ASSIGNMENT (DBMS)

NAME-Abhishek Harsh

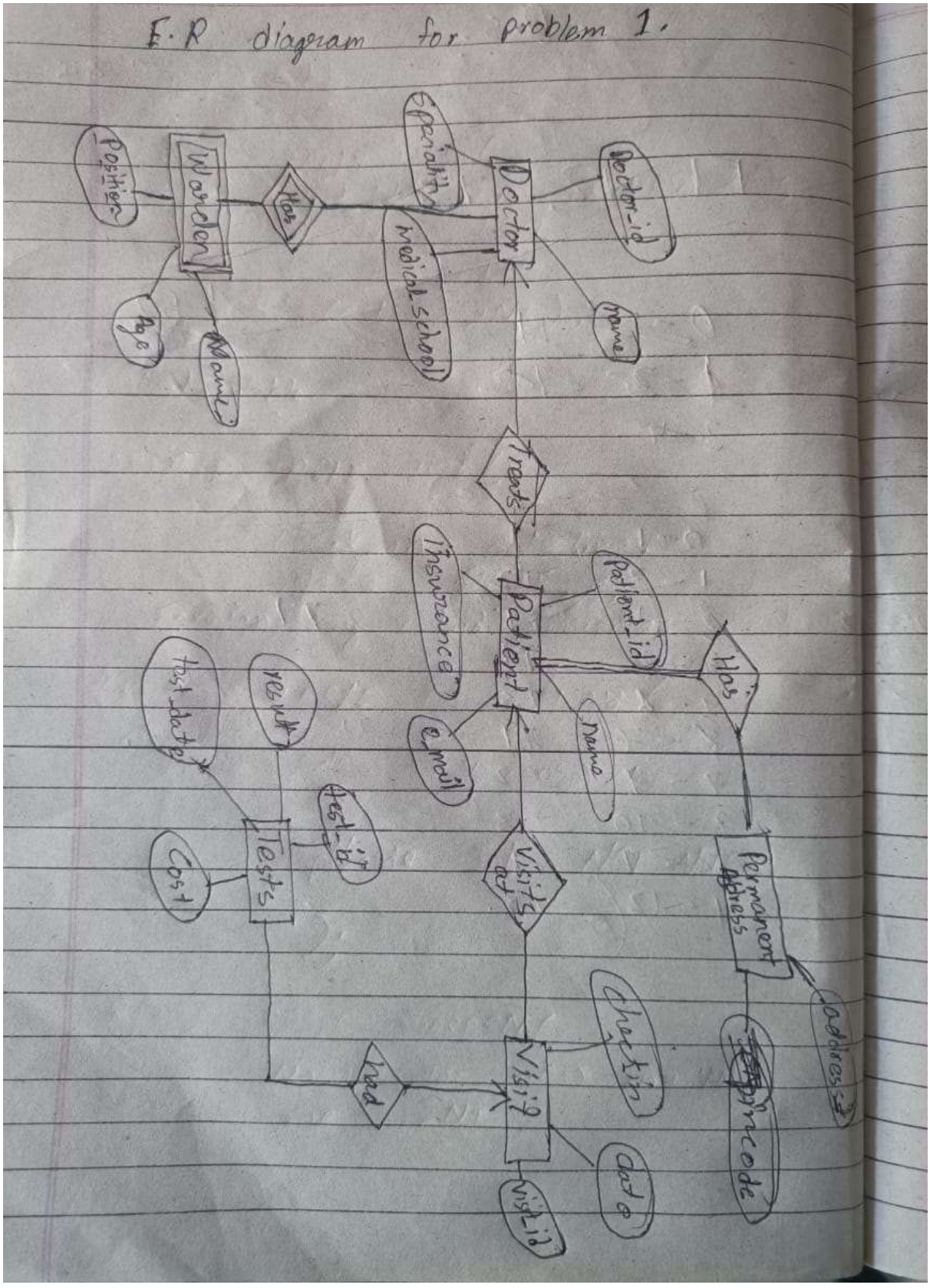
ROLL NUMBER-2021BCS0036

SUBJECT-DATABASE MANAGEMENT SYSTEM

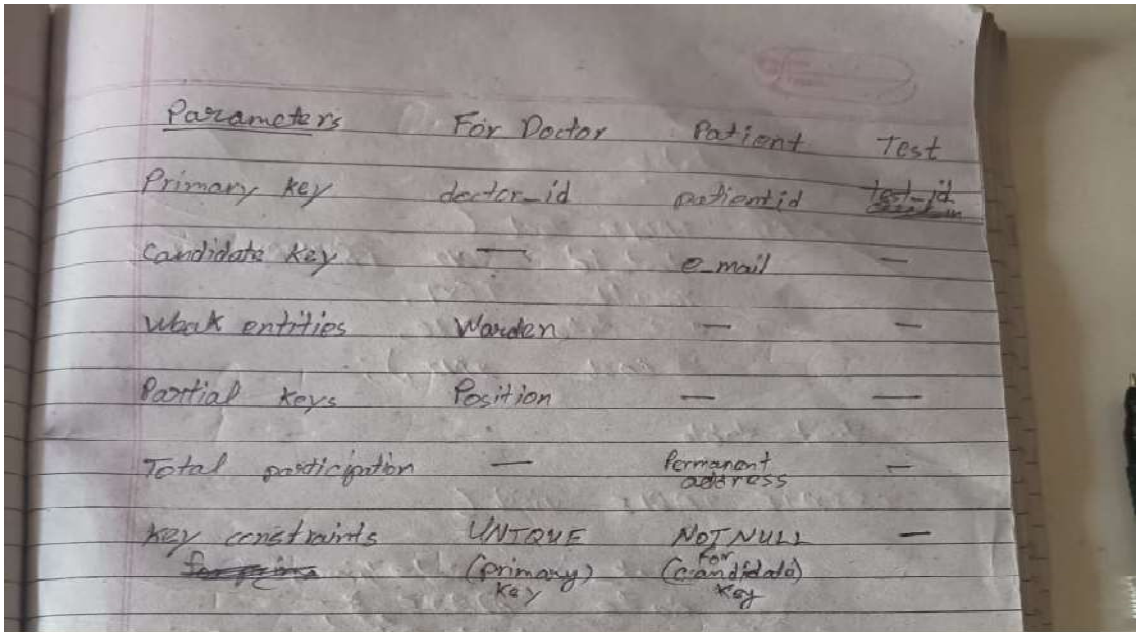
PROBLEM 1:-

Assume that you have been asked to create a database for a medical clinic. In this database you want to store information about Doctors, Patients, Tests, and Visits. For each doctor, we need to store a doctor id, name, specialty, and medical school that got her/his degree. For a patient, we need to store name, patient id, insurance, and a doctor that is the primary physician for this patient. For a test, we store the test name, test id, date of test, and the result. For a visit, we need to store the visit id, the check in and check out days, the patient associated with the visit and the doctor, who can be any doctor, not necessarily the primary physician. We assume that the doctor ids, patient ids, visit ids, and test ids are unique. Make sure that your design allows the same patient to have many visits on different dates and to have the same test many times.

