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

**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.**

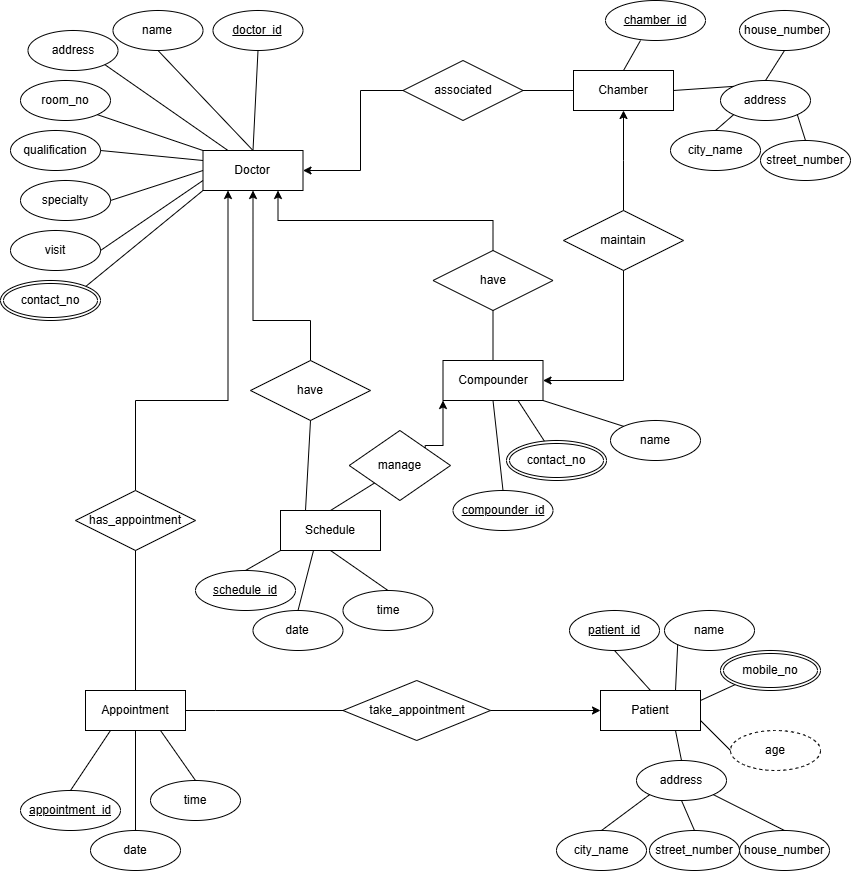
**Midterm Assignment**

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:

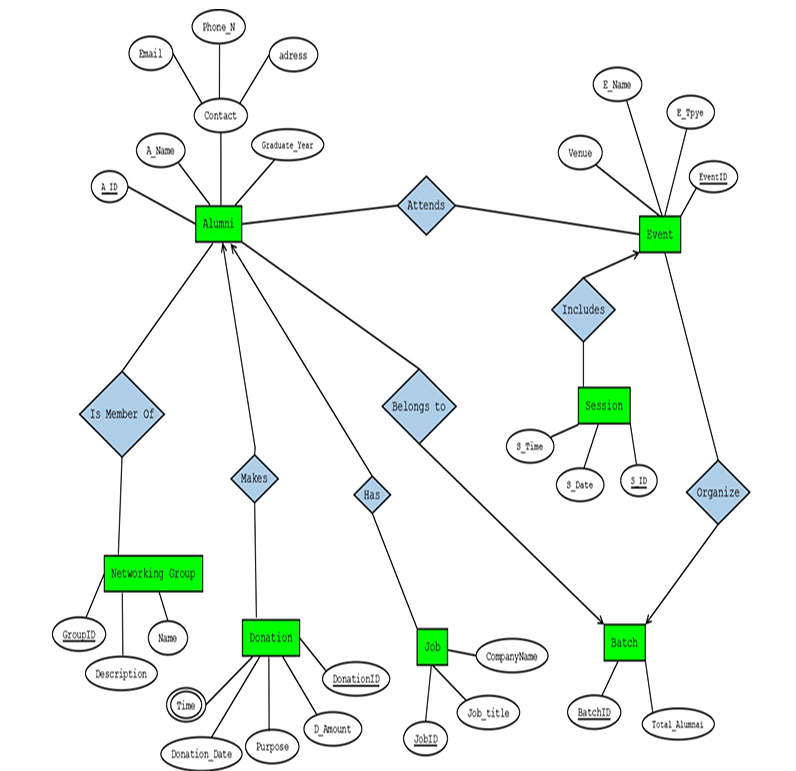


1. **Below an ER Diagram has been given write the scenario.**

***For reference see ERDiagramTutorial.***

1. **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.

