Name	ID	STUDENT SIGN
Sheikh Rubaeid Sanjid	21-45844-3	Rubaeid

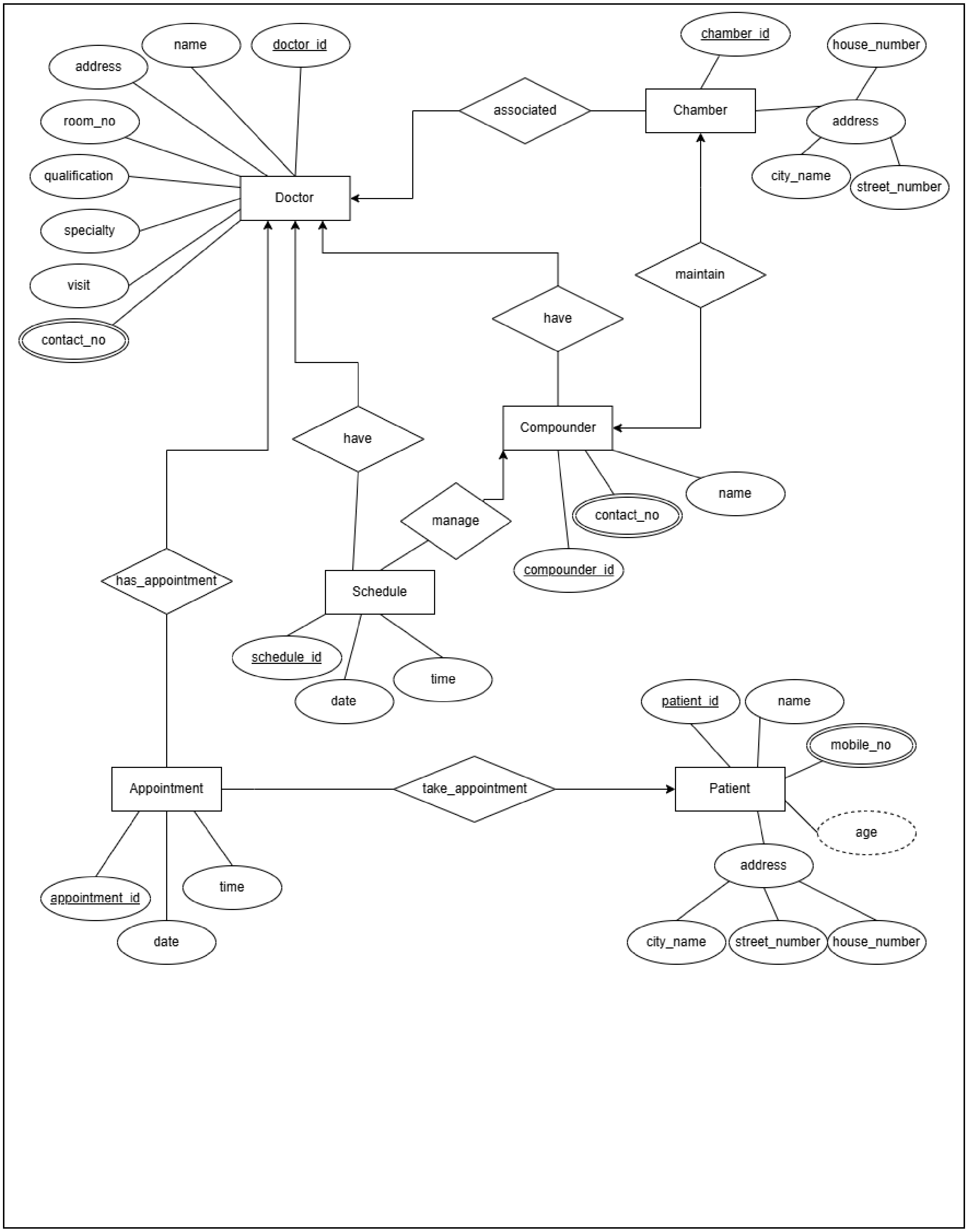
Instructions:

- **Make sure to write your Name, ID and Signature on this document.**
- **First write your signature on a paper then take photo of that signature and use it for signing this document.**
- **After completing the requirements of the midterm assignment by editing this document, upload this document in the link provided in your VUES Student Account.**

Midterm Assignment

1. **Below a scenario has been given draw the ER Diagram.**
   Draw with proper annotations (use DIA, VISIO, MS WORD etc.). ***For reference see ERDiagramTutorial.***
   In a doctor appointment management system, a patient takes appointment from a doctor and a doctor attend appointment of many patients. A doctor is identified by doctor id, name, address, room no, qualification, specialty, visit, and contact no. There may be multiple contact no of a doctor. A Patient is identified by their unique ID and the system stores their name, mobile number, age and address. A patient address is composed of city name, street number, and house number. One doctor can be associated with many chambers, but one chamber is associated with only one doctor and the system stores the id and address of each chamber, including the city name, street number, and house number. Additionally, one doctor has many compounders, but a compounder is associated with only one doctor and a compounder is identified by their ID, name, and contact numbers. A chamber is maintained by only one compounder and a compounder maintain only one chamber. While taking appointment, the date and time of appointment for each patient is also stored. Furthermore, one doctor can have many schedules, but each schedule is associated with only one doctor and the system stores the id, time and date for each schedule and a compounder can manage many schedules for doctor, but a schedule is managed by one compounder.

