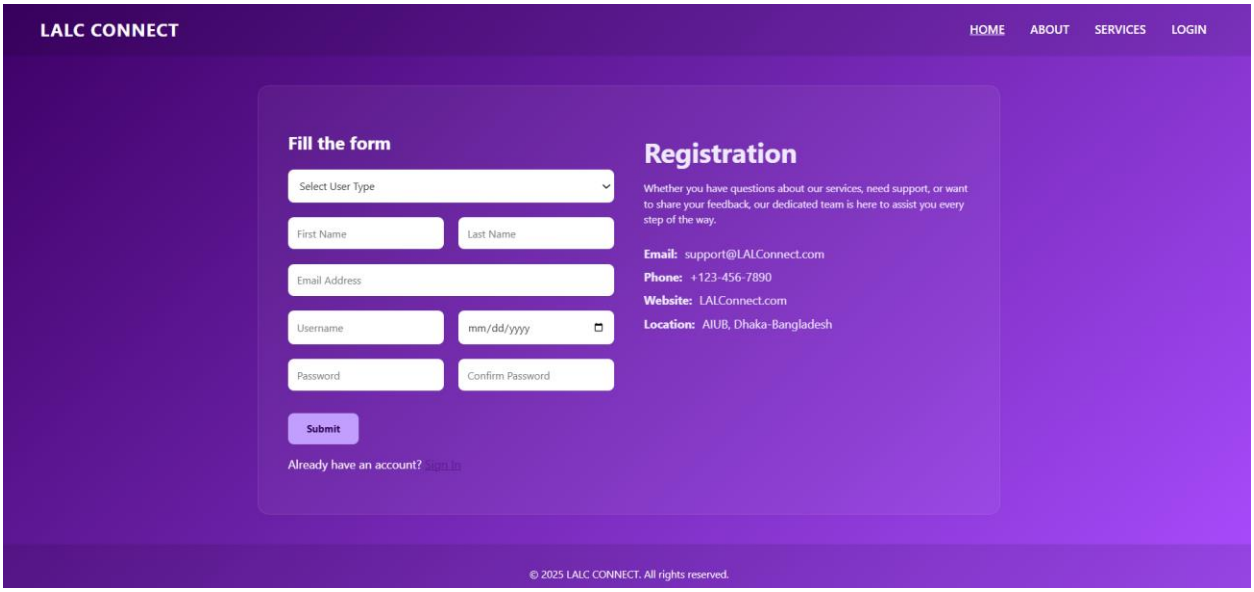
Contents Page

Introduction

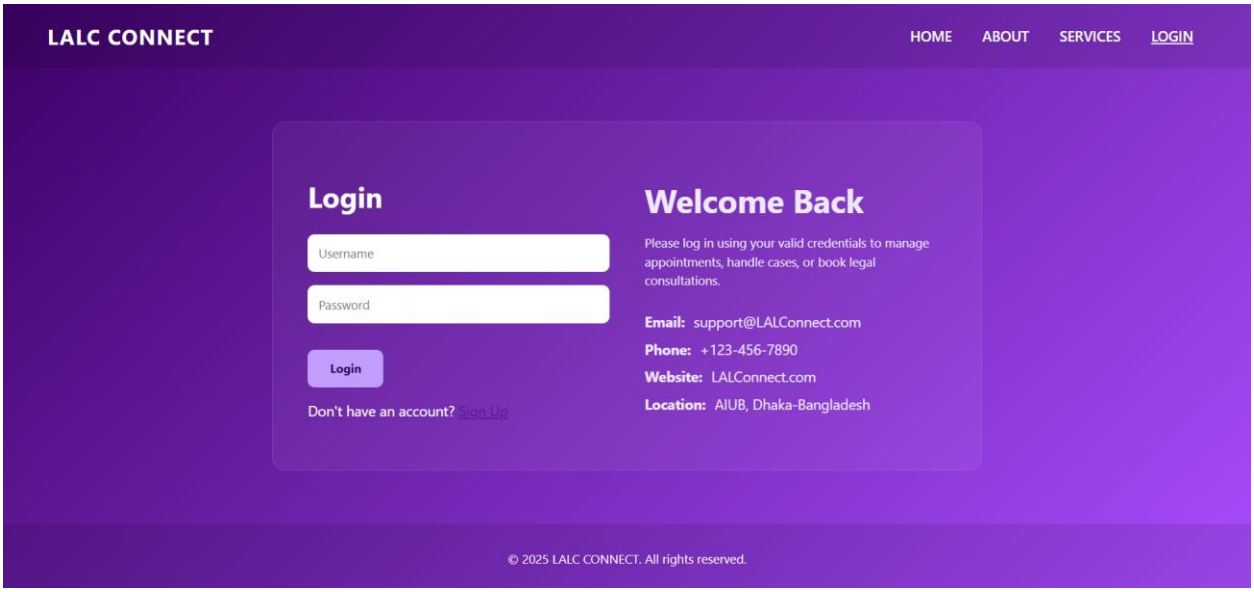
Lawyer Appointment Management System is designed to help people in Bangladesh easily connect with qualified lawyers. It acts as a central system that keeps everything well-organized for both clients and lawyers. The platform helps users find legal support based on specific needs—such as family matters, 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

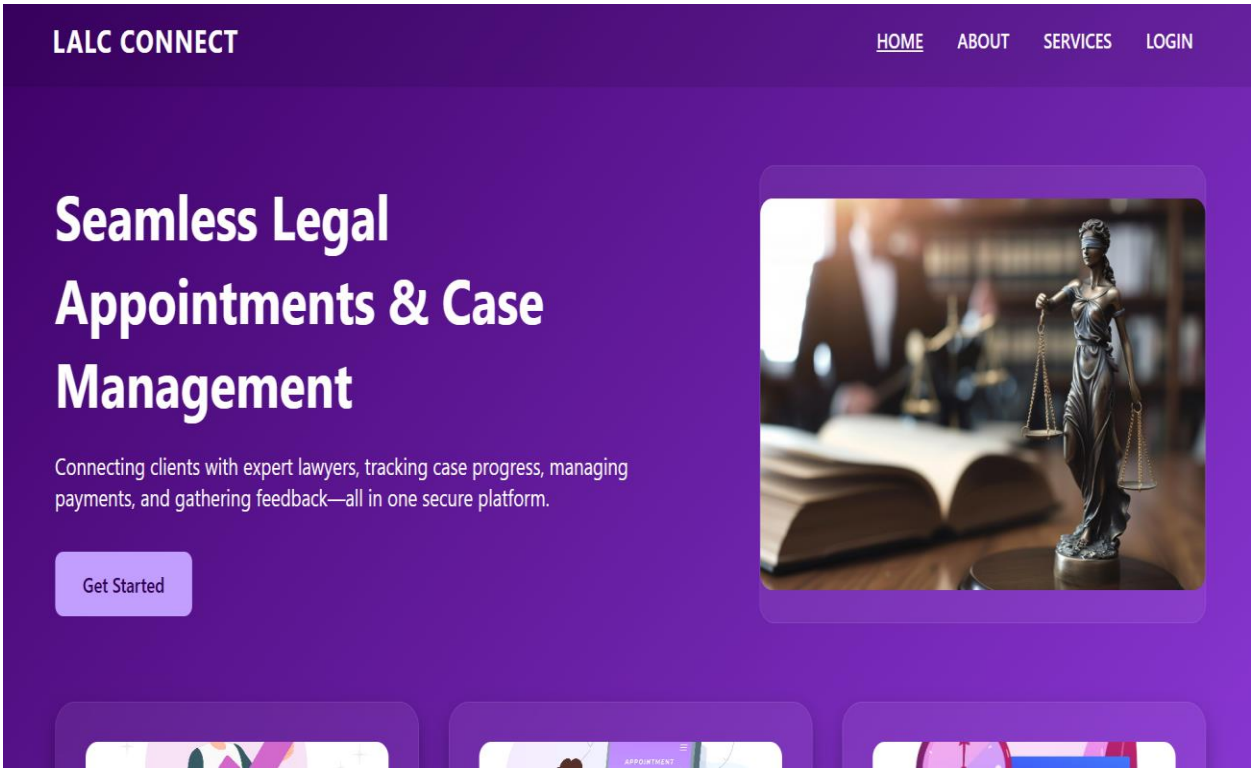
1. Registration Page:

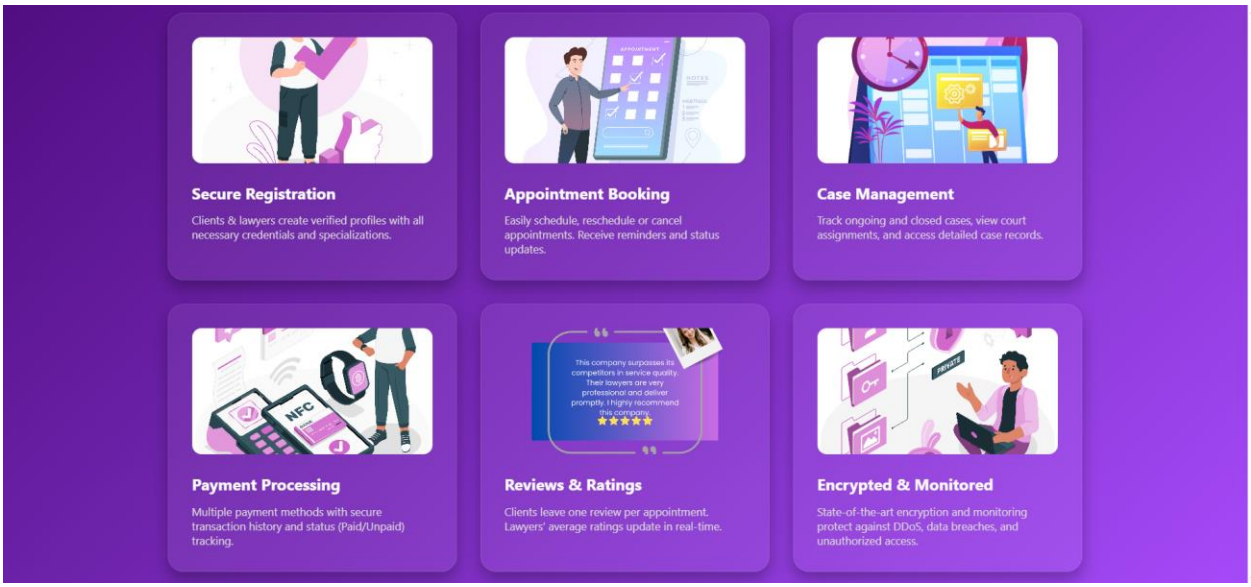


2. Login Page:

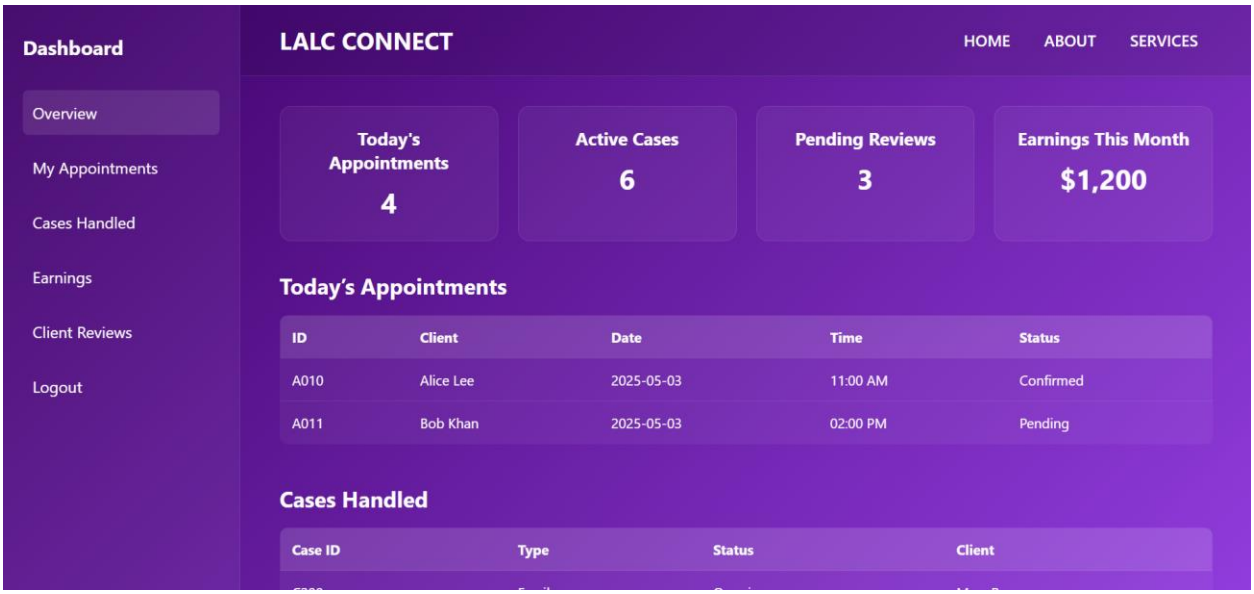

LALC CONNECT
HOME
ABOUT
SERVICES
LOGIN
Login
Username
Password
Login
Don't have an account? Sign Up
Welcome Back
Please log in using your valid credentials to manage appointments, handle cases, or book legal consultations.
Email: support@LALConnect.com
Phone: +123-456-7890
Website: LALConnect.com
Location: AIUB, Dhaka-Bangladesh
© 2025 LALC CONNECT. All rights reserved.

3. Homepage:

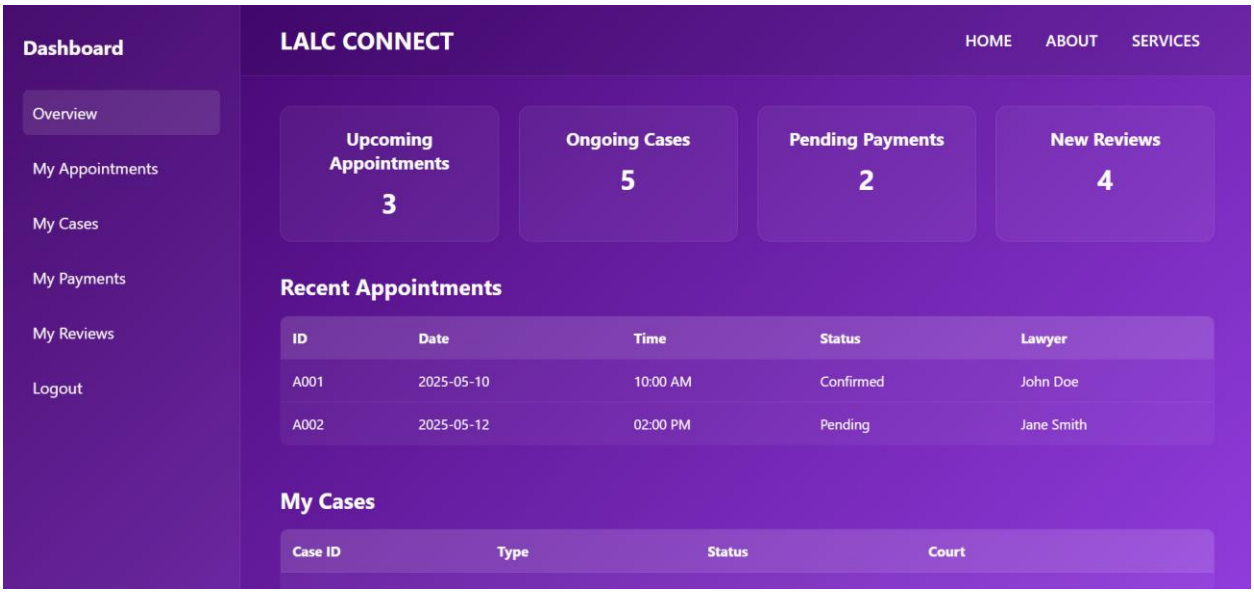

LALC CONNECT
HOME
ABOUT
SERVICES
LOGIN
Seamless Legal Appointments & Case Management
Connecting clients with expert lawyers, tracking case progress, managing payments, and gathering feedback—all in one secure platform.
Get Started



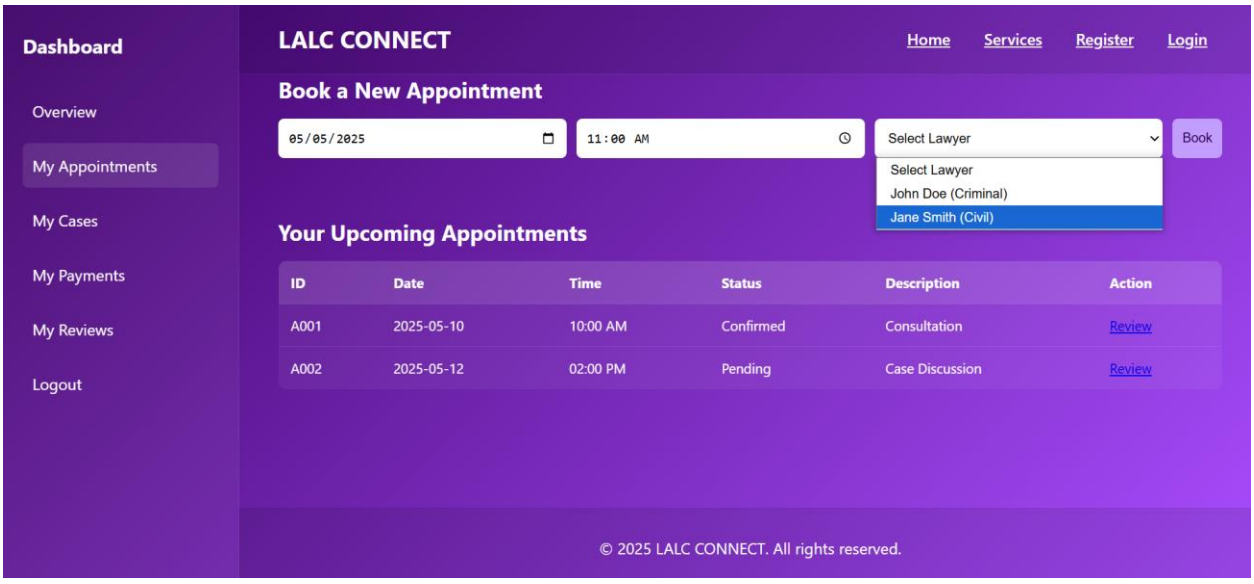
4. Lawyer's Dashboard:



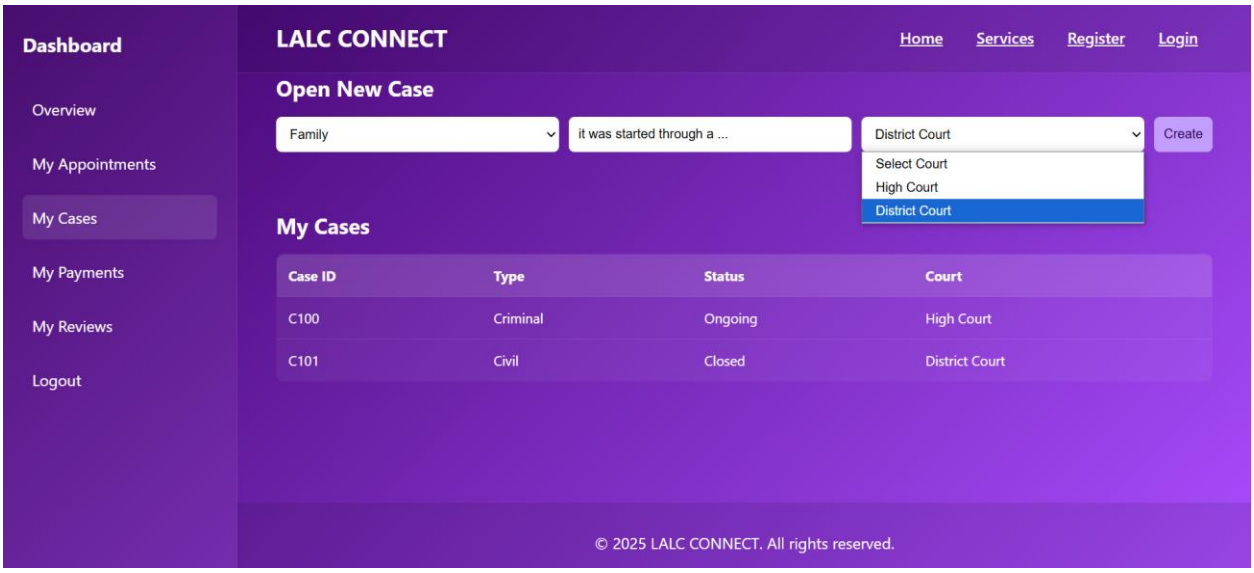
5. Client's Dashboard



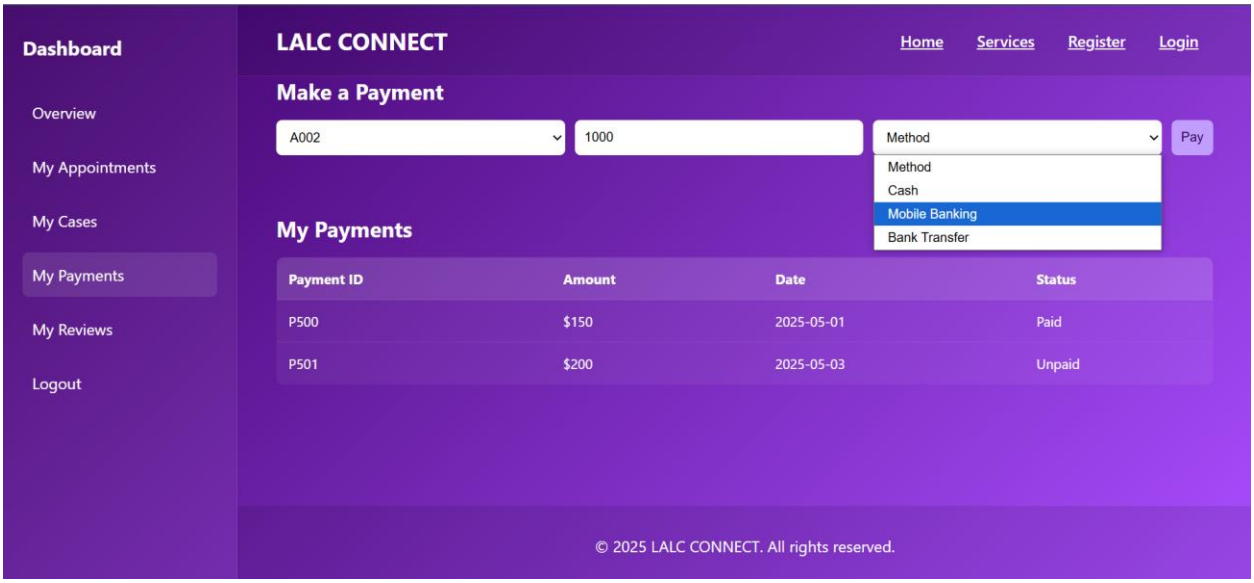
6. Client Appointments: Clients can book appointments.



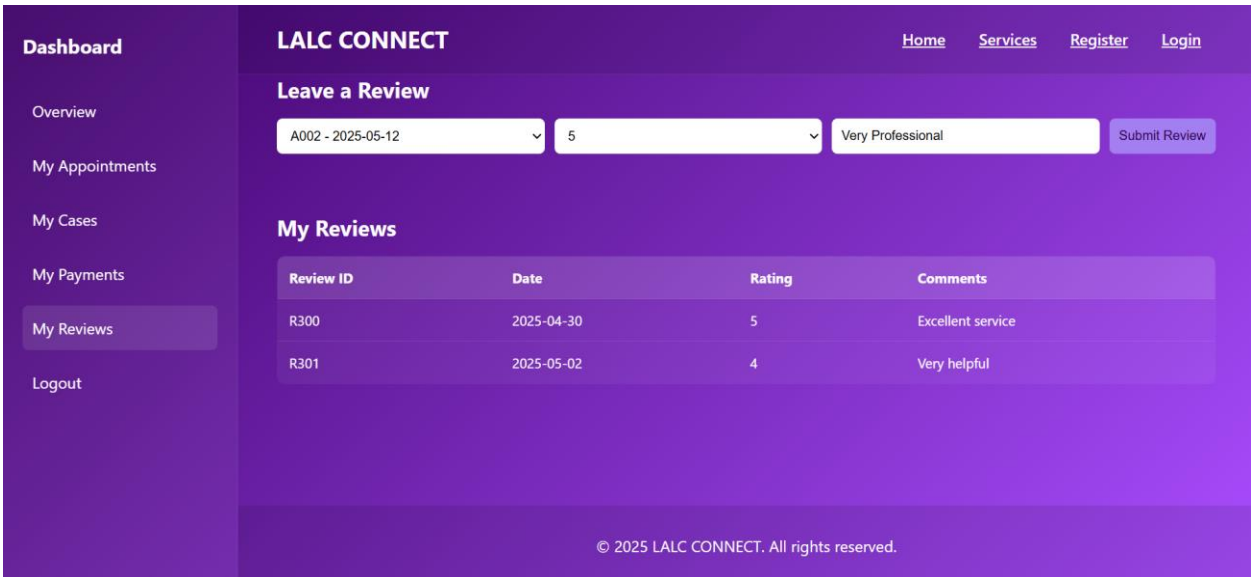
7. Client Cases: Client can open new cases.



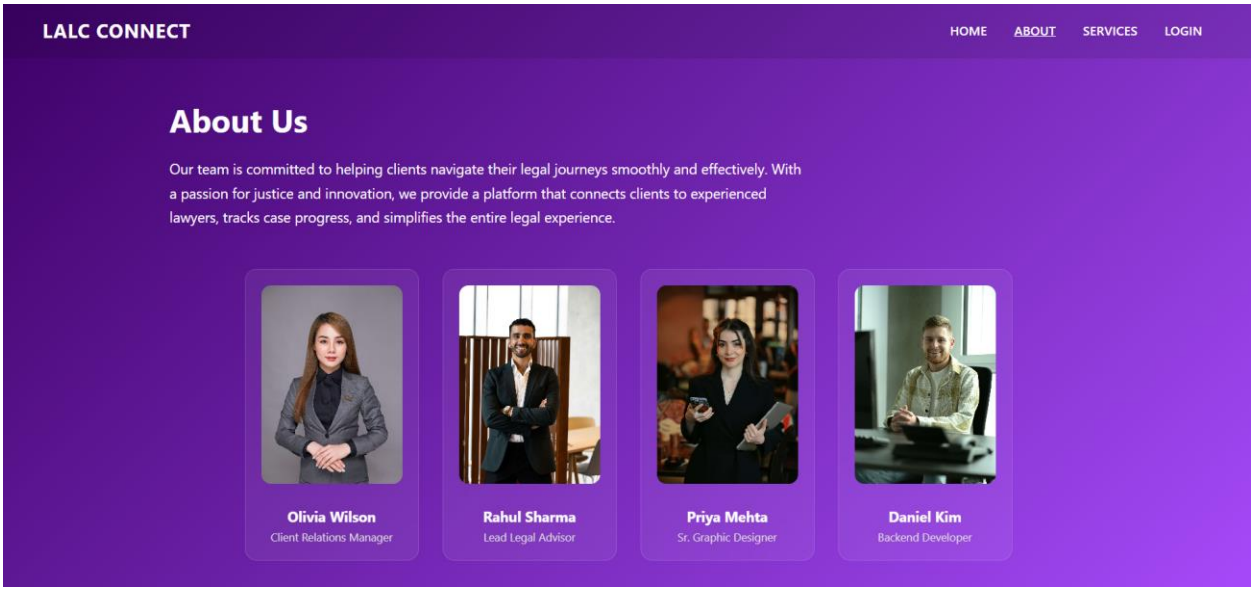
8. Client Payment: Client can pay their appointment/cases payment through their preferred payment method.



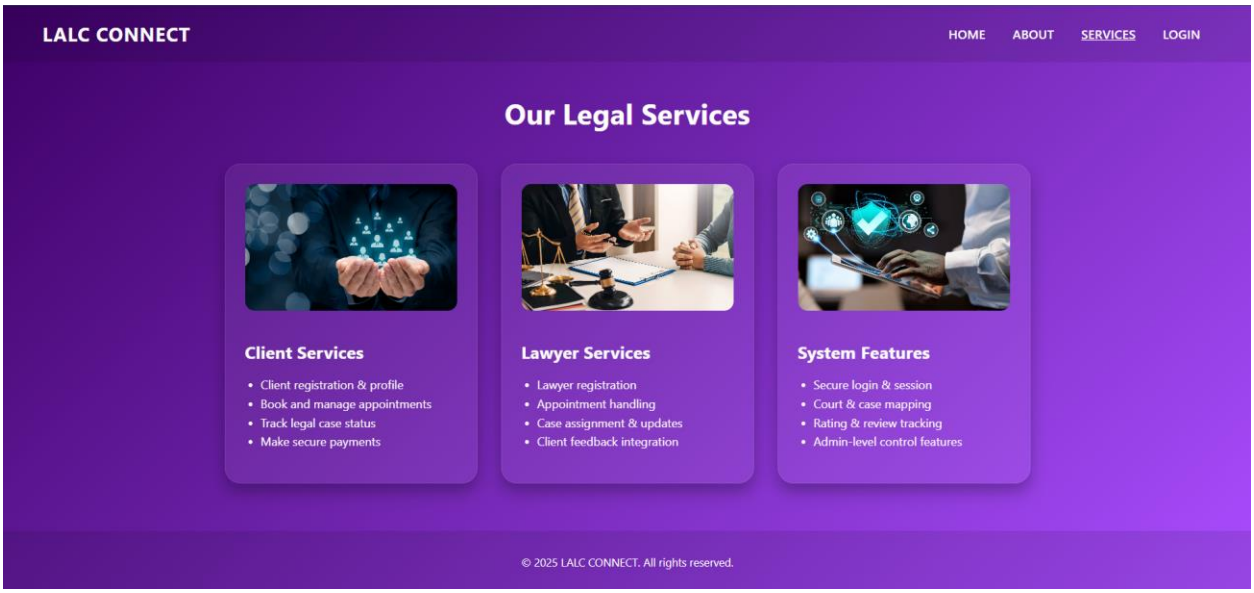
9. Reviews: Client can rate/feedback their specific cases and selected lawyers.



10. About-us Page:



11. Our Sevices Page:

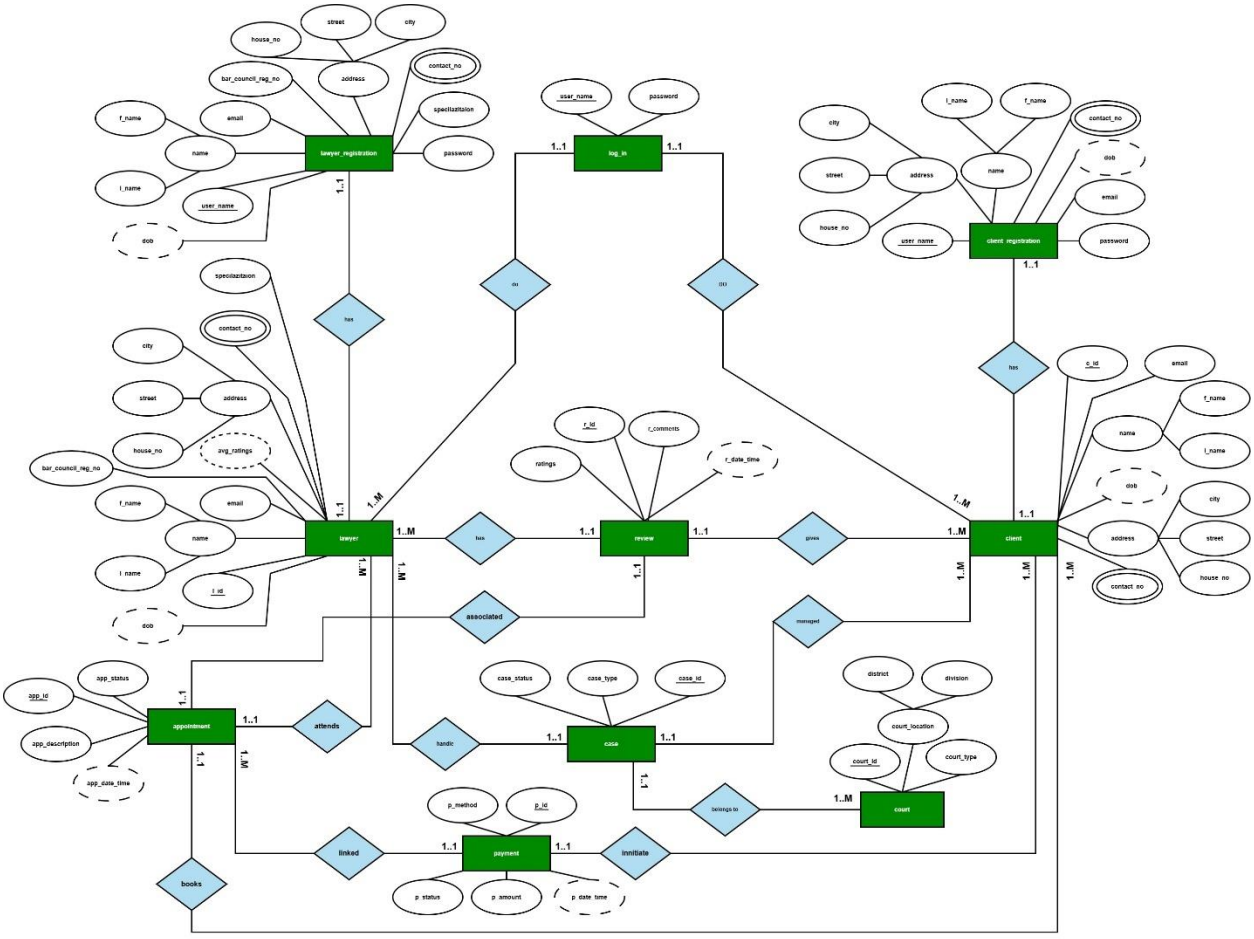


Scenario Description

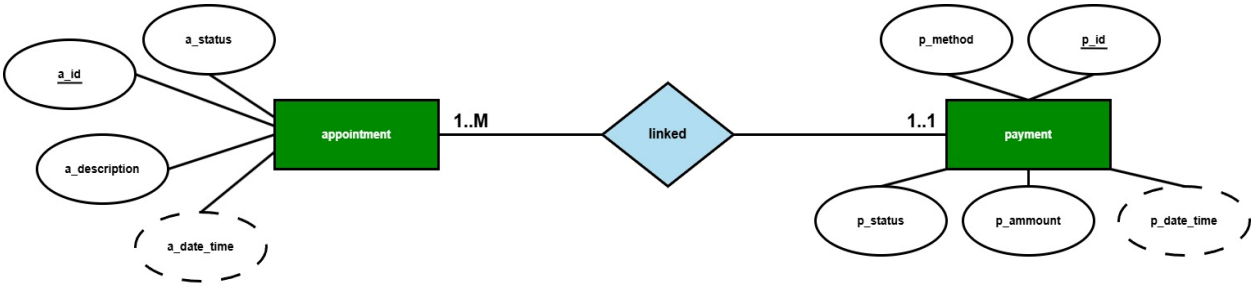
In a lawyer appointment management system, a client can register by providing a unique username, first name, last name, date of birth, contact number, email, and password. Each address consists of a city name, street number, and house number. A lawyer can also register using the same credentials, with 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_ammount, 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_ammount, p_date_time

2NF:

1. app_id, app_status, app_description, app_date_time
2. p_id, p_method, p_status, p_ammount, p_date_time

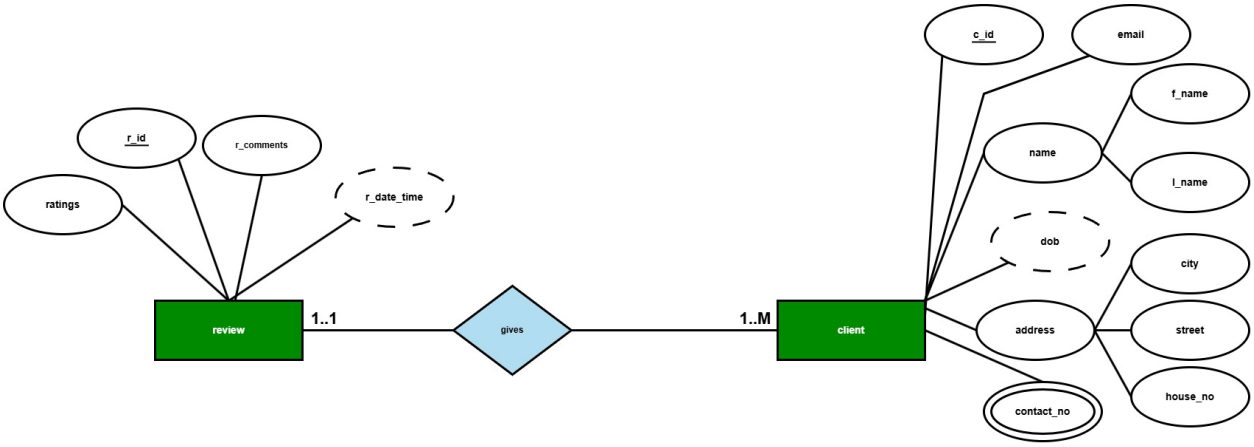
3NF

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_ammount, 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