1. **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.***

A diagram of a company

Description automatically generated

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**

**1** **st Normal Form (1NF)**

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: o **Ticket Table**: (Ticket\_Number, Ticket\_Status, Ticket\_Price, Seat\_Number,

Route\_ID)

o **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**.

**3rd Normal Form (3NF)**

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);

1. **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.***

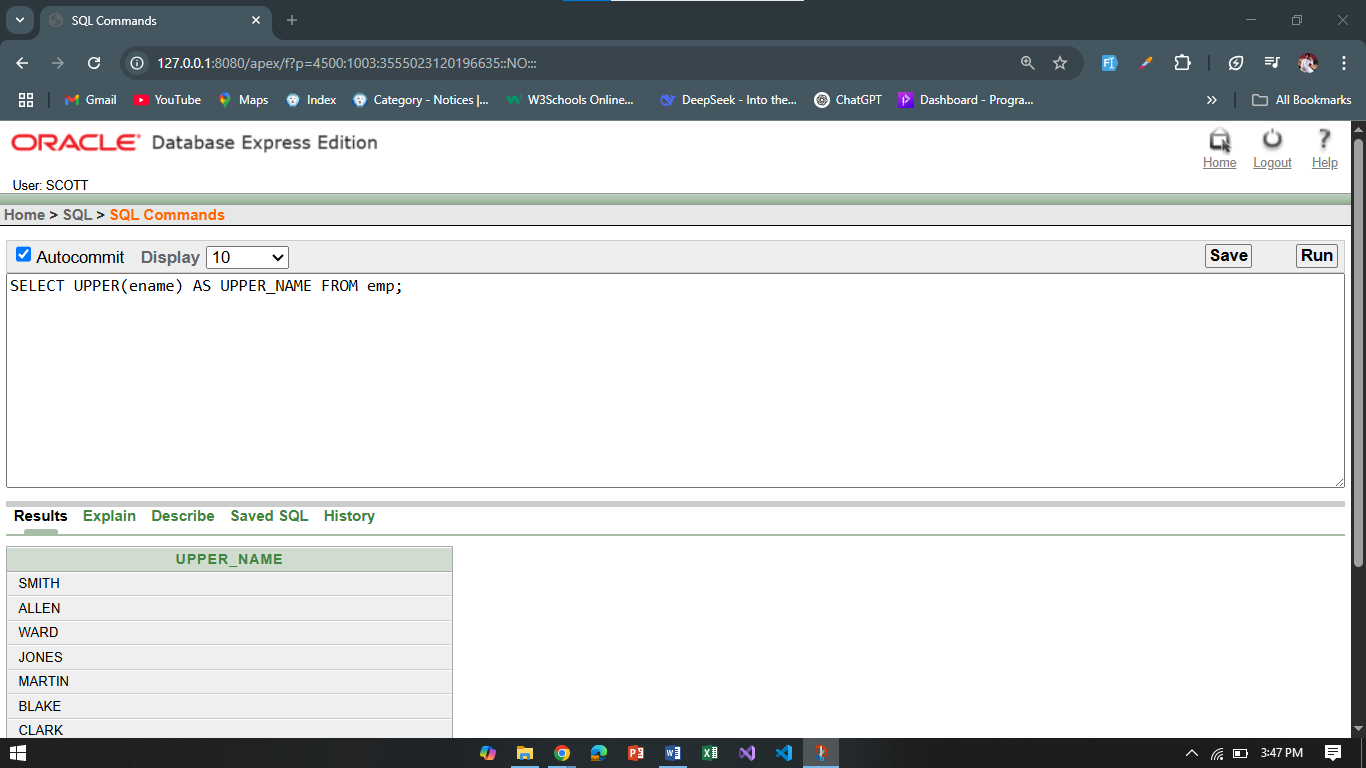
Answer 4:

**SQL**

**-2 single-row function**

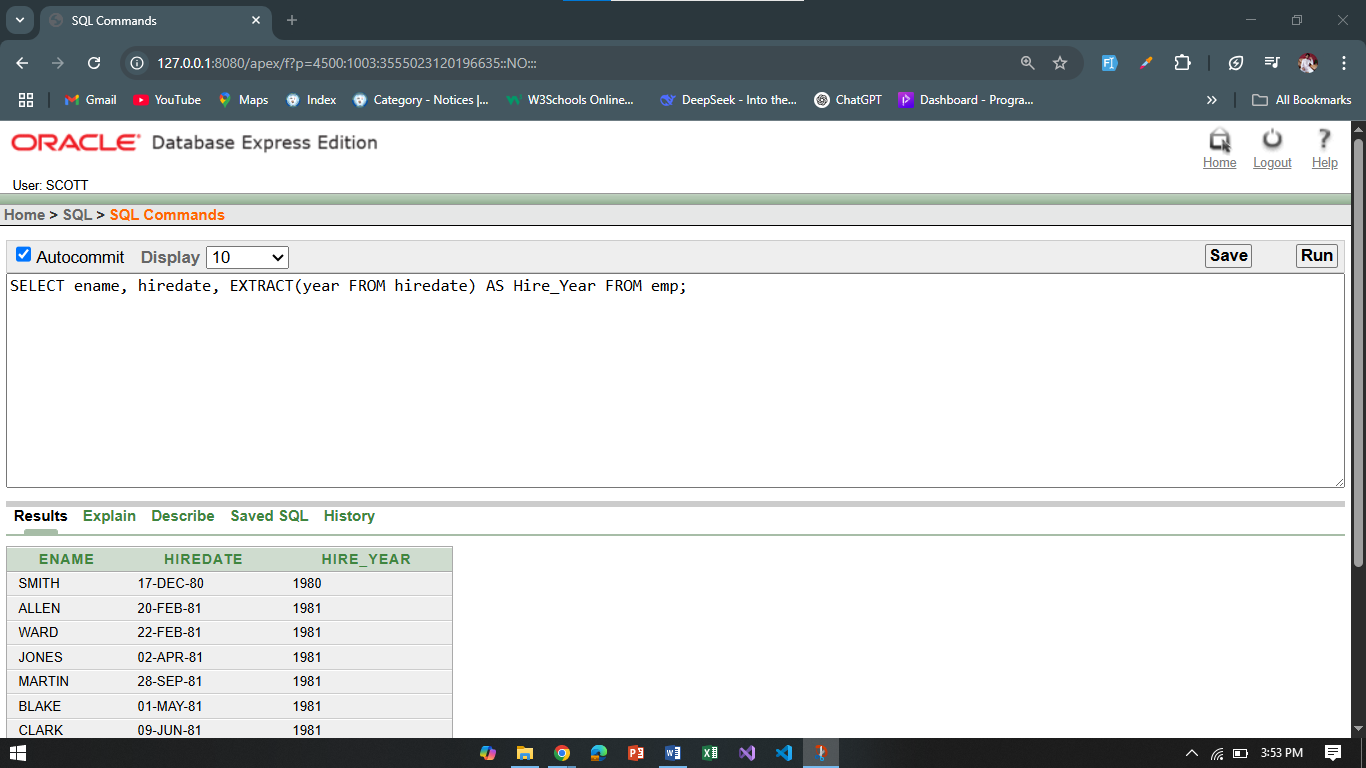
**Q1:** Display the employee name in uppercase from the EMP table.