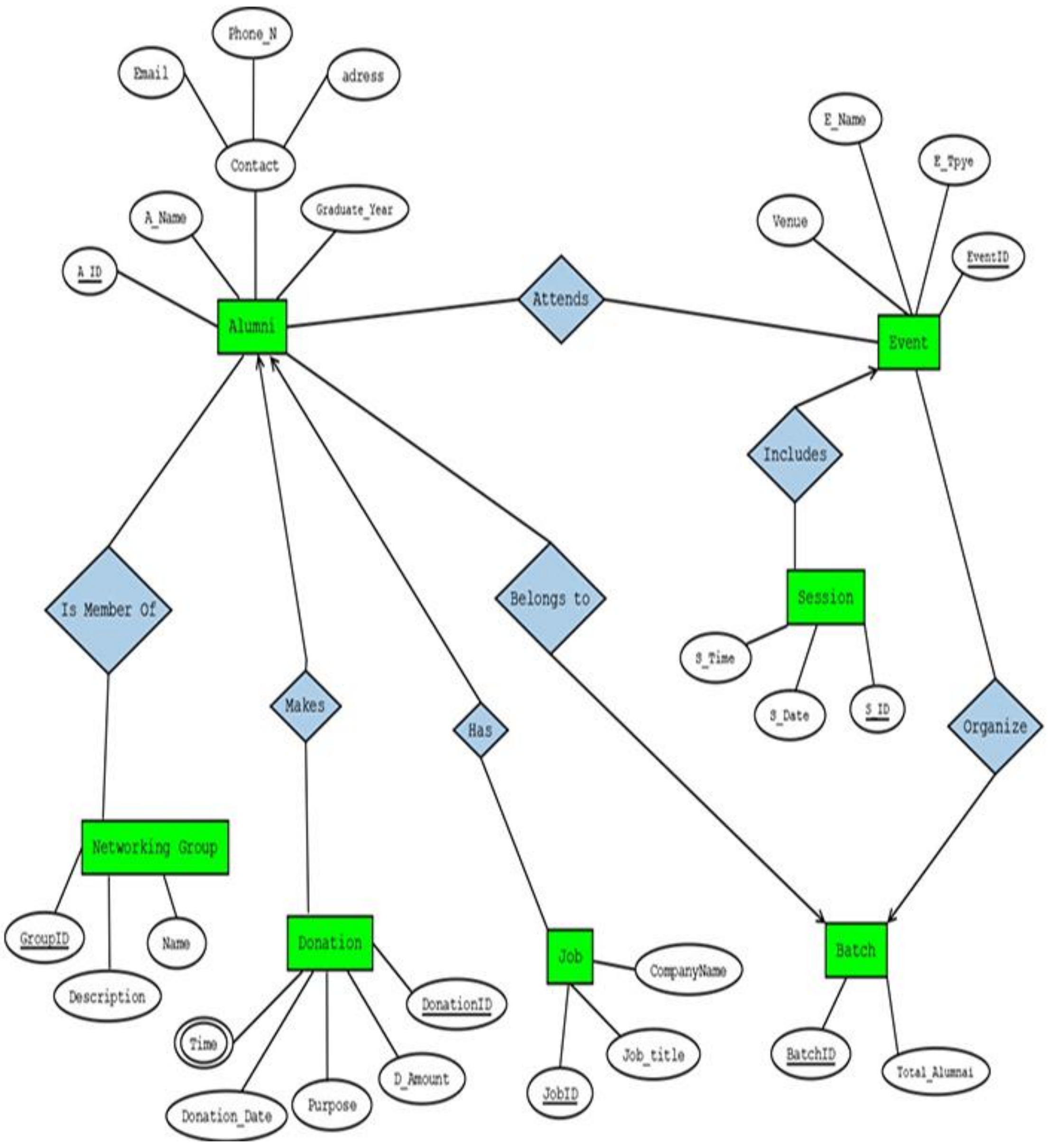
Answer 1:



address
name
doctor_id
room_no
qualification
specialty
visit
contact_no
Doctor
associated
chamber_id
house_number
Chamber
address
city_name
street_number
maintain
have
have
Compounder
name
contact_no
compounder_id
manage
has_appointment
Schedule
schedule_id
date
time
patient_id
name
mobile_no
Appointment
take_appointment
Patient
age
address
appointment_id
time
date
city_name
street_number
house_number

2. **Below an ER Diagram has been given write the scenario.**
   For reference see ERDiagramTutorial.

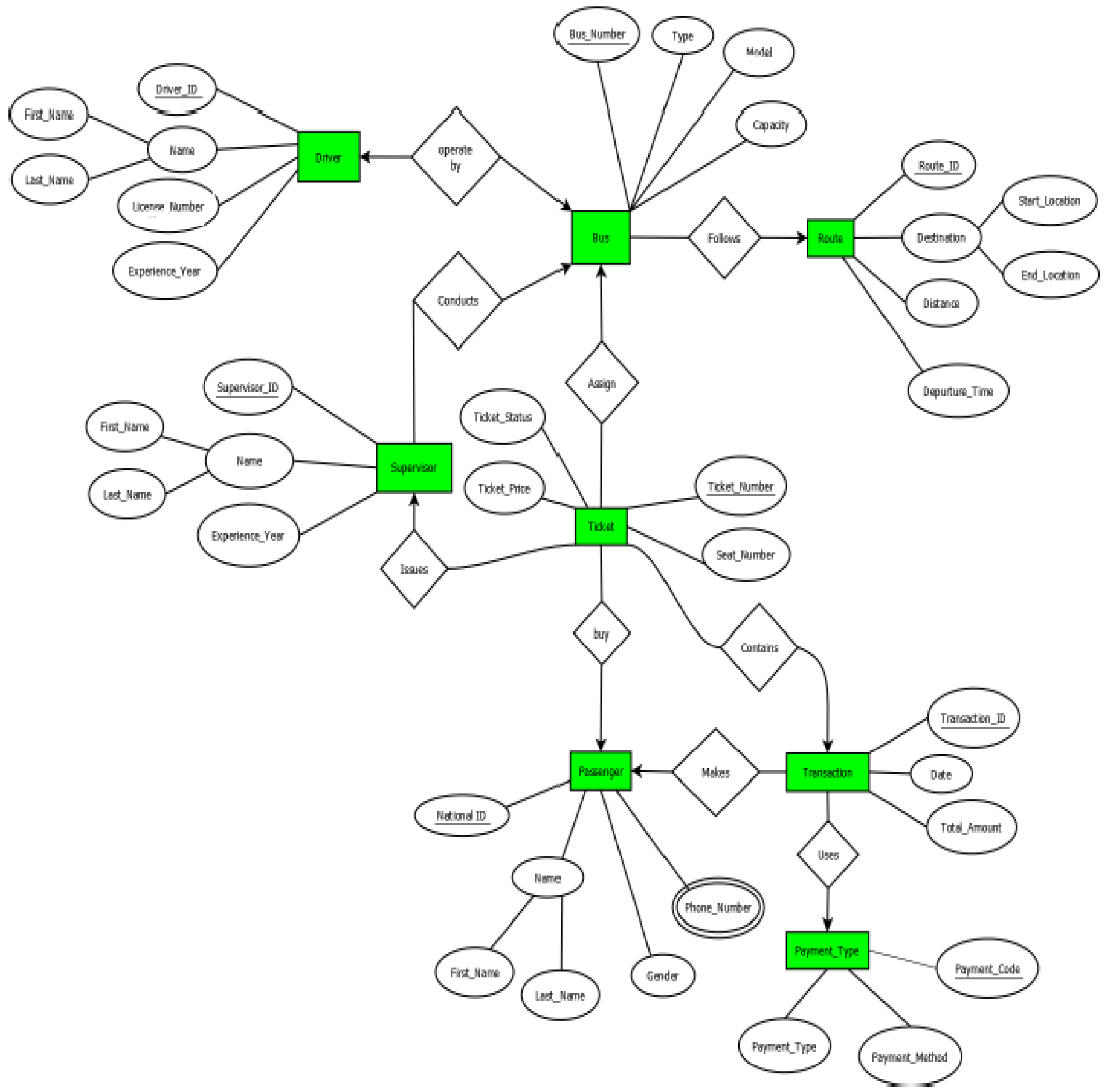
2. **Below an ER Diagram has been given write the scenario.**
   For reference see ERDiagramTutorial.