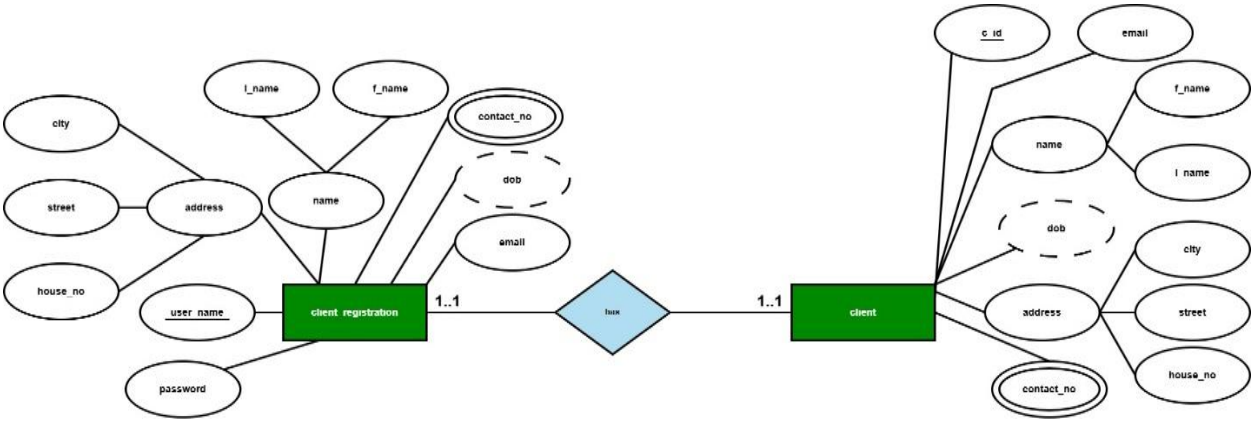
3NF

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

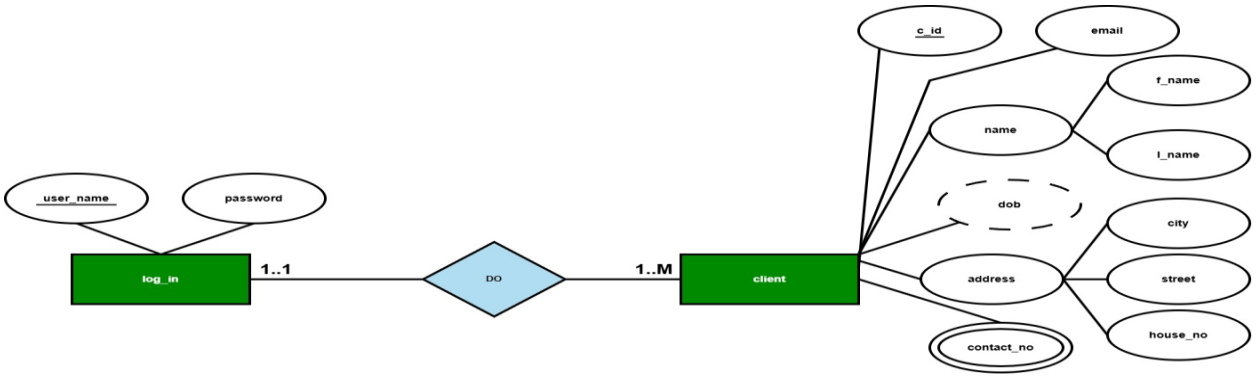
3NF

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

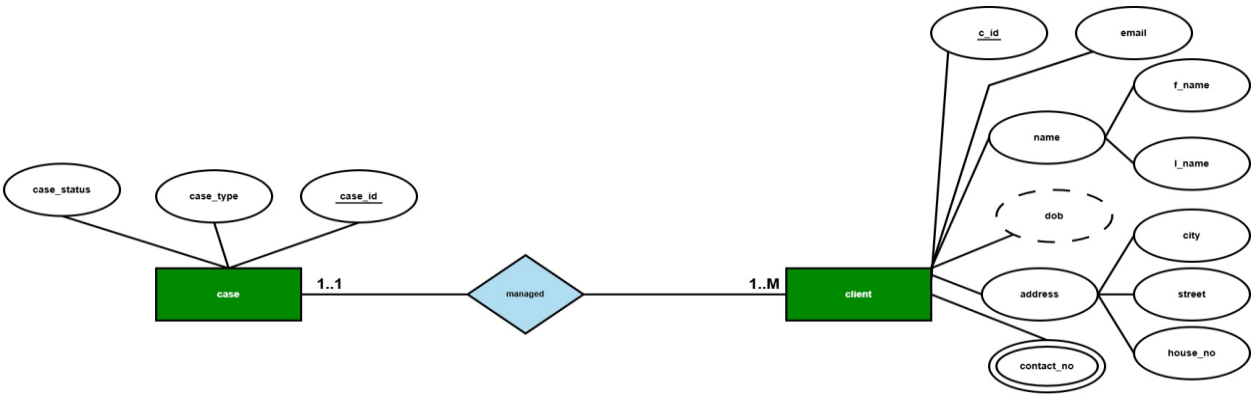
3NF

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

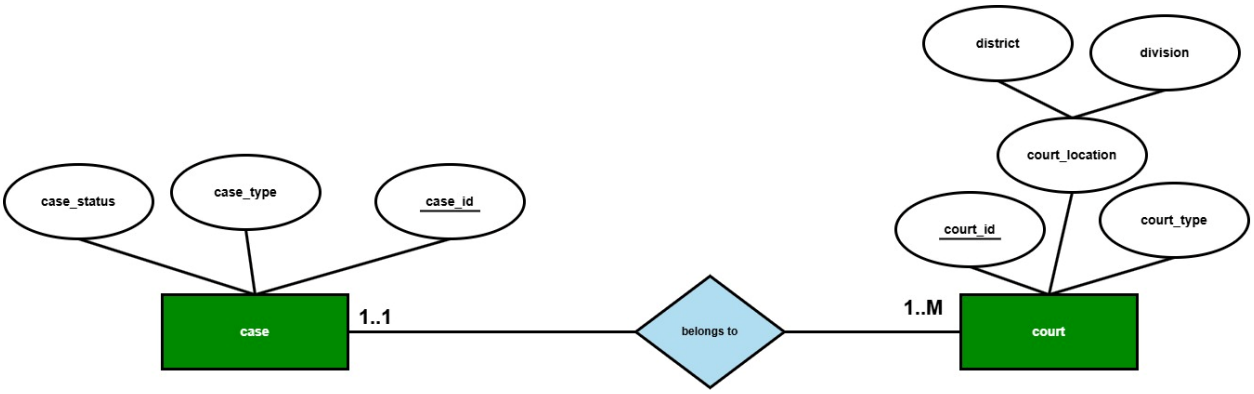
3NF

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

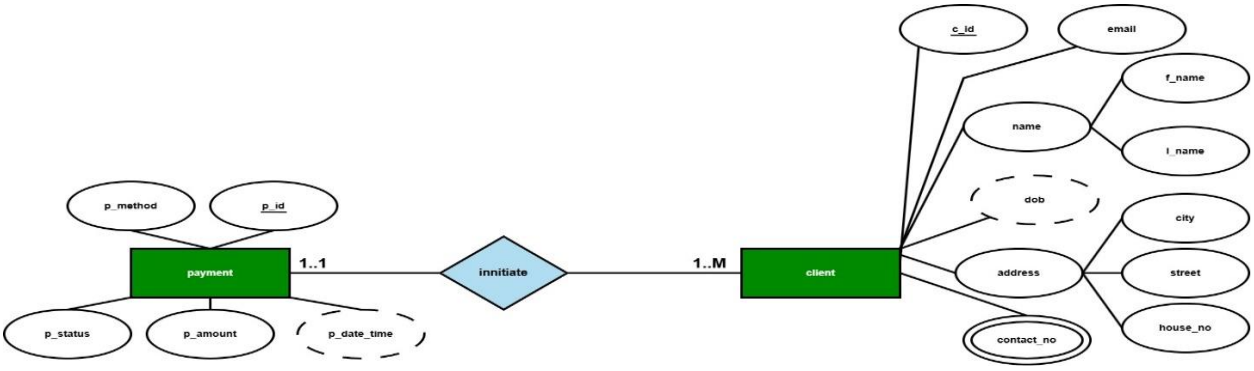
3NF

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

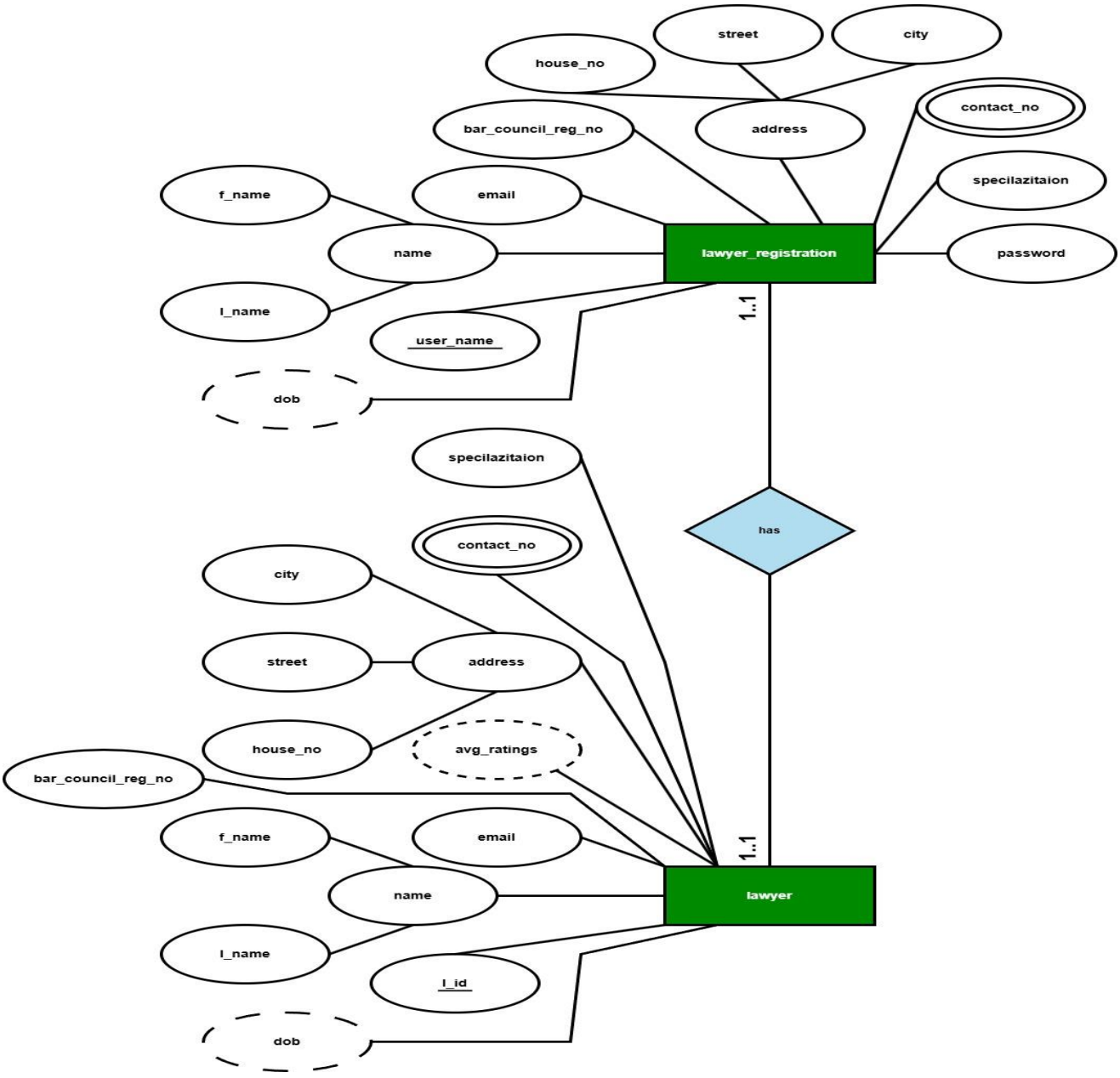
3NF

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

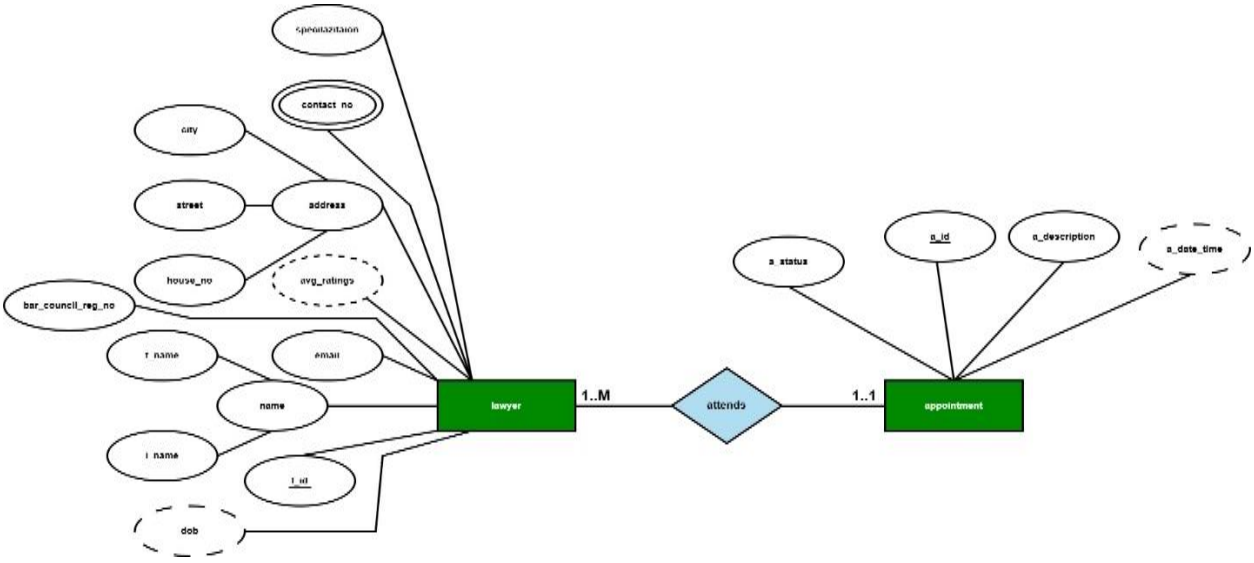
3NF

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

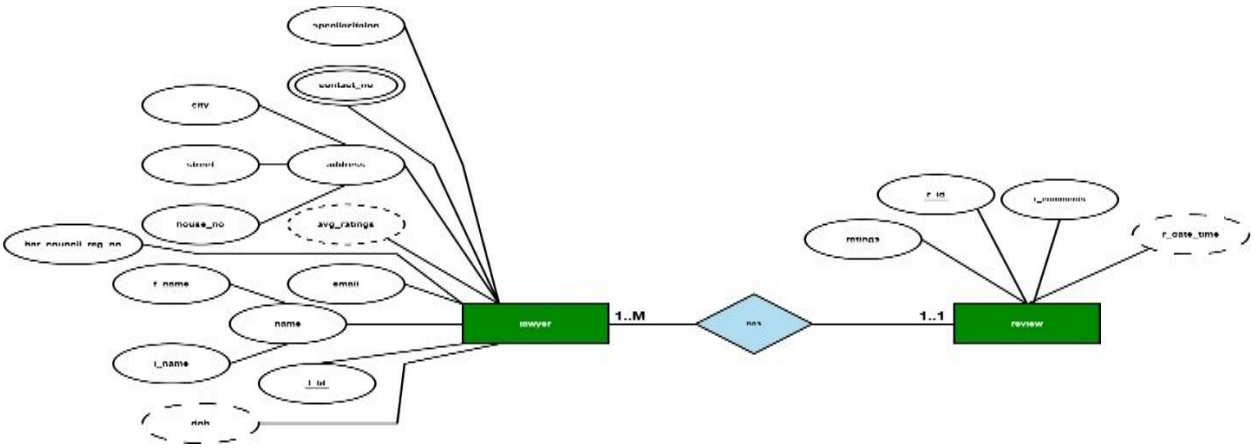
Name and Address are Transitive attribute

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

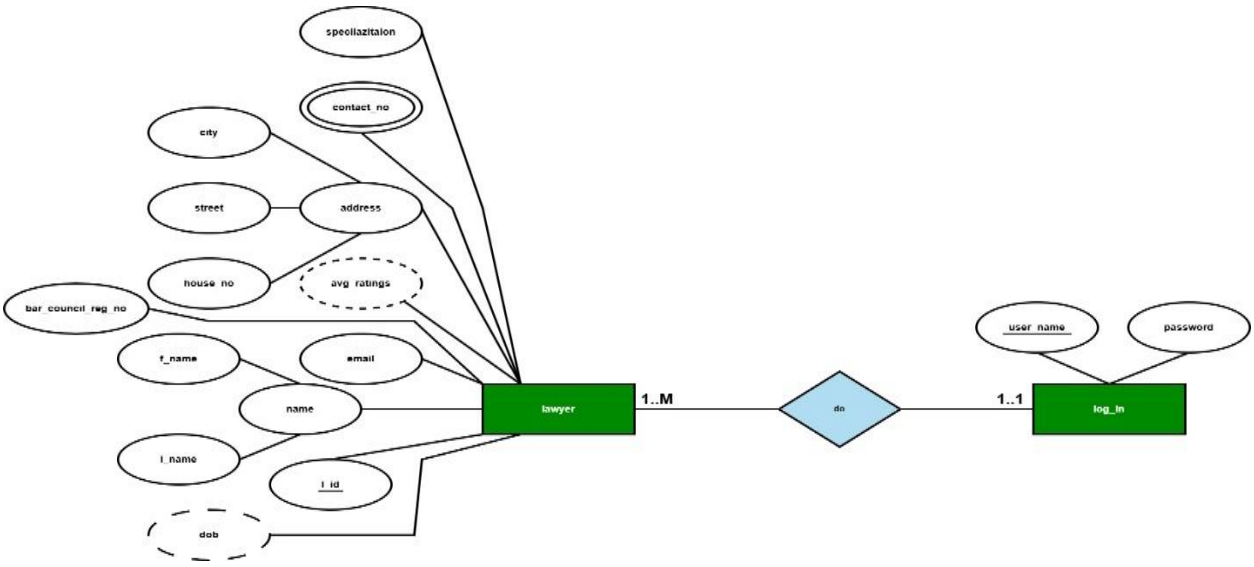
3NF

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

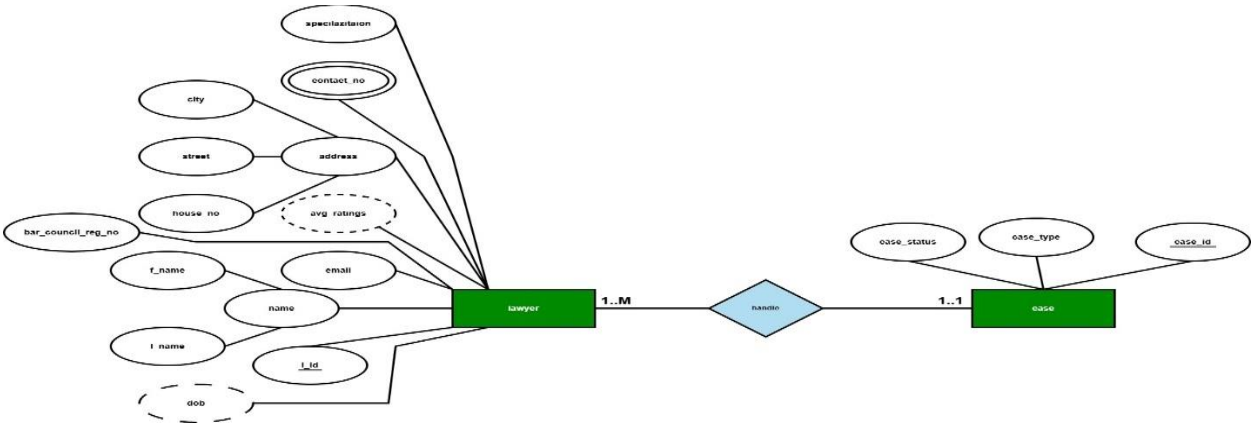
3NF

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

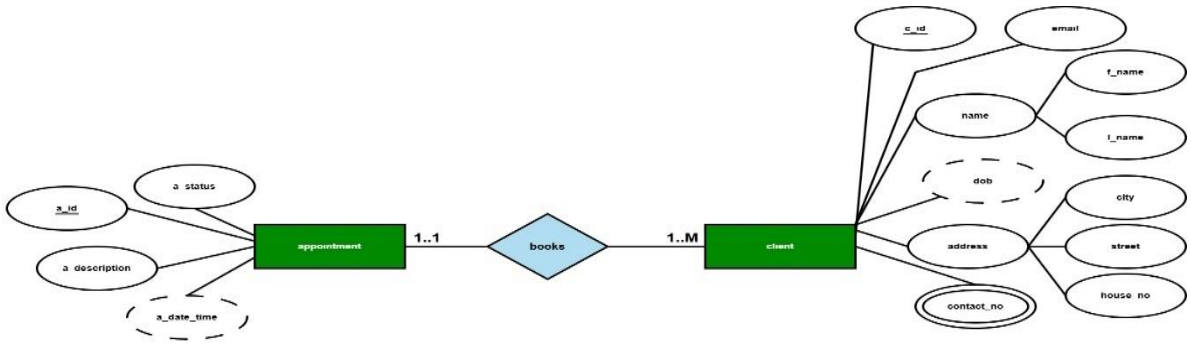
3NF

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

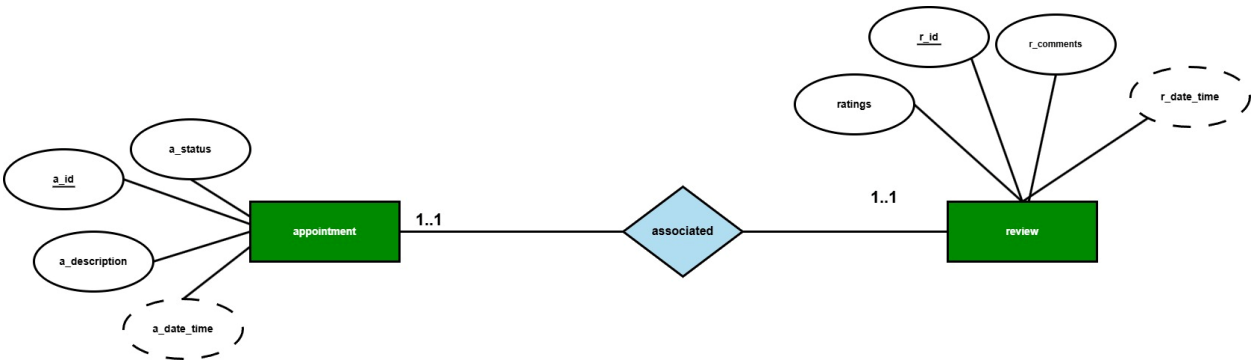
3NF

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;
```

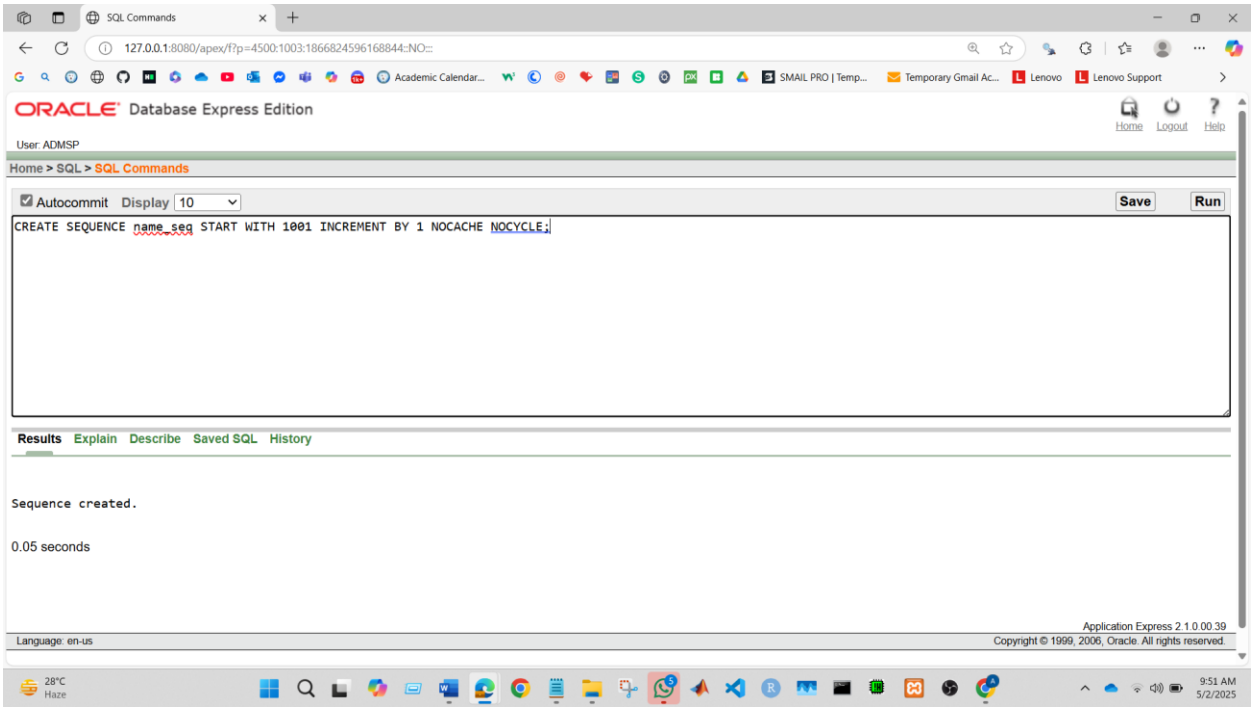


Table:

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

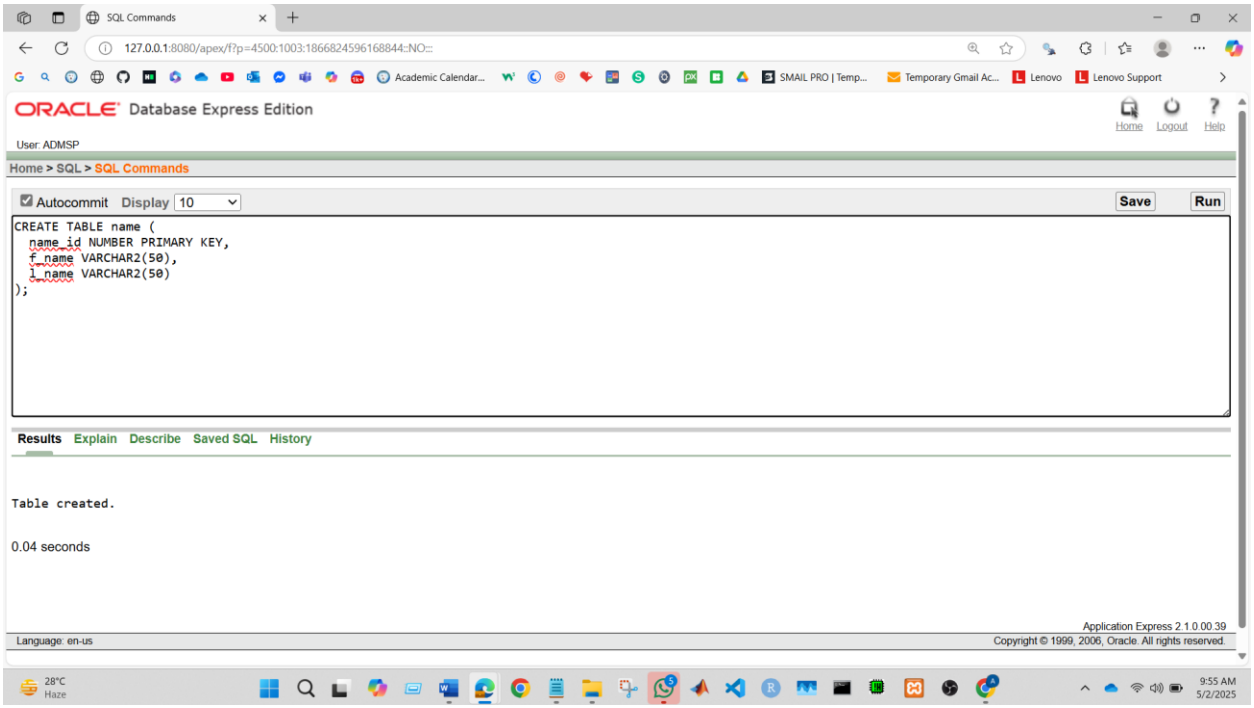


Table 2: address

Sequence:

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

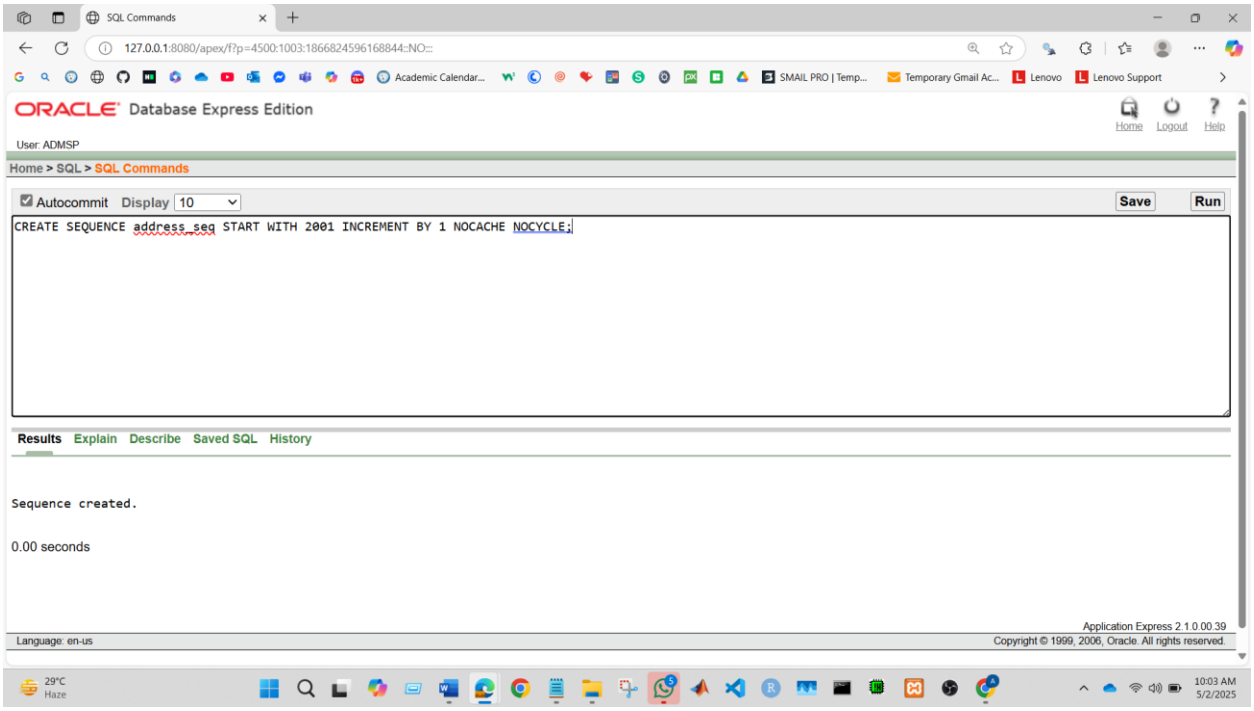


Table:

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

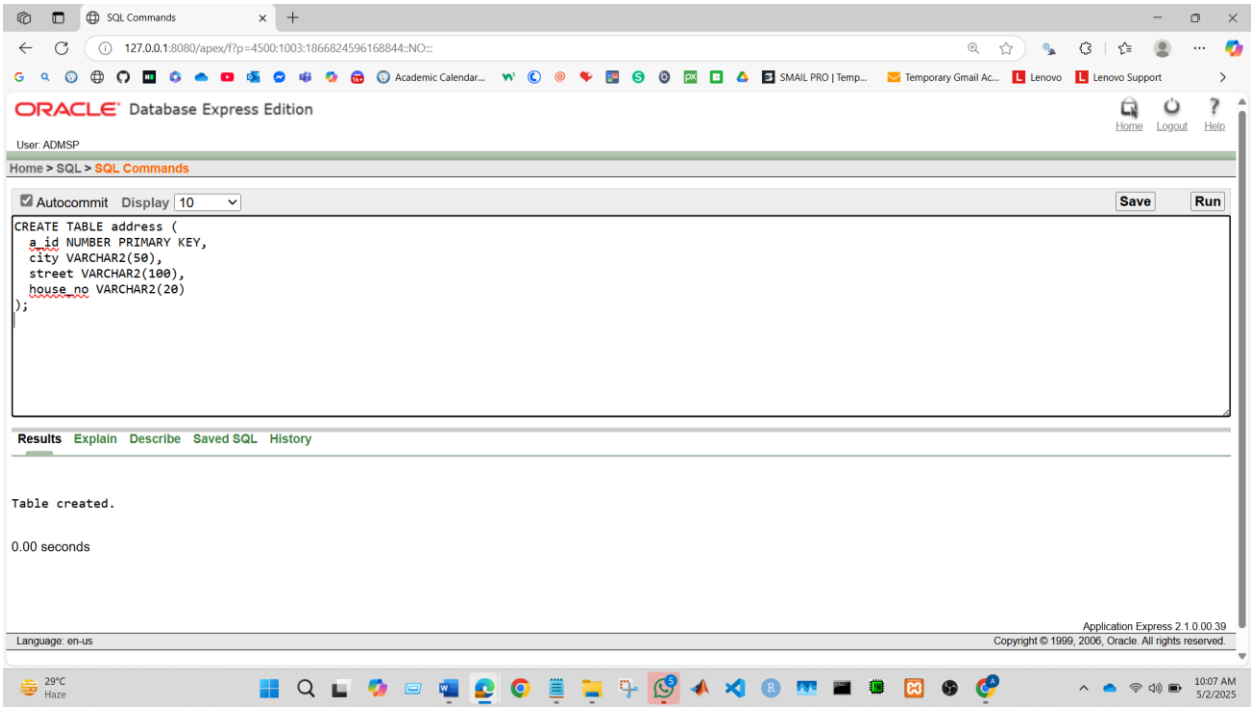


Table 3: user

Sequence:

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

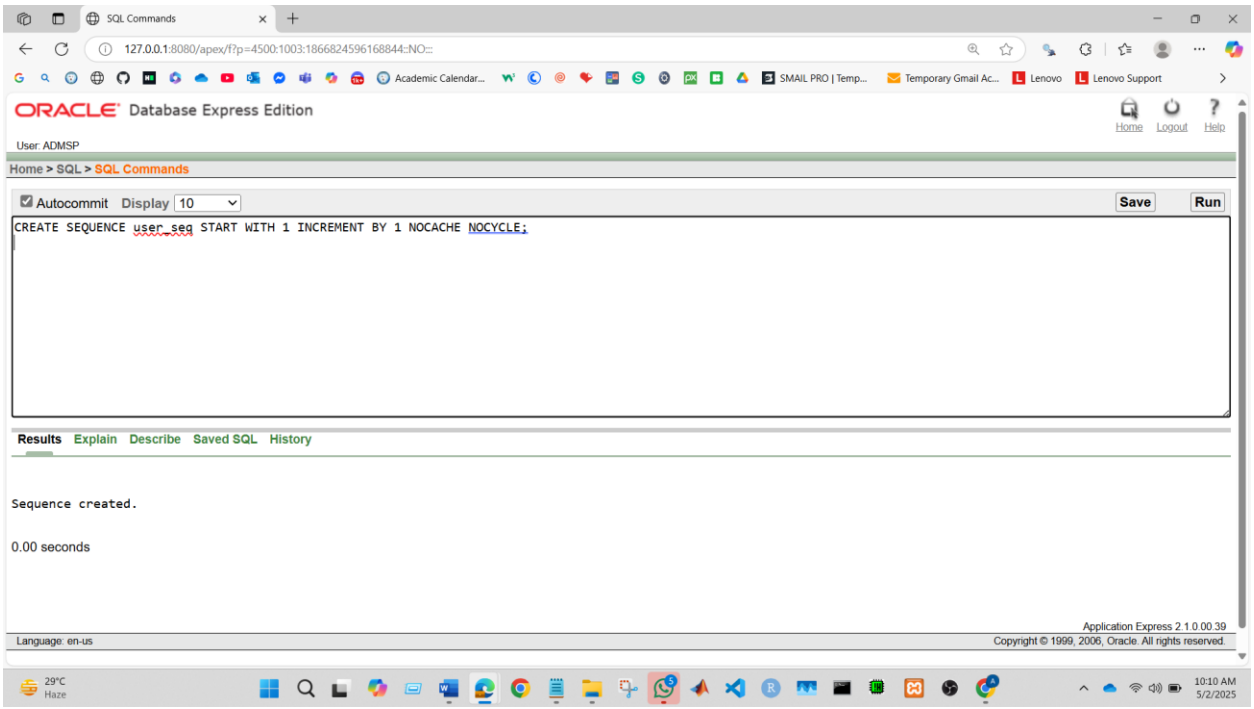


Table:

```
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)
);
```

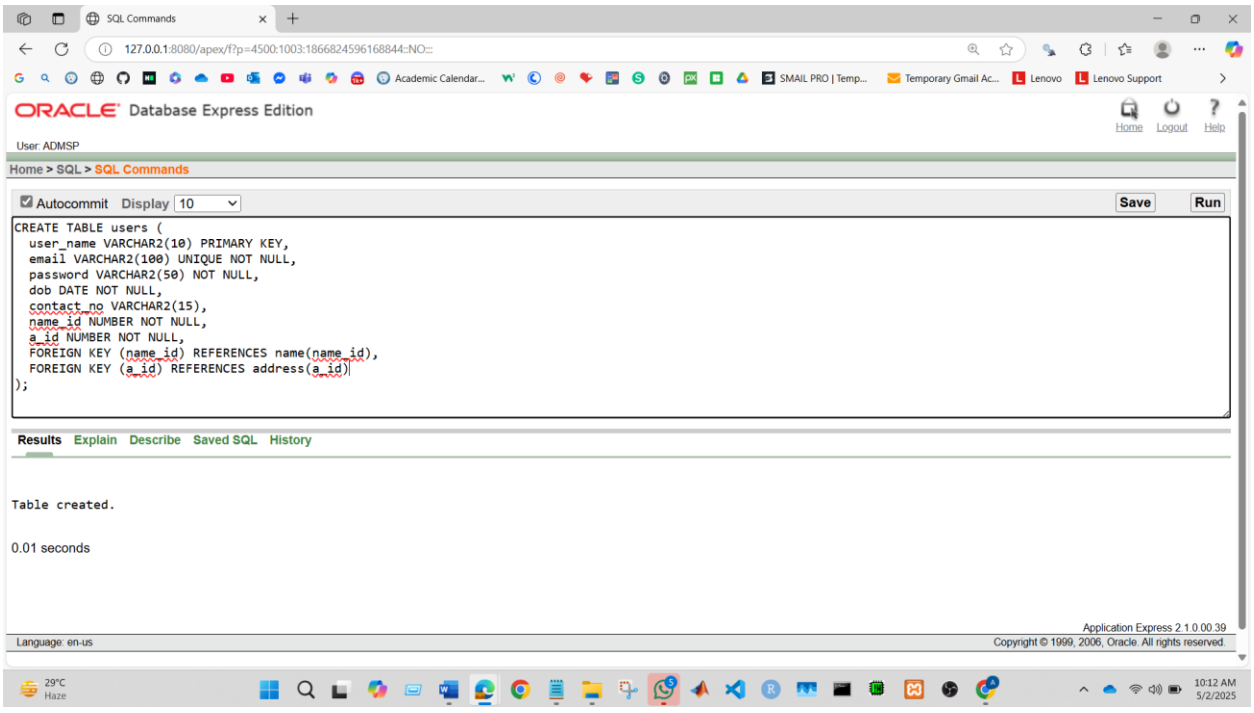


Table 4: client

Sequence:

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

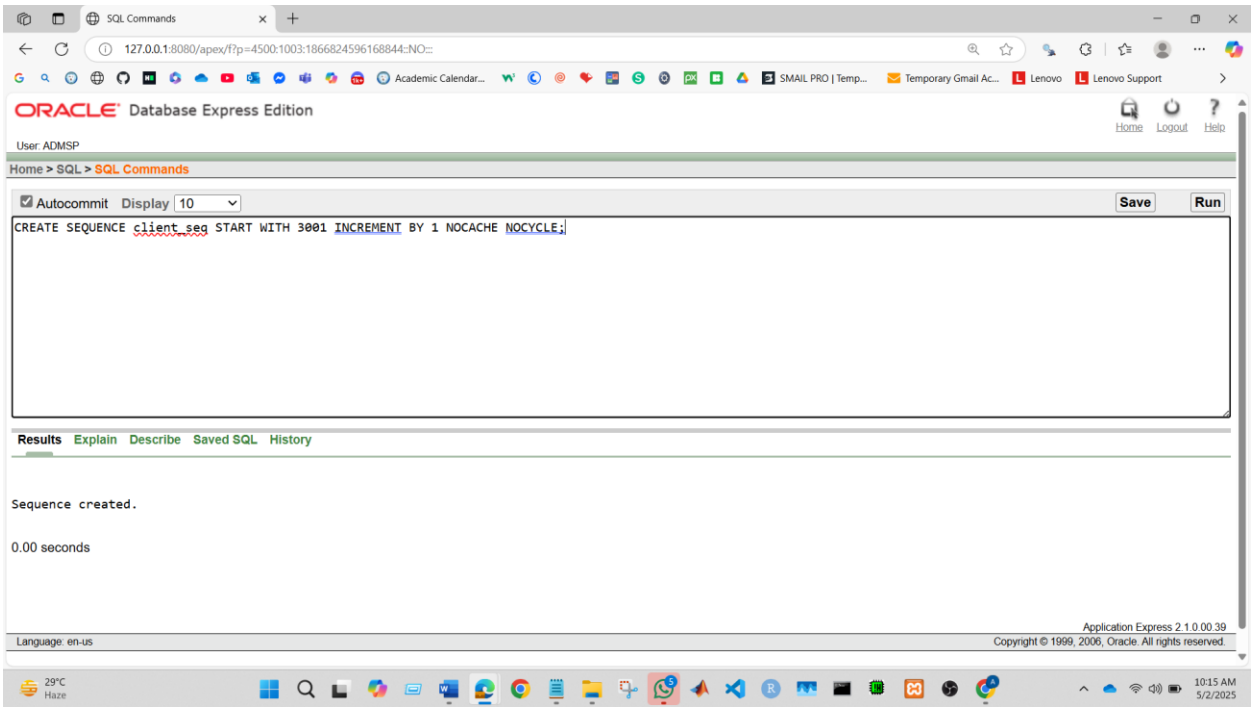


Table:

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

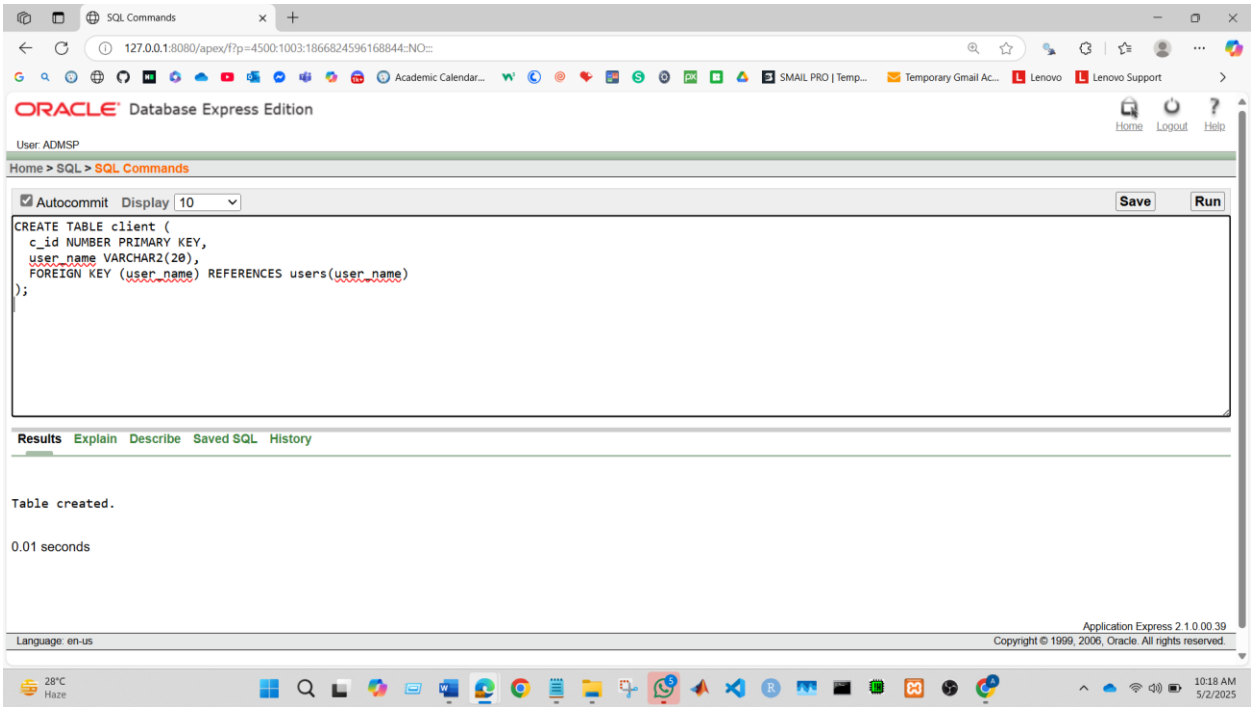


Table 5: lawyer

Sequence:

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

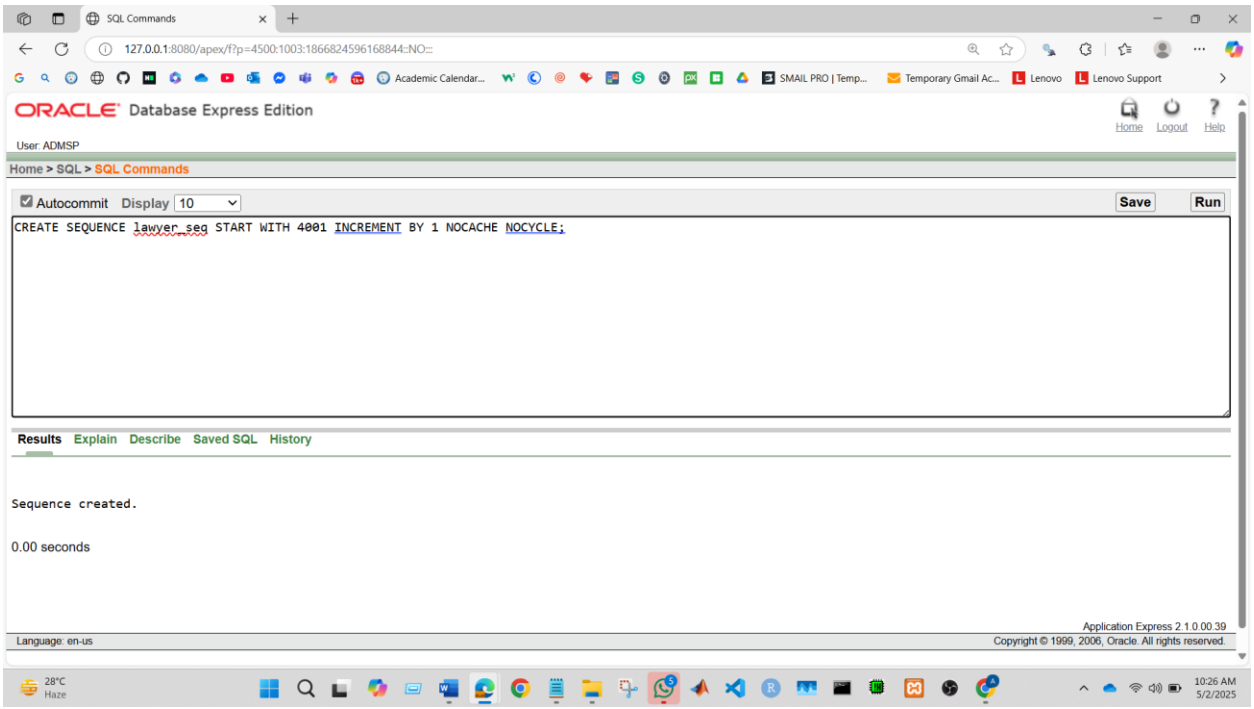


Table:

```
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)
);
```

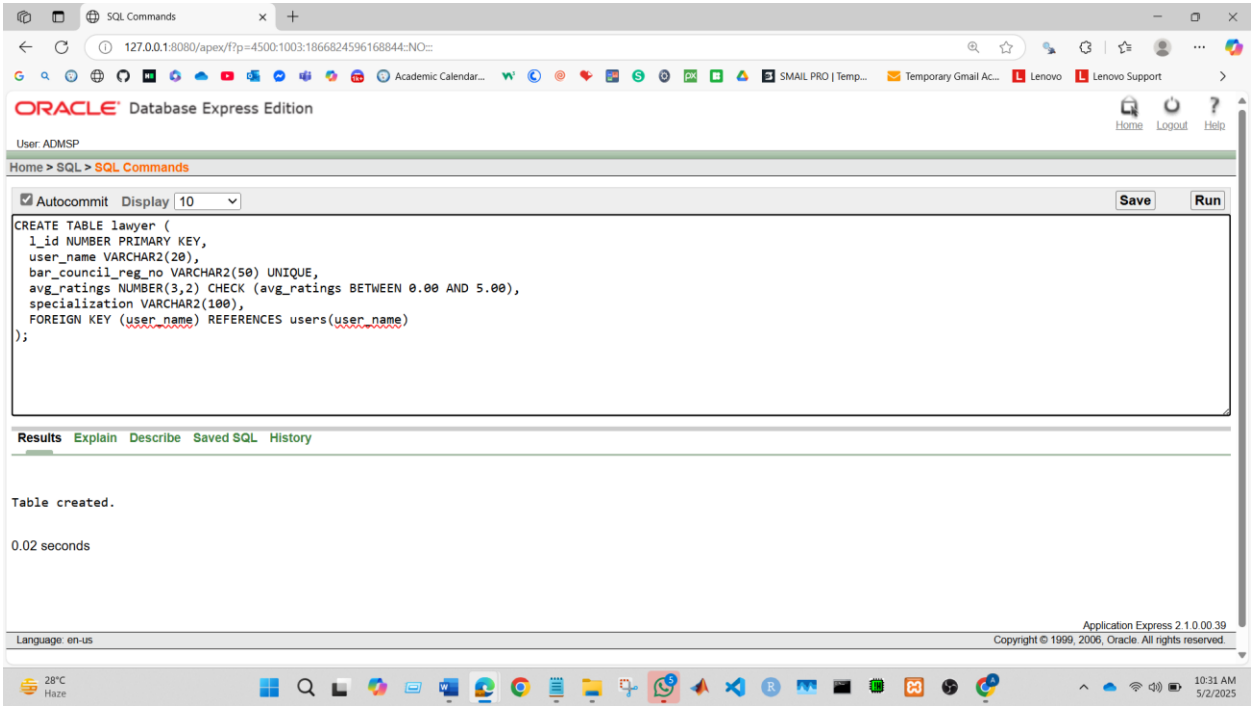


Table 6: location

Sequence:

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

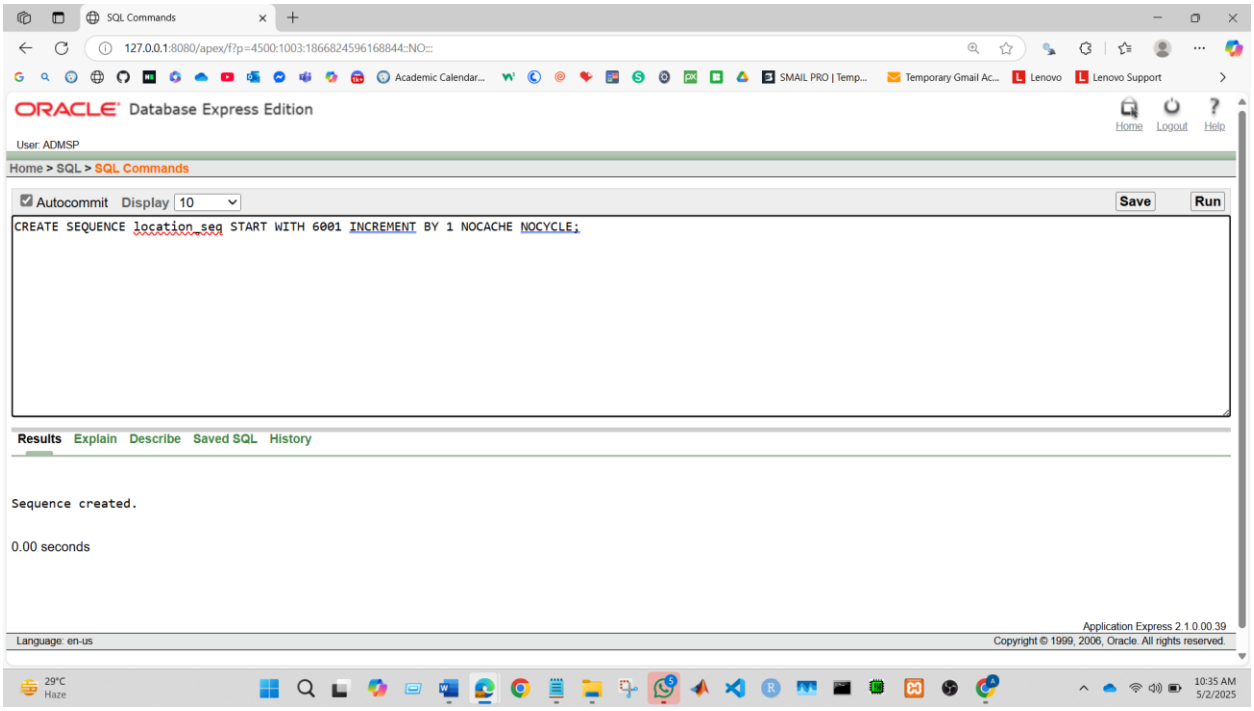


Table:

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

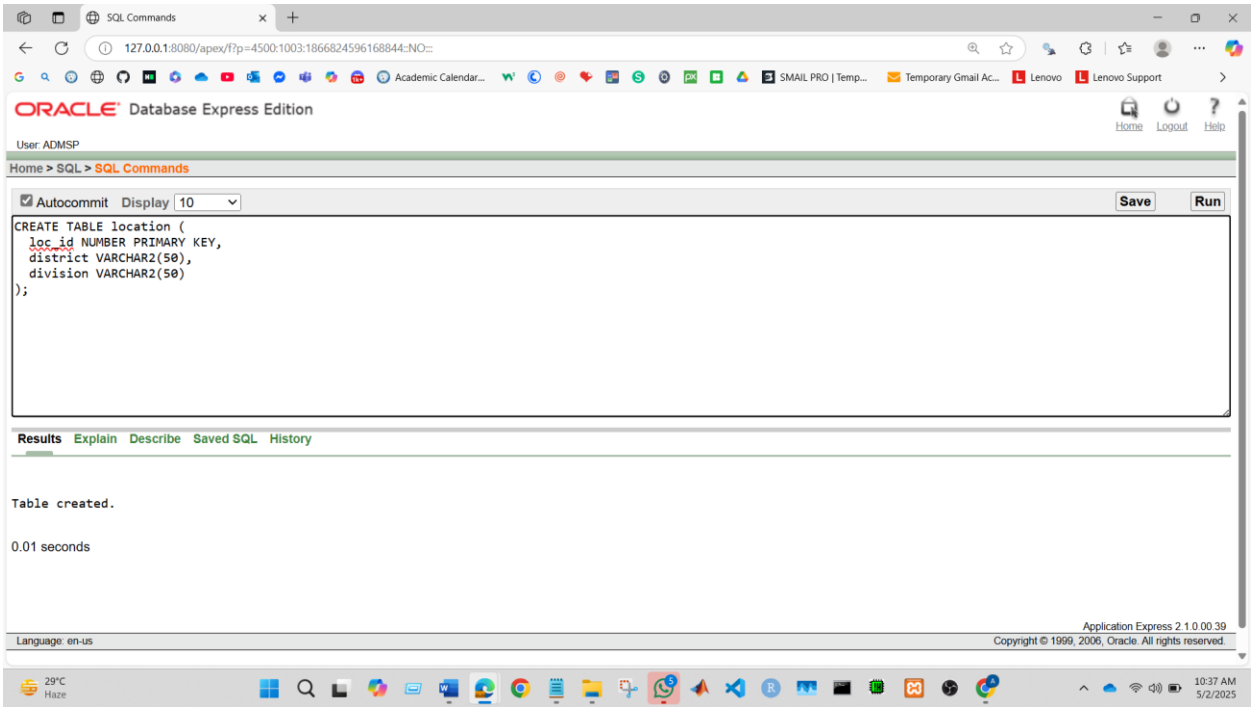


Table 7: court

Sequence:

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

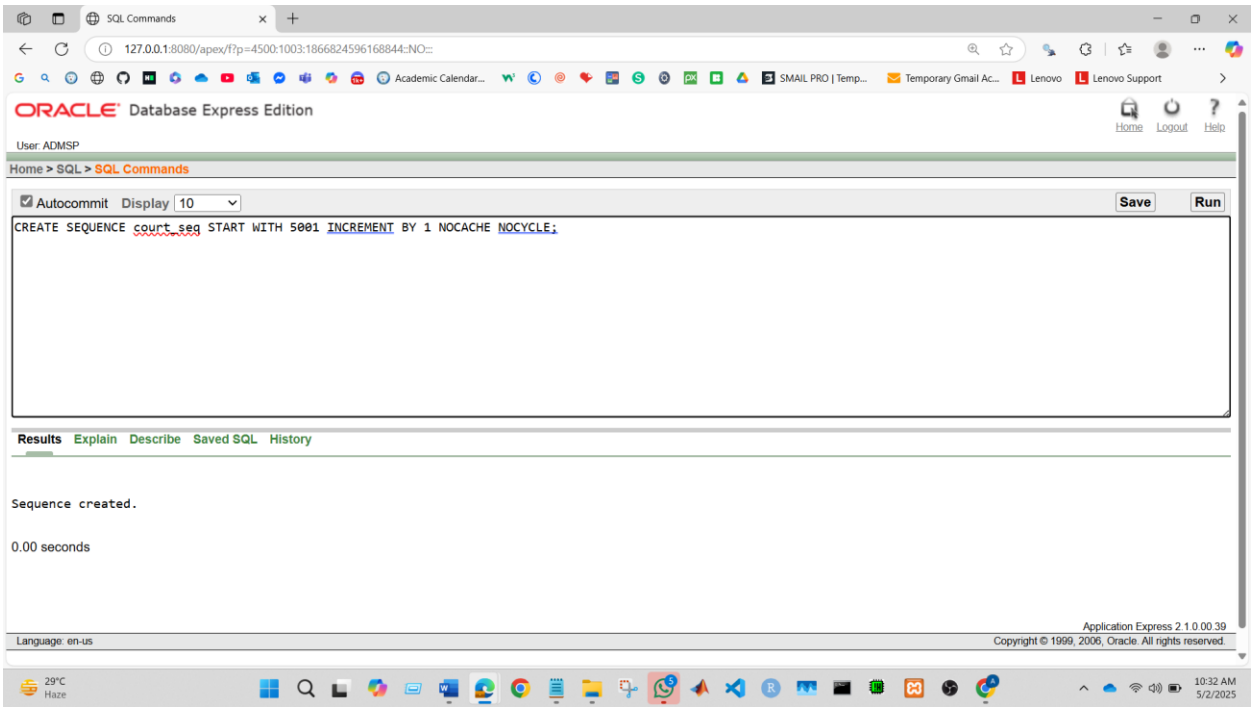


Table:

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

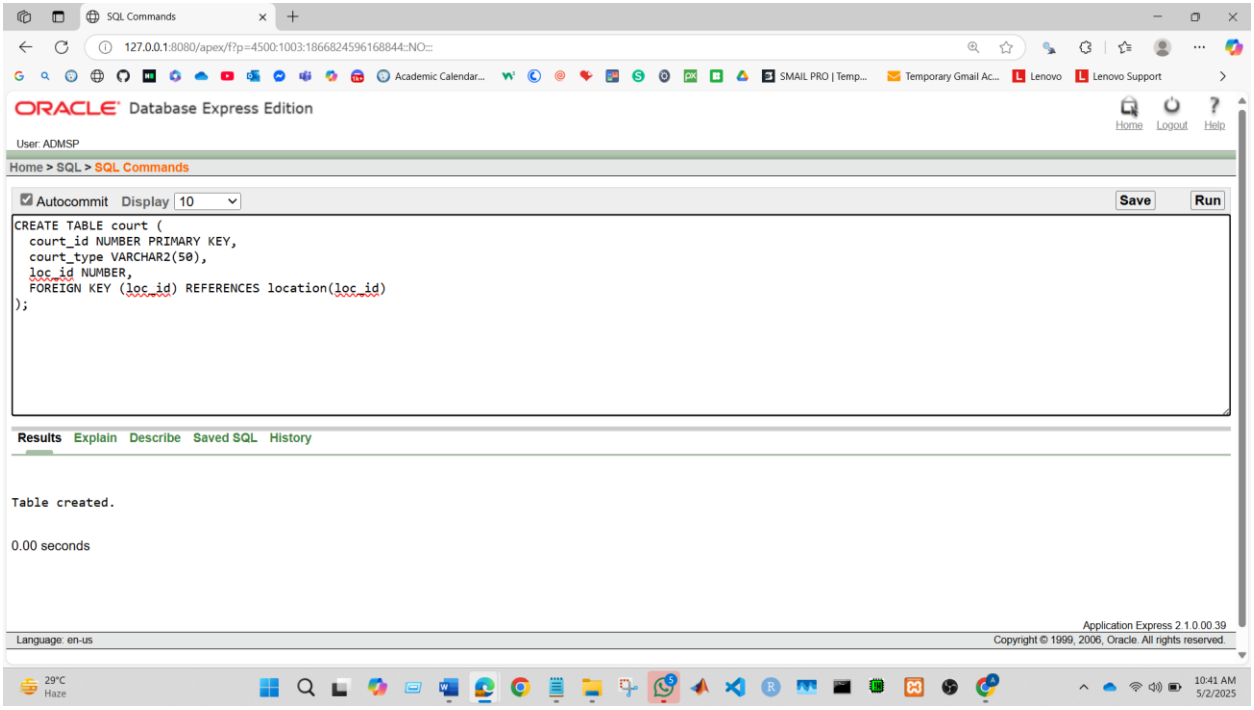


Table 8: case

Sequence:

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

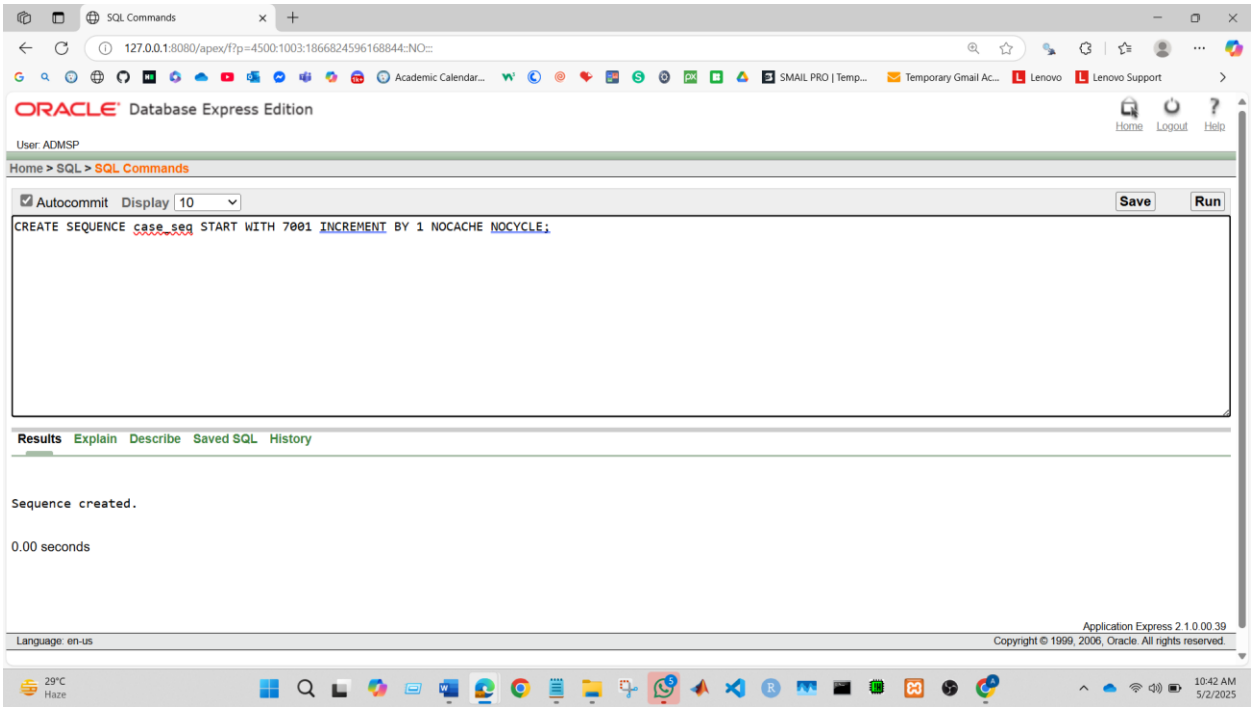


Table:

```
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)
);
```

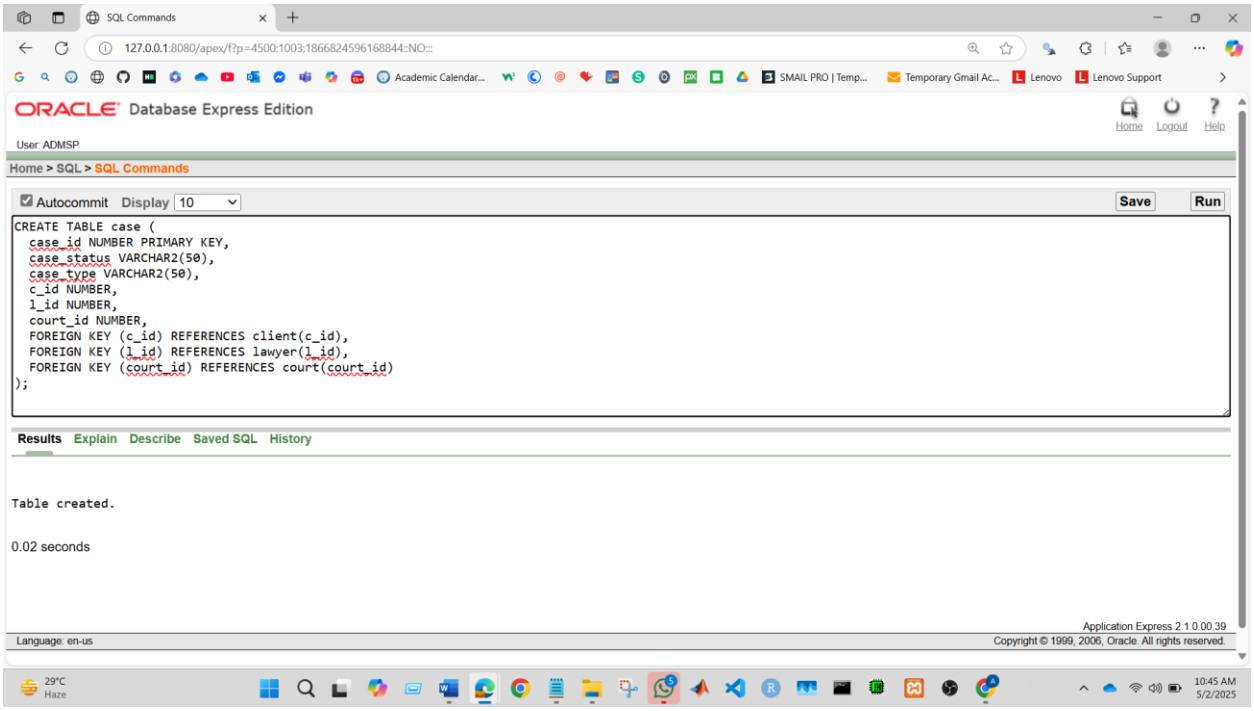


Table 9: appointment

Sequence:

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

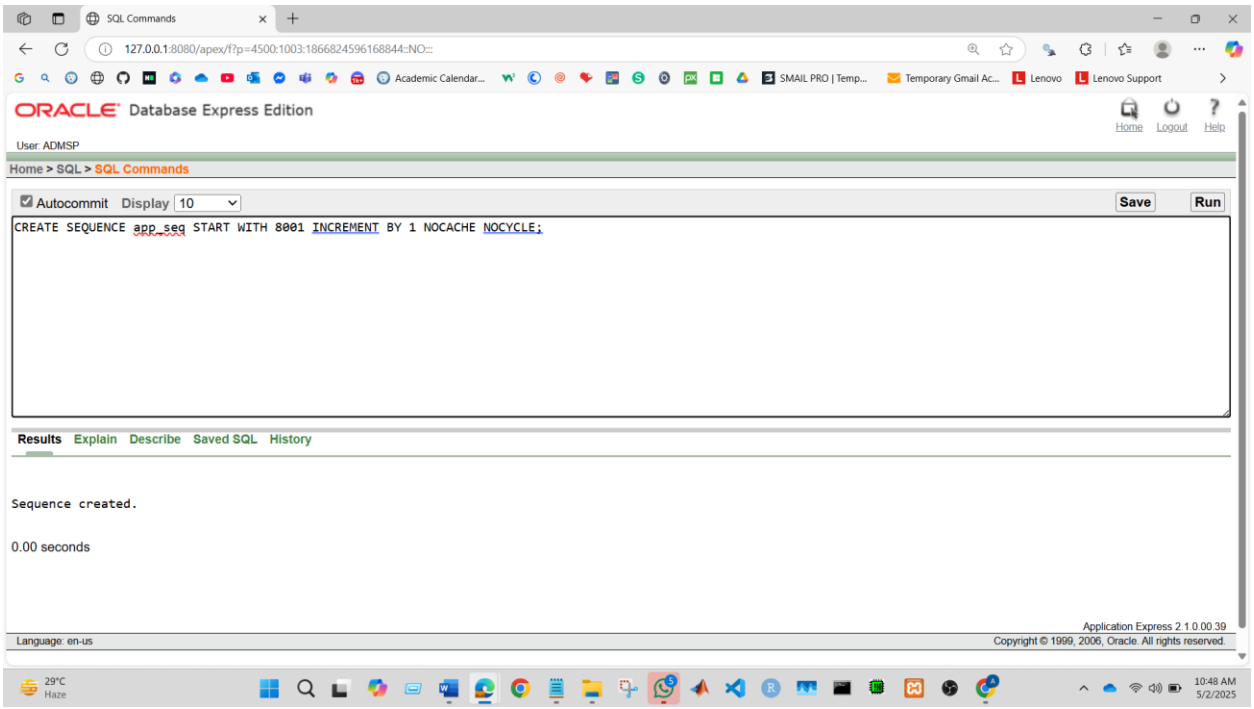


Table:

```
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)
);
```

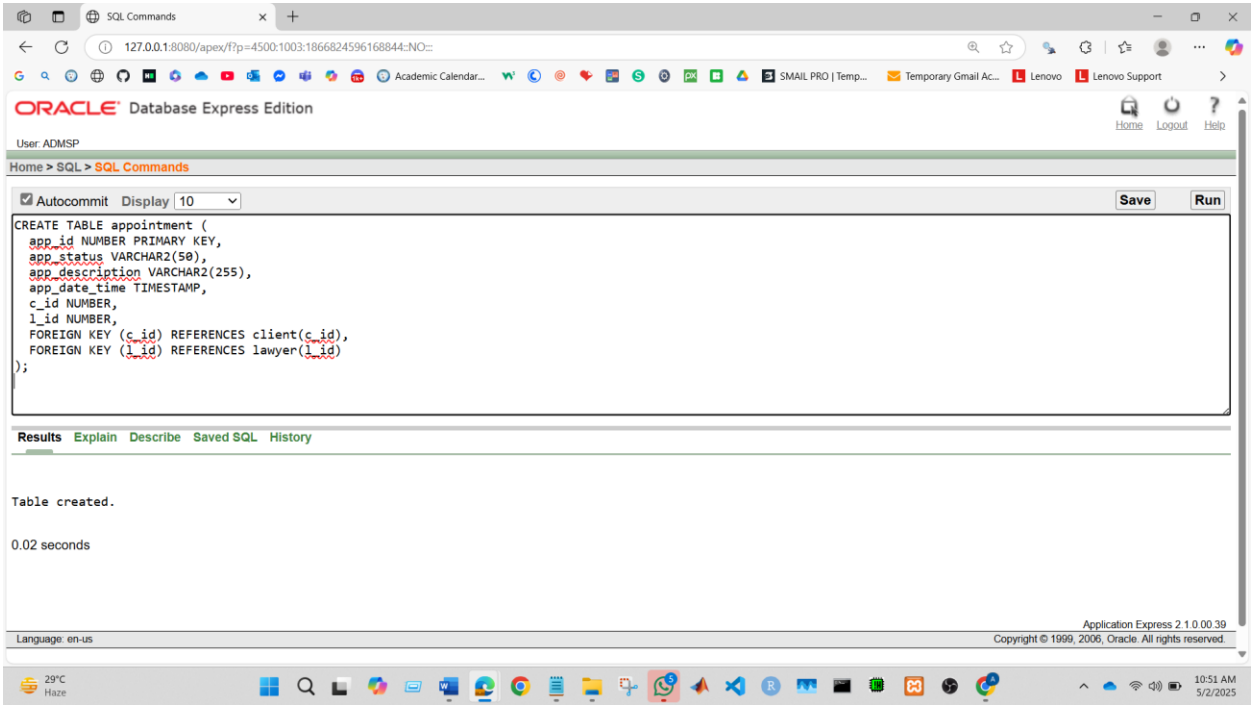


Table 10: payment

Sequence:

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

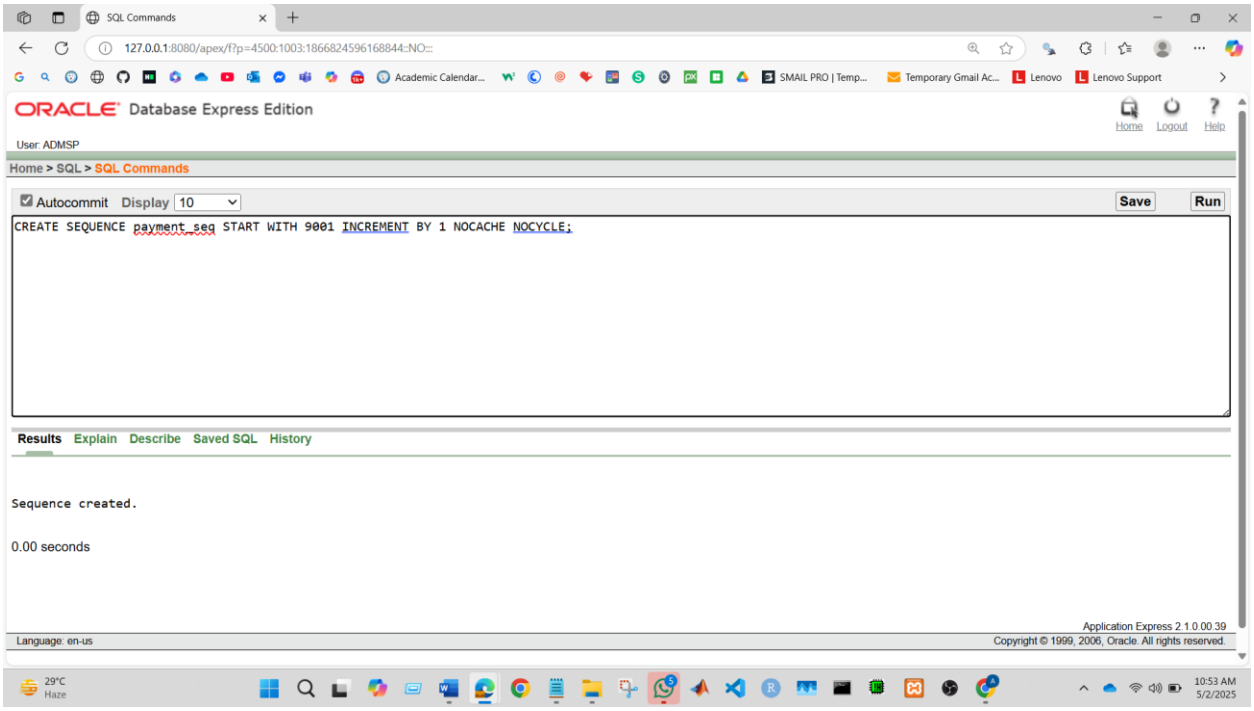


Table:

```
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)
);
```

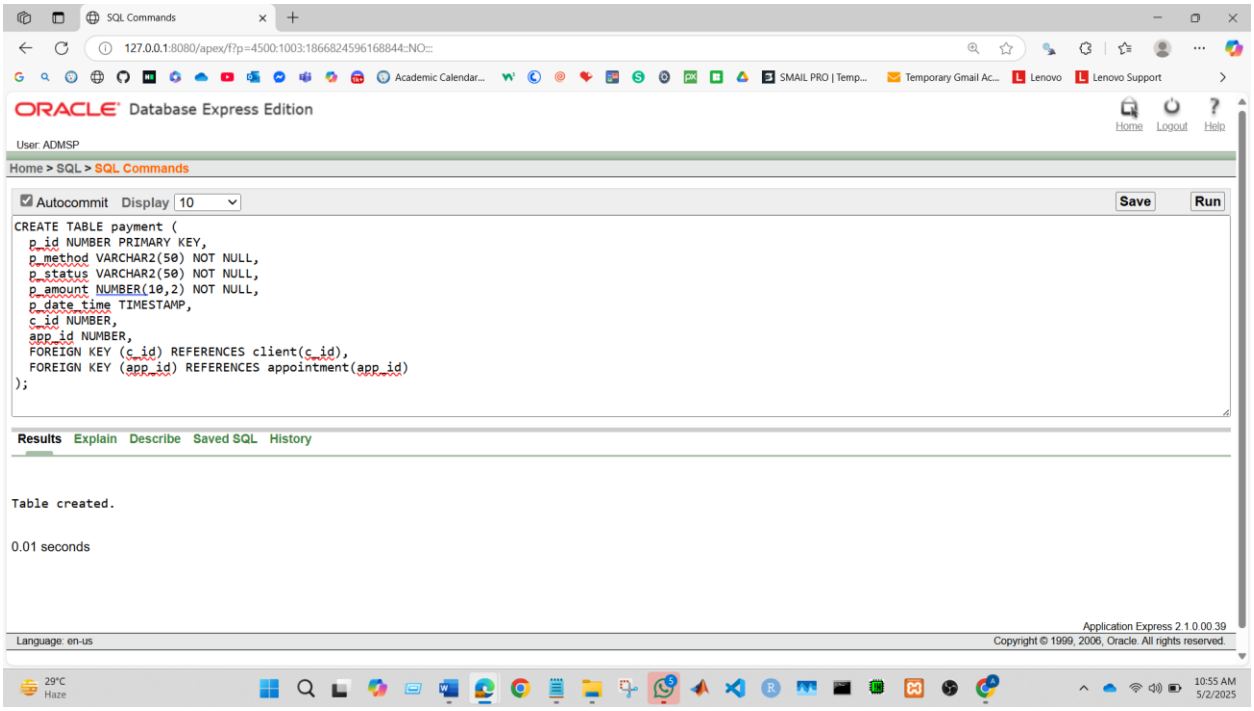


Table 11: rating

Sequence:

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

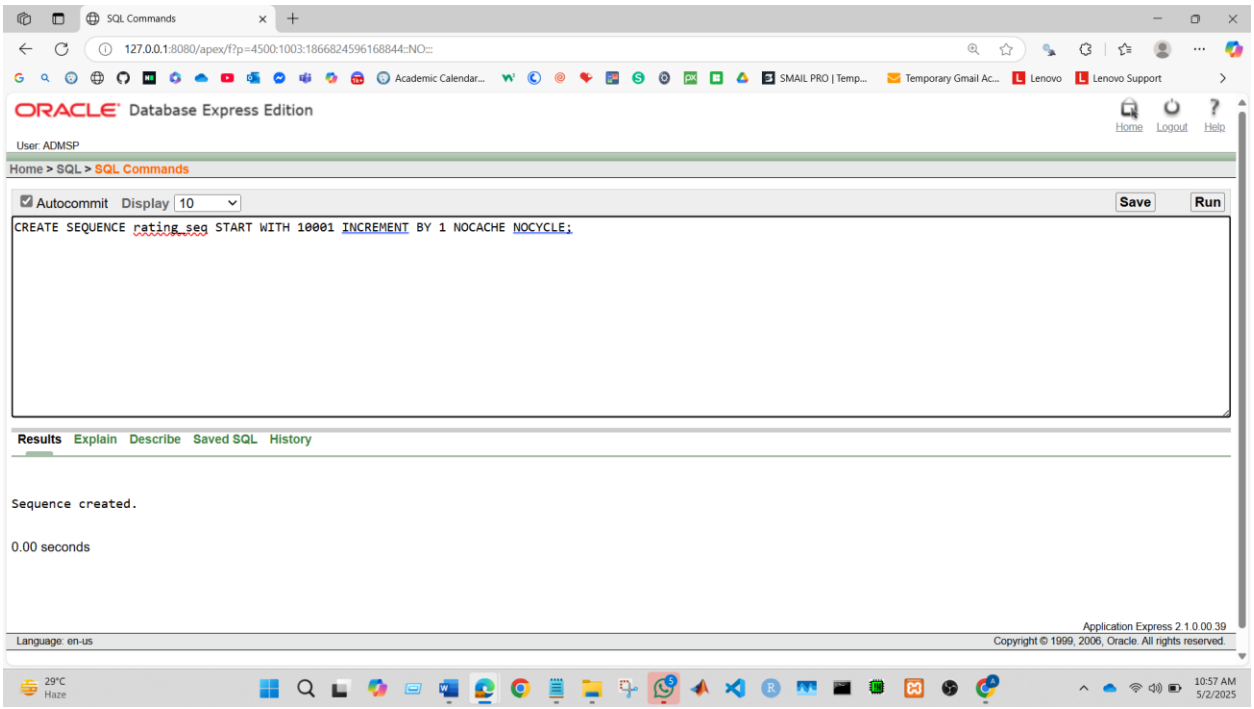
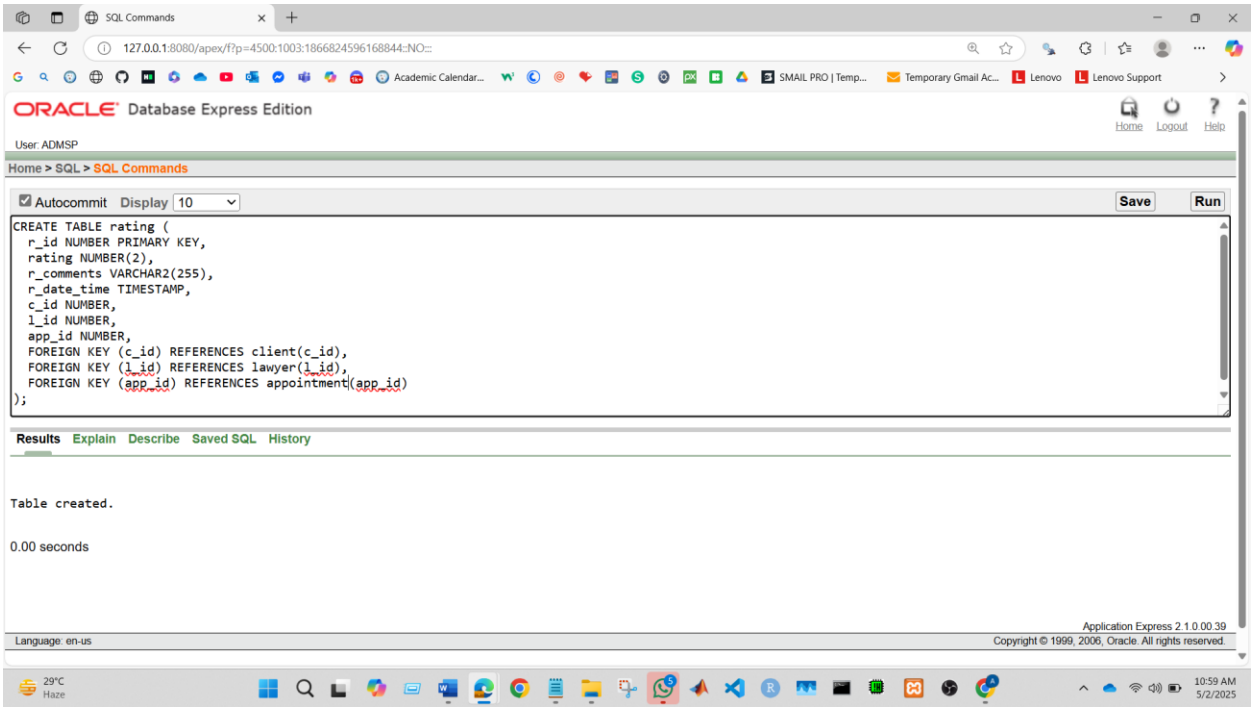


Table:

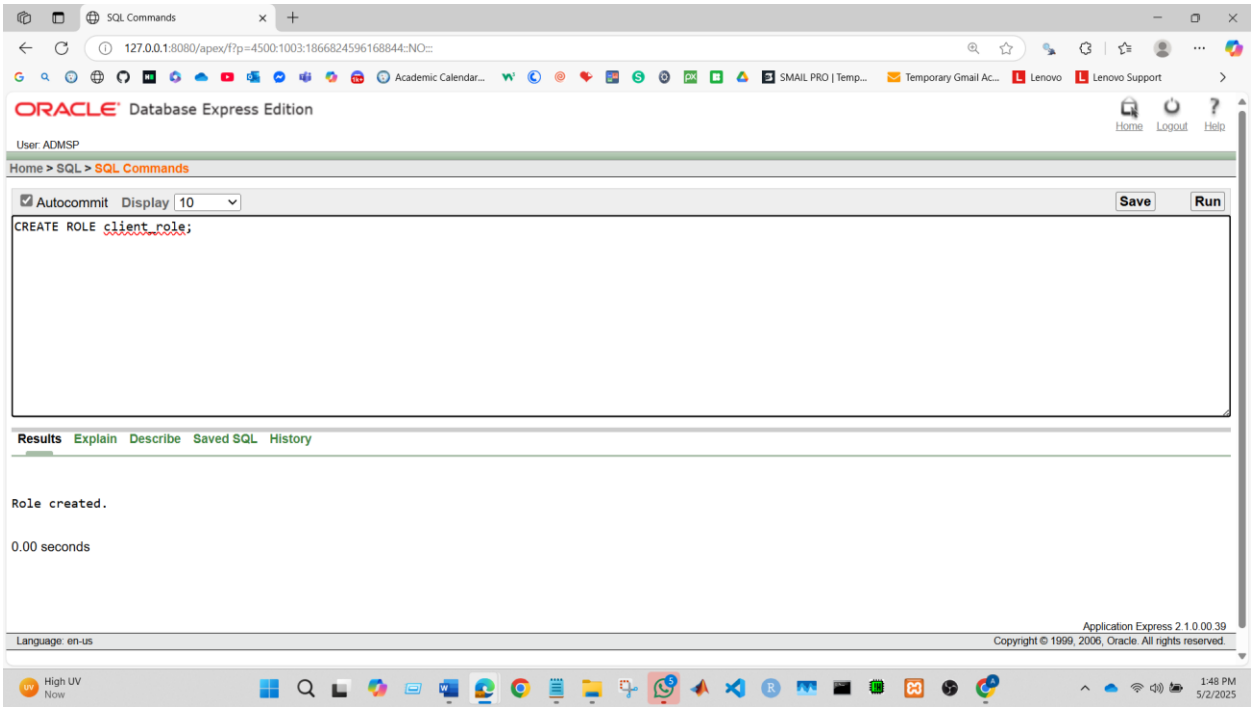
```
CREATE TABLE rating (
  r_id NUMBER PRIMARY KEY,
  rating NUMBER(2),
  r_comments VARCHAR2(255),
  r_date_time TIMESTAMP,
  c_id NUMBER,
  l_id NUMBER,
  app_id NUMBER,
  FOREIGN KEY (c_id) REFERENCES client(c_id),
  FOREIGN KEY (l_id) REFERENCES lawyer(l_id),
  FOREIGN KEY (app_id) REFERENCES appointment(app_id)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



Create Roles

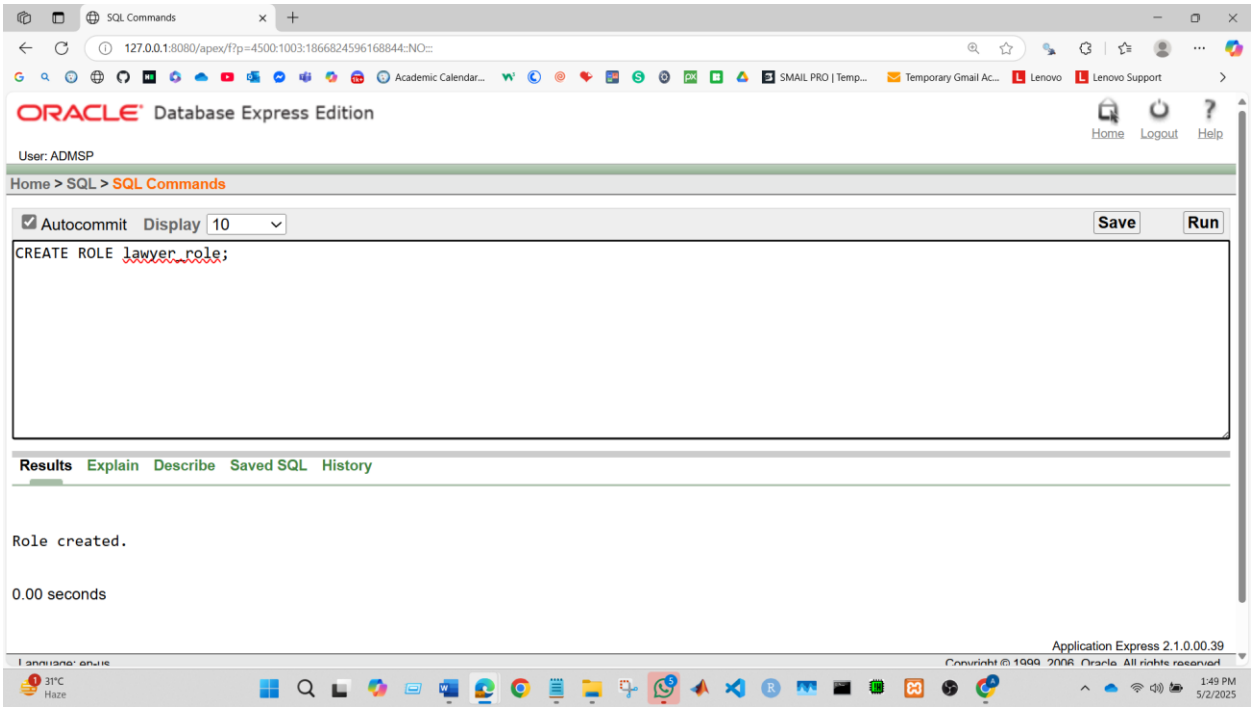
Client Role:

```
CREATE ROLE client_role;
```



Lawyer Role:

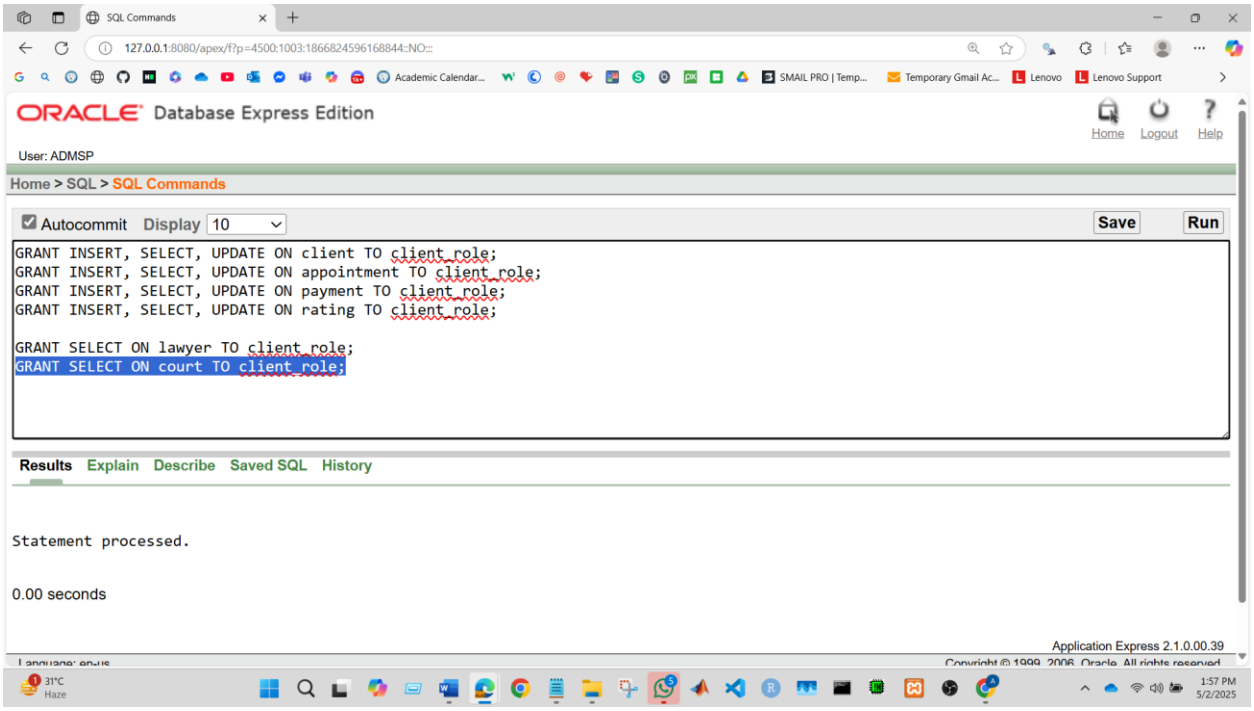
```
CREATE ROLE lawyer_role;
```



Grant Privileges to Roles:

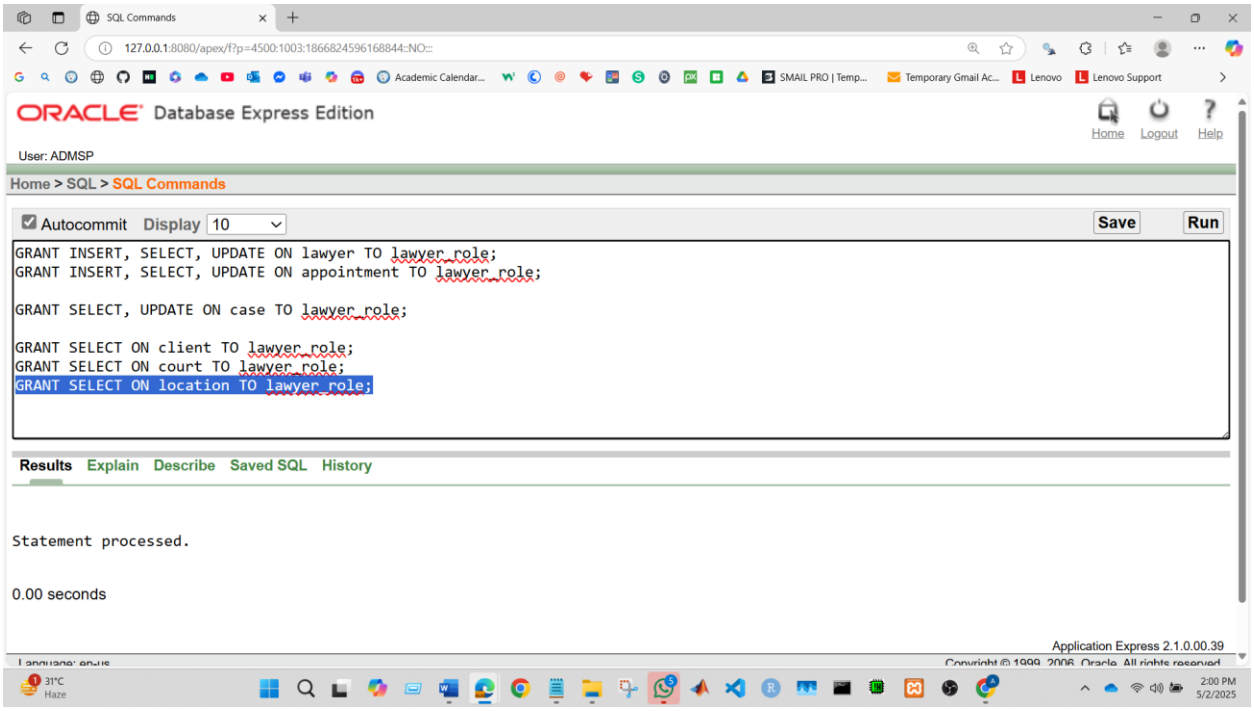
Client Privileges:

```
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;
```



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;
```


SQL Commands
127.0.0.1:8080/apex/f?p=4500:1003:186682459616884 4::NO:::
ORACLE Database Express Edition
Home
Logout
Help
User: ADMSP
Home > SQL > SQL Commands
Autocommit
Display 10
Save
Run
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;
Results
Explain
Describe
Saved SQL
History
Statement processed.
0.00 seconds
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
```

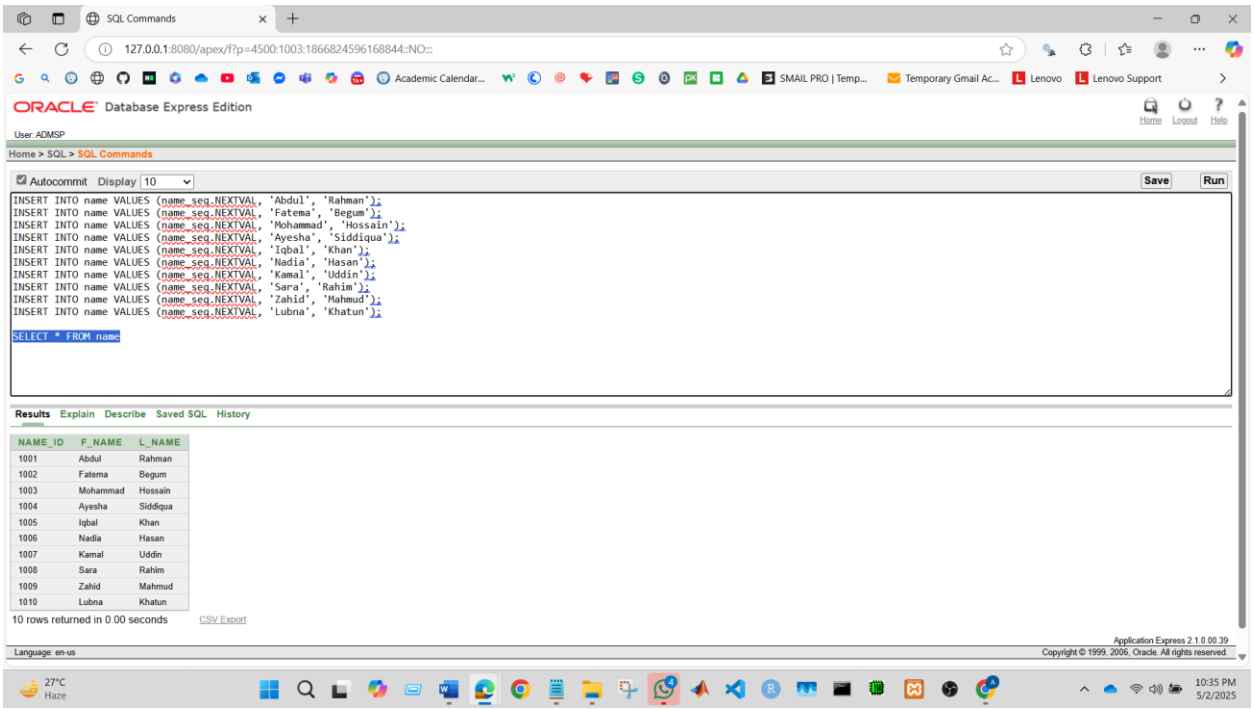


Table 2: address

```
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
```

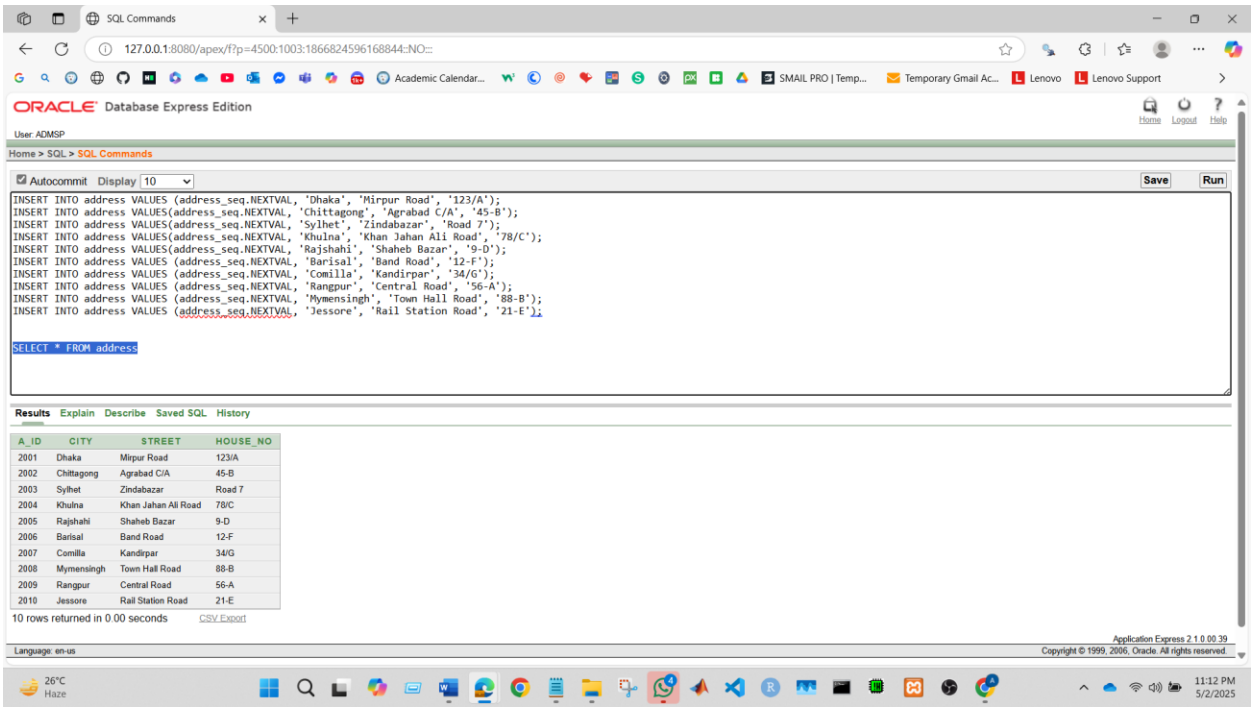


Table 3: user

```
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', 'bdpass456', 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
```