**Answer:** SELECT UPPER(ename) AS UPPER\_NAME FROM emp;



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

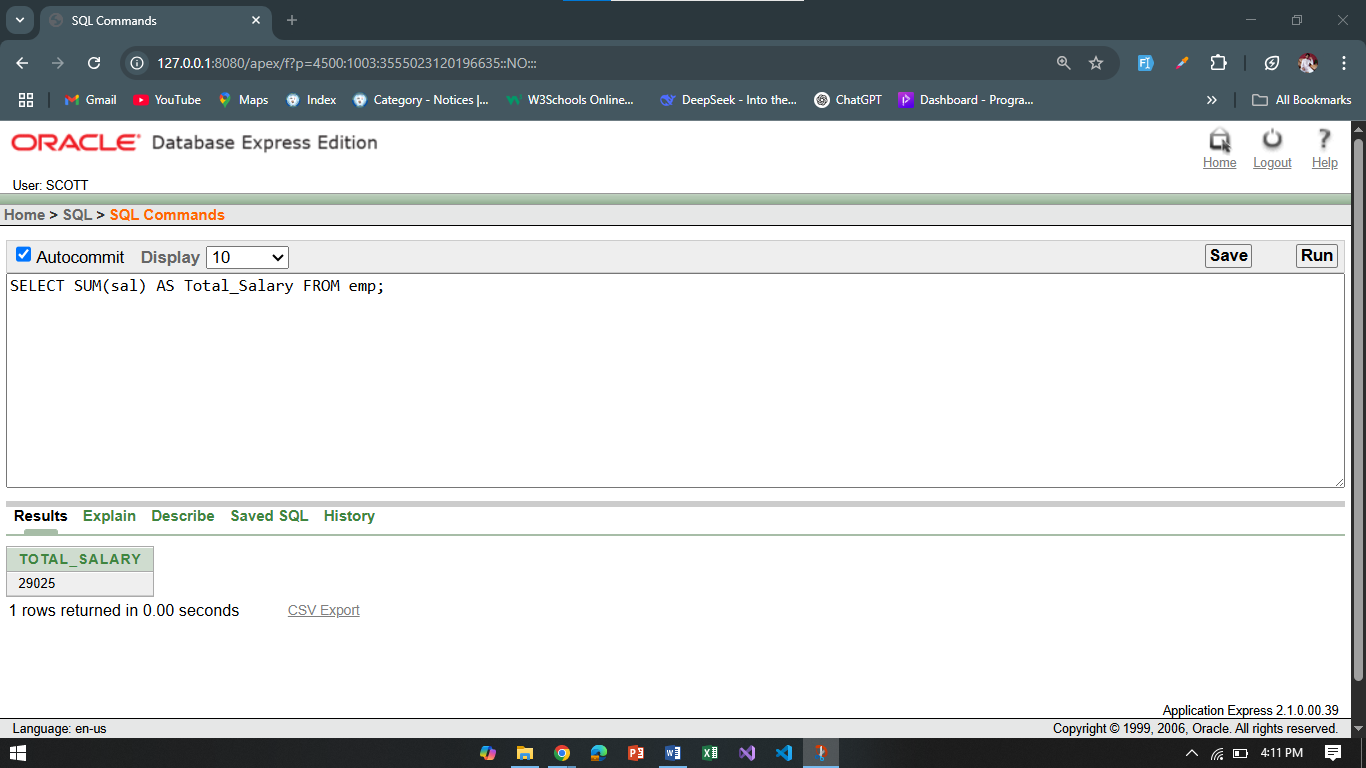
**Answer:** SELECT ename, hiredate, EXTRACT(year FROM hiredate) AS Hire\_Year FROM emp;