Answer 2:

In an alumni management system, each alumni is uniquely identified by an alumni ID and the system stores their name, contact information including email, phone number, and address, as well as their year of graduation. Each alumni may hold a job, and a job is associated with only one alumni. Job details such as job ID, company name, and job title are stored by the system. An alumni can make multiple donations to the institution. Each donation is uniquely identified by a donation ID and includes the donation amount, purpose, date, and time. An alumni may also be a member of one or more networking groups. Each networking group is uniquely identified by a group ID and includes a name and description. A networking group can have many alumni members. Each alumni belongs to exactly one batch, and a batch may include many alumni. Each batch is identified by a batch ID and includes the total number of alumni in that batch. The system also records events that are attended by alumni. An event can be attended by multiple alumni, and an alumni can attend multiple events. Each event is identified by an event ID and includes the event name, type, and venue. An event may include multiple sessions, but each session belongs to only one event. A session is identified by a session ID and includes a date and time. Furthermore, each event is organized by exactly one batch, and a batch can organize many events.

3. **Normalize the ER Diagram given below up to 3rd Normal Form and finalize the tables that needs to be created. Then (in Oracle using SQL) write down the queries that are required to create all the tables with necessary constraints. Also insert at least 3 rows of data in each created table.**
For reference see NormalizationTutorial and BasicSQLTutorial.



Answer Box (Normalization steps in detail as shown in Normalization Tutorial Slide + all the queries required to create the tables and insert data after Normalization):

Step 1: Analysing the ER Diagram

From the diagram, we can identify the following entities and their attributes:

- **Driver**: Driver_ID (PK), First_Name, Last_Name, License_Number, and Experience_Year
- **Bus**: Bus_Number (PK), Type, Model, and Capacity
- **Route**: Route_ID (PK), Start_Location, End_Location, Distance, and Departure_Time
- **Supervisor**: Supervisor_ID (PK), First_Name, Last_Name, and Experience_Year
- **Ticket**: Ticket_Number (PK), Ticket_Status, Ticket_Price, and Seat_Number
- **Passenger**: Passenger_ID (PK), First_Name, Last_Name, Phone_Number, and Gender
- **Transaction**: Transaction_ID (PK), Date, and Total_Amount
- **Payment_Type**: Payment_Code (PK), Payment_Method, and Payment_Type

Step 2: Normalization

<u>1 st Normal Form (1NF)</u>

In 1NF, we ensure that:

- Each field contains atomic (indivisible) values.
- There are no repeating groups or arrays in the attributes.

Entities in 1NF:

- Driver (Driver_ID, First_Name, Last_Name, License_Number, Experience_Year)
  - No multivalued attributes, so it’s in 1NF.
- Bus (Bus_Number, Type, Model, Capacity)
  - No multivalued attributes, so it’s in 1NF.
- Route (Route_ID, Start_Location, End_Location, Distance, Departure_Time)
  - No multivalued attributes, so it’s in 1NF.
- Supervisor (Supervisor_ID, First_Name, Last_Name, Experience_Year)
  - No multivalued attributes, so it’s in 1NF.

- Ticket (Ticket_Number, Ticket_Status, Ticket_Price, Seat_Number, Route_ID)
    - No multivalued attributes, so it's in 1NF.
- Passenger (Passenger_ID, First_Name, Last_Name, Gender)
    - No multivalued attributes, so it's in 1NF.
- Transaction (Transaction_ID, Date, Total_Amount, Passenger_ID, Ticket_Number)
    - No multivalued attributes, so it's in 1NF.
- Payment_Type (Payment_Code, Payment_Method, Payment_Type)
    - No multivalued attributes, so it's in 1NF.

2nd Normal Form (2NF)

In **2NF**, we remove partial dependencies, meaning:

- Non-key attributes must depend on the entire primary key, not just part of it.

Entities in 2NF:

- **Driver Table:**
    - Primary Key: Driver_ID
    - Attributes: Driver_ID, First_Name, Last_Name, License_Number, Experience_Year
    - No partial dependencies. It's already in **2NF**.
- **Bus Table:**
    - Primary Key: Bus_Number
    - Attributes: Bus_Number, Type, Model, Capacity
    - No partial dependencies. It's already in **2NF**.
- **Route Table:**
    - Primary Key: Route_ID
    - Attributes: Route_ID, Start_Location, End_Location, Distance, Departure_Time
    - No partial dependencies. It's already in **2NF**.
- **Supervisor Table:**
    - Primary Key: Supervisor_ID
    - Attributes: Supervisor_ID, First_Name, Last_Name, Experience_Year
    - No partial dependencies. It's already in **2NF**.
- **Ticket Table:**
    - Primary Key: Ticket_Number
    - Attributes: Ticket_Number, Ticket_Status, Ticket_Price, Seat_Number, Route_ID
    - Partial Dependency: Route_ID is partially related to the primary key (Ticket_Number).

- **Solution:** Split the table into two: ○ **Ticket Table**: (Ticket_Number, Ticket_Status, Ticket_Price, Seat_Number, Route_ID)
  - ○ **Route Table**: (Route_ID, Start_Location, End_Location, Distance, Departure_Time)
- **Passenger Table:**
  - Primary Key: Passenger_ID
  - Attributes: Passenger_ID, First_Name, Last_Name, Gender
  - No partial dependencies. It's already in **2NF**.
- **Transaction Table:**
  - Primary Key: Transaction_ID
  - Attributes: Transaction_ID, Date, Total_Amount, Passenger_ID, Ticket_Number
  - No partial dependencies. It's already in **2NF**.

- **Payment_Type Table:**

  - Primary Key: Payment_Code
  - Attributes: Payment_Code, Payment_Method, Payment_Type
  - No partial dependencies. It's already in **2NF**.

<u>3rd Normal Form (3NF)</u>

In 3NF, we remove transitive dependencies, meaning:

- Non-key attributes should not depend on other non-key attributes

Entities in 3NF:

- Driver Table: No transitive dependencies. It's in 3NF.
- Bus Table: No transitive dependencies. It's in 3NF.
- Route Table: No transitive dependencies. It's in 3NF.
- Supervisor Table: No transitive dependencies. It's in 3NF.
- Ticket Table: No transitive dependencies. It's in 3NF.
- Passenger Table: No transitive dependencies. It's in 3NF.
- Transaction Table: No transitive dependencies. It's in 3NF.
- Payment_Type Table: No transitive dependencies. It's in 3NF.

Final Tables After Normalization

- **Driver Table**: Driver_ID (Primary Key), First_Name, Last_Name, License_Number, Experience_Year
- **Bus Table**: Bus_Number (Primary Key), Type, Model, Capacity
- **Route Table**: Route_ID (Primary Key), Start_Location, End_Location, Distance, Departure_Time

- **Supervisor Table:** Supervisor_ID (Primary Key), First_Name, Last_Name, Experience_Year.
- **Ticket Table**: Ticket_Number (Primary Key), Ticket_Status, Ticket_Price, Seat_Number, Route_ID (Foreign Key references Route(Route_ID)).

- **Passenger Table**: Passenger_ID (Primary Key), First_Name, Last_Name, Gender.
- **Transaction Table**: Transaction_ID (Primary Key), Date, Total_Amount, Passenger_ID (Foreign Key references Passenger(Passenger_ID)), Ticket_Number (Foreign Key references Ticket(Ticket_Number)).
- **Payment_Type Table**: Payment_Code (Primary Key), Payment_Method, Payment_Type.