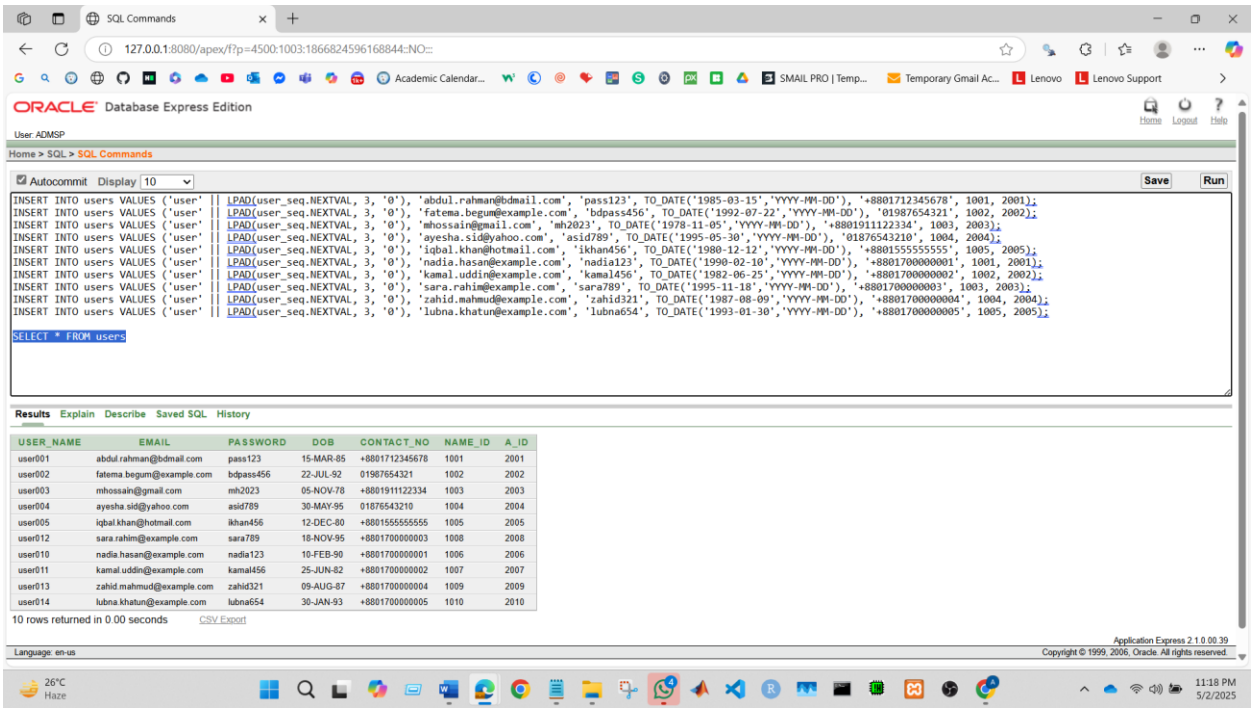


Table 4 client:

```
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
```

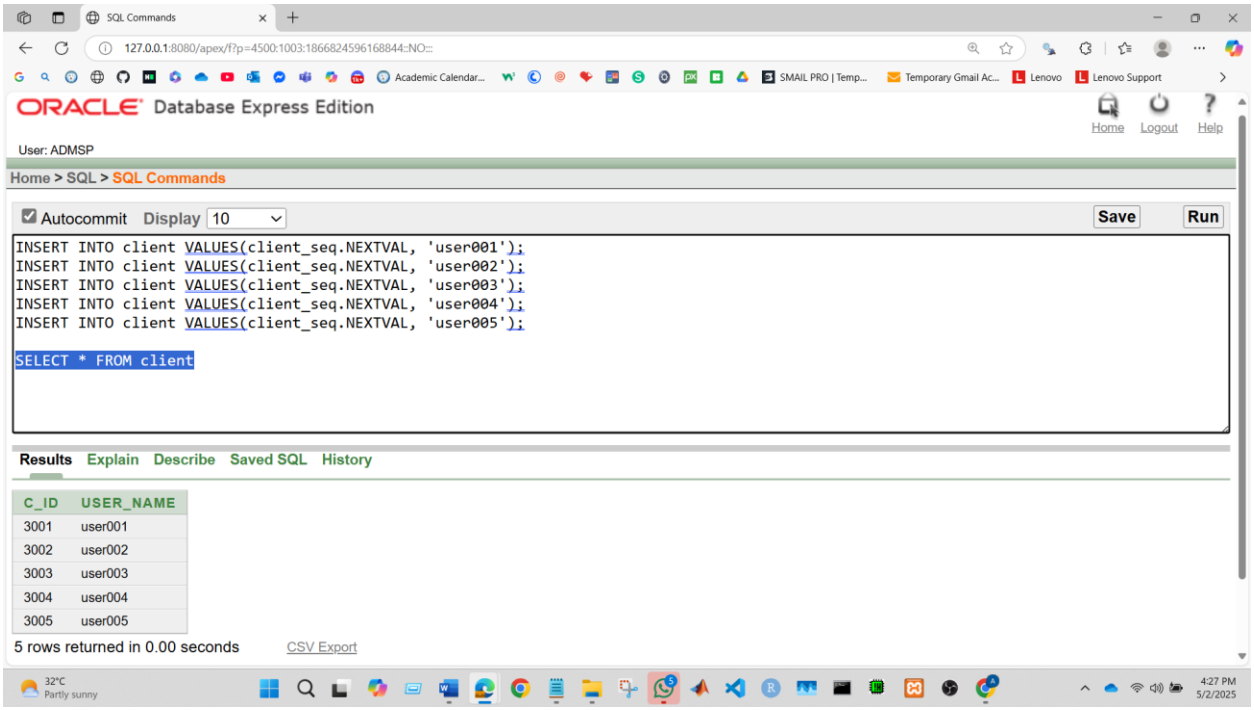


Table 5: lawyer

```
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
```

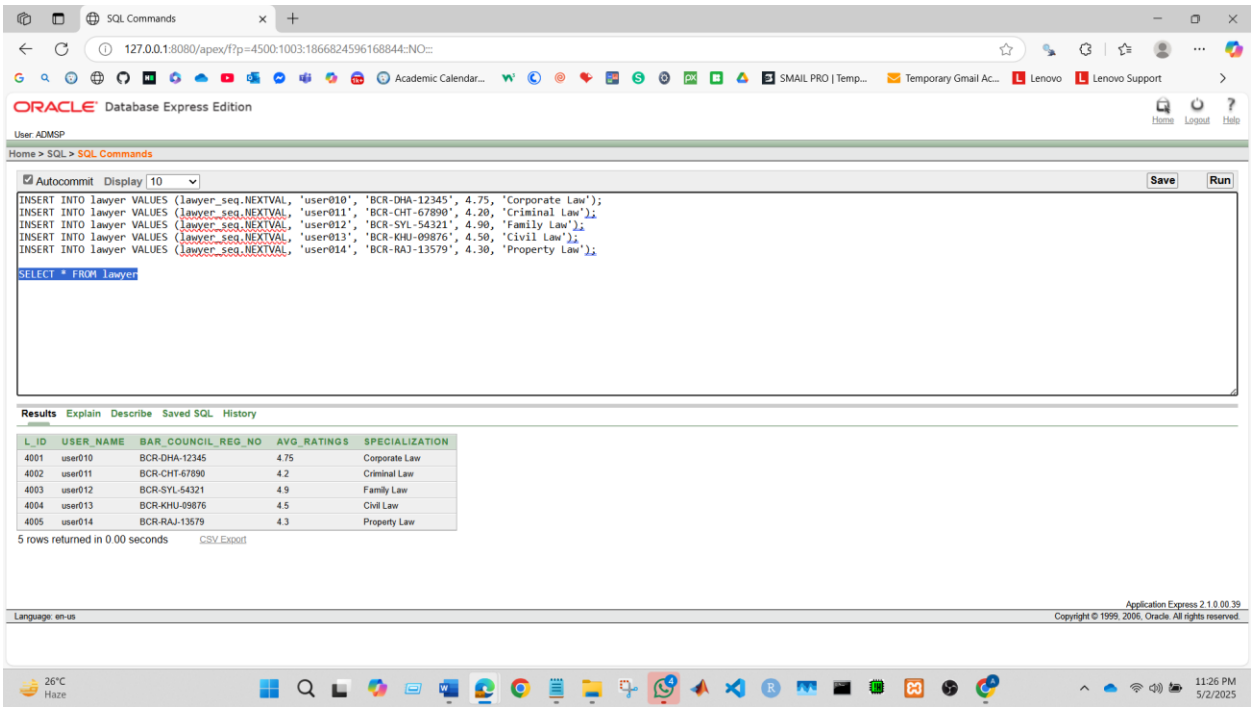


Table 6: location

```
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
```

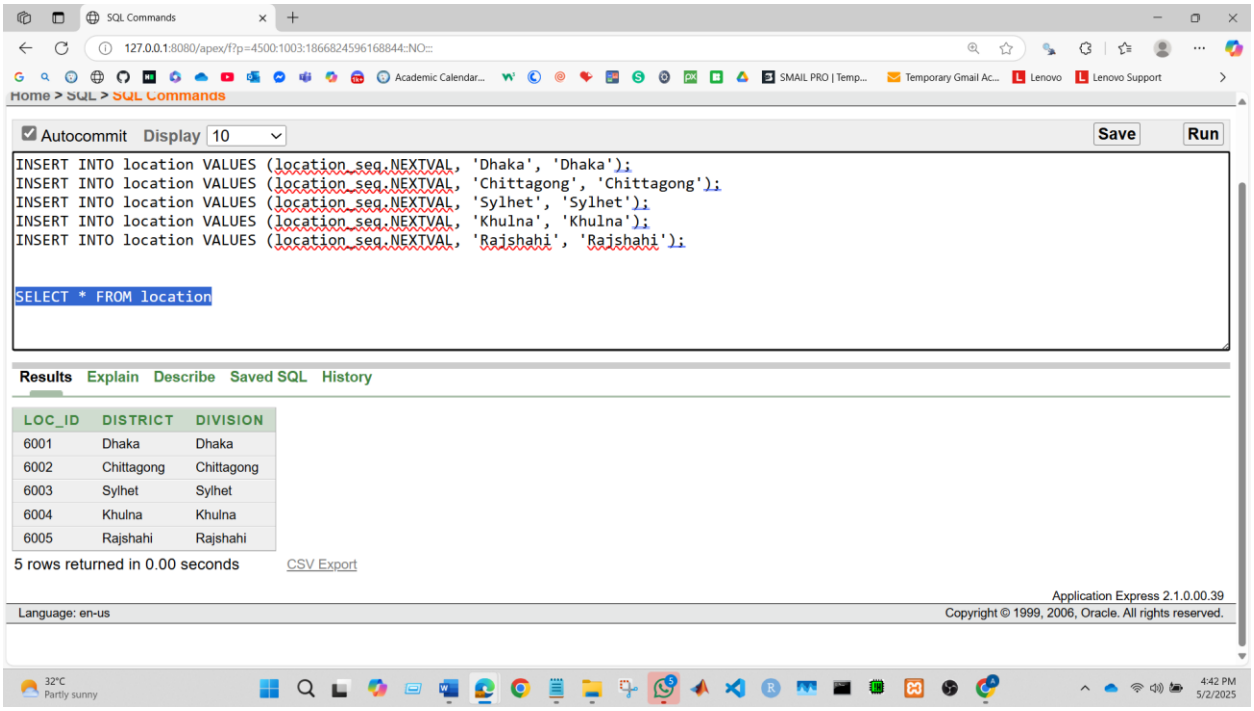


Table 7: court

```
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
```

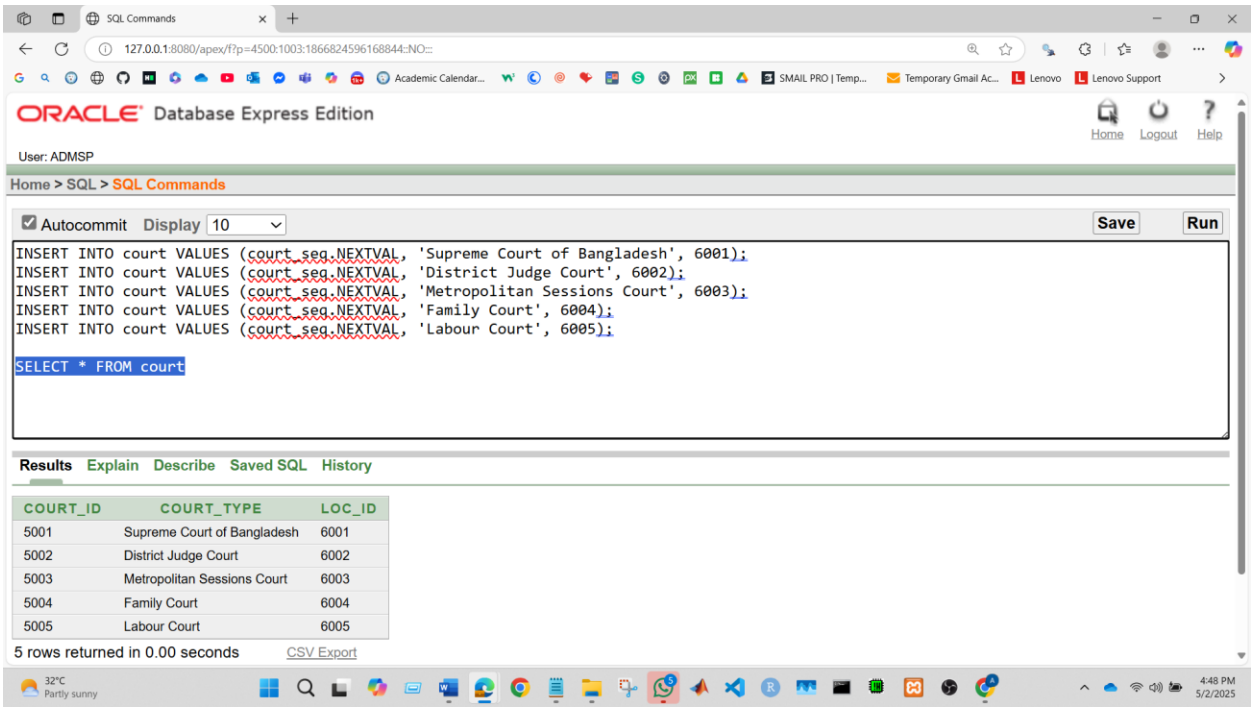


Table 8: case

```
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
```

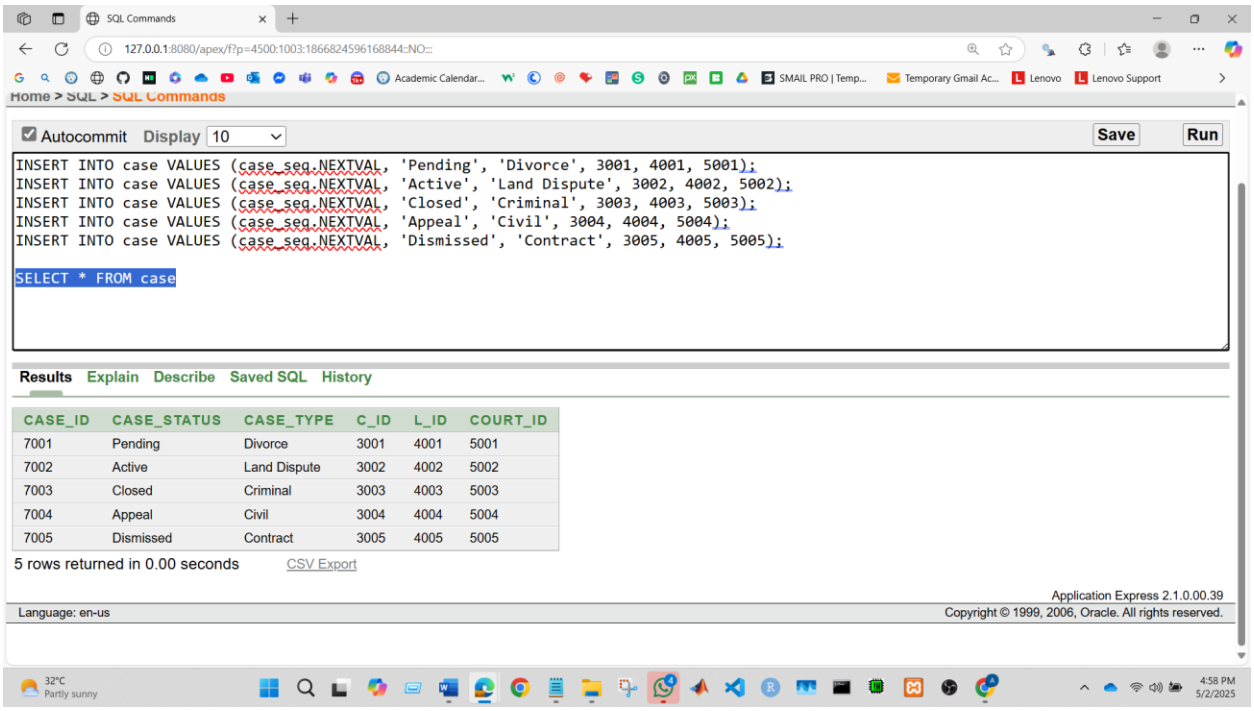


Table 9: appointment

```
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
```

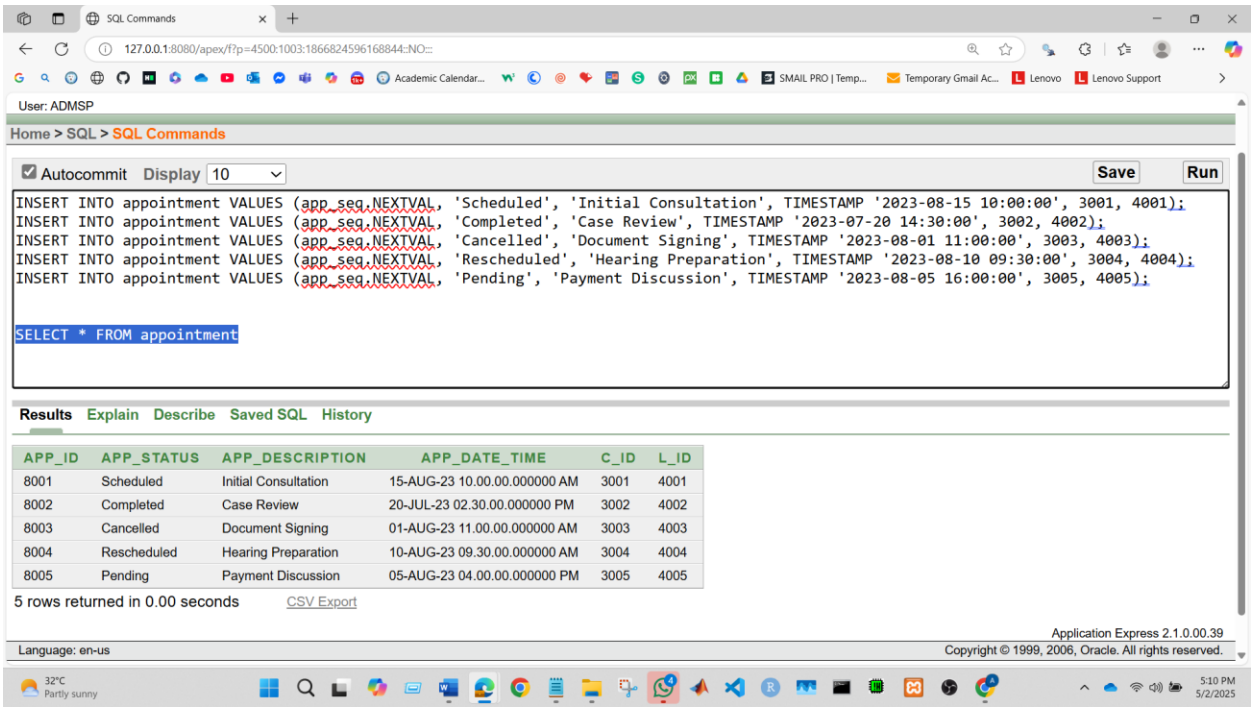


Table 10: payment

```
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
```

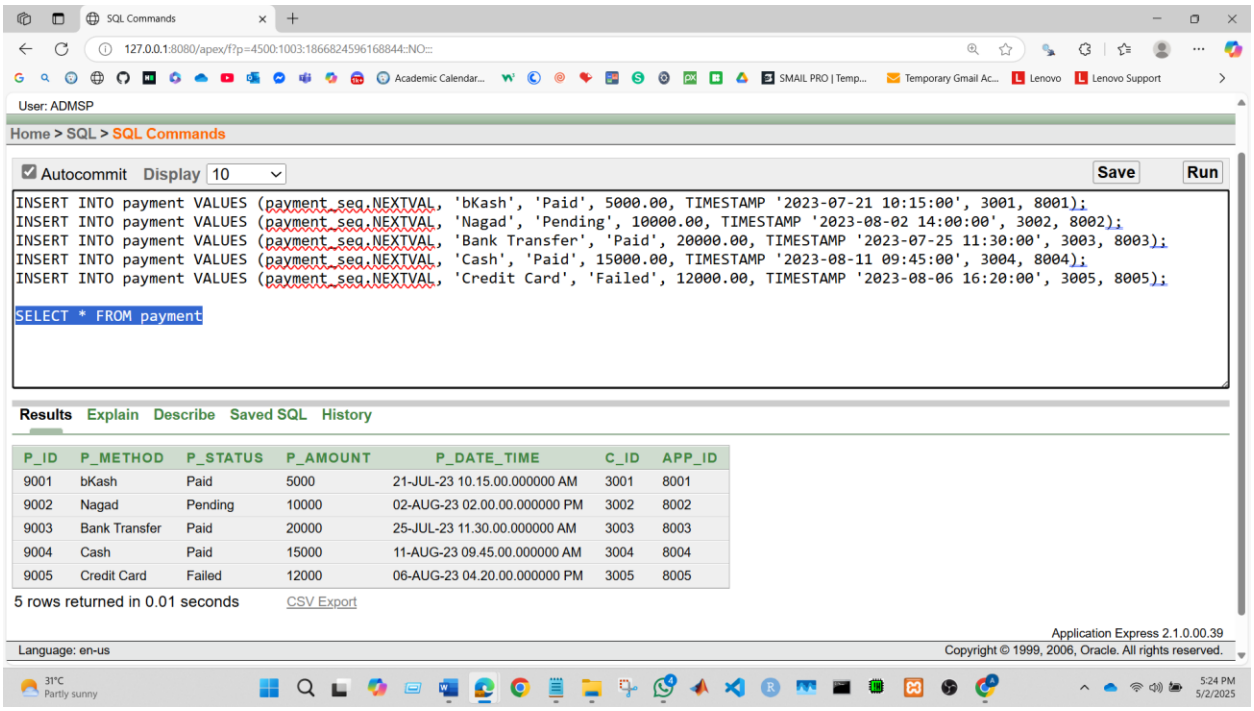
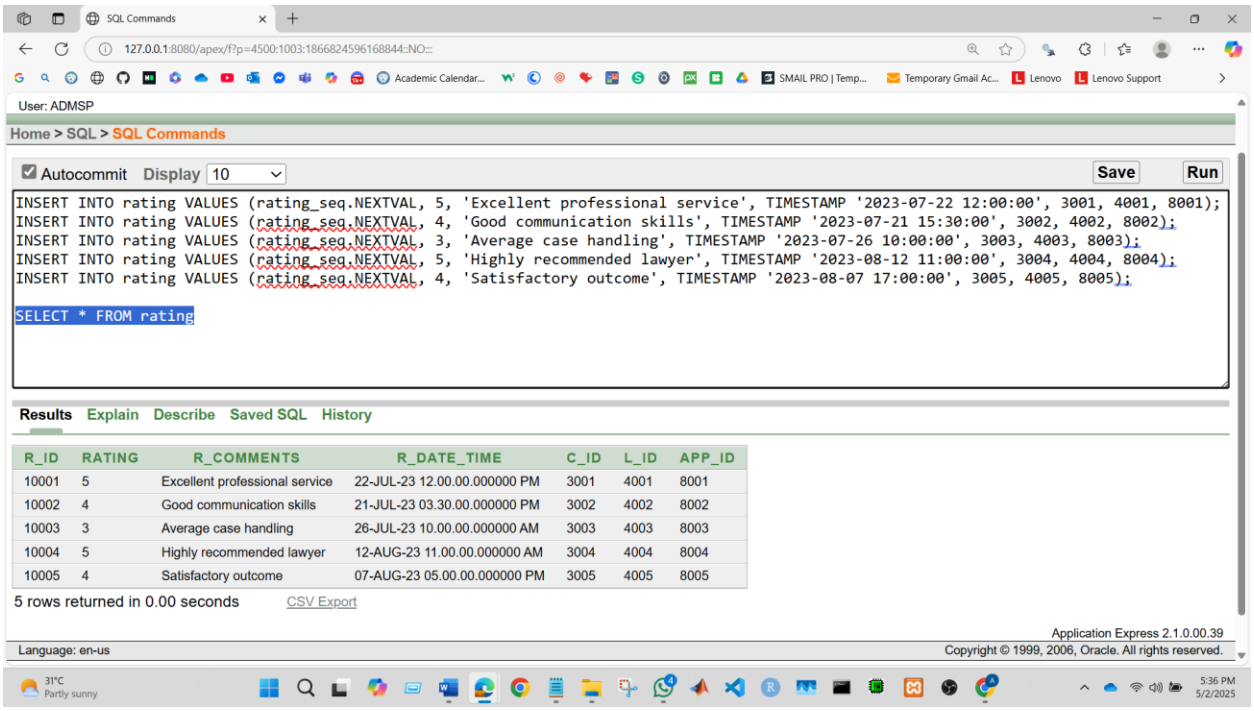


Table 11: rating

```
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=4500:1003:18668245961688844::NO:::
User: ADMSP
Home > SQL > SQL Commands
Autocommit Display 10
Save
Run
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
Application Express 2.1.0.00.39
Language: en-us
Copyright © 1999, 2006, Oracle. All rights reserved.

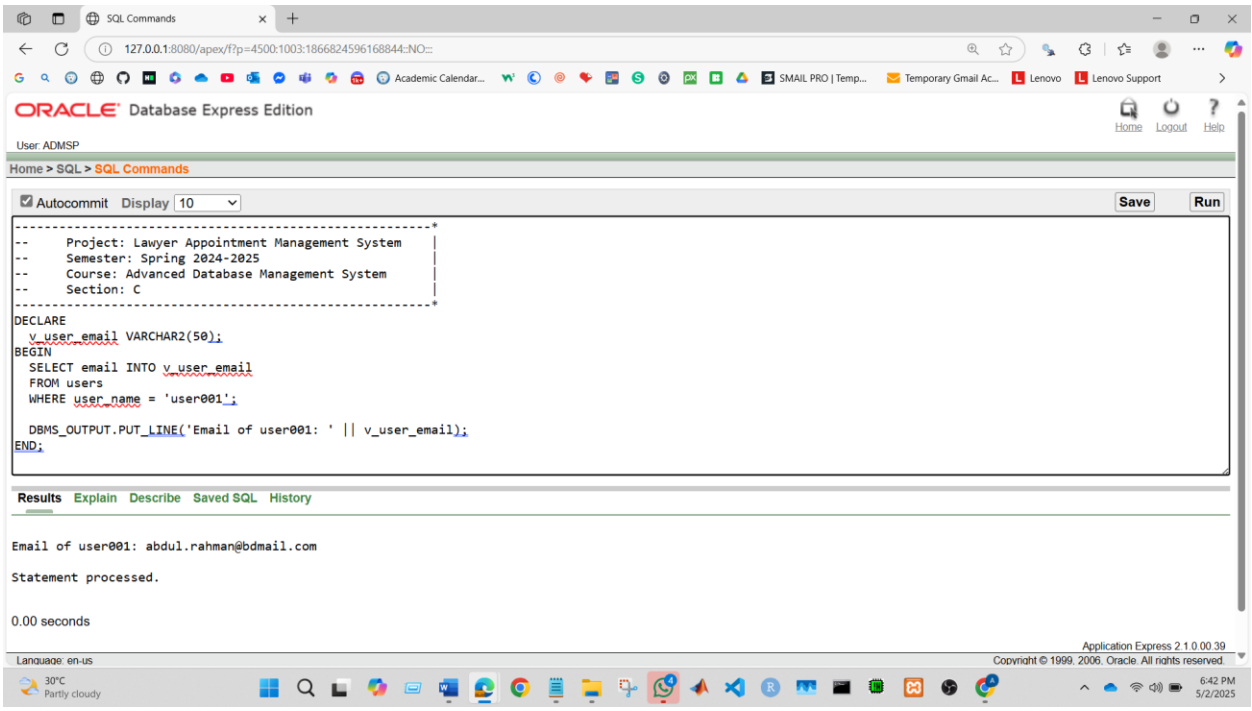
Query Writing

Basic PL/SQL

Variables:

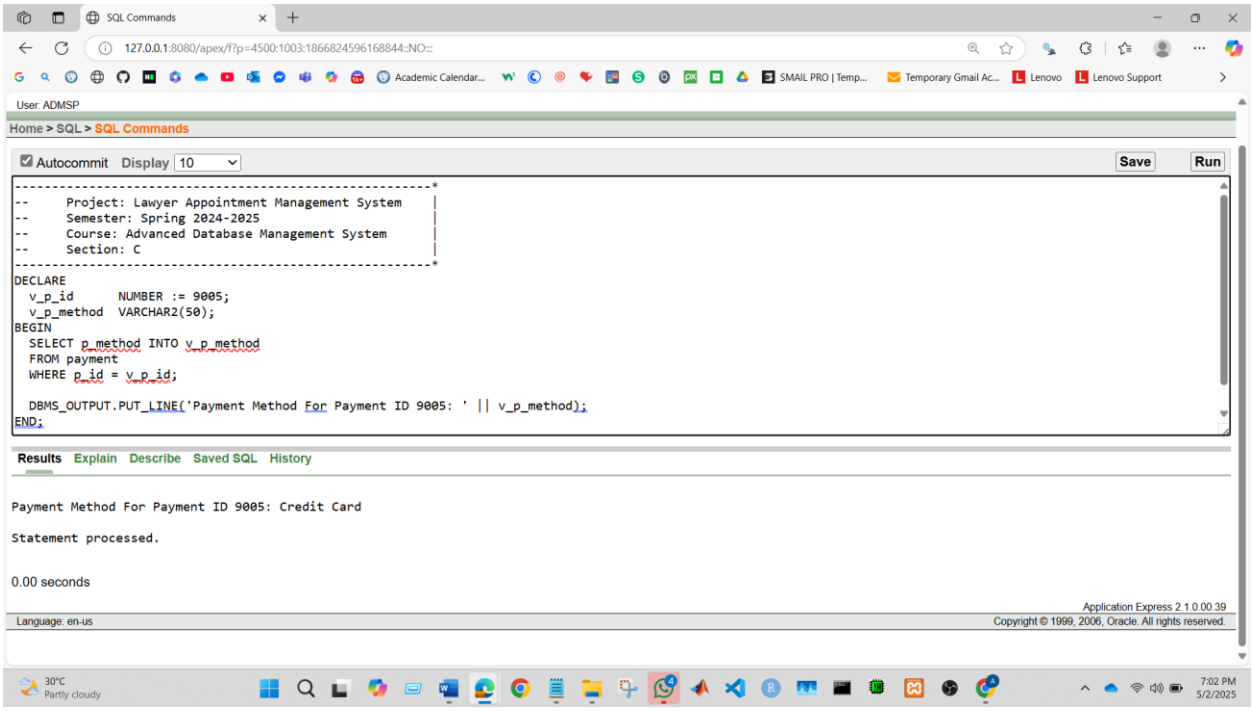
1. Display the email of a user with user_name = 'user001' using variables.

```
-------------------------------------------------------*
--      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;
```



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;
```

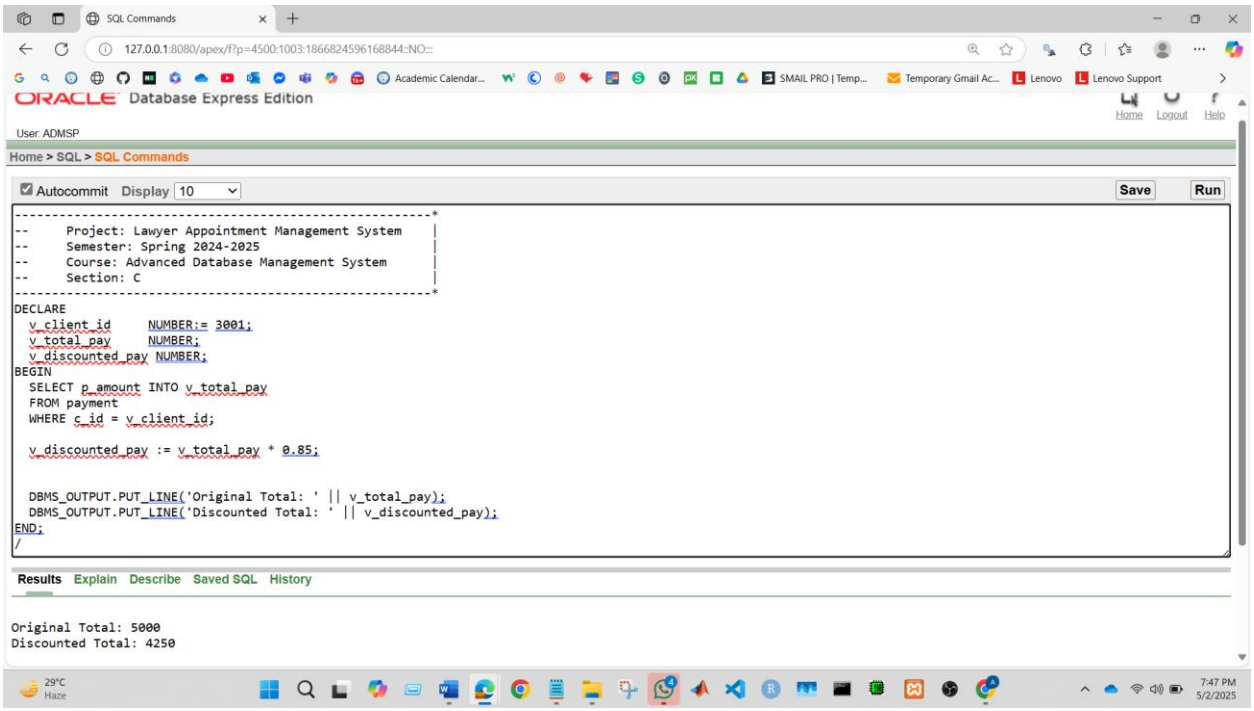


Operators:

1. 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;
```

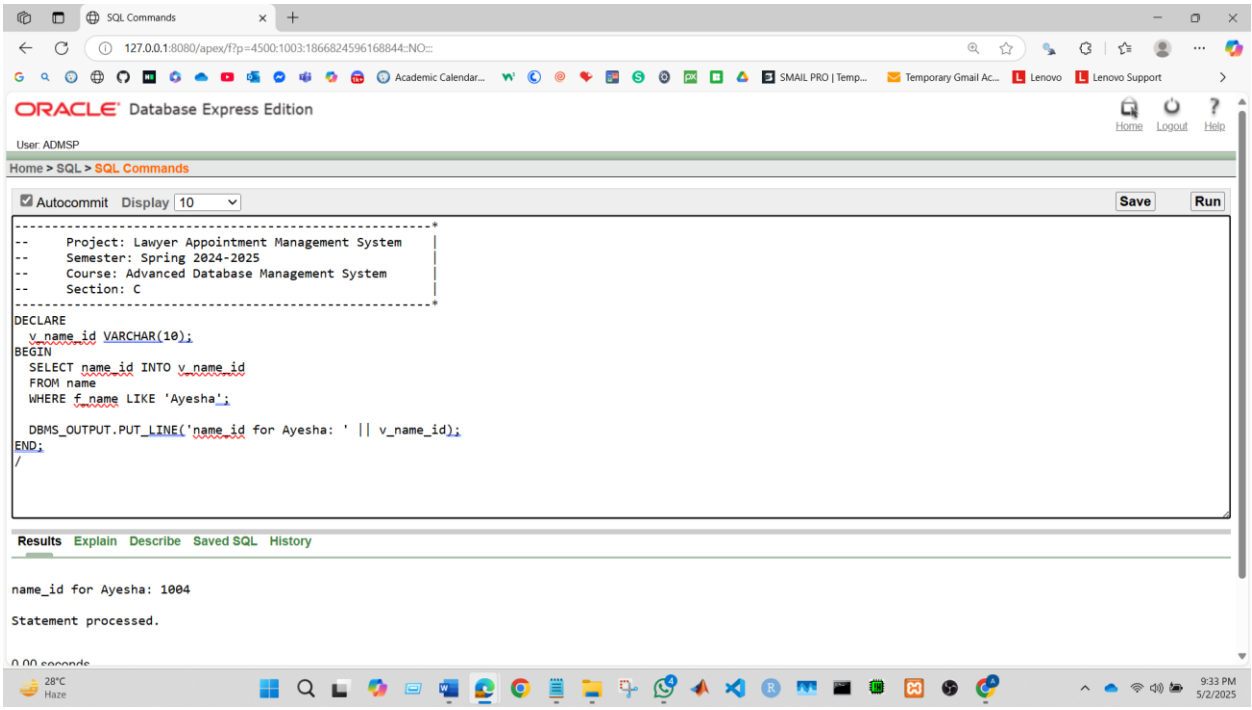
```
  DBMS_OUTPUT.PUT_LINE('Original Total: ' || v_total_pay);
  DBMS_OUTPUT.PUT_LINE('Discounted Total: ' || v_discounted_pay);
END;
/
```



2. 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;
/
```

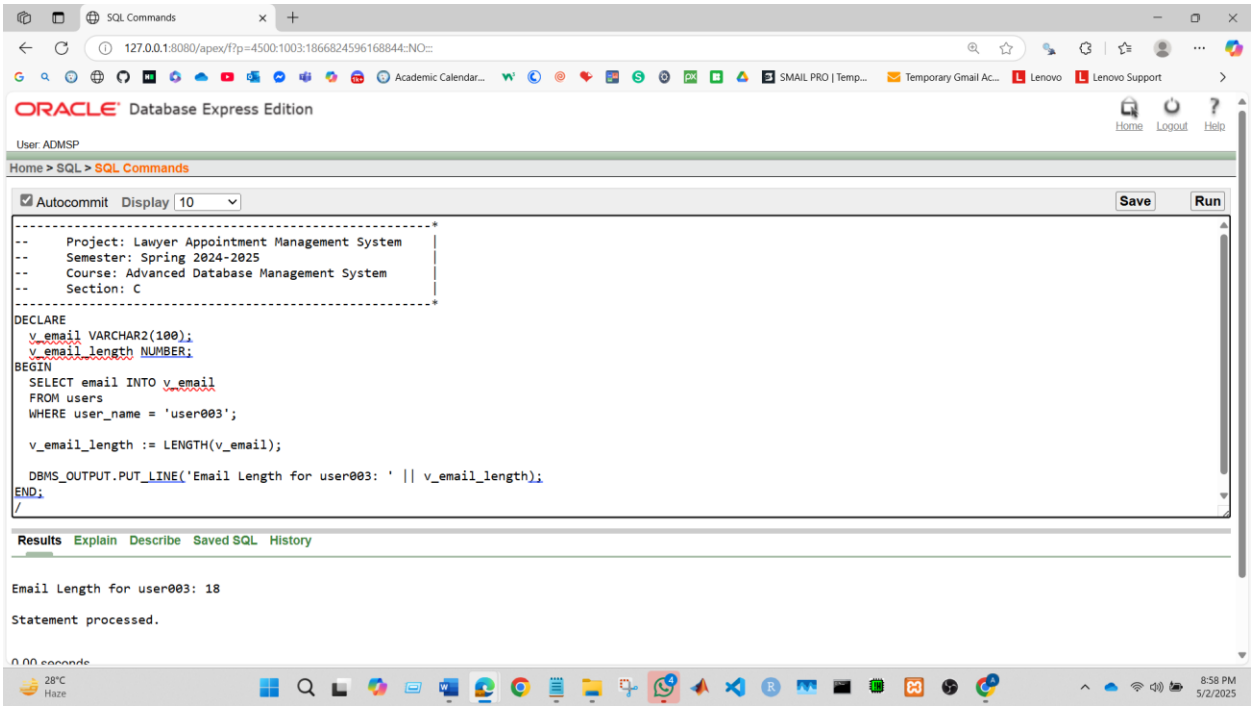


Single-Row Function:

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);
```

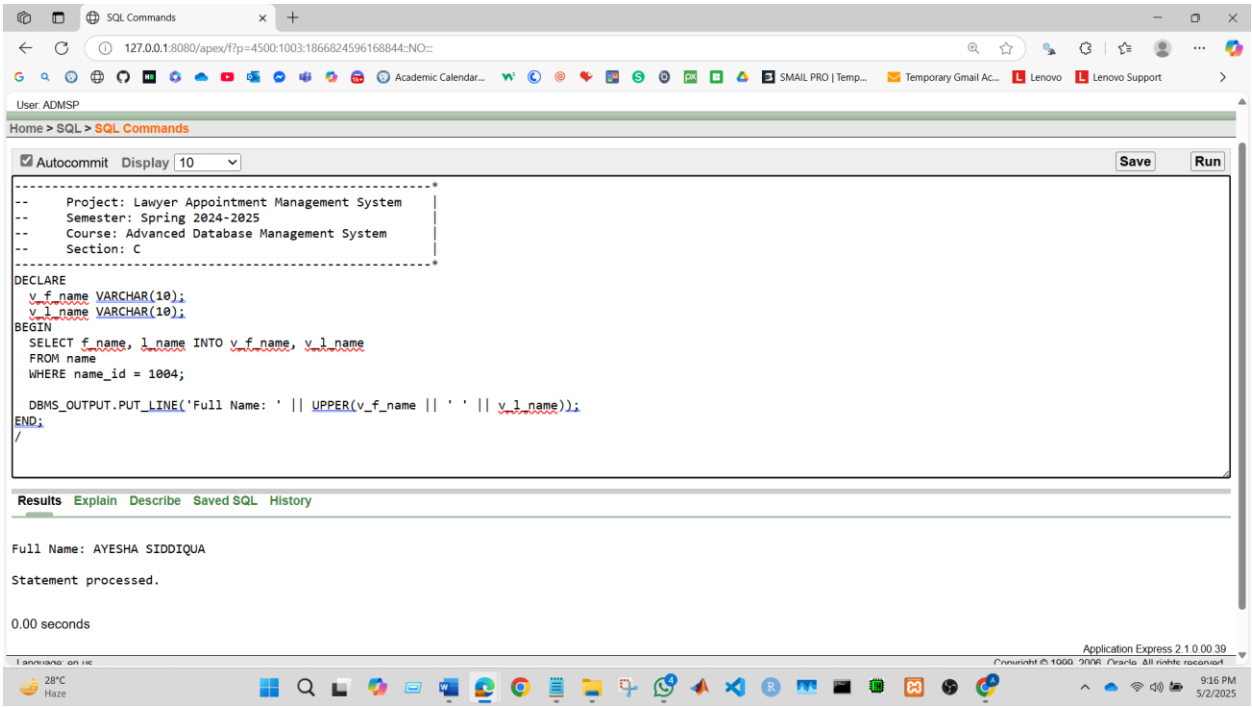
```
  DBMS_OUTPUT.PUT_LINE('Email Length for user003: ' || v_email_length);
END;
/
```



2. 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;
/
```