**-2 group function**

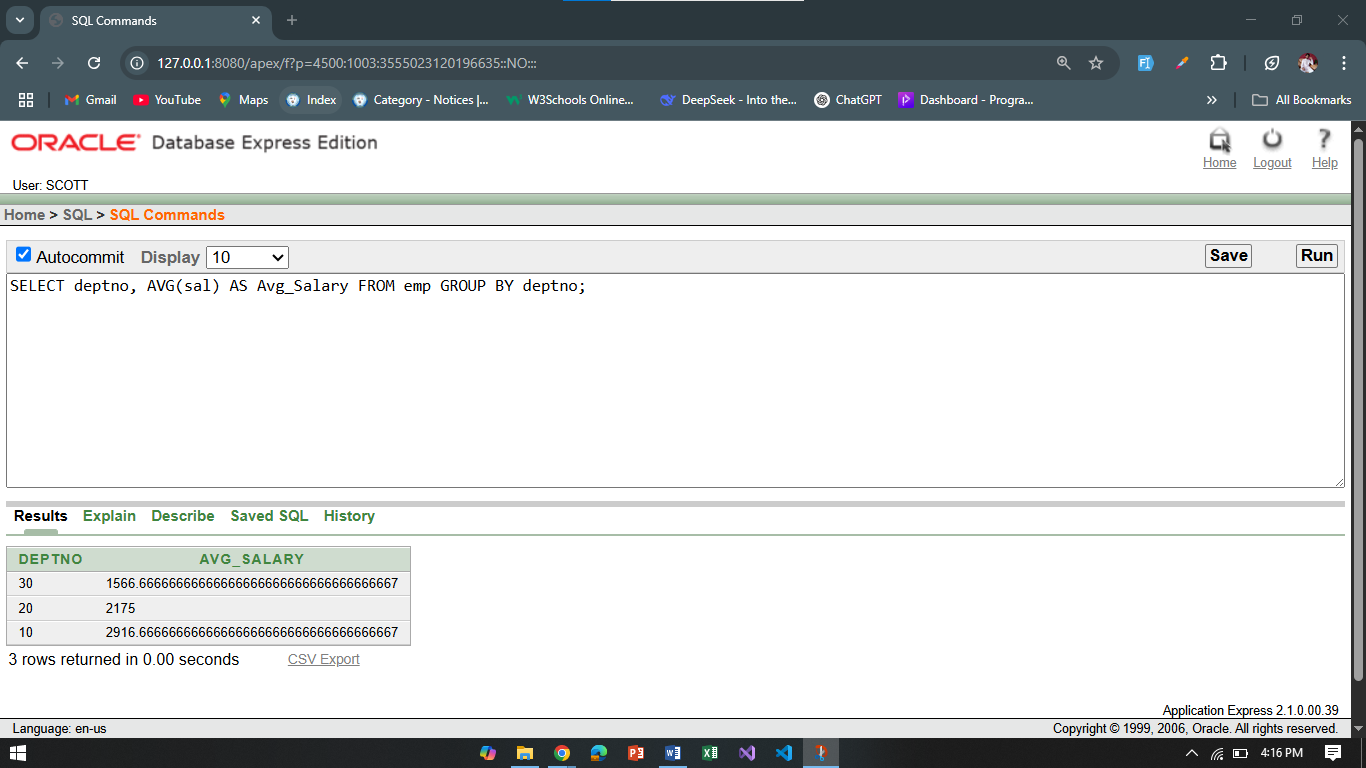
**Q1:** Find the total salary paid to all employees.

**Answer:** SELECT SUM(sal) AS Total\_Salary FROM emp;



**Q2:** Display the average salary department wise.

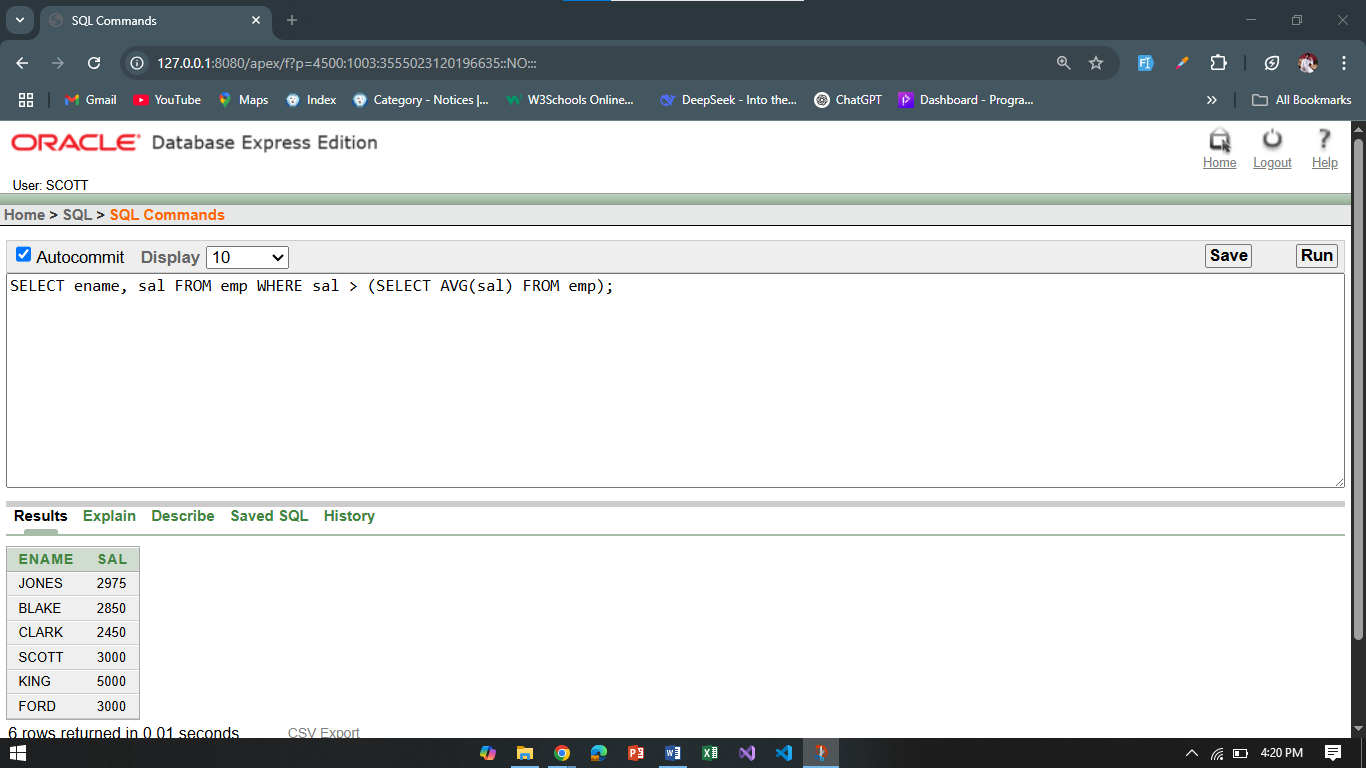
**Answer:** SELECT deptno, AVG(sal) AS Avg\_Salary FROM emp GROUP BY deptno;