Final Table List (With Their Attributes)

- **Driver**: Driver_ID, First_Name, Last_Name, License_Number, Experience_Year
- **Bus**: Bus_Number, Type, Model, Capacity
- **Route**: Route_ID, Start_Location, End_Location, Distance, Departure_Time
- **Supervisor**: Supervisor_ID, First_Name, Last_Name, Experience_Year
- **Ticket**: Ticket_Number, Ticket_Status, Ticket_Price, Seat_Number, Route_ID
- **Passenger**: Passenger_ID, First_Name, Last_Name, Gender
- **Transaction**: Transaction_ID, Date, Total_Amount, Passenger_ID, Ticket_Number
- **Payment_Type**: Payment_Code, Payment_Method, Payment_Type

Queries to create all the tables:

```
CREATE TABLE Driver (

   Driver_ID INT PRIMARY KEY,

   First_Name VARCHAR(50),

   Last_Name VARCHAR(50),

   License_Number VARCHAR(20) UNIQUE,

   Experience_Year INT

);
```

```
CREATE TABLE Bus (
    Bus_Number VARCHAR(10) PRIMARY KEY,
    Type VARCHAR(20),
    Model VARCHAR(30),
    Capacity INT
);

CREATE TABLE Route (
    Route_ID INT PRIMARY KEY,
    Start_Location VARCHAR2(50),
    End_Location VARCHAR2(50),
    Distance NUMBER(6, 2),
    Departure_Time DATE
);

CREATE TABLE Supervisor (
    Supervisor_ID INT PRIMARY KEY,
    First_Name VARCHAR(50),
    Last_Name VARCHAR(50),
    Experience_Year INT
);

CREATE TABLE Passenger (
    Passenger_ID INT PRIMARY KEY,
    First_Name VARCHAR(50),
    Last_Name VARCHAR(50),
    Gender CHAR(1)
);
```

```
CREATE TABLE Ticket (

    Ticket_Number INT PRIMARY KEY,

    Ticket_Status VARCHAR(20),

    Ticket_Price DECIMAL(8,2),

    Seat_Number VARCHAR(10),

    Route_ID INT,

    FOREIGN KEY (Route_ID) REFERENCES Route(Route_ID)

);


CREATE TABLE Payment_Type (

    Payment_Code INT PRIMARY KEY,

    Payment_Method VARCHAR(50),

    Payment_Type VARCHAR(50)

);


CREATE TABLE Transaction_Table (

    Transaction_ID INT PRIMARY KEY,

    Trans_Date DATE,

    Total_Amount NUMBER(10,2),

    Passenger_ID INT,

    Ticket_Number INT,

    FOREIGN KEY (Passenger_ID) REFERENCES Passenger(Passenger_ID),

    FOREIGN KEY (Ticket_Number) REFERENCES Ticket(Ticket_Number)

);
```

Insert 3 rows of data in each table:

```
INSERT INTO Driver VALUES (1, 'John', 'Doe', 'LIC1234', 5);

INSERT INTO Driver VALUES (2, 'Alice', 'Smith', 'LIC5678', 8);

INSERT INTO Driver VALUES (3, 'Bob', 'Johnson', 'LIC9101', 3);


INSERT INTO Bus VALUES ('B001', 'AC', 'VolvoX', 40);

INSERT INTO Bus VALUES ('B002', 'Non-AC', 'TataExpress', 50);

INSERT INTO Bus VALUES ('B003', 'AC', 'Mercedes', 35);


INSERT INTO Route VALUES (101, 'City Center', 'Airport', 25.5, TO_DATE('08:00:00', 'HH24:MI:SS'));

INSERT INTO Route VALUES (102, 'North Station', 'Downtown', 15.0, TO_DATE('10:30:00', 'HH24:MI:SS'));

INSERT INTO Route VALUES (103, 'West End', 'University', 18.75, TO_DATE('07:15:00', 'HH24:MI:SS'));


INSERT INTO Supervisor VALUES (1, 'Emily', 'White', 10);

INSERT INTO Supervisor VALUES (2, 'Mark', 'Brown', 7);

INSERT INTO Supervisor VALUES (3, 'Sara', 'Black', 12);


INSERT INTO Passenger VALUES (201, 'Michael', 'Jordan', 'M');

INSERT INTO Passenger VALUES (202, 'Lisa', 'Kudrow', 'F');

INSERT INTO Passenger VALUES (203, 'Tom', 'Hanks', 'M');


INSERT INTO Ticket VALUES (301, 'Confirmed', 250.00, 'A1', 101);

INSERT INTO Ticket VALUES (302, 'Pending', 200.00, 'B2', 102);

INSERT INTO Ticket VALUES (303, 'Confirmed', 180.00, 'C3', 103);
INSERT INTO Payment_Type VALUES (1, 'Credit Card', 'Online');

INSERT INTO Payment_Type VALUES (2, 'Cash', 'Offline');

INSERT INTO Payment_Type VALUES (3, 'Mobile Payment', 'Online');



INSERT INTO Transaction_Table VALUES (401, TO_DATE('2025-04-10', 'YYYY-MM-DD'), 250.00, 201, 301);

INSERT INTO Transaction_Table VALUES (402, TO_DATE('2025-04-11', 'YYYY-MM-DD'), 200.00, 202, 302);

INSERT INTO Transaction_Table VALUES (403, TO_DATE('2025-04-12', 'YYYY-MM-DD'), 180.00, 203, 303);
```

4. **Query Writing (Write down the question and the answer. Give full screenshot of the Oracle 10g Homepage that contains the answer and result)**
   -All screenshots MUST include the DATE and TIME feature from the screen of the machine (PC, Laptop etc.) used
   SQL
   -2 single-row function
   -2 group function
   -2 subquery
   -2 joining
   For reference see BasicSQLTutorial and AdvanceSQLTutorial.