1) Create the E-R diagrAM

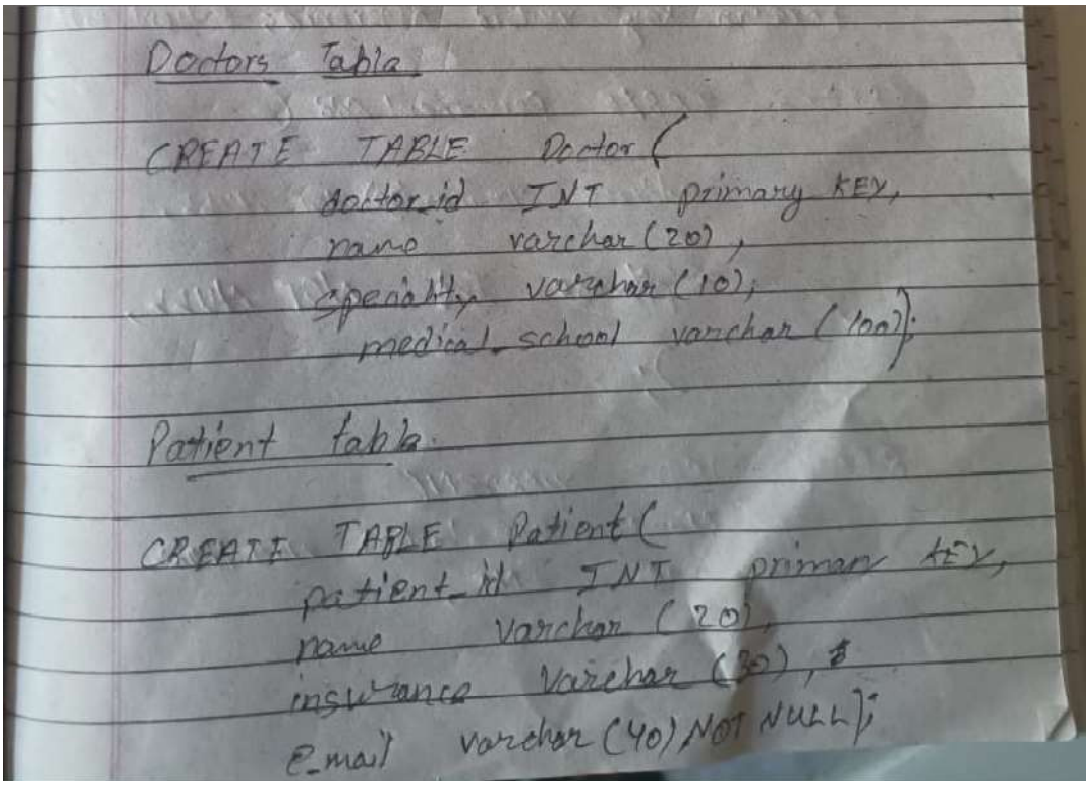


2) List the primary keys, candidate keys, weak entities (if any), partial keys (if any), total particiapation and any key constraints.



Parameters	For Doctor	Patient	Test
Primary key	doctor_id	patientid	test_id
Candidate key	—	e_mail	—
Weak entities	Warden	—	—
Partial keys	Position	—	—
Total participation	—	Permanent address	—
Key constraints	UNIQUE (primary key)	NOT NULL for (candidate key)	—

3) Turn the E-R diagram into tables (relational model schema). Provide the SQL CREATE TABLE statement for each table in the relational schema.