**-2 subquery**

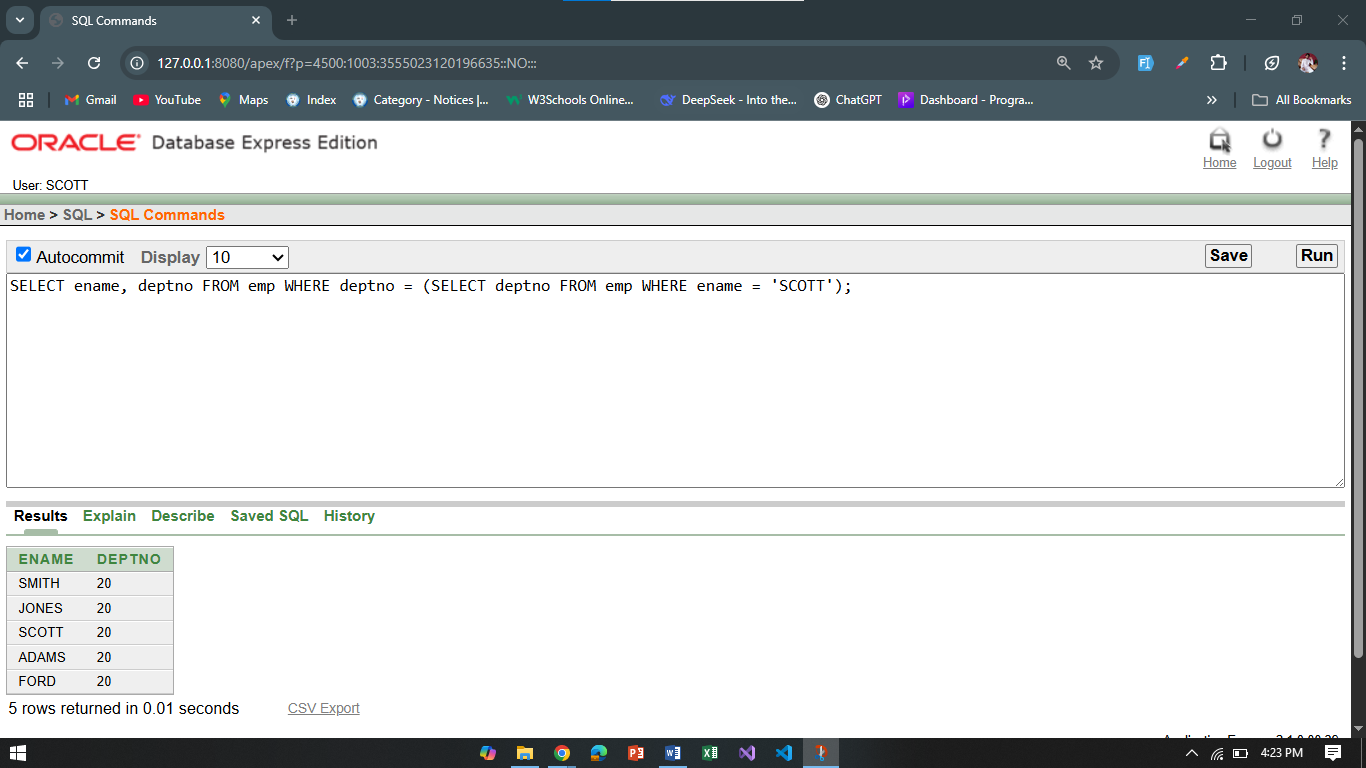
**Q1:** Display employees who earn more than the average salary.