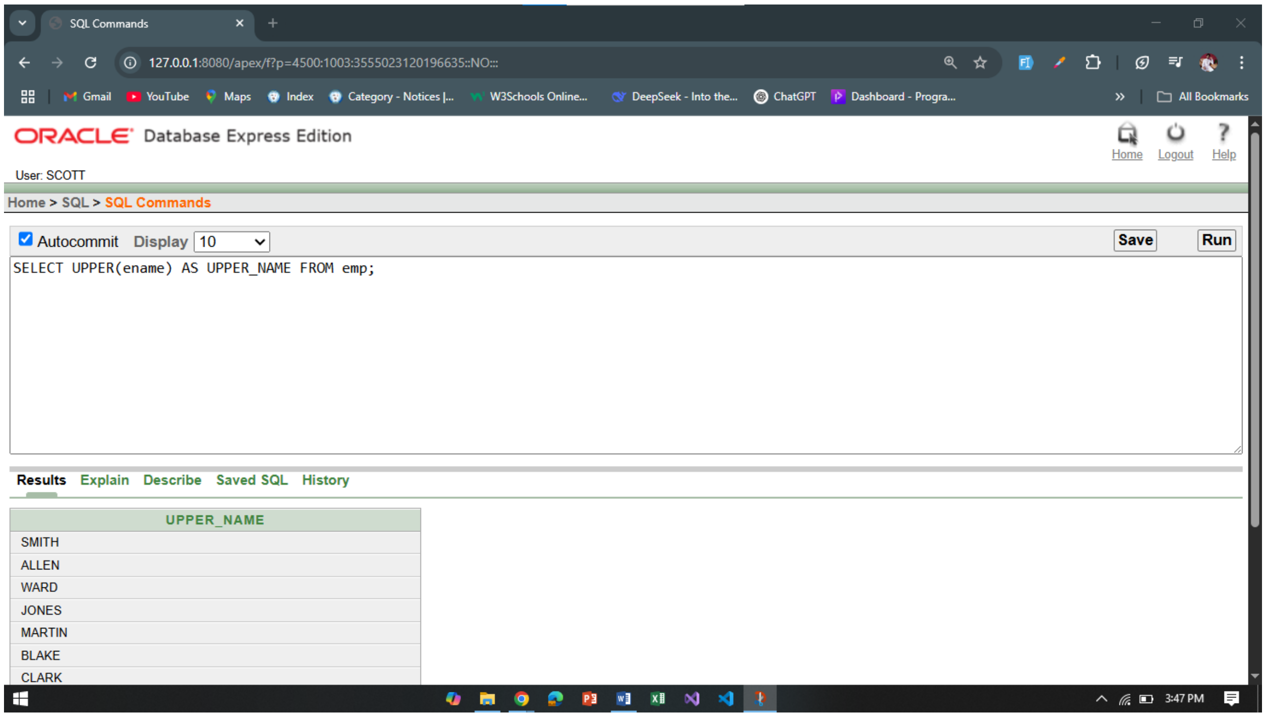
Answer 4:

<u>SQL</u>

-2 single-row function

Q1: Display the employee name in uppercase from the EMP table.

Answer: SELECT UPPER(ename) AS UPPER_NAME FROM emp;

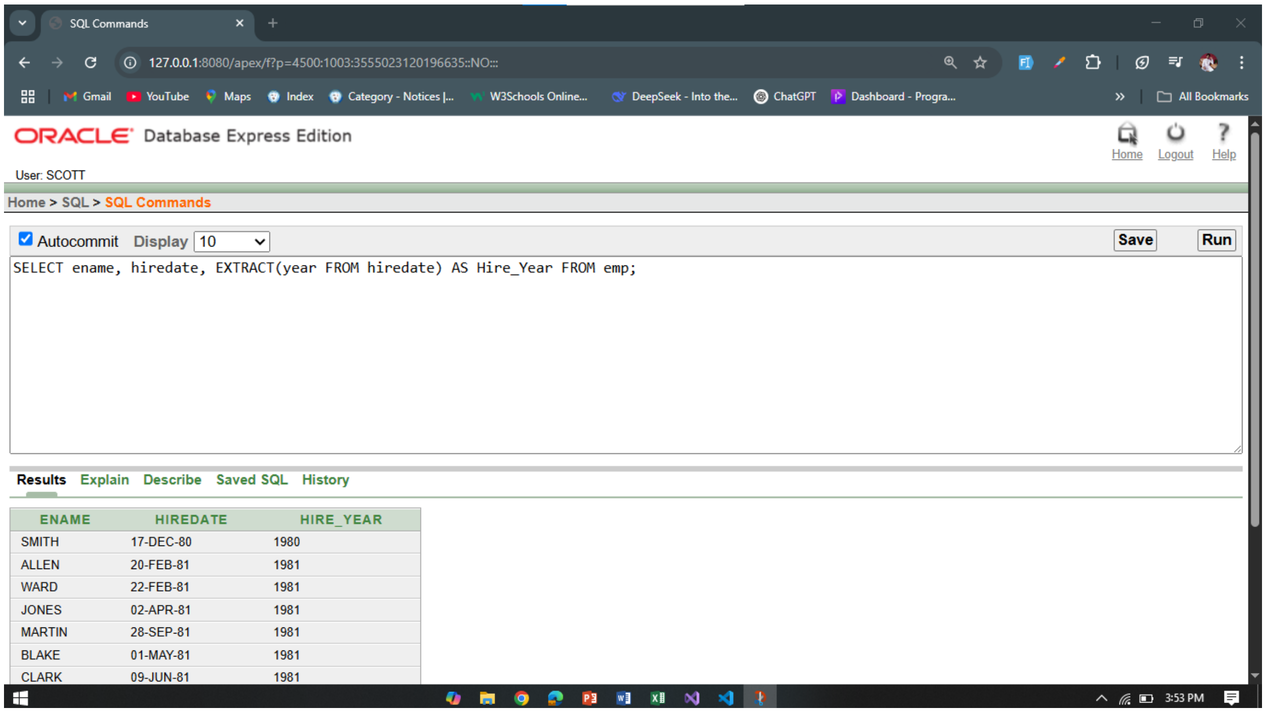

SQL Commands
127.0.0.1:8080/apex/f?p=4500:1003:3555023120196635::NO:::
ORACLE Database Express Edition
Home Logout Help
User: SCOTT
Home > SQL > SQL Commands
Autocommit Display 10
Save Run
Results Explain Describe Saved SQL History

```
SELECT UPPER(ename) AS UPPER_NAME FROM emp;
```

UPPER_NAME
SMITH
ALLEN
WARD
JONES
MARTIN
BLAKE
CLARK

Q2: Show the hire date of all employees and extract the year of hiring.

Answer: SELECT ename, hiredate, EXTRACT(year FROM hiredate) AS Hire_Year FROM emp;


SQL Commands
127.0.0.1:8080/apex/f?p=4500:1003:3555023120196635::NO:::
ORACLE Database Express Edition
Home Logout Help
User: SCOTT
Home > SQL > SQL Commands
Autocommit Display 10
Save Run
Results Explain Describe Saved SQL History