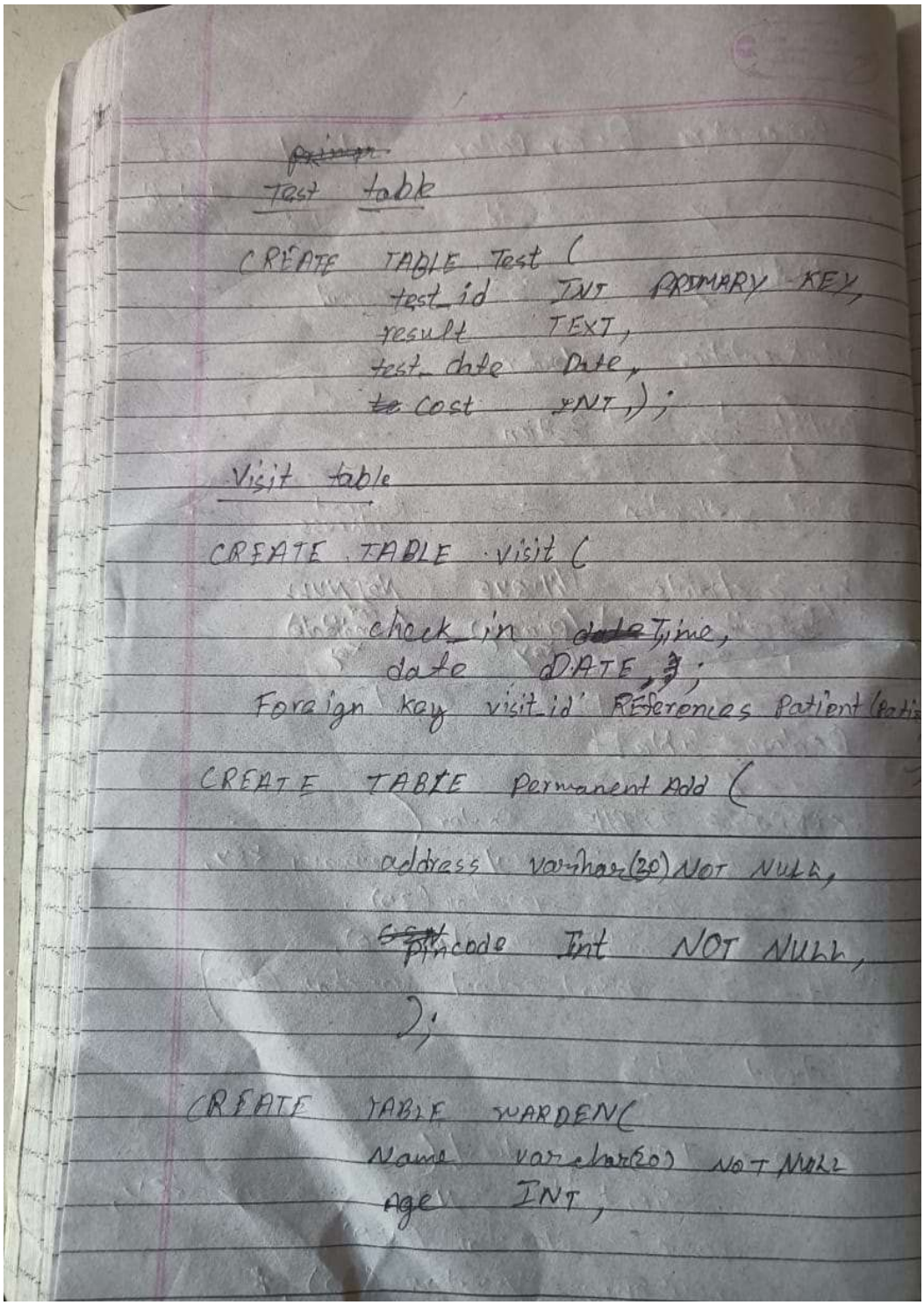


Doctors Table

```
CREATE TABLE Doctor (
    doctor_id INT primary KEY,
    name varchar(20),
    speciality varchar(10),
    medical_school varchar(100));
```

Patient table

```
CREATE TABLE Patient (
    patient_id INT primary KEY,
    name Varchar(20),
    insurance Varchar(30),
    E_mail varchar(40) NOT NULL);
```



Test table

```
CREATE TABLE Test (
    test_id    INT PRIMARY KEY,
    result     TEXT,
    test_date  Date,
    cost       INT,);
```

Visit table

```
CREATE TABLE visit (
    check_in  Time,
    date      DATE,);
Foreign key visit_id' References Patient (Pati
```

```
CREATE TABLE Permanent Add (
    address  varchar(20) NOT NULL,
    Pincode  Int NOT NULL,
    );
```

```
CREATE TABLE WARDEN(
    Name  varchar(20) NOT NULL
    Age   INT,
```

PROBLEM:- 2)

Suppose that you need to design a database for an airport. The relevant information that must be stored is:

• Every airplane has a registration number, and each airplane is of specific model.

• The airport accommodates a number of airplane models, and each model is identified by a model number (e.g. A-320, B-767) and has a capacity and a weight.

• A number of technicians work at the airport. You need to store the name, SSN, address, phone number, and salary of each technician.