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



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

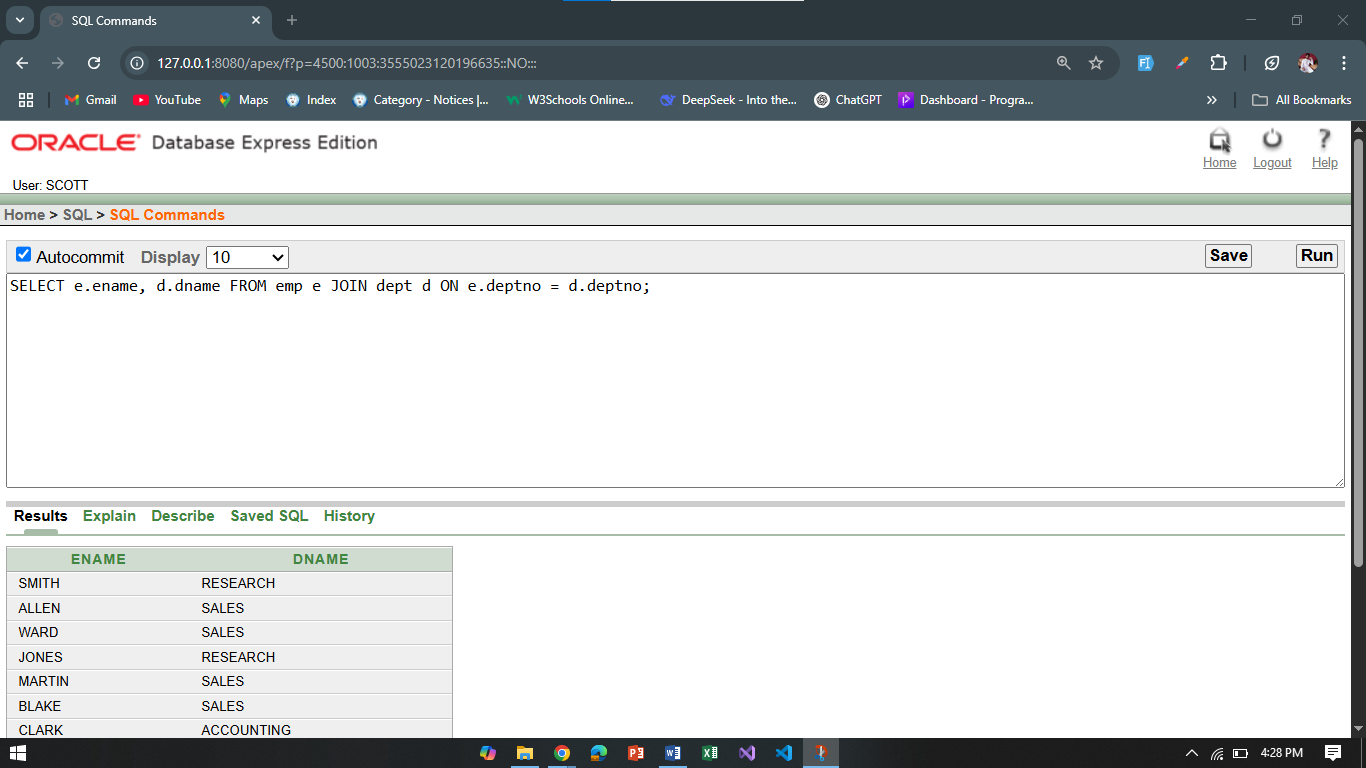
**Answer:** SELECT ename, deptno FROM emp WHERE deptno = (SELECT deptno FROM emp WHERE ename = 'SCOTT');



**-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;



**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;