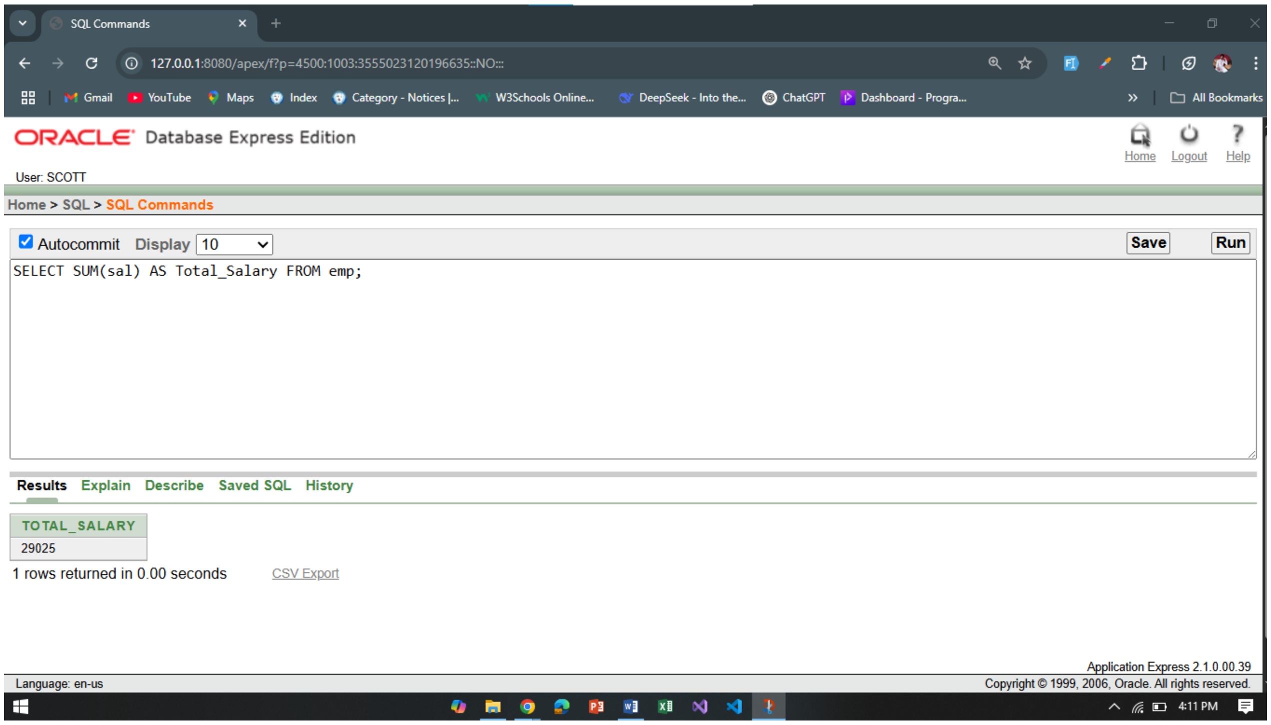
```
SELECT ename, hiredate, EXTRACT(year FROM hiredate) AS Hire_Year FROM emp;
```

ENAME	HIREDATE	HIRE_YEAR
SMITH	17-DEC-80	1980
ALLEN	20-FEB-81	1981
WARD	22-FEB-81	1981
JONES	02-APR-81	1981
MARTIN	28-SEP-81	1981
BLAKE	01-MAY-81	1981
CLARK	09-JUN-81	1981

-2 group function

Q1: Find the total salary paid to all employees.

Answer: SELECT SUM(sal) AS Total_Salary FROM emp;



SQL Commands
127.0.0.1:8080/apex/f?p=4500:1003:3555023120196635::NO:::
ORACLE Database Express Edition
Home Logout Help
User: SCOTT
Home > SQL > SQL Commands
Autocommit Display 10
Save Run
Results Explain Describe Saved SQL History
CSV Export
Language: en-us
Application Express 2.1.0.00.39
Copyright © 1999, 2006, Oracle. All rights reserved.

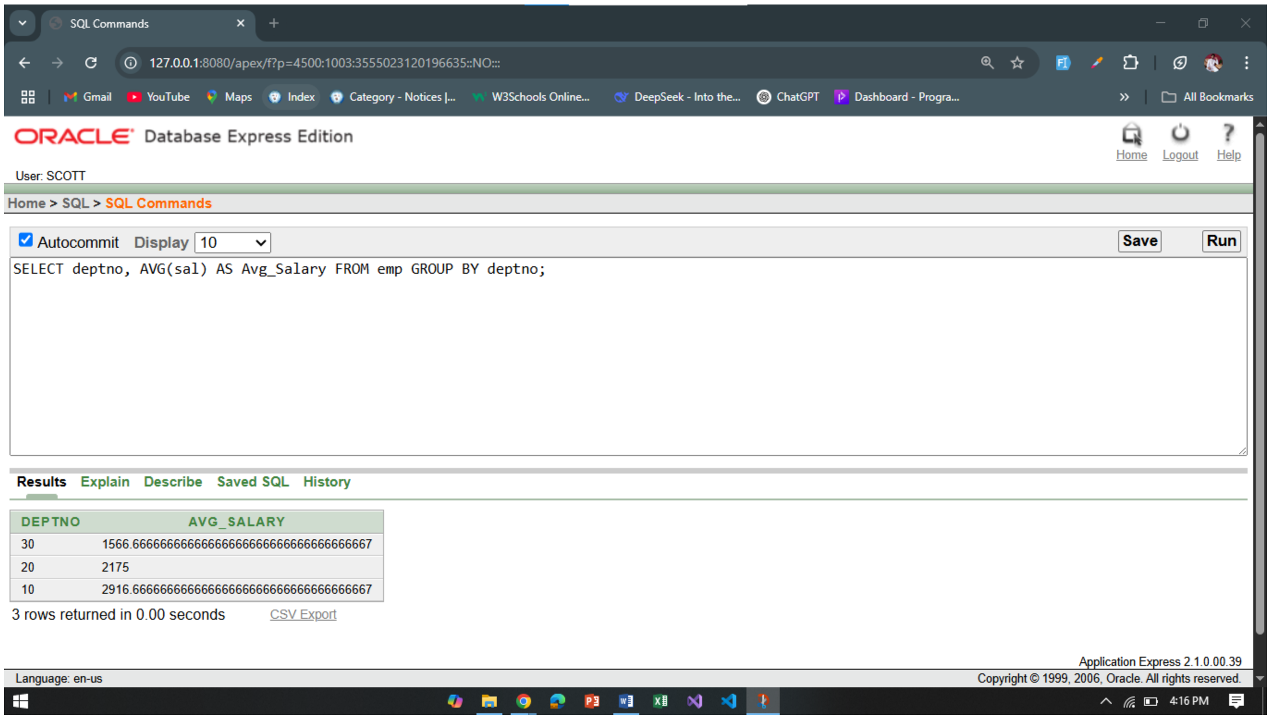
```
SELECT SUM(sal) AS Total_Salary FROM emp;
```

TOTAL_SALARY
29025

1 rows returned in 0.00 seconds

Q2: Display the average salary department wise.