• Each technician is an expert on one or more plane model(s).

• Traffic controllers work also at the airport. We need to store name, SSN, address, and phone number.

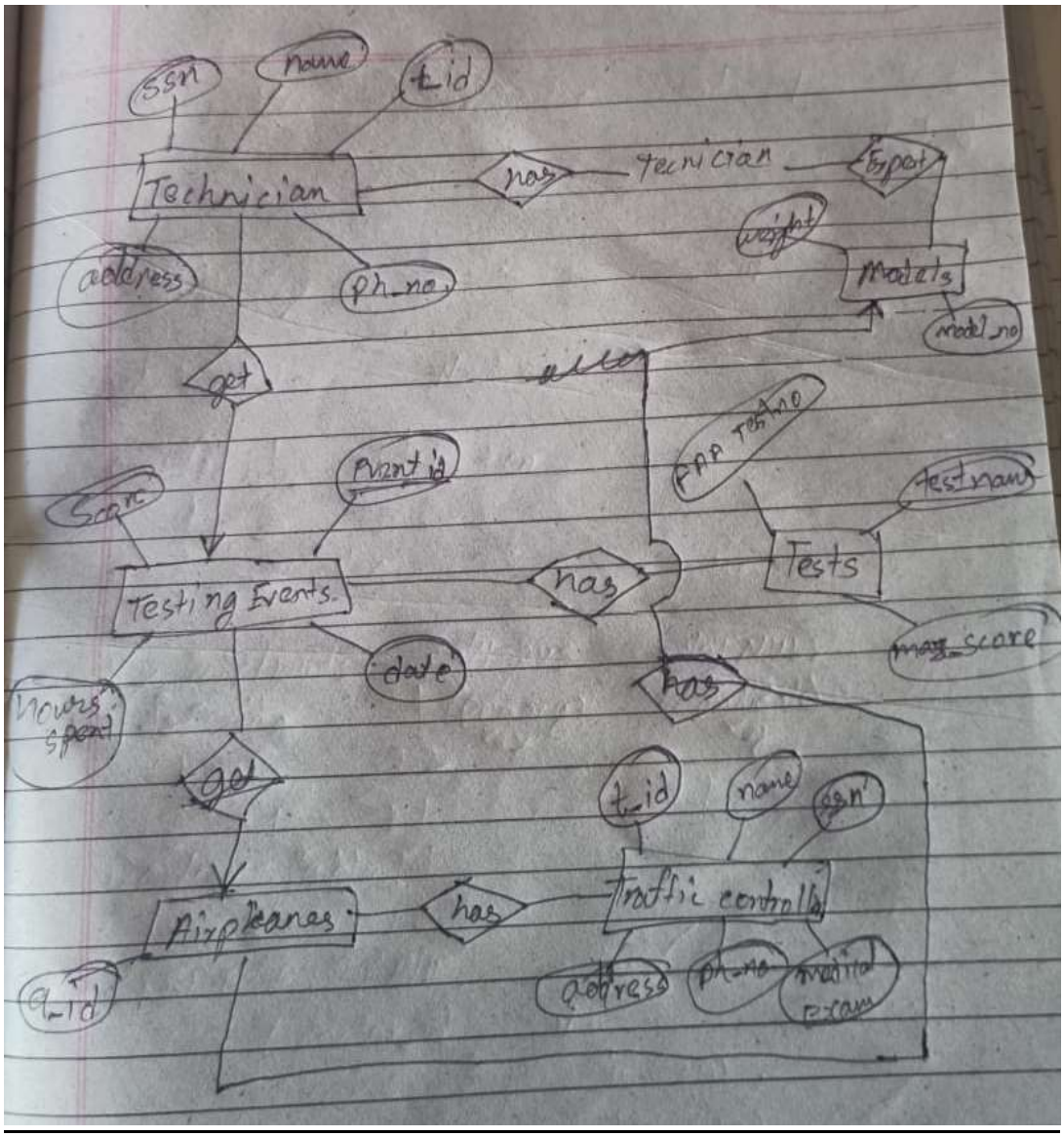
• Traffic controllers must have an annual medical examination. For each traffic controller, you must store the date of the most recent exam.