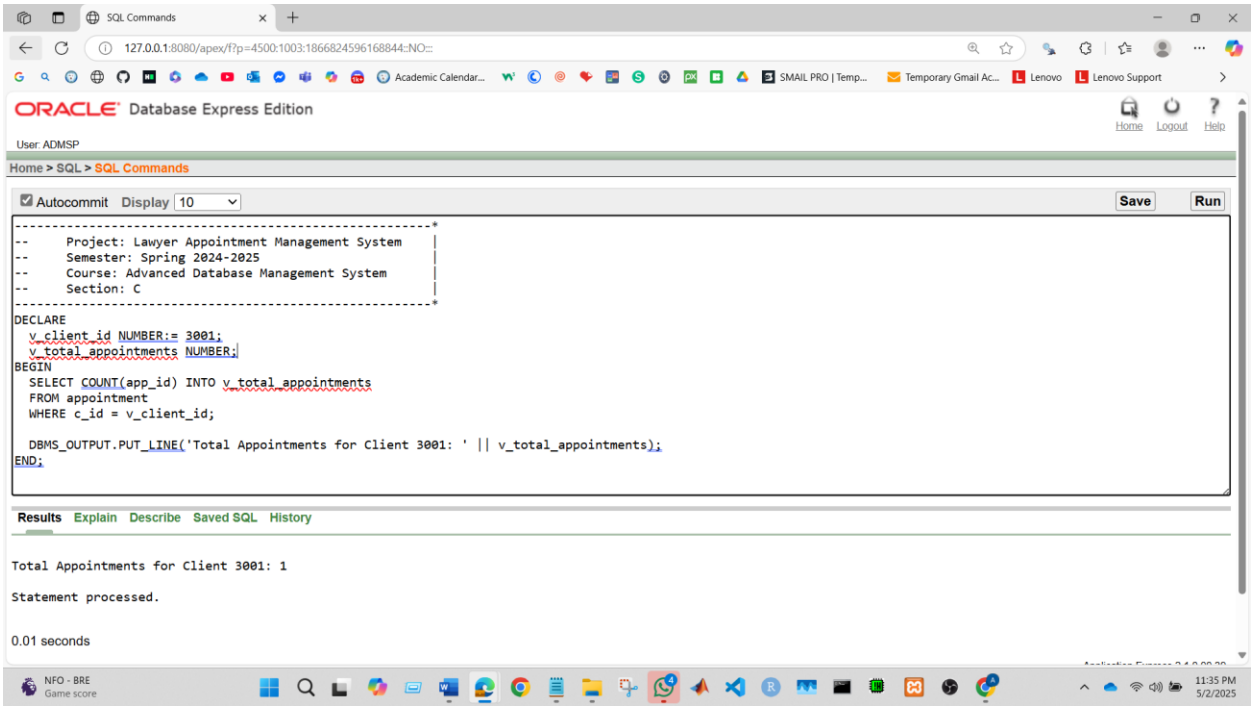


Group Function:

1. 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;
```

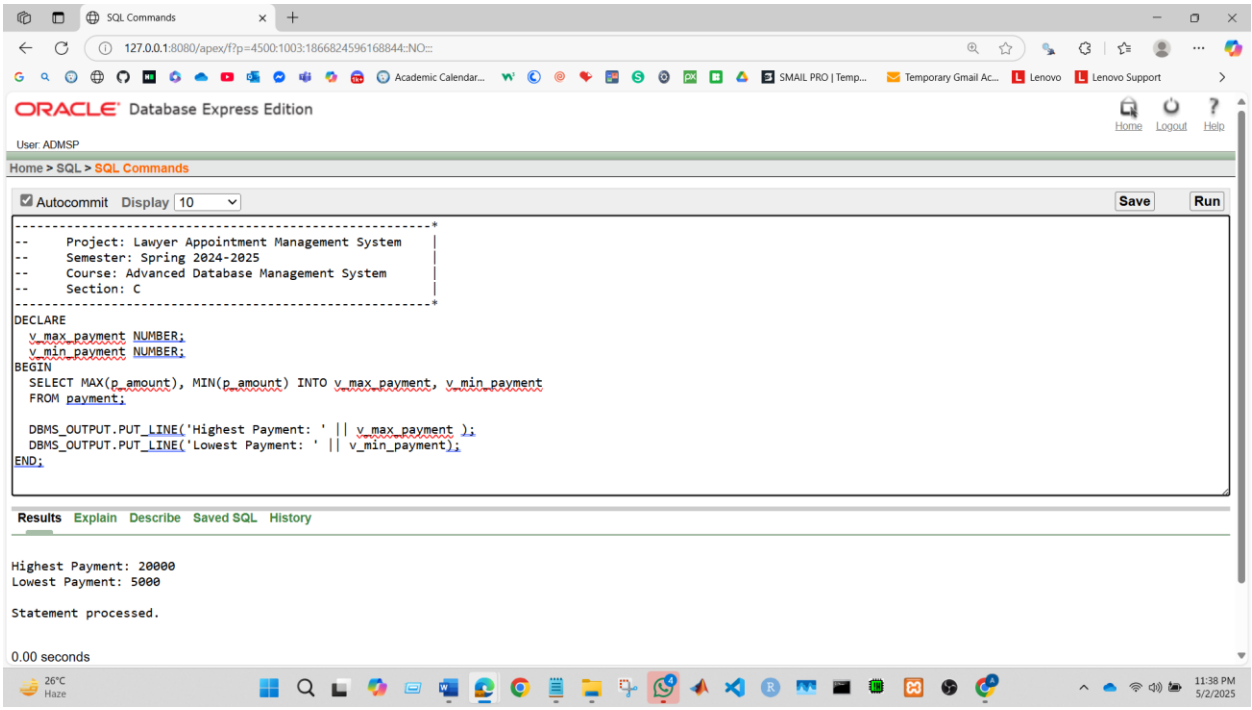
```
  DBMS_OUTPUT.PUT_LINE('Total Appointments for Client 3001: ' ||
v_total_appointments);
END;
```



2. 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;
/
```

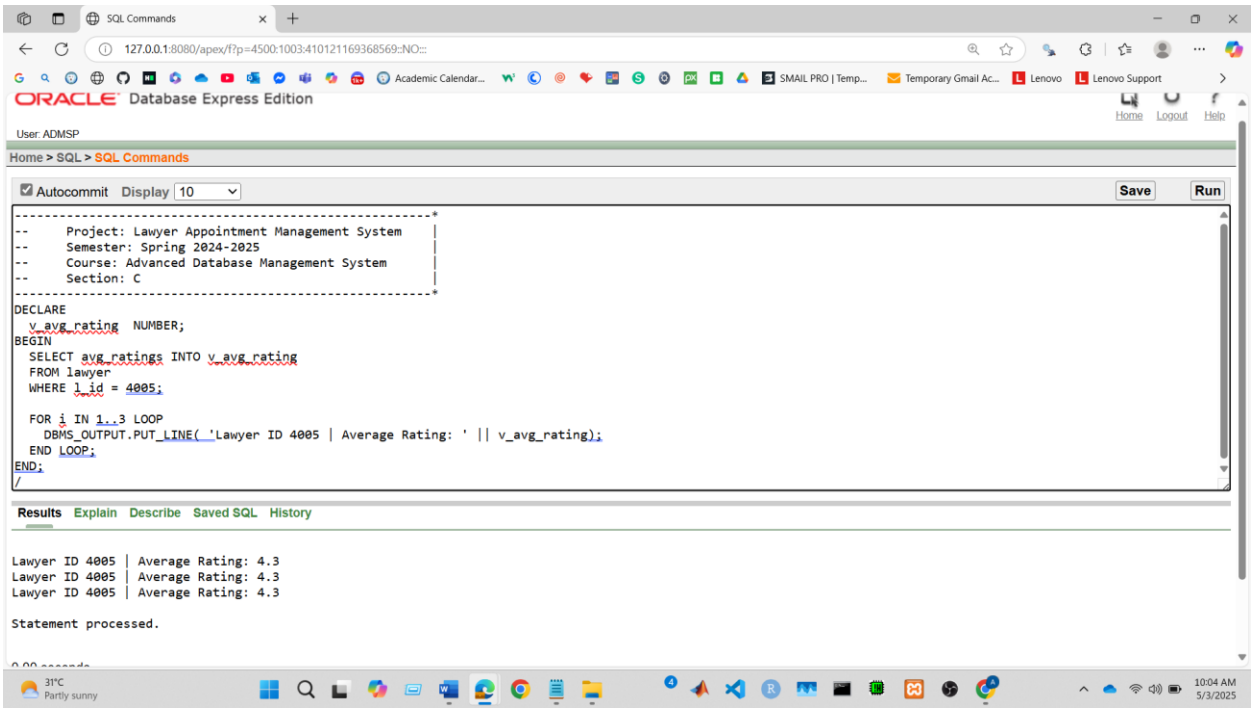


Loop:

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
```

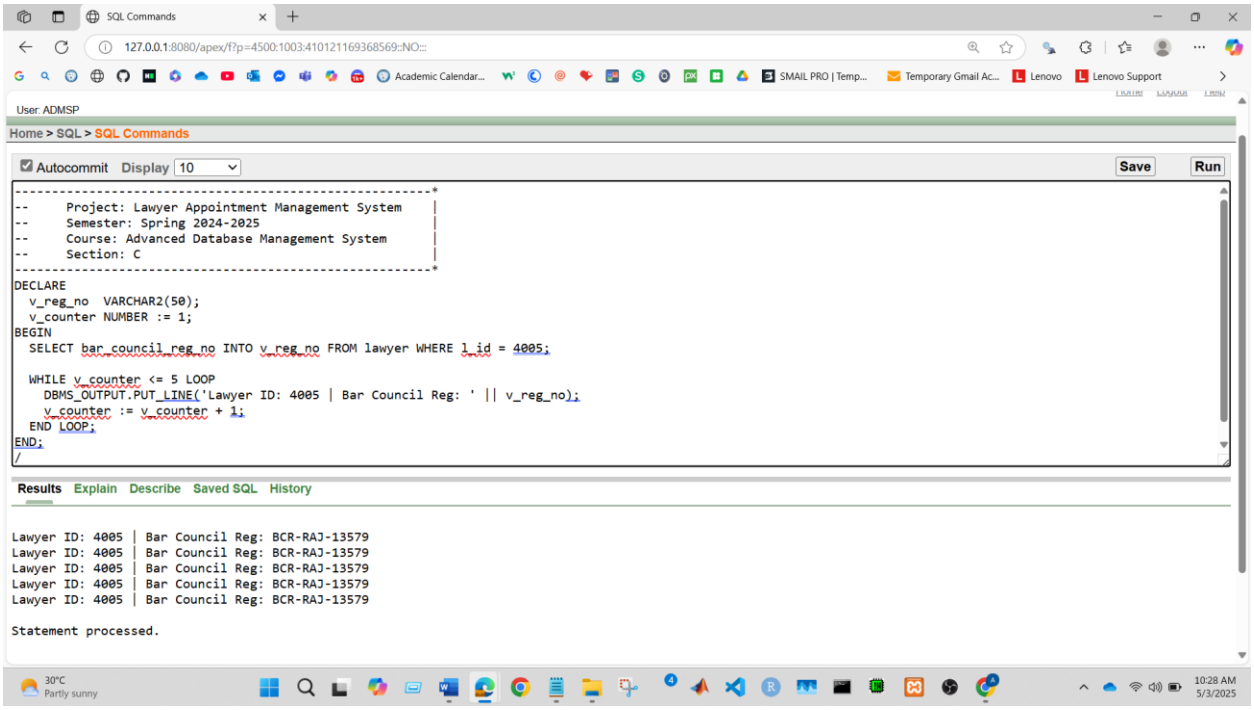
```
    DBMS_OUTPUT.PUT_LINE( 'Lawyer ID 4005 | Average Rating: ' ||
v_avg_rating);
  END LOOP;
END;
/
```



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;
/
```

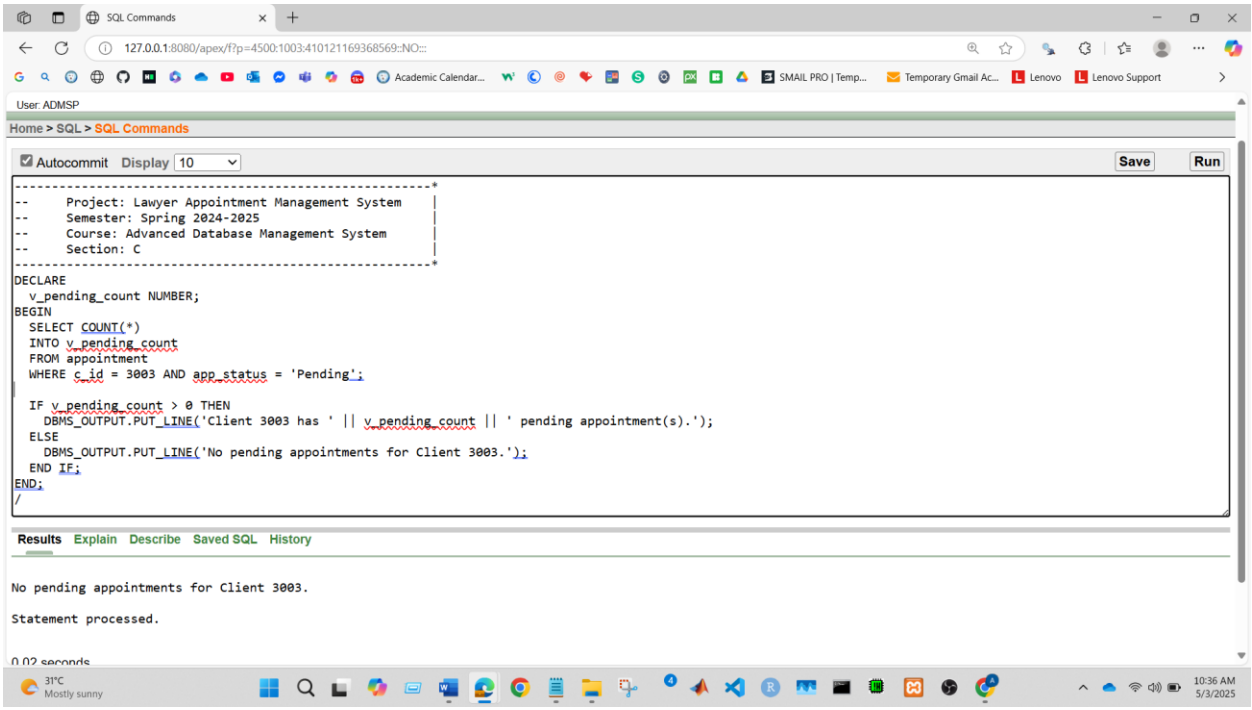


Conditional Statements:

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
```

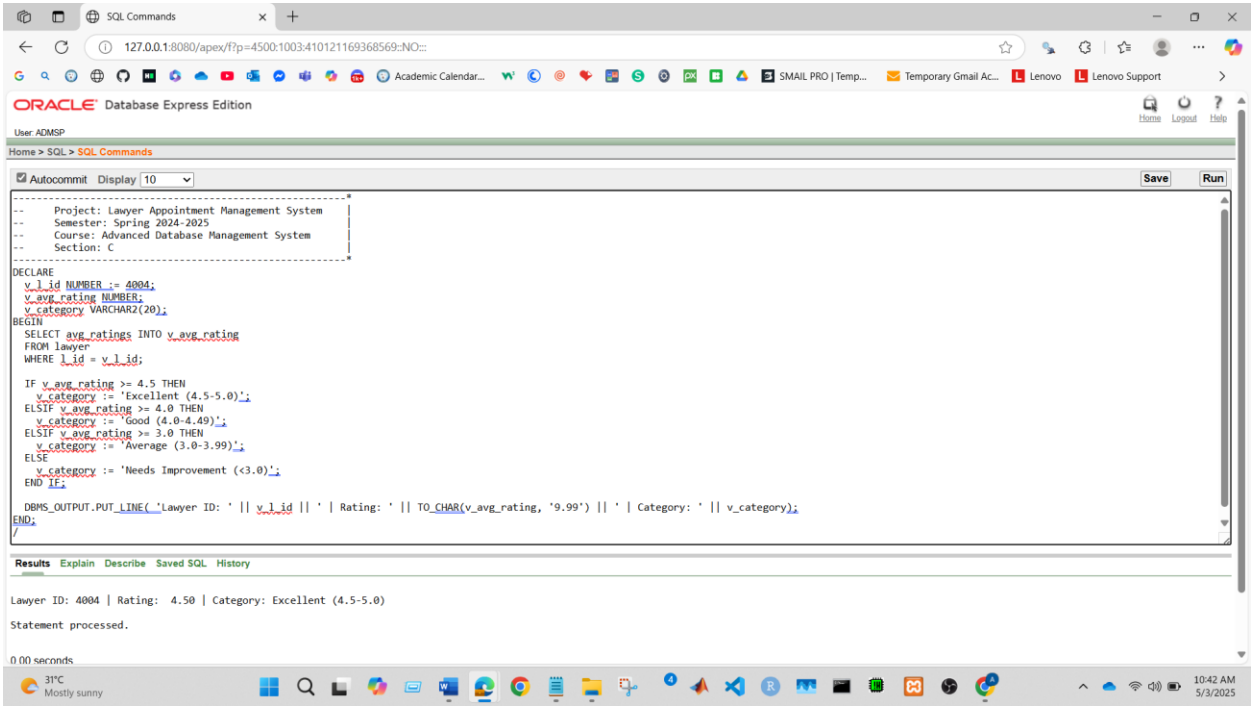
```
  FROM appointment
  WHERE c_id = 3003 AND app_status = 'Pending';

  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;
/
```



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;
```

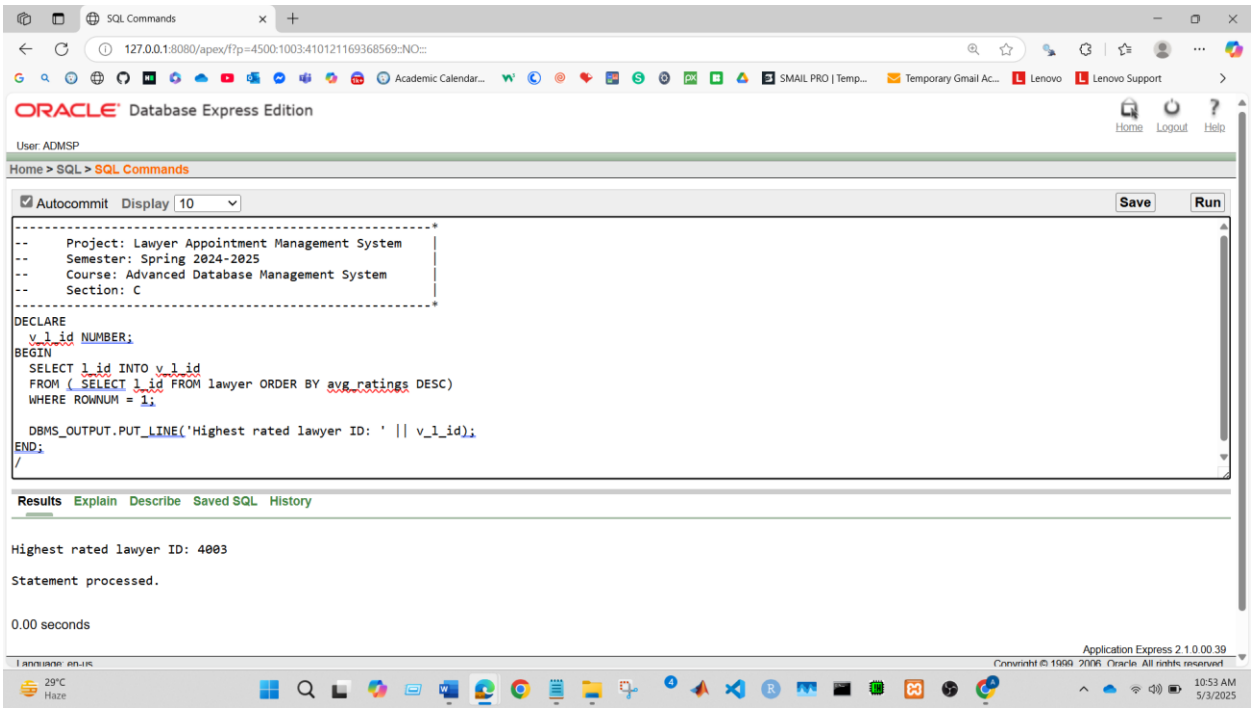
```
  v_avg_rating NUMBER;
  v_category VARCHAR2(20);
BEGIN
  SELECT avg_ratings INTO v_avg_rating
  FROM lawyer
  WHERE l_id = v_l_id;

  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;
/
```



Subquery:

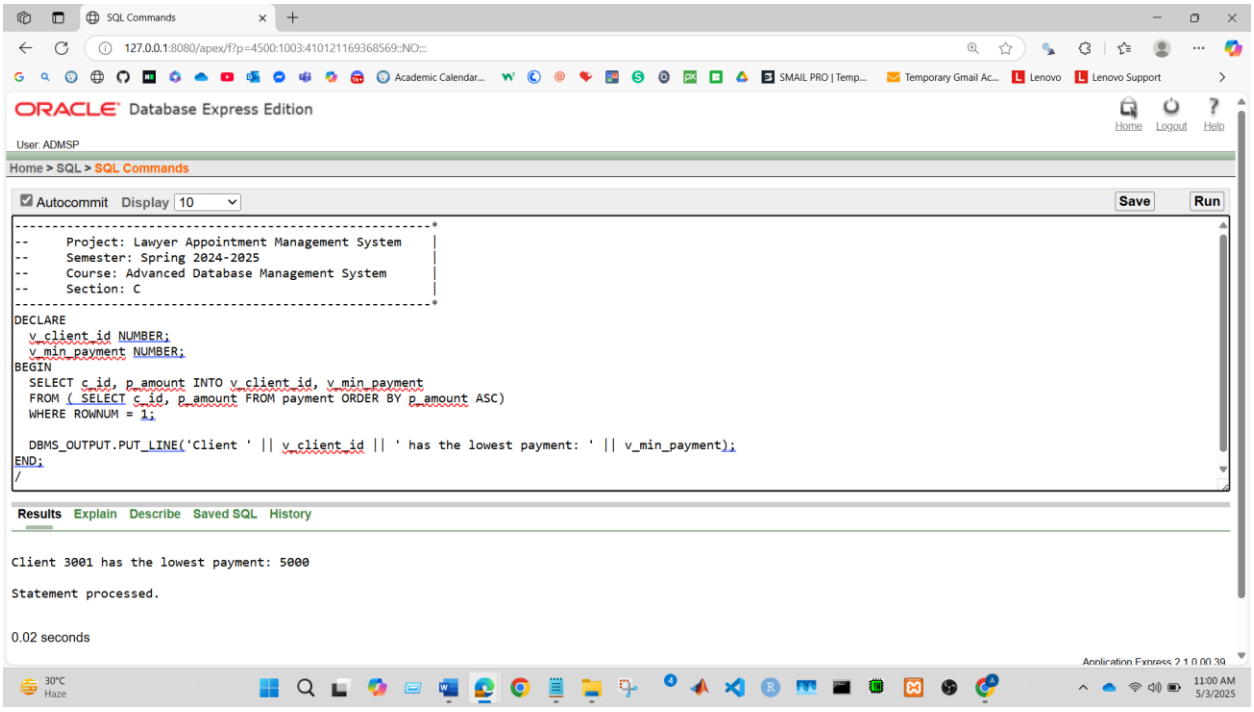
1. Show the highest rated lawyer id.

```
------------------------------------------------------------*
--      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;
/
```



2. Display the client ID with the lowest payment.

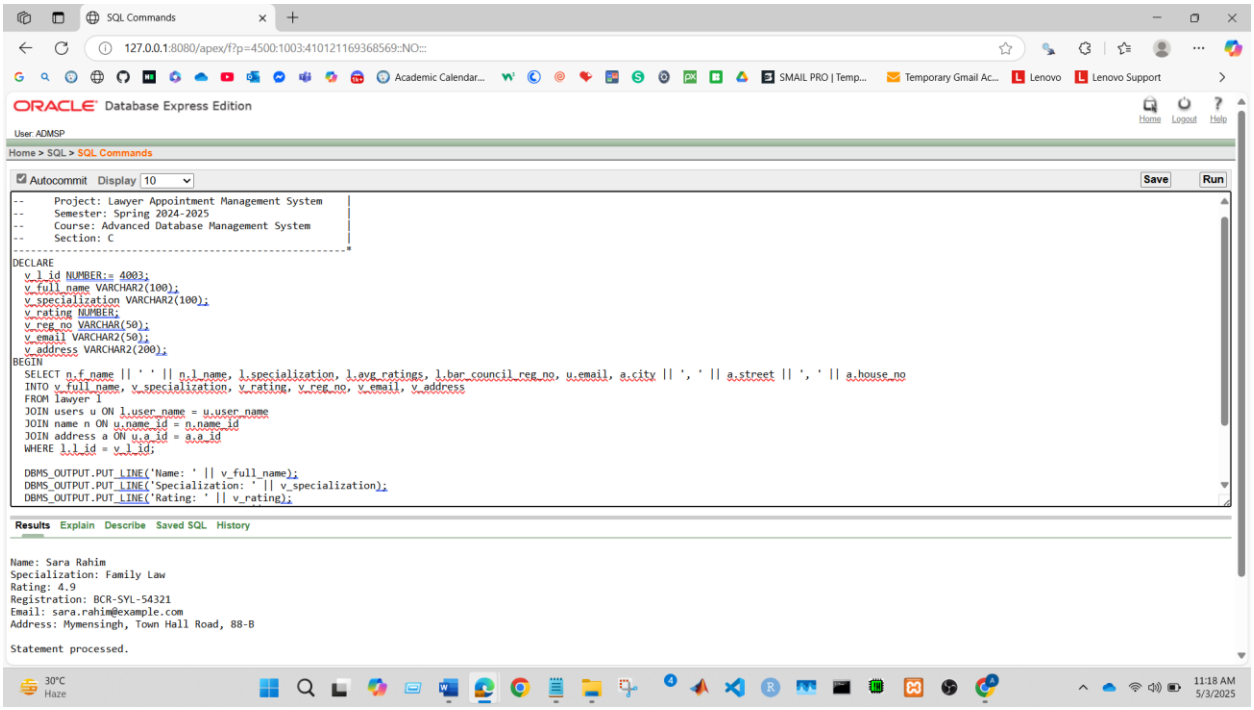
```
------------------------------------------------------------*
--      Project: Lawyer Appointment Management System       |
--      Semester: Spring 2024-2025                          |
--      Course: Advanced Database Management System         |
--      Section: C                              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



Joining:

1. **Display detailed information for a specific lawyer (l_id = 4003) using joins.**

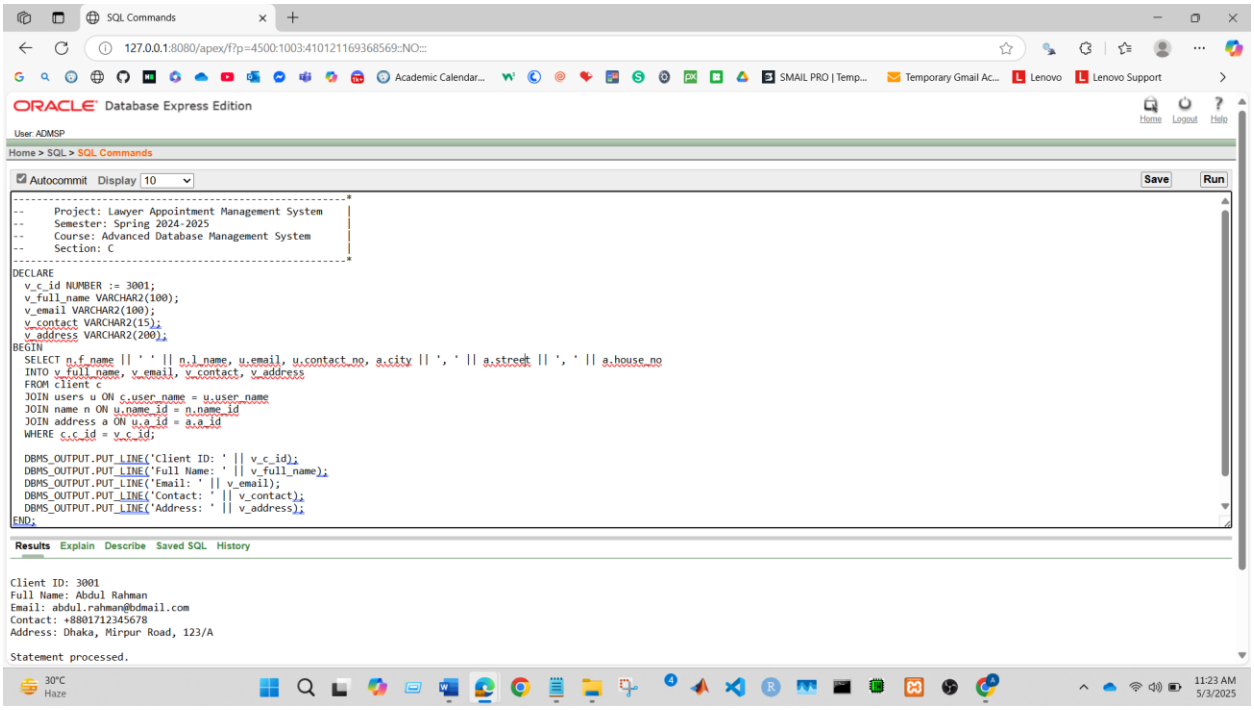
```
------------------------------------------------------*
--      Project: Lawyer Appointment Management System    |
--      Semester: Spring 2024-2025                       |
--      Course: Advanced Database Management System      |
--      Section: C                                       |
------------------------------------------------------*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



2. Display detailed information for a clieny (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;
/
```

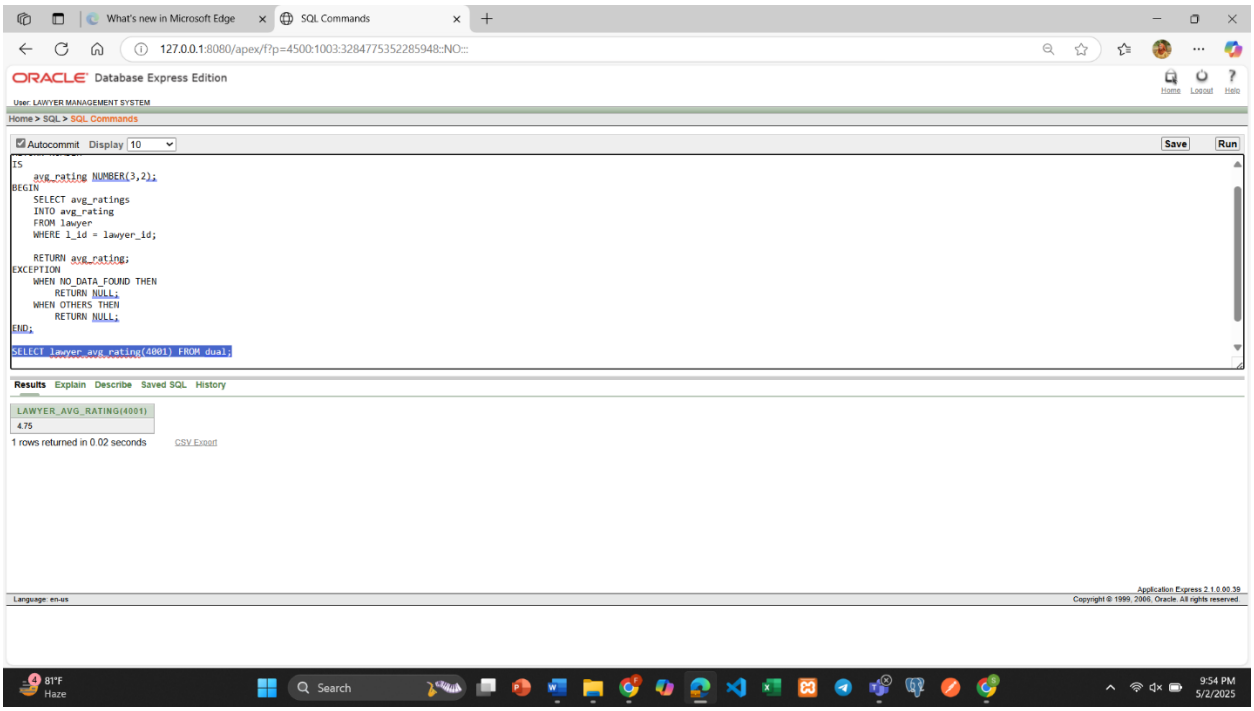


Advance PL/SQL

Stored Function:

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

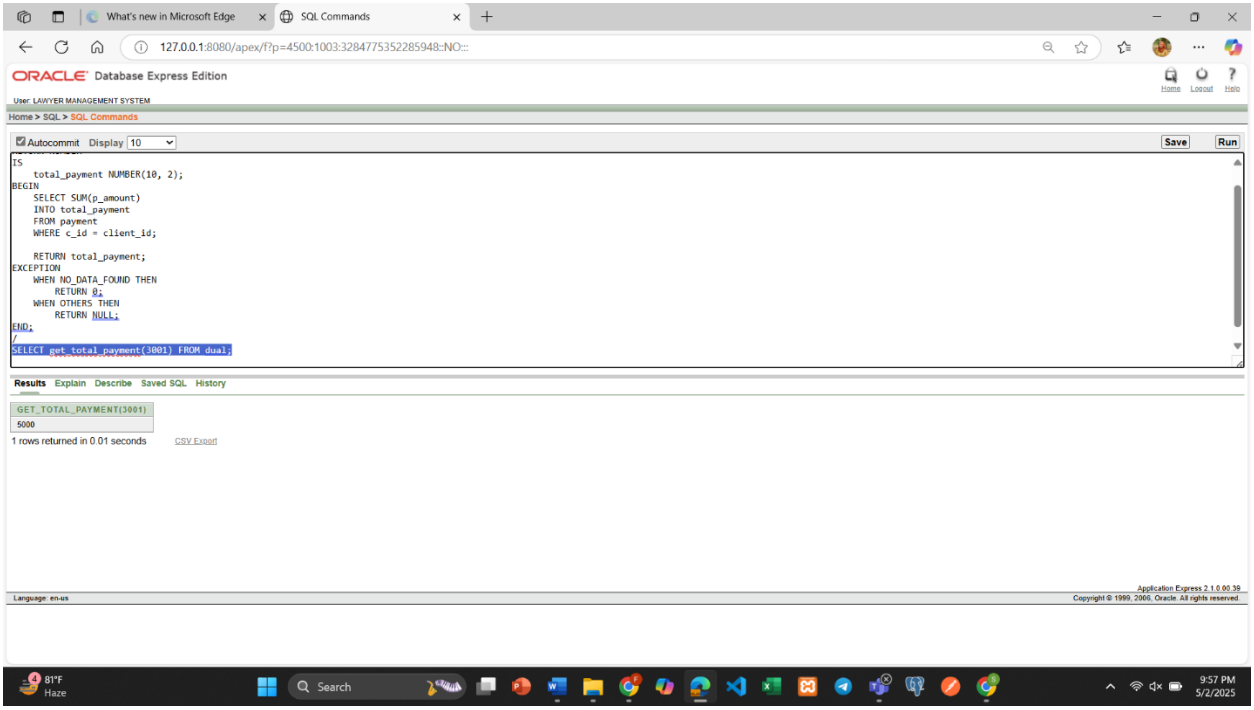
```
------------------------------------------------------*
--      Project: Lawyer Appointment Management System    |
--      Semester: Spring 2024-2025                       |
--      Course: Advanced Database Management System      |
--      Section: C                                       |
------------------------------------------------------*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