Answer: SELECT deptno, AVG(sal) AS Avg_Salary FROM emp GROUP BY deptno;



SQL Commands
127.0.0.1:8080/apex/f?p=4500:1003:3555023120196635::NO:::
ORACLE Database Express Edition
Home Logout Help
User: SCOTT
Home > SQL > SQL Commands
Autocommit Display 10
Save Run
Results Explain Describe Saved SQL History
CSV Export
Language: en-us
Application Express 2.1.0.00.39
Copyright © 1999, 2006, Oracle. All rights reserved.

```
SELECT deptno, AVG(sal) AS Avg_Salary FROM emp GROUP BY deptno;
```

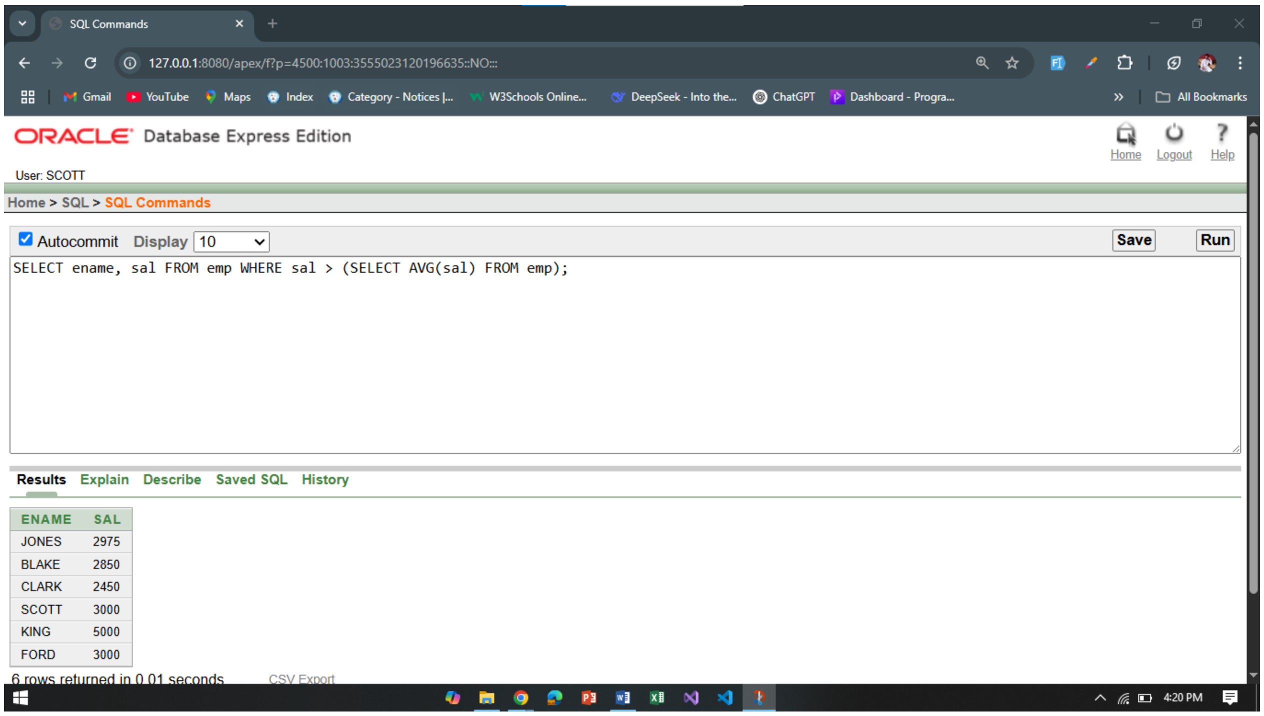
DEPTNO	AVG_SALARY
30	1566.66666666666666666666666666666666667
20	2175
10	2916.66666666666666666666666666666666667

3 rows returned in 0.00 seconds

-2 subquery

Q1: Display employees who earn more than the average salary.

Answer: SELECT ename, sal FROM emp WHERE sal > (SELECT AVG(sal) FROM emp);



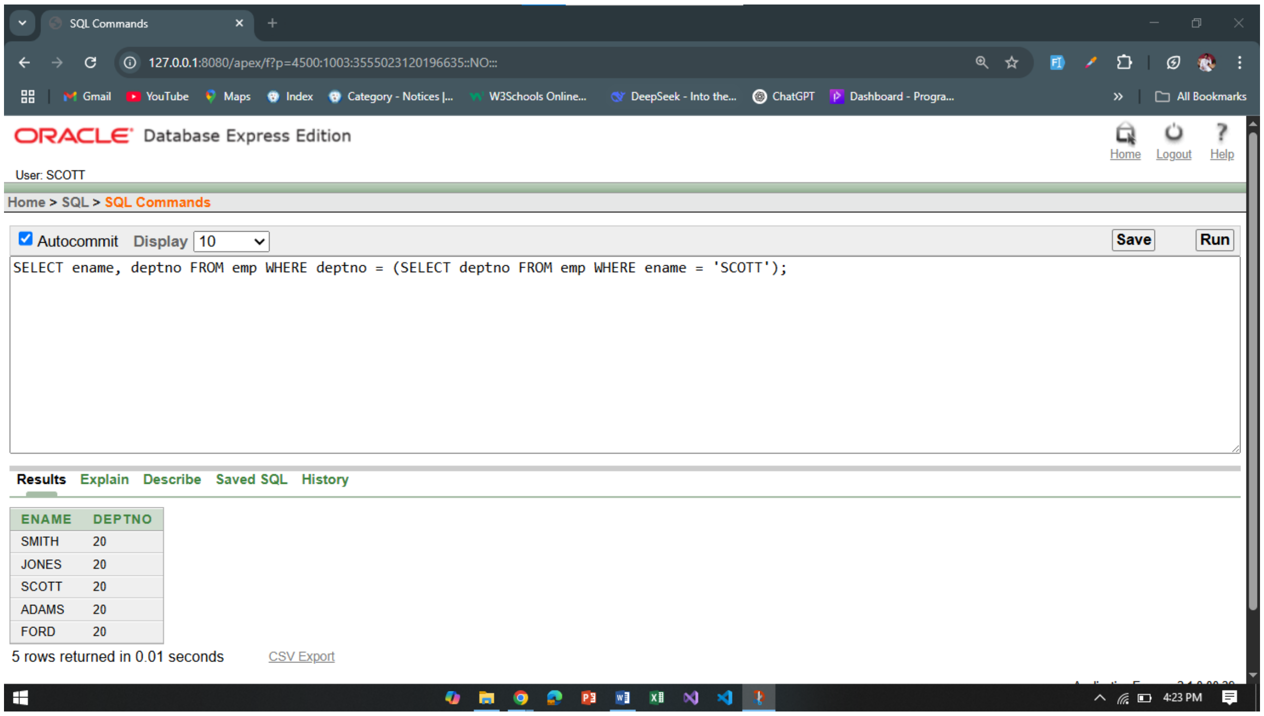
```
SELECT ename, sal FROM emp WHERE sal > (SELECT AVG(sal) FROM emp);
```

ENAME	SAL
JONES	2975
BLAKE	2850
CLARK	2450
SCOTT	3000
KING	5000
FORD	3000

6 rows returned in 0.01 seconds

Q2: Find employees who work in the same department as ‘SCOTT’.