• The airport has a number of tests that are used periodically to ensure that airplanes are still airworthy. Each test has a Federal Aviation Administration (FAA) test number, a name, and a maximum possible score.

• The FAA requires the airport to keep track of each time a given airplane is tested by a given technician using a giving test. For each testing event, the information needed is the date, the number of hours

the technician spent doing the test, and the score the airplane received on the test.

1) Give an E/R diagram for this database. List the primary keys, candidate keys, weak entities (if 1 any), partial keys (if any), total participation and any key constraints.



1. Airplanes Table

- Primary key: registration_number
- Candidate keys: registration_number
- Weak entities: None
- Partial keys: None
- Total participation: None
- Key constraints: registration_number is unique

2. AirplaneModels Table

- Primary key: model_number
- Candidate keys: model_number
- Weak entities: None
- Partial keys: None
- Total participation: None
- Key constraints: model_number is unique

3. Technicians Table

- Primary key: technician_id
- Candidate keys: technician_id
- Weak entities: None
- Partial keys: None
- Total participation: None
- Key constraints: technician_id is unique

4. TechnicianExpertise Table

- Primary key: technician_id, model_number
- Candidate keys: technician_id, model_number
- Weak entities: None
- Partial keys: technician_id, model_number
- Total participation: None
- Key constraints: None

5. TrafficControllers Table

- Primary key: traffic_controller_id
- Candidate keys: traffic_controller_id
- Weak entities: None
- Partial keys: None
- Total participation: None
- Key constraints: traffic_controller_id is unique

6. Tests Table

- Primary key: FAA_test_number
- Candidate keys: FAA_test_number
- Weak entities: None
- Partial keys: None
- Total participation: None
- Key constraints: FAA_test_number is unique

7. TestingEvents Table

- Primary key: event_id
- Candidate keys: event_id
- Weak entities: None
- Partial keys: None
- Total participation: None
- Key constraints: event_id is unique

2)Turn the E-R diagram into tables (relational model schema). Indicate primary and candidate keys and foreign keys. Give the SQL statements to create the corresponding relations in a relational DBMS.

1. Airplanes Table:

```
CREATE TABLE Airplanes (
    registration_number TEXT PRIMARY KEY,
    model TEXT NOT NULL,
    FOREIGN KEY (model) REFERENCES
AirplaneModels(model_number)
);
```

2. AirplaneModels Table:

```
CREATE TABLE AirplaneModels (
    model_number TEXT PRIMARY KEY,
    capacity INTEGER NOT NULL,
    weight INTEGER NOT NULL
);
```

3. Technicians Table:

```
CREATE TABLE Technicians (
    technician_id INTEGER PRIMARY KEY,
    name TEXT NOT NULL,
    SSN TEXT NOT NULL,
    address TEXT NOT NULL,
    phone_number TEXT NOT NULL,
    salary REAL NOT NULL
);
```

4. Technical Expertise Table:

```
CREATE TABLE TechnicianExpertise (
    technician_id INTEGER PRIMARY KEY,
    model_number TEXT PRIMARY KEY,
    FOREIGN KEY (technician_id) REFERENCES
Technicians(technician_id),
    FOREIGN KEY (model_number) REFERENCES
AirplaneModels(model_number)
);
```

5. TrafficControllers Table:

```
CREATE TABLE TrafficControllers (
    traffic_controller_id INTEGER PRIMARY KEY,
    name TEXT NOT NULL,
    SSN TEXT NOT NULL,
    address TEXT NOT NULL,
    phone_number TEXT NOT NULL,
    medical_examination_date DATE NOT NULL
);
```

6. Tests Table:

```
CREATE TABLE Tests (
    FAA_test_number TEXT PRIMARY KEY,
    test_name TEXT NOT NULL,
    maximum_score INTEGER NOT NULL
);
```