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;
```

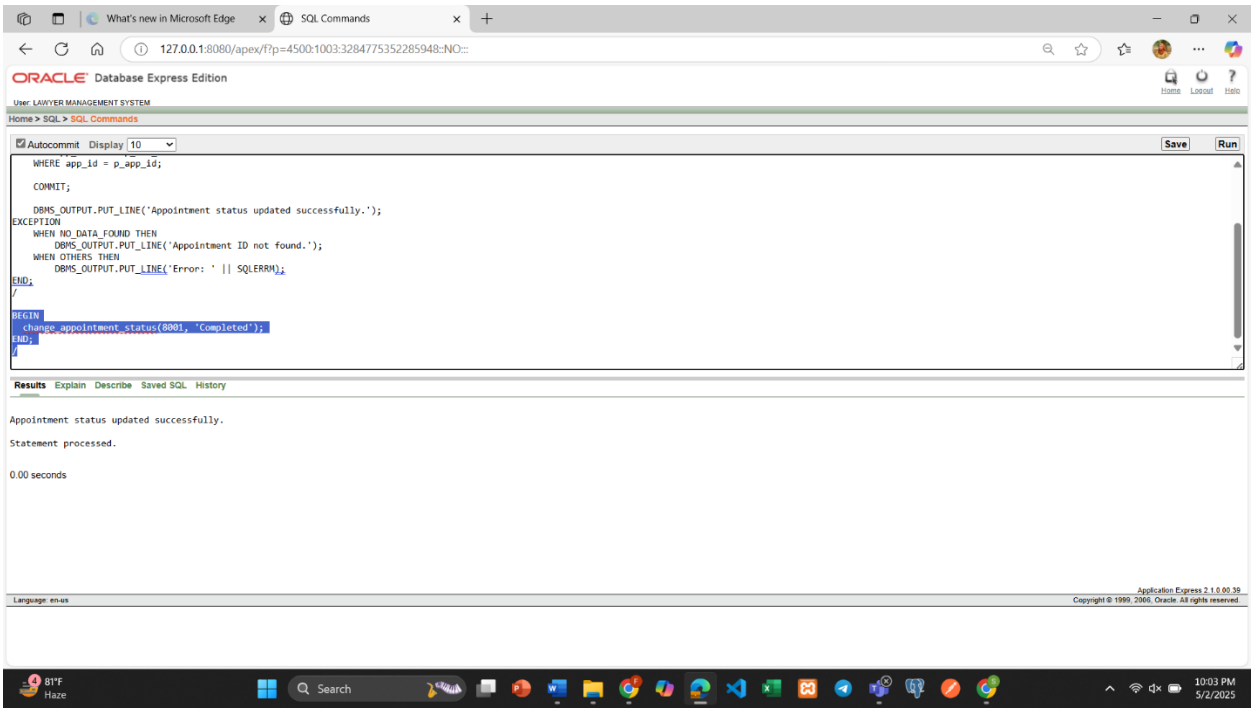


Stored Procedure:

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;
/
```



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;
/
```

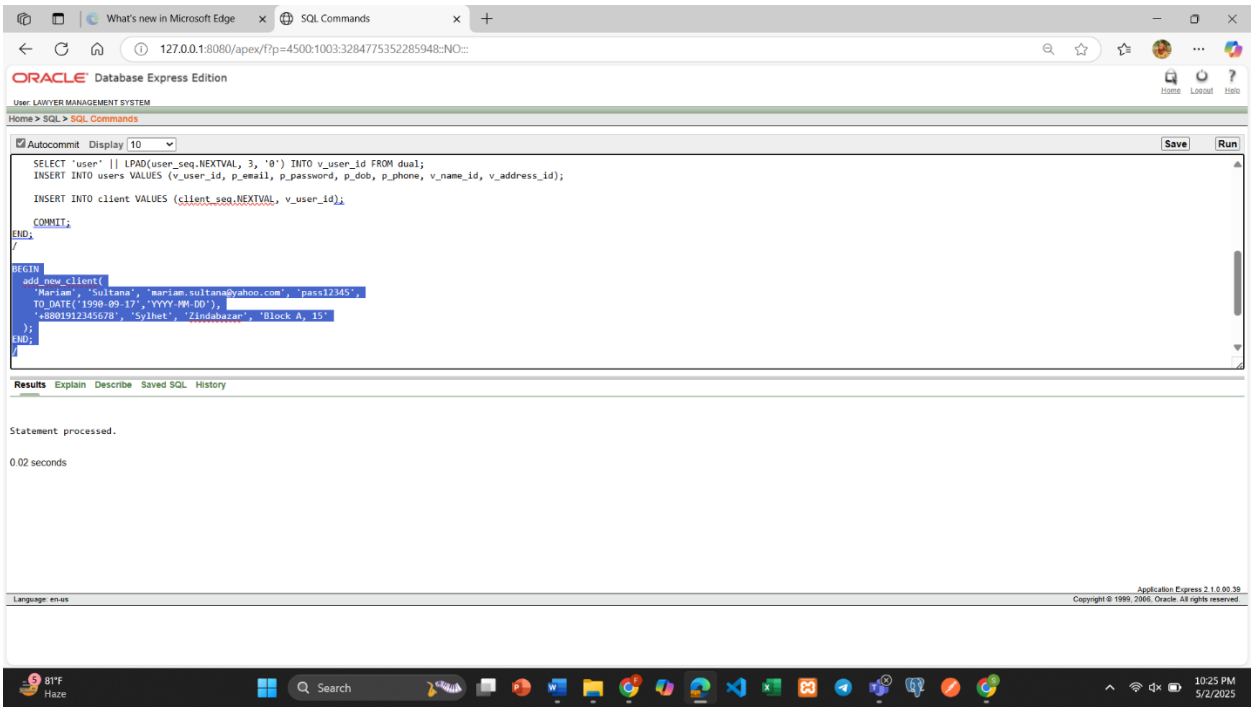
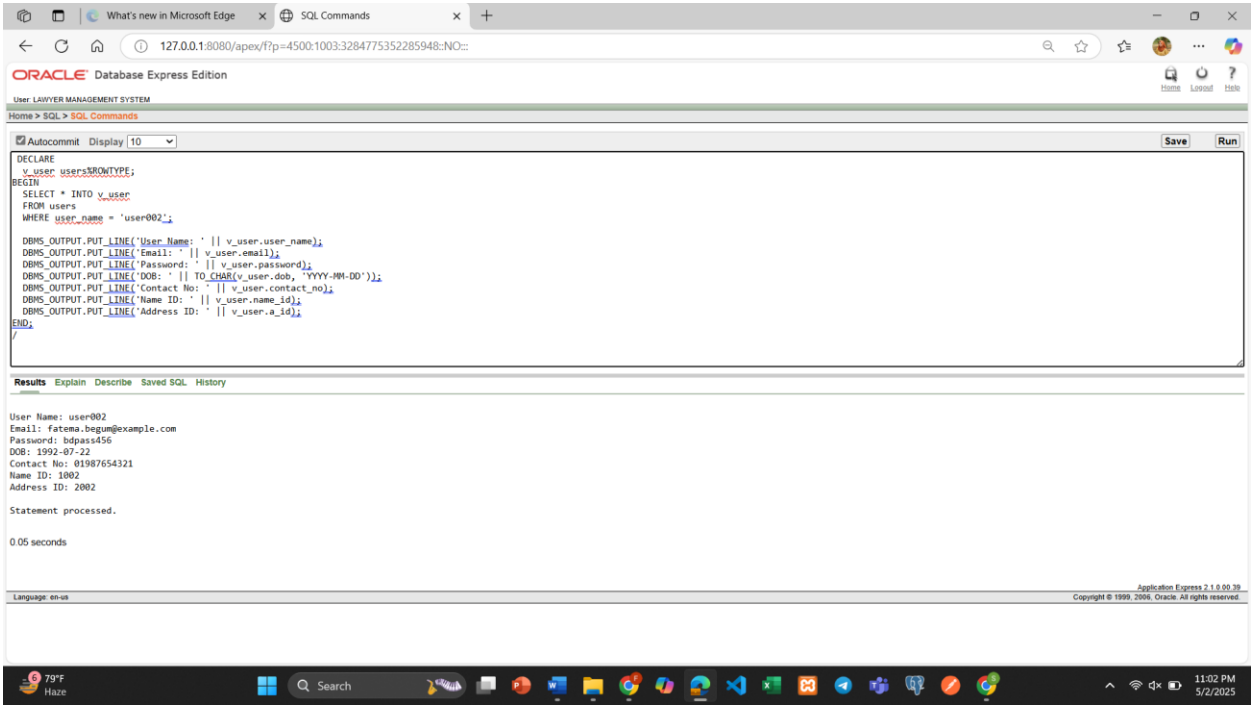


Table-Based Record:

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);
```

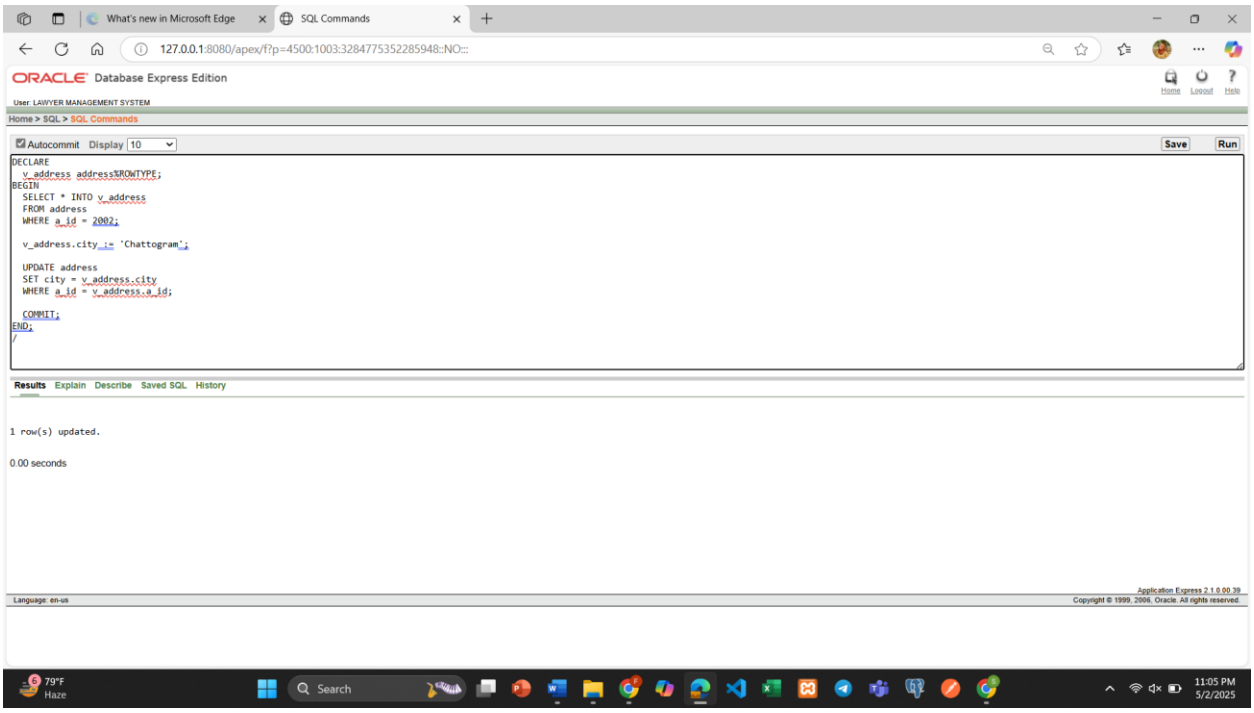
```
  DBMS_OUTPUT.PUT_LINE('Name ID: ' || v_user.name_id);
  DBMS_OUTPUT.PUT_LINE('Address ID: ' || v_user.a_id);
END;
```



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;
/
```

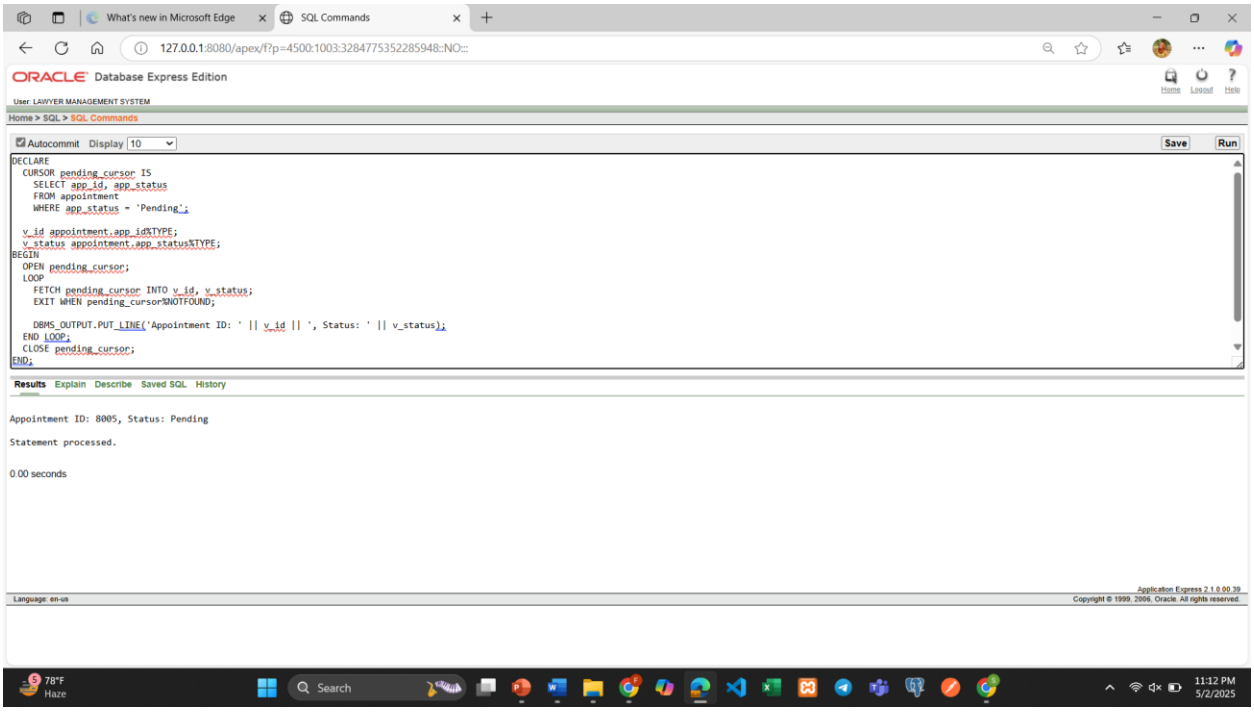


Explicit Cursor:

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;
```

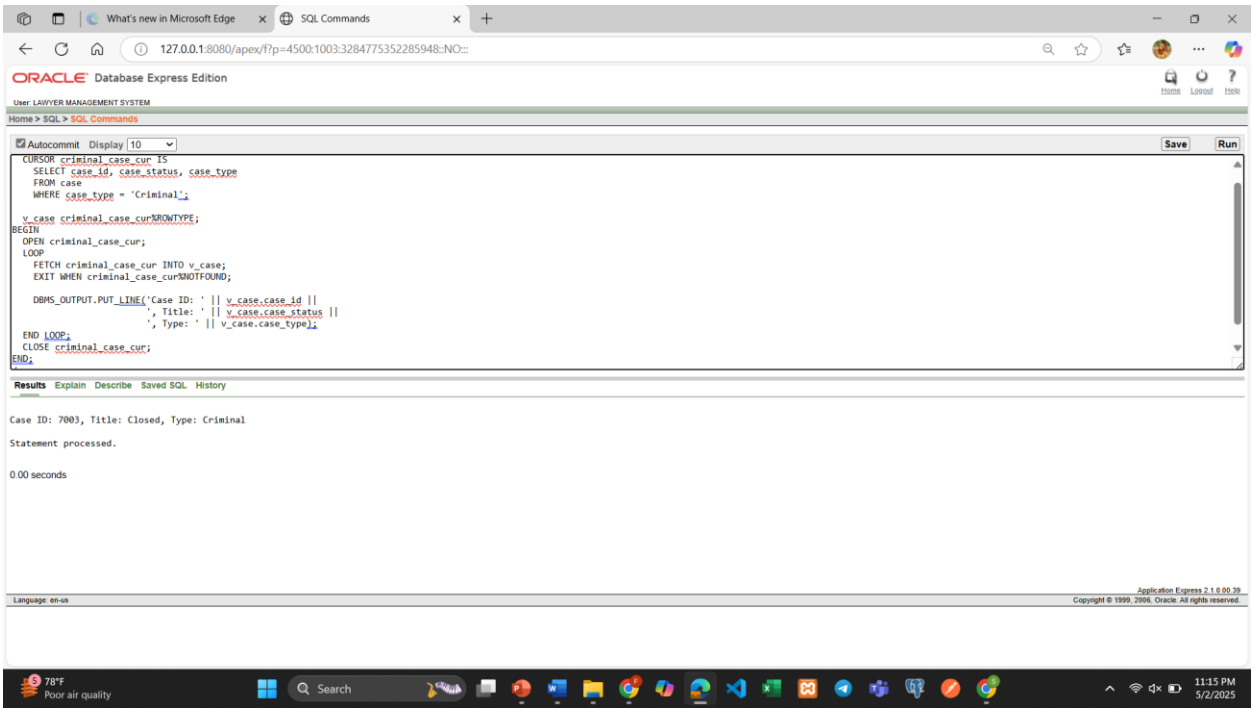
```
  v_status appointment.app_status%TYPE;
BEGIN
  OPEN pending_cursor;
  LOOP
    FETCH pending_cursor INTO v_id, v_status;
    EXIT WHEN pending_cursor%NOTFOUND;

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



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
```

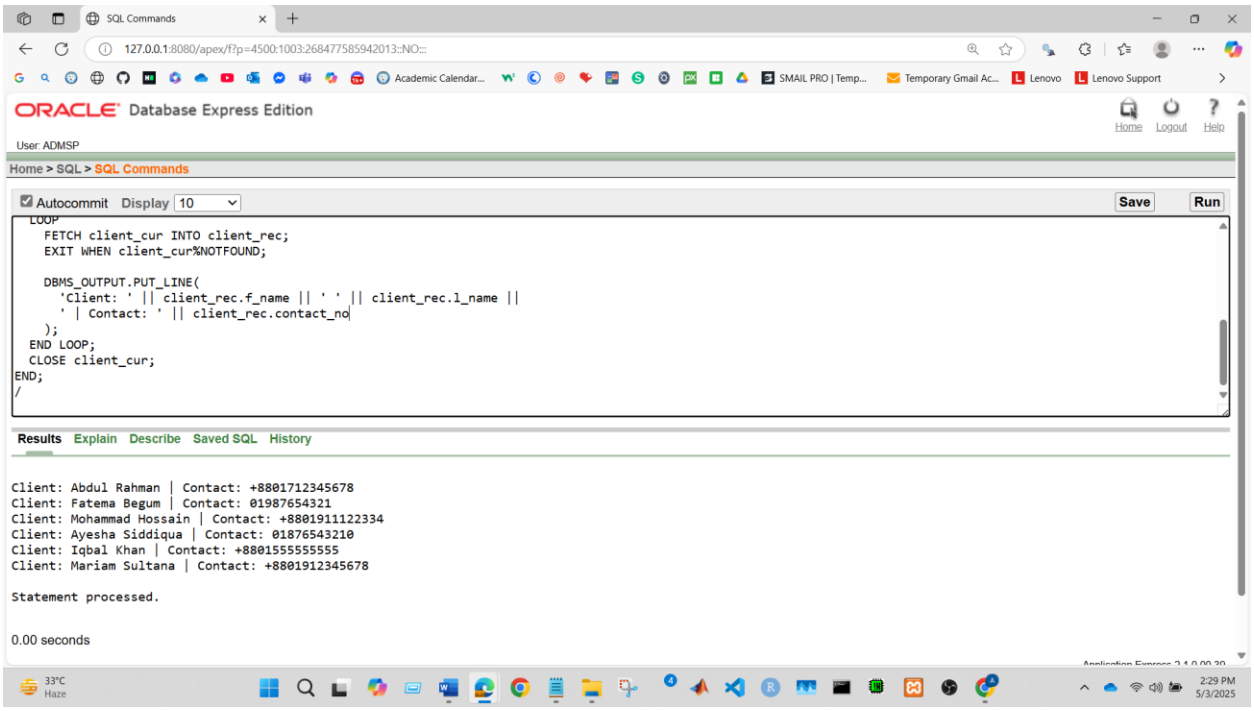
```
  CURSOR criminal_case_cur IS
    SELECT case_id, case_status, case_type
    FROM case
    WHERE case_type = 'Criminal';

  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;
/
```



Cursor-Based Record:

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

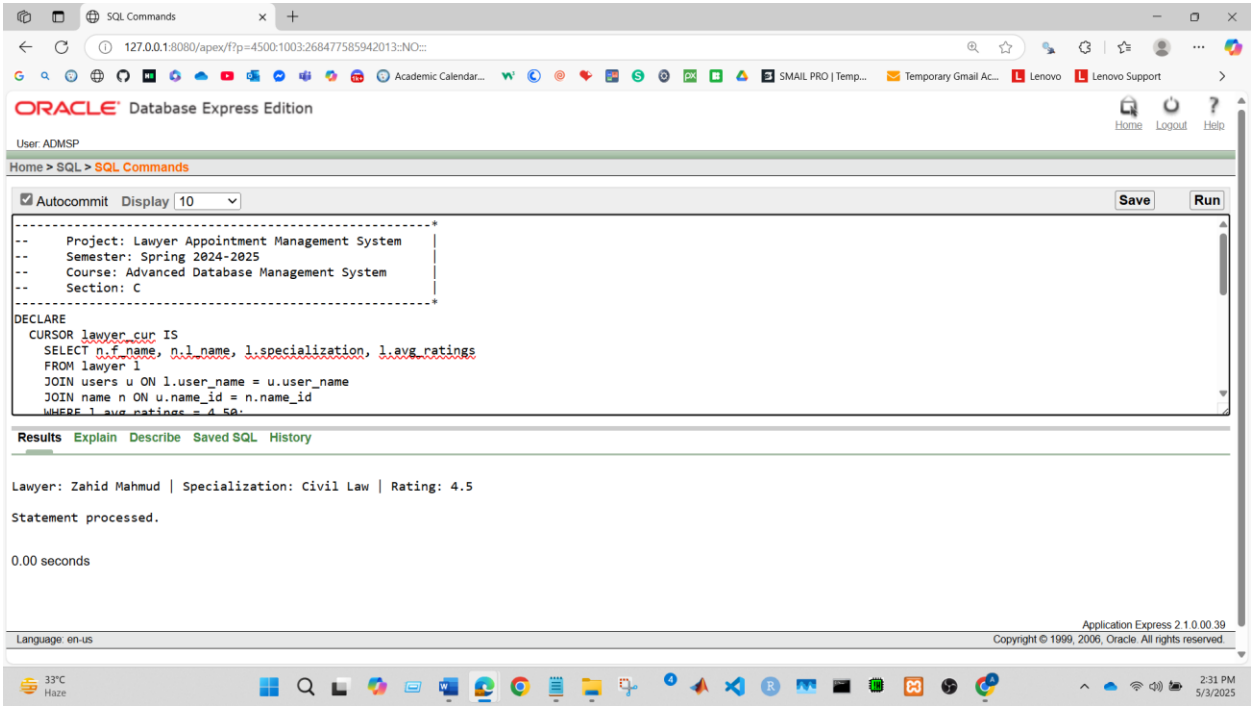
```
------------------------------------------------------------*
--     Project: Lawyer Appointment Management System       |
--     Semester: Spring 2024-2025                          |
--     Course: Advanced Database Management System         |
--     Section: C                                  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



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;
```

```
    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;
/
```

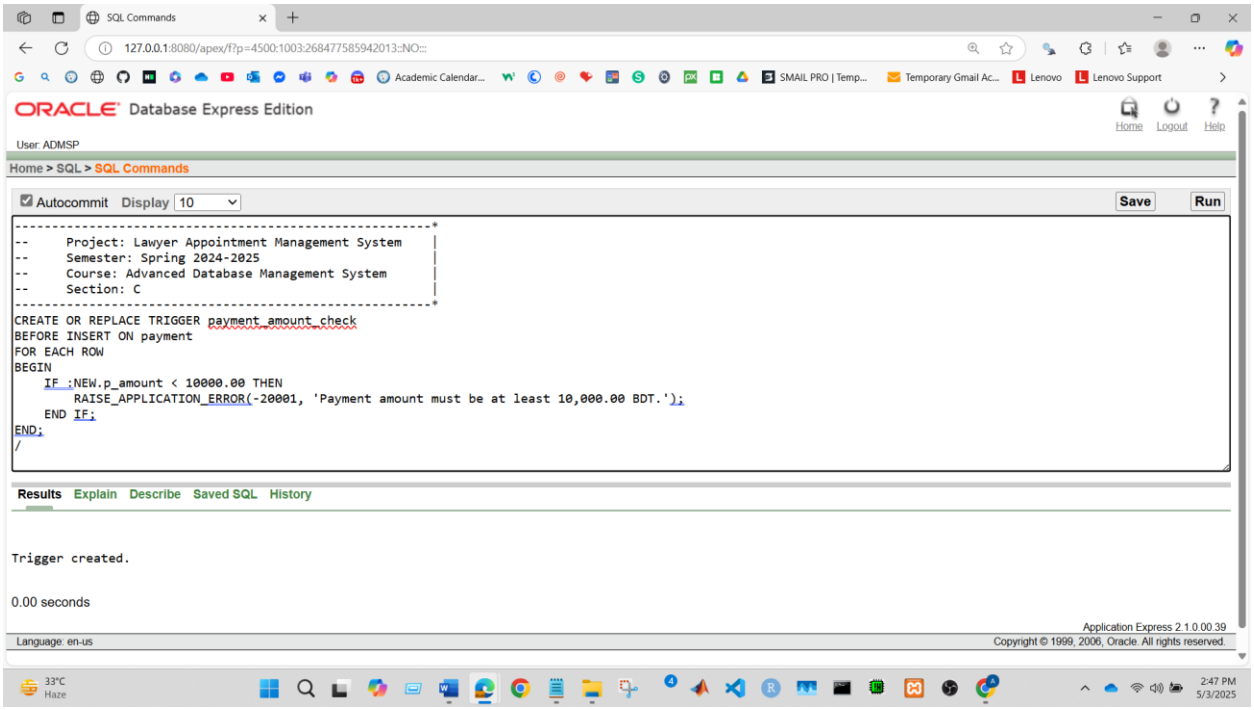


Row-Level Trigger:

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
```

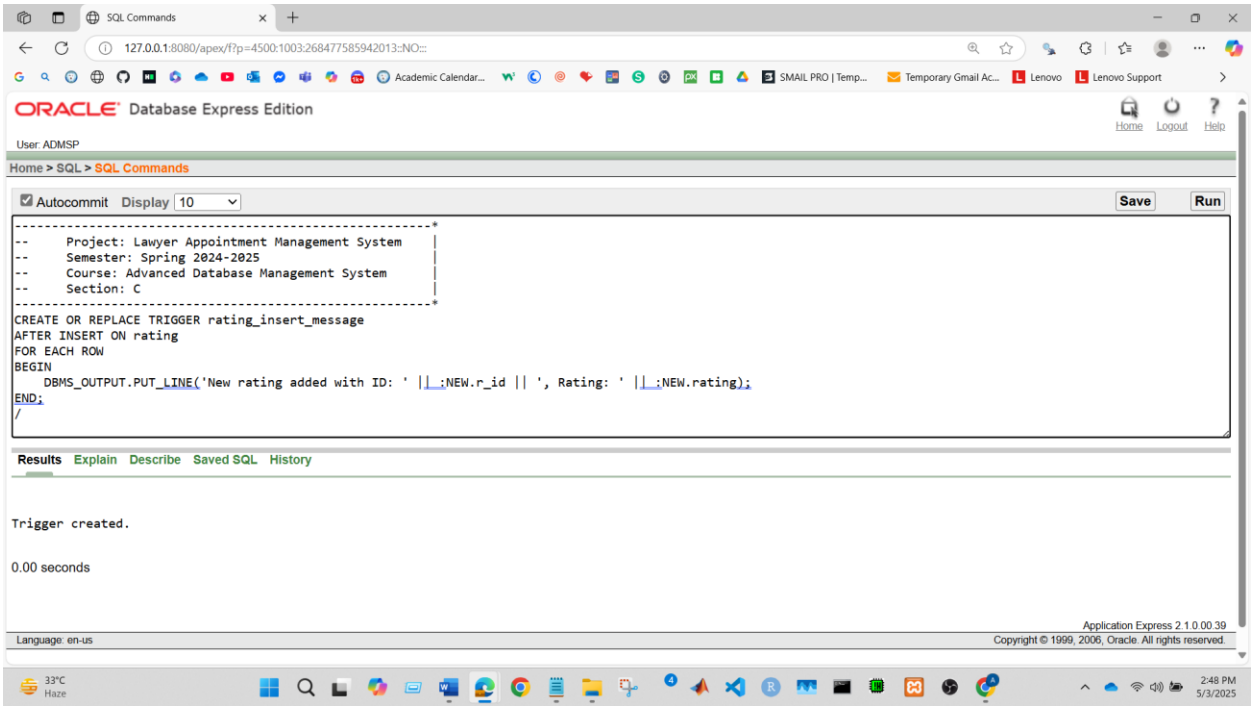
```
BEFORE INSERT ON payment
FOR EACH ROW
BEGIN
    IF :NEW.p_amount < 10000.00 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Payment amount must be at least
10,000.00 BDT.');
    END IF;
END;
/
```



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
```

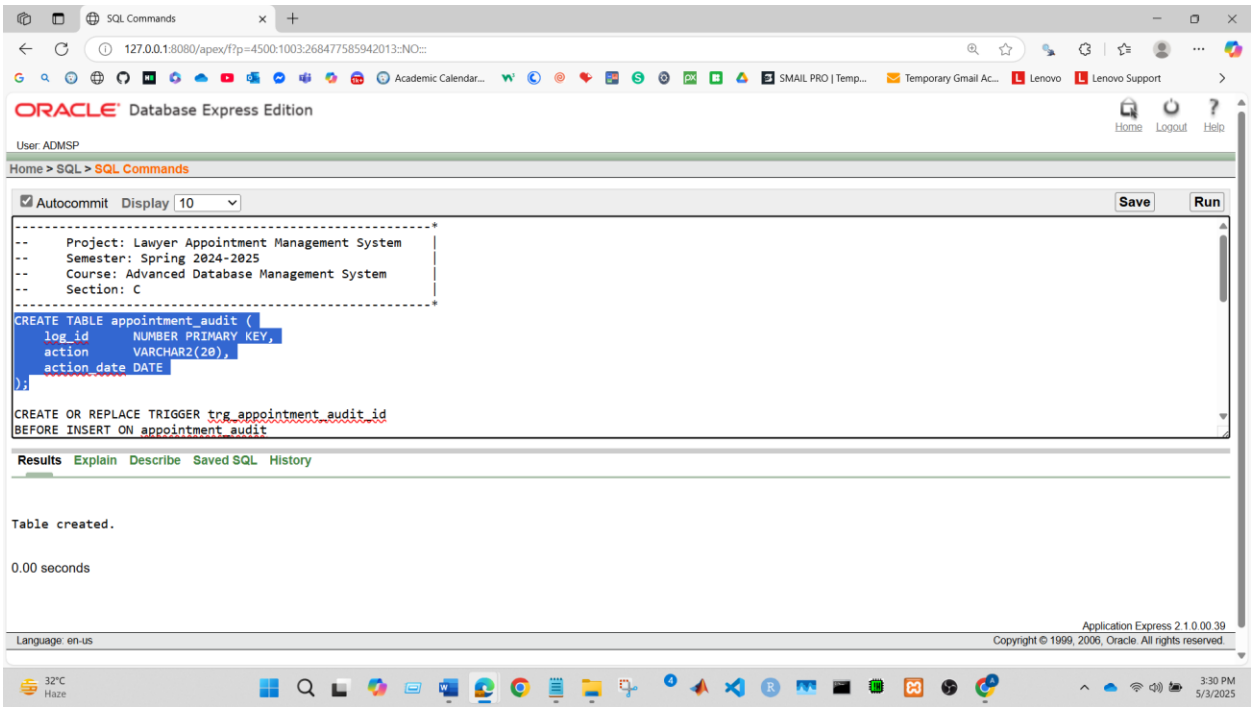
```
    DBMS_OUTPUT.PUT_LINE('New rating added with ID: ' || :NEW.r_id || ',
Rating: ' || :NEW.rating);
END;
/
```



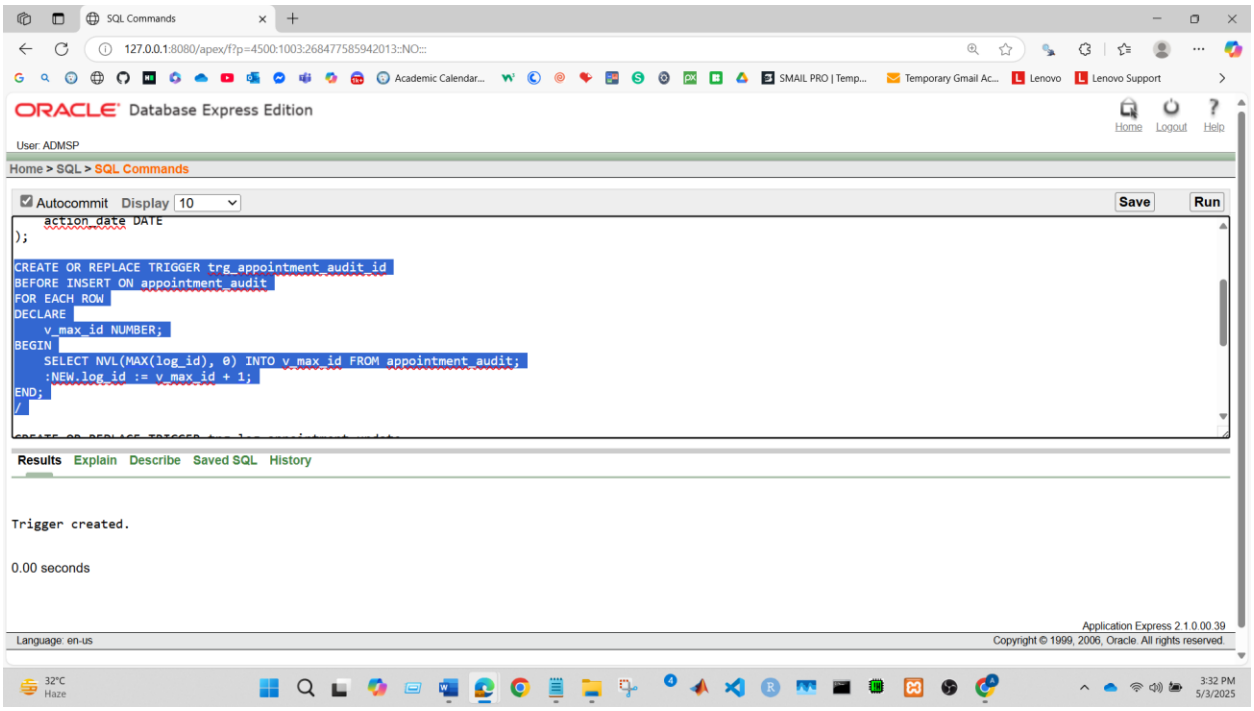
Statement-Level Trigger:

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

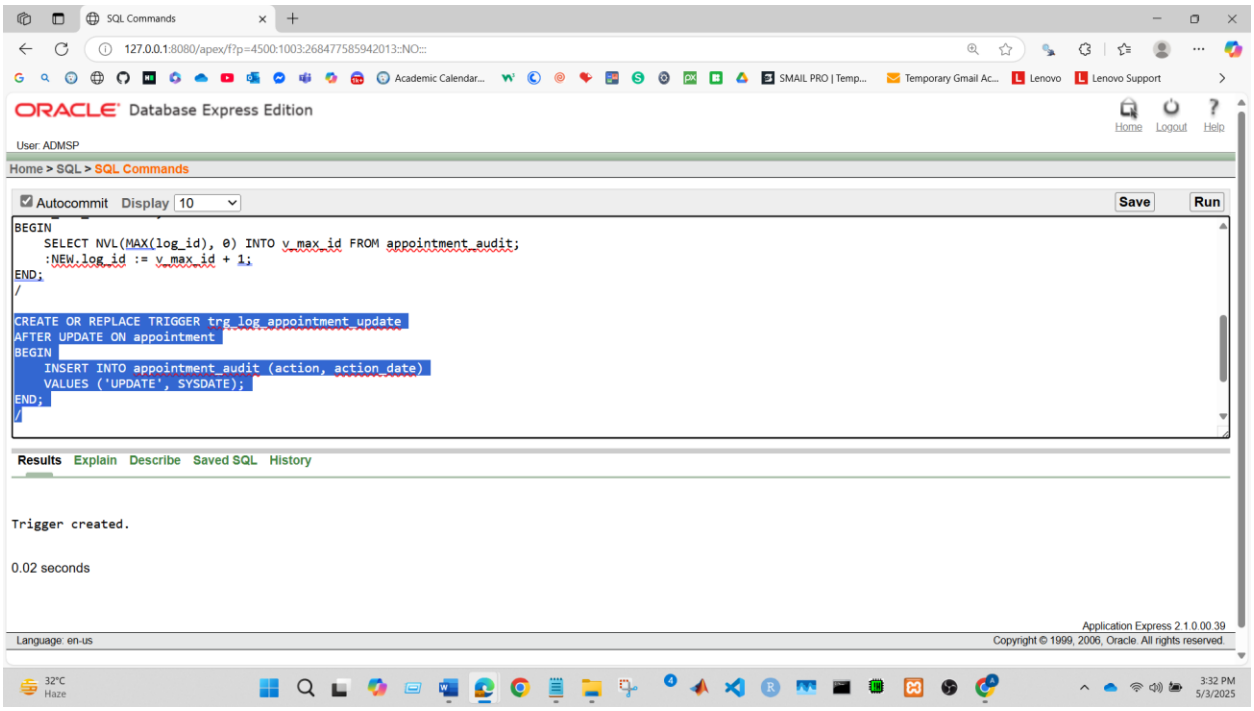
```
-------------------------------------------------------*
--      Project: Lawyer Appointment Management System    |
--      Semester: Spring 2024-2025                       |
--      Course: Advanced Database Management System      |
--      Section: C                                       |
-------------------------------------------------------*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



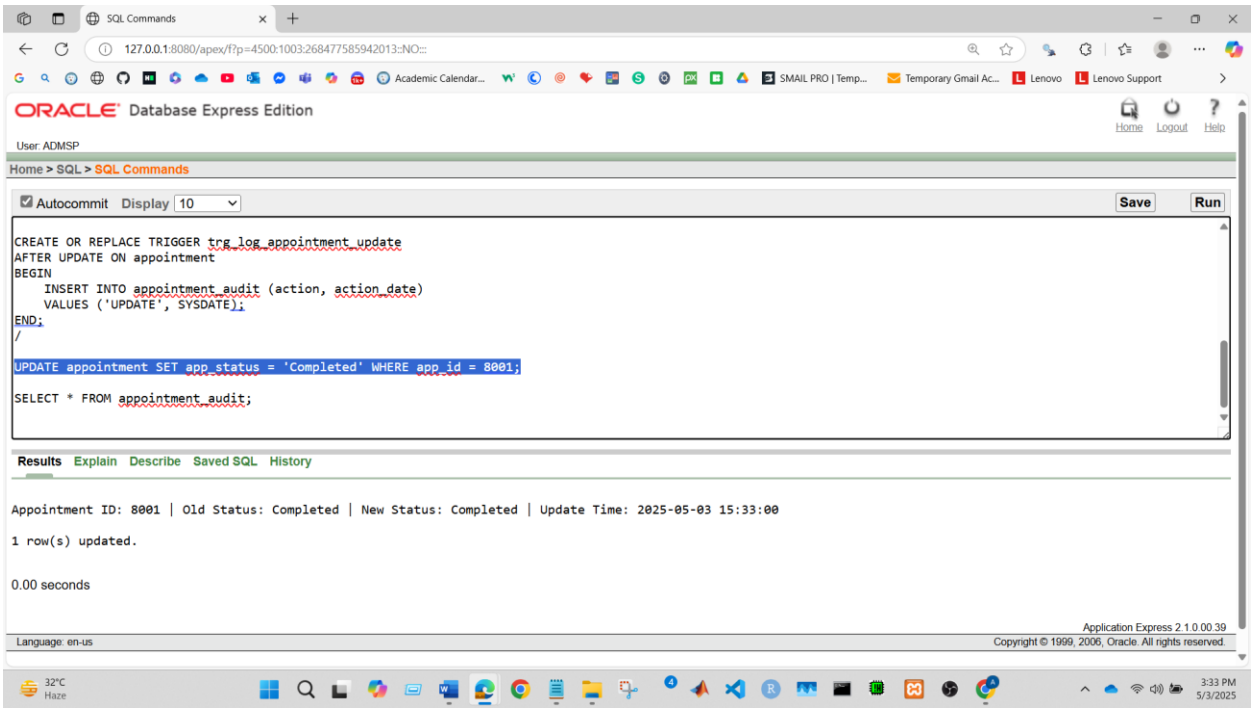

```
CREATE OR REPLACE TRIGGER trg_appointment_audit_id
BEFORE INSERT ON appointment_audit
FOR EACH ROW
DECLARE
    v_max_id NUMBER;
BEGIN
    SELECT NVL(MAX(log_id), 0) INTO v_max_id FROM appointment_audit;
    :NEW.log_id := v_max_id + 1;
END;
/
```



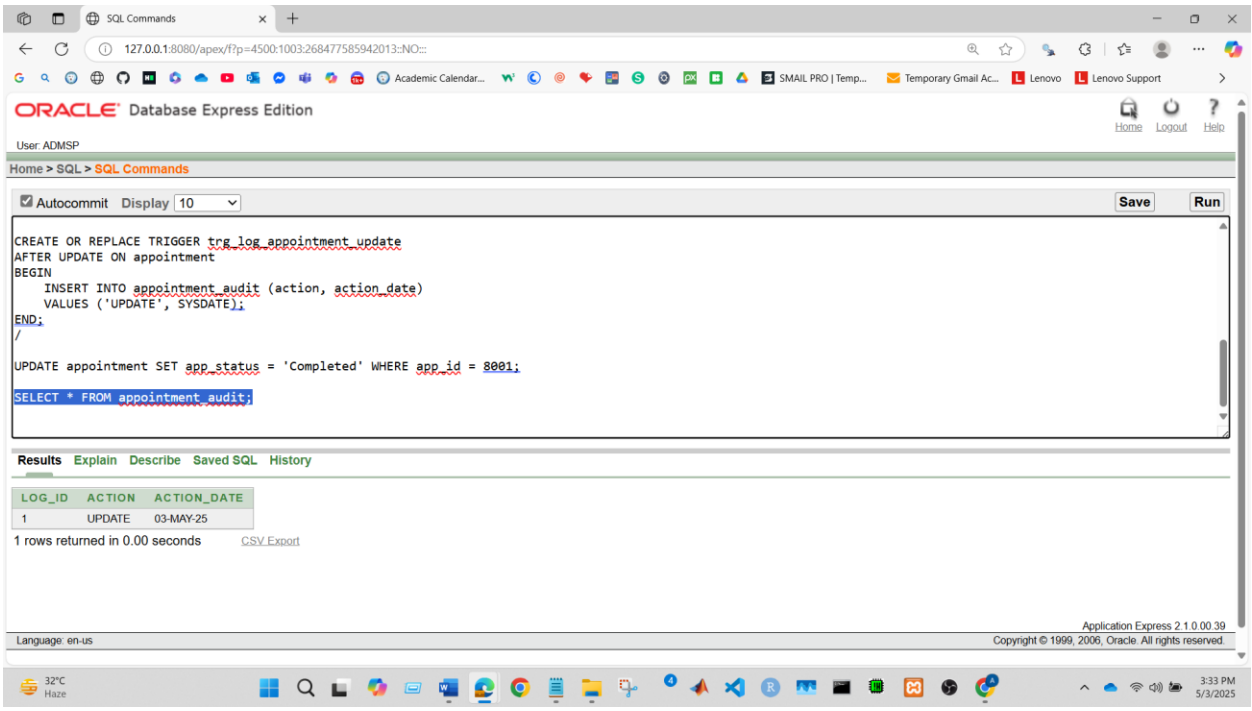

```
CREATE OR REPLACE TRIGGER trg_log_appointment_update
AFTER UPDATE ON appointment
BEGIN
    INSERT INTO appointment_audit (action, action_date)
    VALUES ('UPDATE', SYSDATE);
END;
/
```




```
UPDATE appointment SET app_status = 'Completed' WHERE app_id = 8001;
```

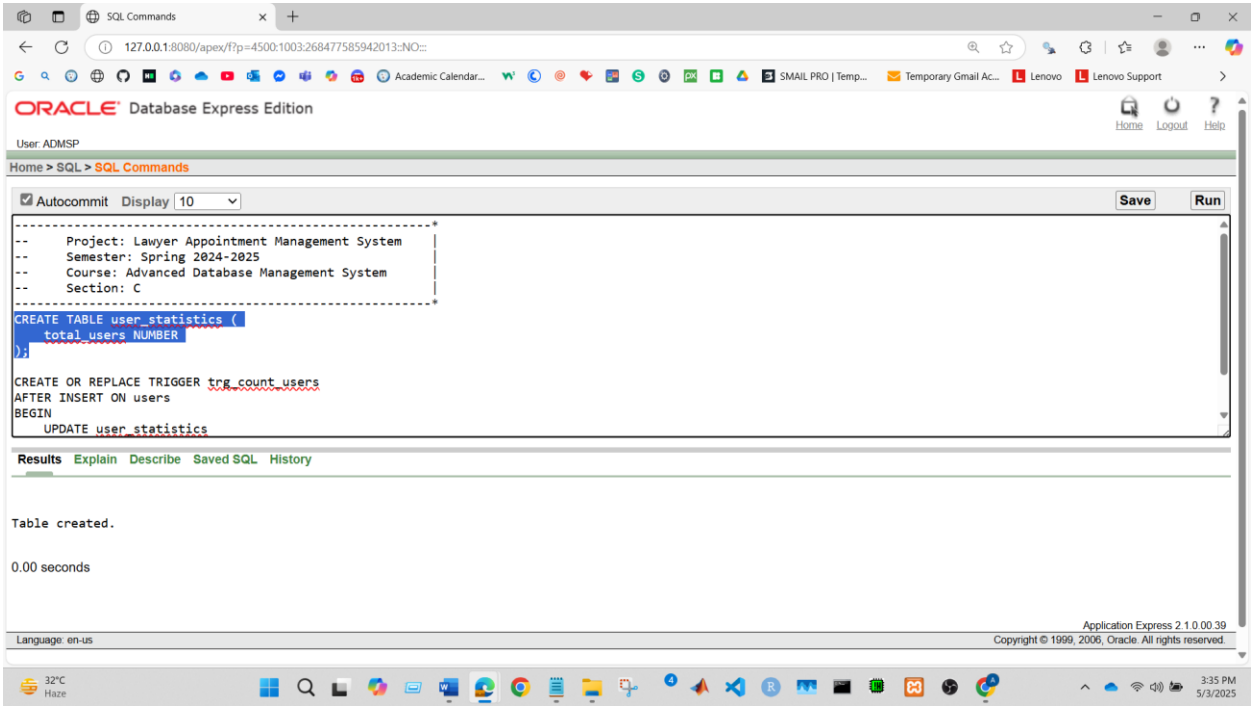



```
SELECT * FROM appointment_audit;
```



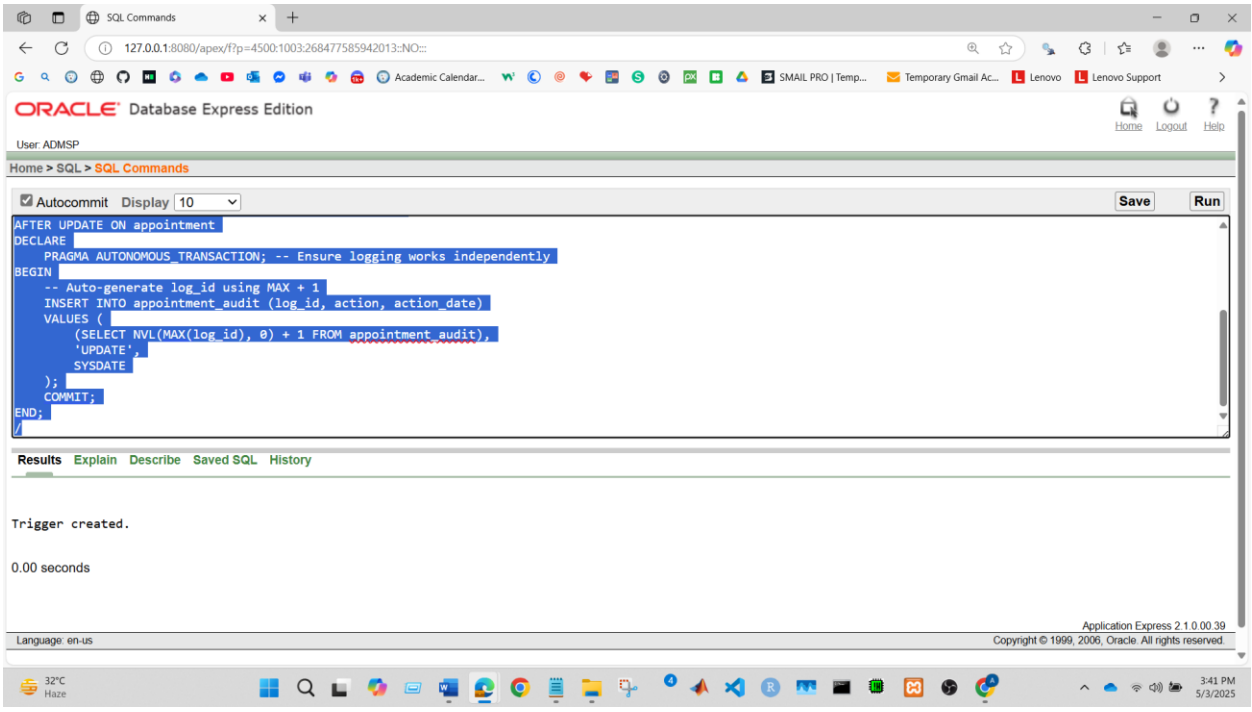
2. Make a trigger to count how many users are added after an insert from users table.

```
CREATE TABLE user_statistics (
    total_users NUMBER
);
```




```
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;
/
```

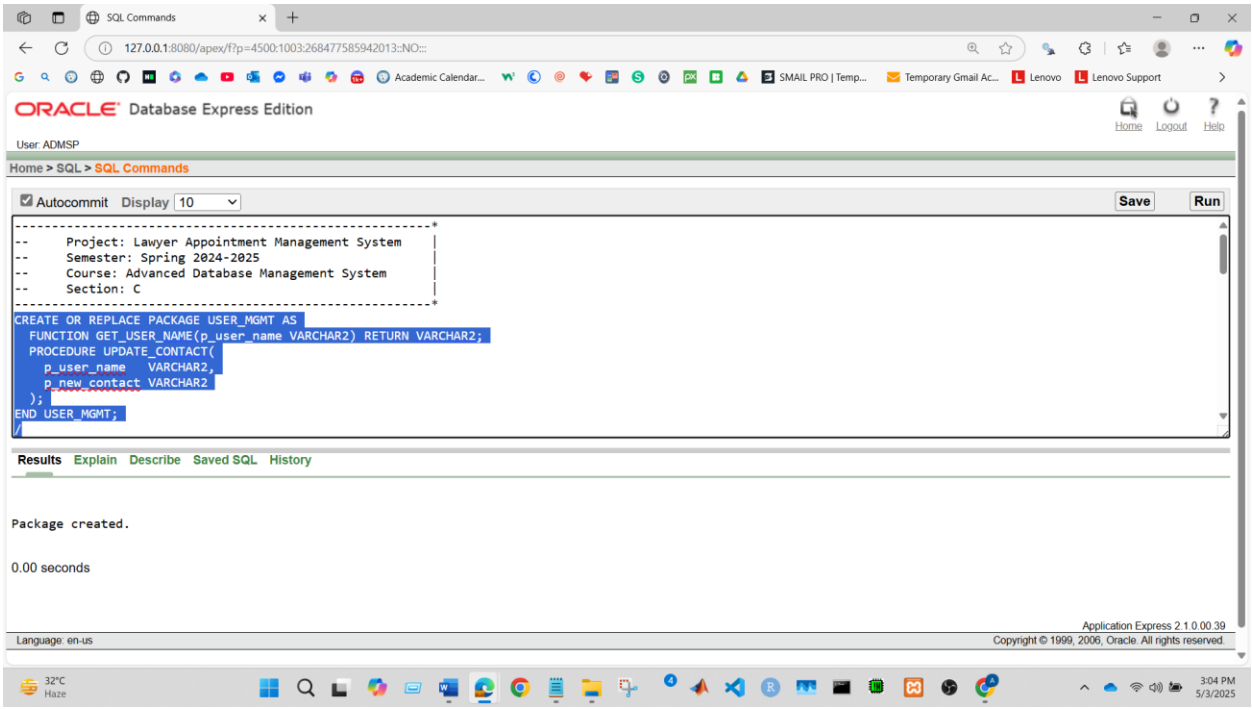


Package:

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

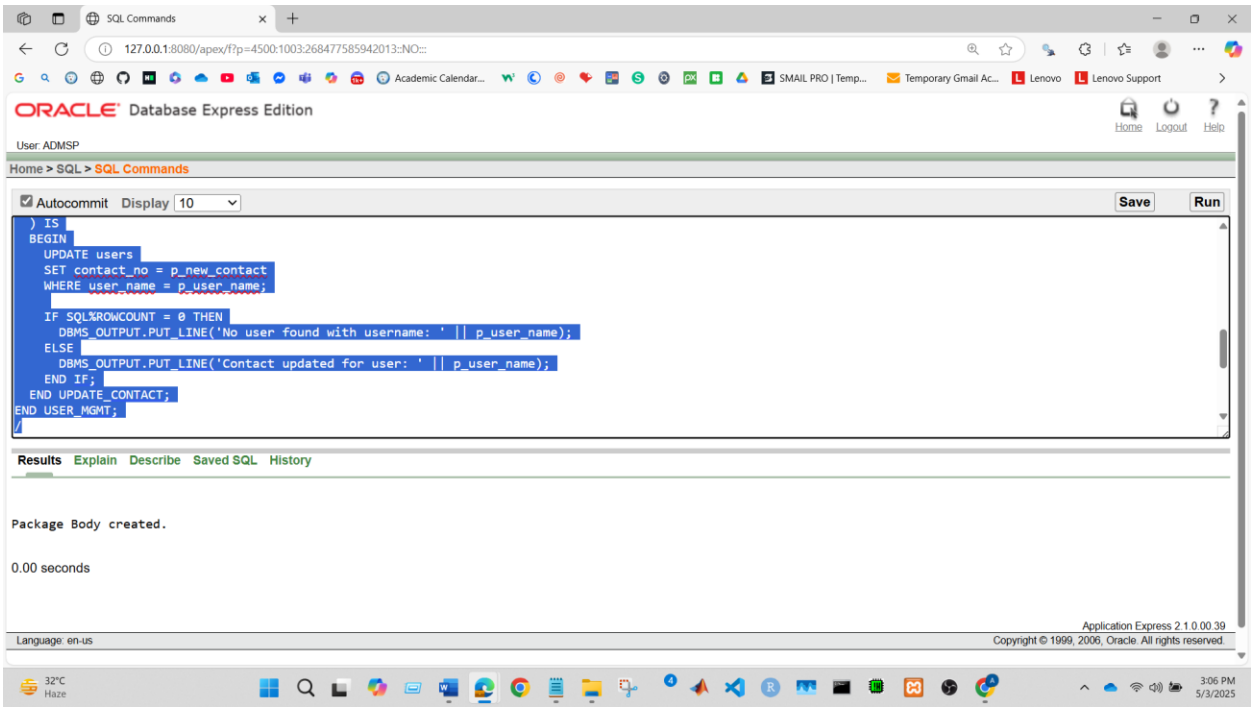
```
-------------------------------------------------------*
--      Project: Lawyer Appointment Management System    |
--      Semester: Spring 2024-2025                       |
```

```
--      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;
/
```

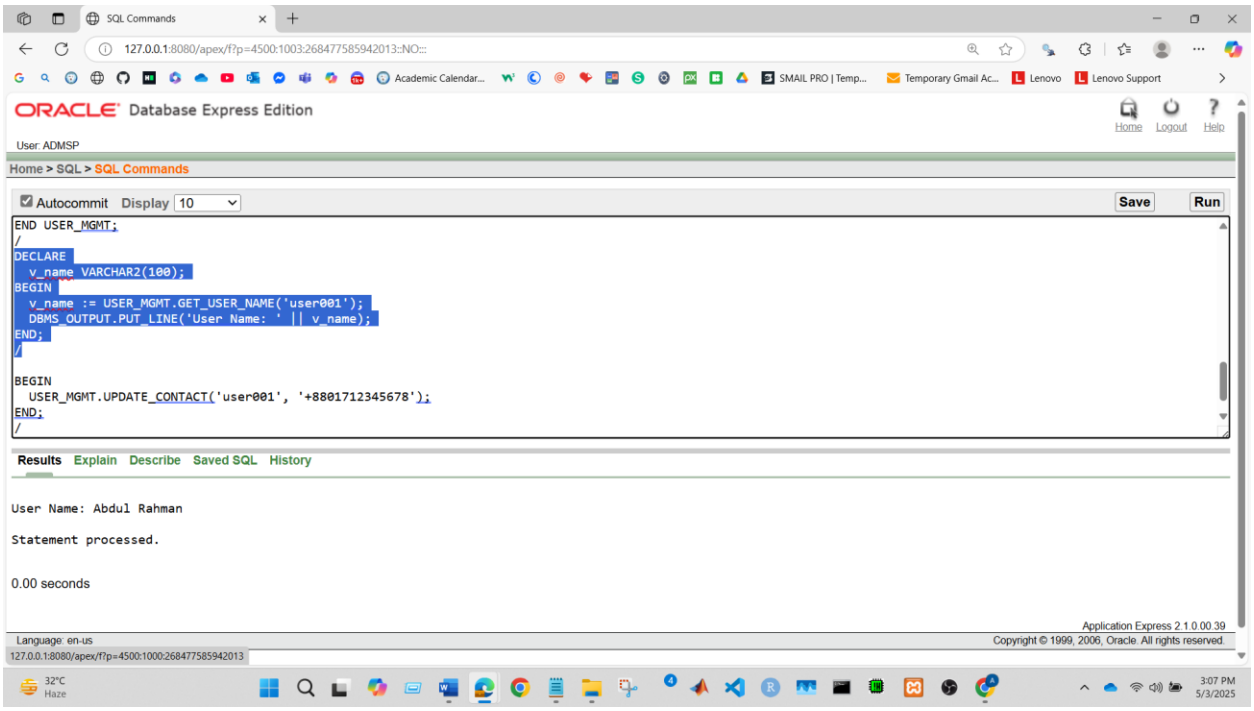



```
CREATE OR REPLACE PACKAGE BODY USER_MGMT AS
  FUNCTION GET_USER_NAME(p_user_name VARCHAR2) RETURN VARCHAR2 IS
    v_full_name VARCHAR2(100);
  BEGIN
    SELECT n.f_name || ' ' || n.l_name
    INTO v_full_name
    FROM users u
    JOIN name n ON u.name_id = n.name_id
    WHERE u.user_name = p_user_name;

    RETURN v_full_name;
```

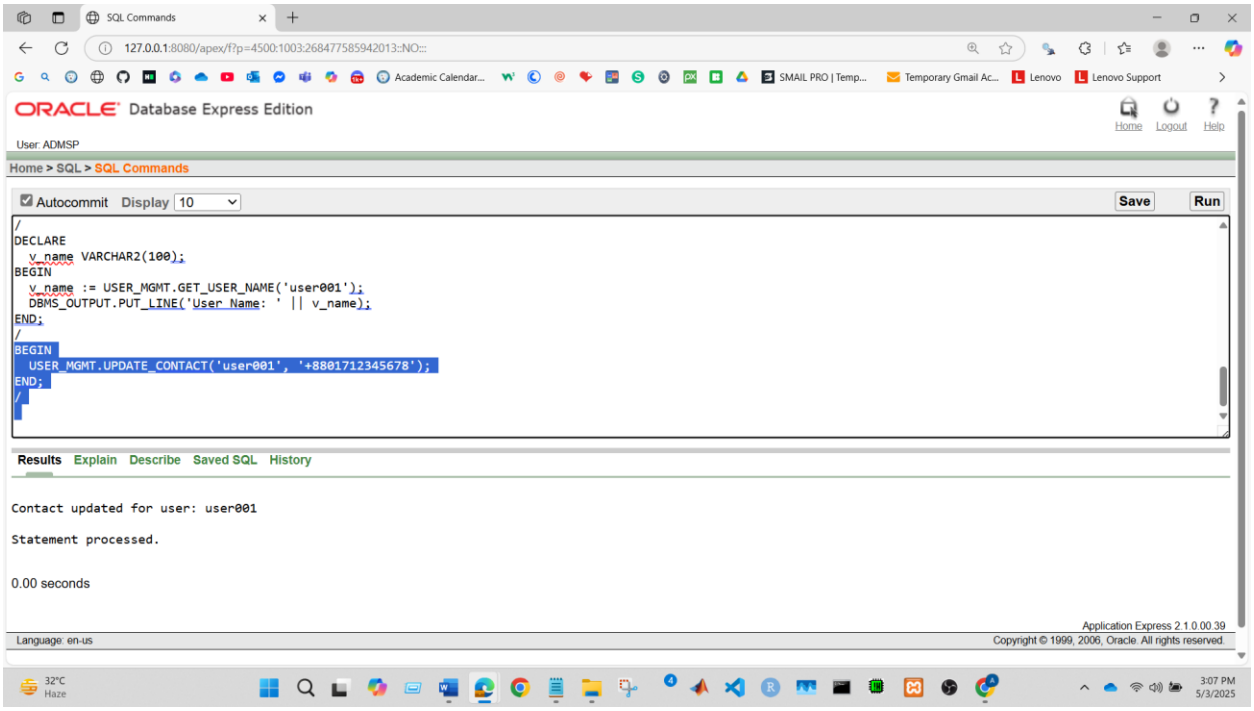
```
  EXCEPTION
    WHEN NO_DATA_FOUND THEN
      RETURN 'User not found.';
  END GET_USER_NAME;

  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;
/
```




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

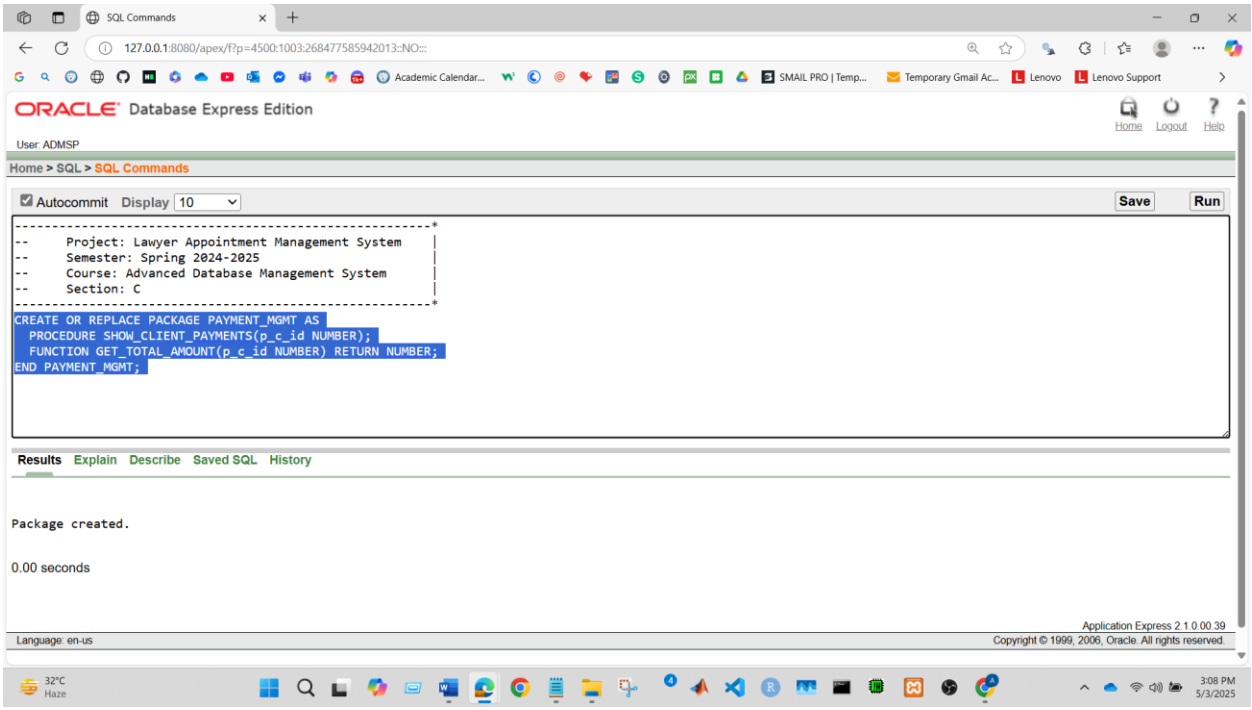



```
BEGIN
  USER_MGMT.UPDATE_CONTACT('user001', '+8801712345678');
END;
/
```



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.

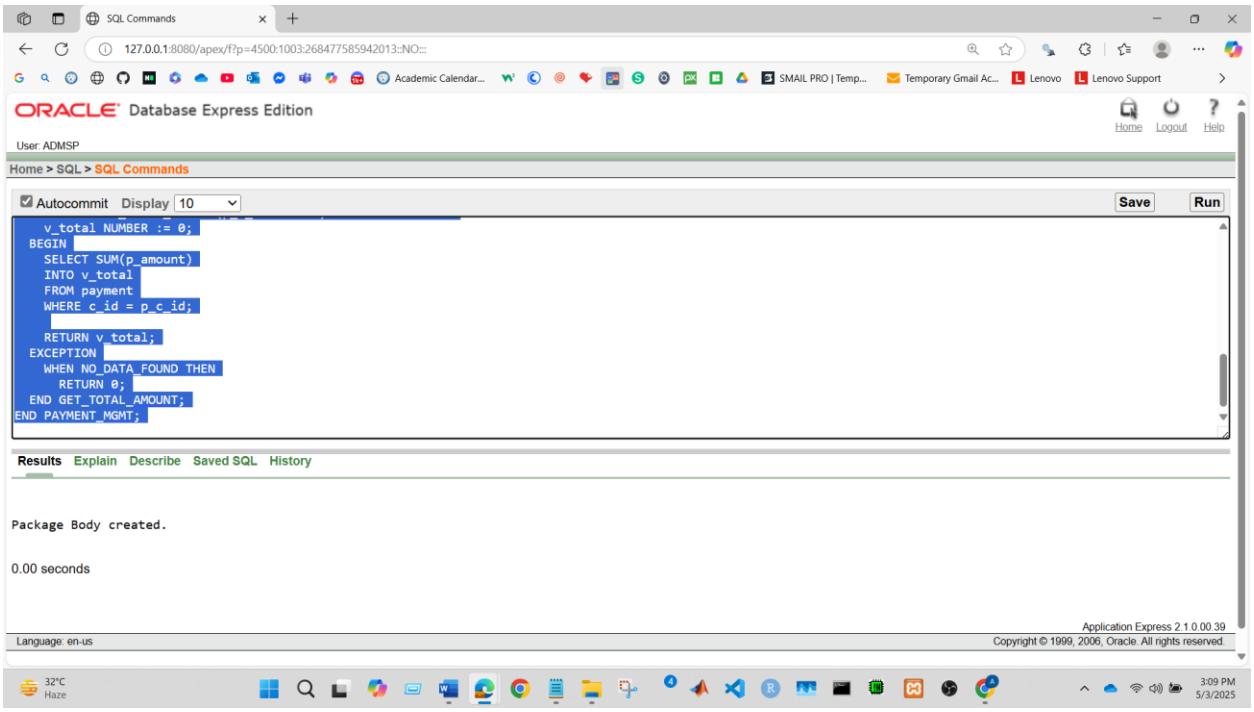
```
------------------------------------------------------------*
--      Project: Lawyer Appointment Management System      |
--      Semester: Spring 2024-2025                         |
--      Course: Advanced Database Management System        |
--      Section: C                                 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

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;
```




```
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**

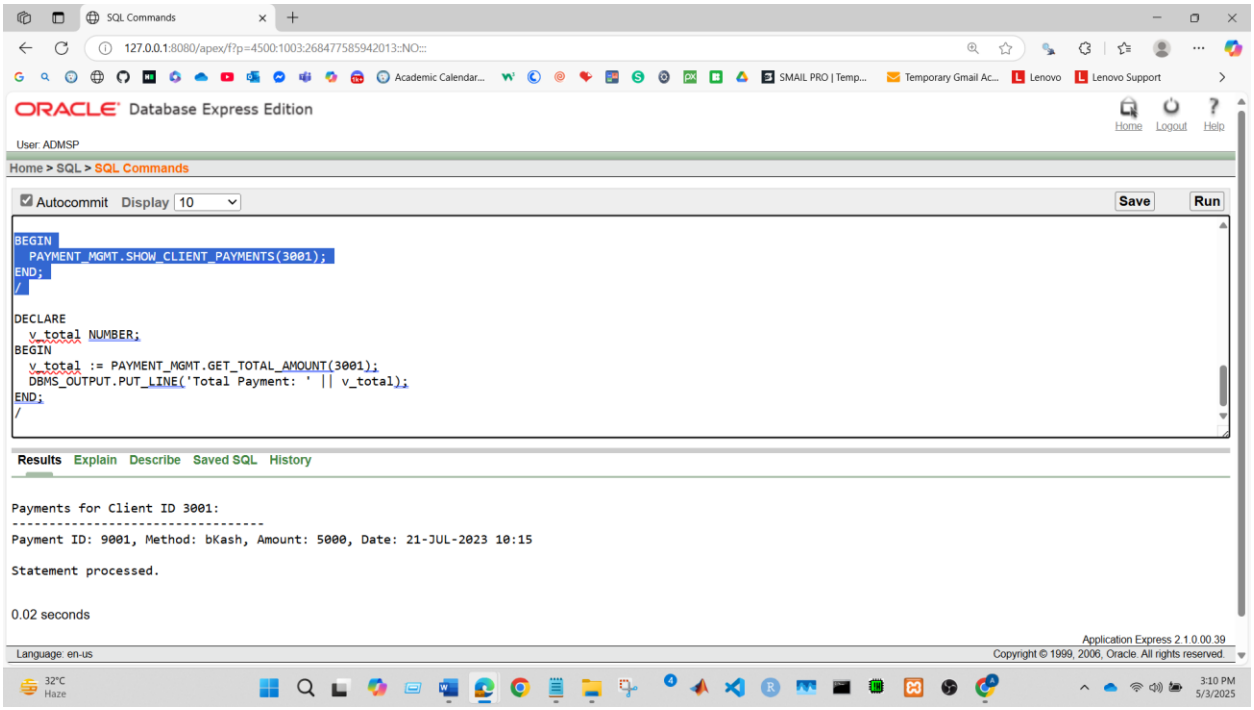


BEGIN

PAYMENT_MGMT.SHOW_CLIENT_PAYMENTS(3001);

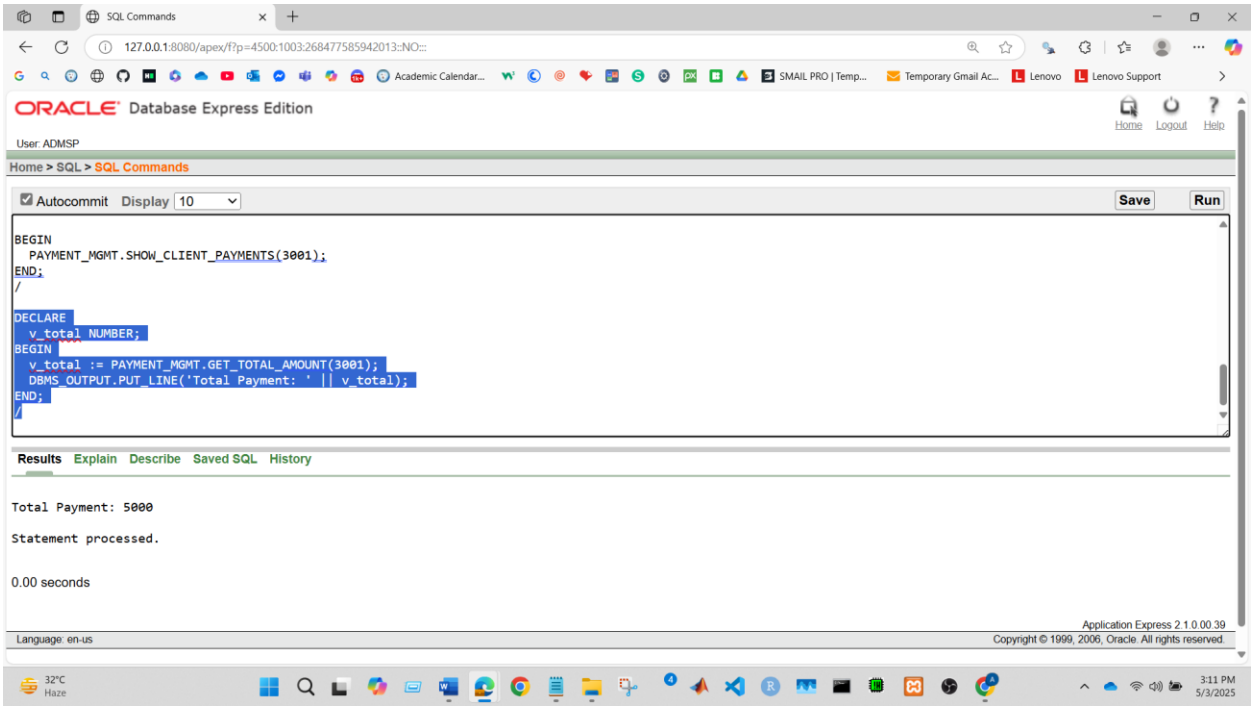
END;

/



DECLARE

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



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.