Answer: SELECT ename, deptno FROM emp WHERE deptno = (SELECT deptno FROM emp WHERE ename = 'SCOTT');



```
SELECT ename, deptno FROM emp WHERE deptno = (SELECT deptno FROM emp WHERE ename = 'SCOTT');
```

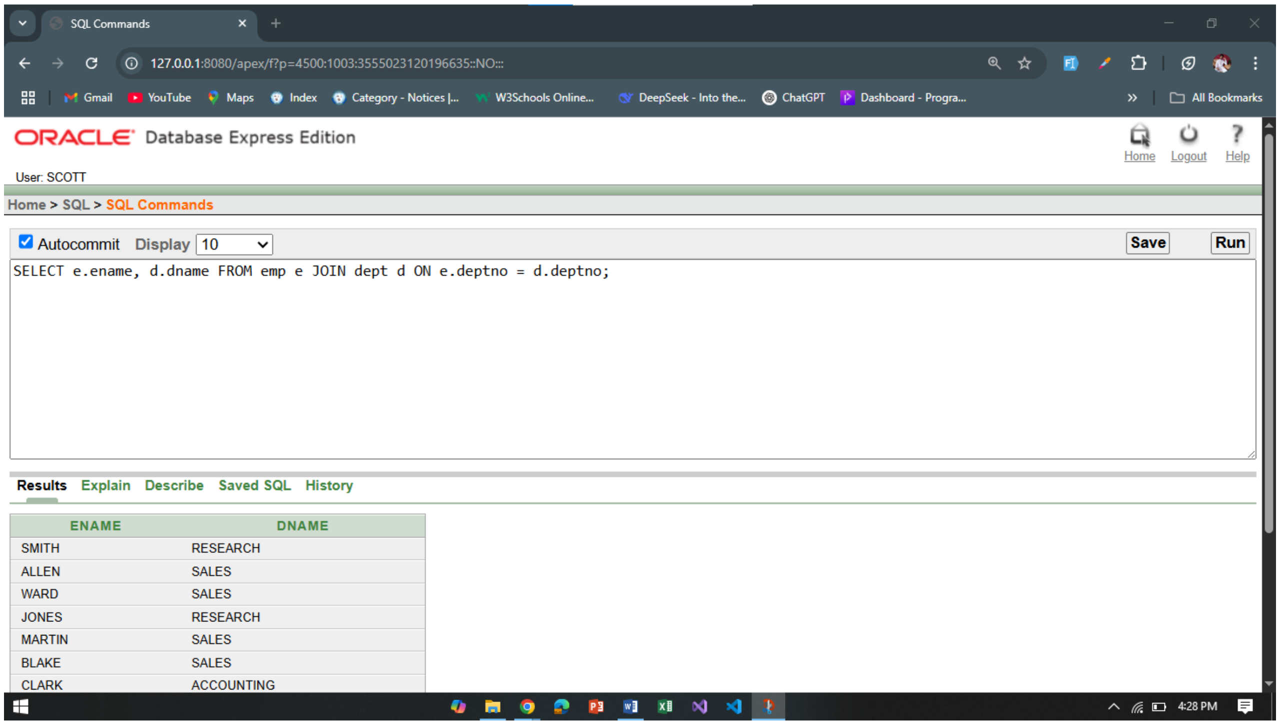
ENAME	DEPTNO
SMITH	20
JONES	20
SCOTT	20
ADAMS	20
FORD	20

5 rows returned in 0.01 seconds

-2 joining

Q1: List all employees along with their department names.

Answer: SELECT e.ename, d.dname FROM emp e JOIN dept d ON e.deptno = d.deptno;

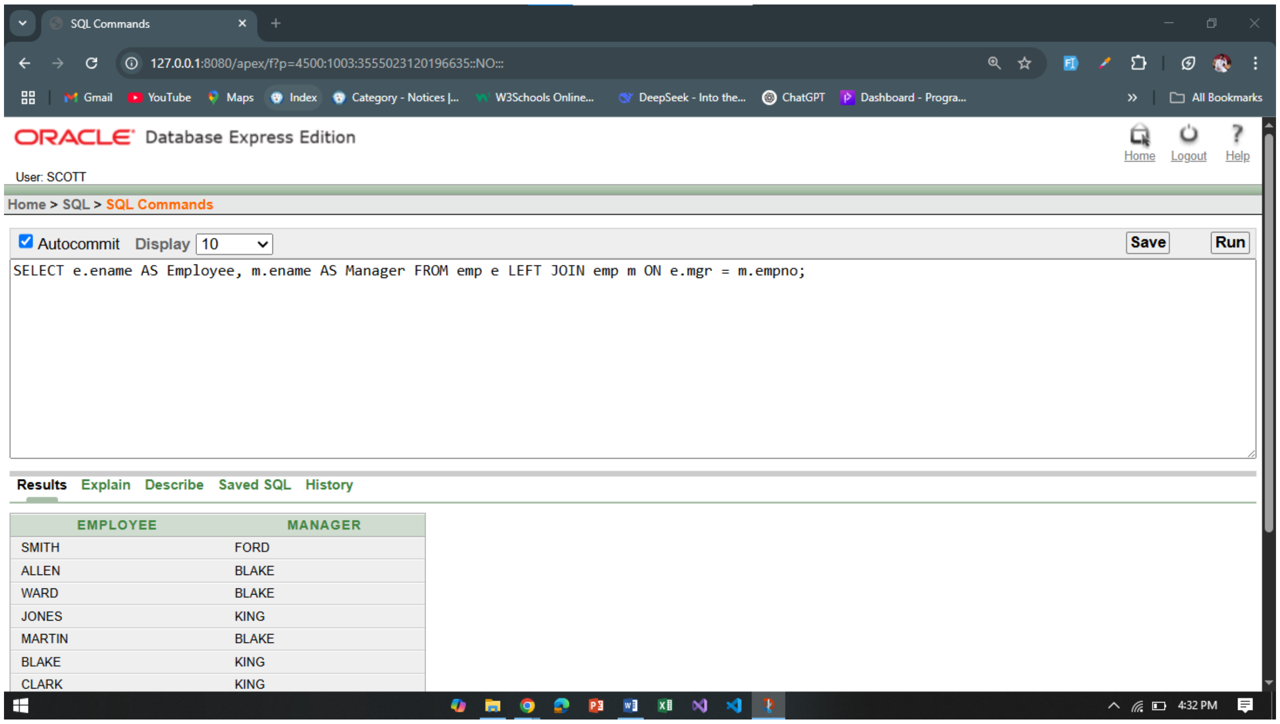


```
SELECT e.ename, d.dname FROM emp e JOIN dept d ON e.deptno = d.deptno;
```

ENAME	DNAME
SMITH	RESEARCH
ALLEN	SALES
WARD	SALES
JONES	RESEARCH
MARTIN	SALES
BLAKE	SALES
CLARK	ACCOUNTING

Q2: Display employee name, their manager name (self-join).

Answer: SELECT e.ename AS Employee, m.ename AS Manager FROM emp e LEFT JOIN emp m ON e.mgr = m.empno;



```
SELECT e.ename AS Employee, m.ename AS Manager FROM emp e LEFT JOIN emp m ON e.mgr = m.empno;
```

EMPLOYEE	MANAGER
SMITH	FORD
ALLEN	BLAKE
WARD	BLAKE
JONES	KING
MARTIN	BLAKE
BLAKE	KING
CLARK	KING