7. TestingEvents Table:

```
CREATE TABLE TestingEvents (
    event_id INTEGER PRIMARY KEY,
    date DATE NOT NULL,
    technician_id INTEGER NOT NULL,
    FAA_test_number TEXT NOT NULL,
    registration_number TEXT NOT NULL,
    hours_spent REAL NOT NULL,
    score INTEGER NOT NULL,
```

```
    FOREIGN KEY (technician_id) REFERENCES
Technicians(technician_id),
    FOREIGN KEY (FAA_test_number) REFERENCES
Tests(FAA_test_number),
    FOREIGN KEY (registration_number) REFERENCES
Airplanes(registration_number)
);
```

PROBLEM:-3)

Consider the following database schema that is part of a larger database and stores information about electronic products (that in our case can be either PC, Laptop or Printer.) Product (maker, model, type) PC (model, speed, ram, hd, price) Laptop (model, speed, ram, hd, screen, price) Printer (model, color, type, price) The Product stores the maker, the model and the type that can be either PC, Laptop, or Printer.

Each of the other three relations store specific information about each product.

Maker and model, or model and type are unique.

Give an expression in relational algebra for each of the queries below:-

1. Which manufacturers make laptops with hard disks at least 100GB?

Ans. Π maker (σ hd $\geq$ 100GB (Laptop $\bowtie$ Product))
(maker = manufacturer, hd = hard disk)

2. Find the manufacturers that make Laptops but not PCs.

Ans. Π maker ((σ type='Laptop' (Product)) - (σ type='PC' (Product)))
(maker = manufacturer)

3. Find the hard disk sizes that occur in two or more PCs.

Ans. Π hd (σ hd ∈ (Π hd (PC ⋈ Product)) (PC ⋈ Product))
(hd = hard disk)

4. Find the manufacturer(s) of the PC(s) with the highest available speed.

Ans. Π maker (σ speed = max(speed) (PC ⋈ Product))
(maker=manufacturer)

PROBLEM :- 4)

Consider a database with the following schema:

Employee(SSN, name, salary, DNo)

Department(DNo, DeptName, MgrSSN)

Project(PNo, location, ProjName)

HourLog(SSN, PNo, hours)

The Employee relation provides a list of employees with their SSN, name, salary, and department number (DNo). The SSN is unique for each employee. Each employee belongs to only one department. The Department relation contains a list of the departments for the company. Its schema includes a unique department number called DNo. It also includes the name of the department (DeptName) and the social security number of the department's manager (MgrSSN). Each department has only one manager. The Project relation includes a unique project number (PNo), location and the project name (ProjName). An employee can be assigned to any number (including zero) projects. Each project has at least one person assigned to it. Finally, the HourLog relation lists for each project the number of hours of work for each employee who is assigned to that project.

The key of this relation is SSN and PNo. Write the following queries in Relational Algebra.

You may use assignment of intermediate results for long queries.

1. Find the name and the SSN of everyone who works more than 100 hours on a project located in Boston.

Ans. Π name, SSN (σ location='Boston' $\wedge$ hours $>$ 100 (HourLog $\bowtie$ Project $\bowtie$ Employee))

2. Find the name and SSN of everyone who works for department number 1 and also works on project number 2.

Ans. Π name, SSN (σ DNo=1 (Employee $\bowtie$ Department)) $\bowtie$ Π name, SSN (σ PNo=2 (HourLog $\bowtie$ Employee))

3. Find the name and the SSN of everyone who works on at least two projects.

Ans. Π name, SSN (σ SSN $\in$ (Π SSN (σ count(*) $>$ 1 (HourLog)) (HourLog $\bowtie$ Project $\bowtie$ Employee))

4. Find the name and the SSN of everyone who works on all projects.

Ans. Π name, SSN (σ SSN $\in$ (Π SSN (σ count(*) = count(Π PNo (Project)) (HourLog)) (HourLog $\bowtie$ Project $\bowtie$ Employee))