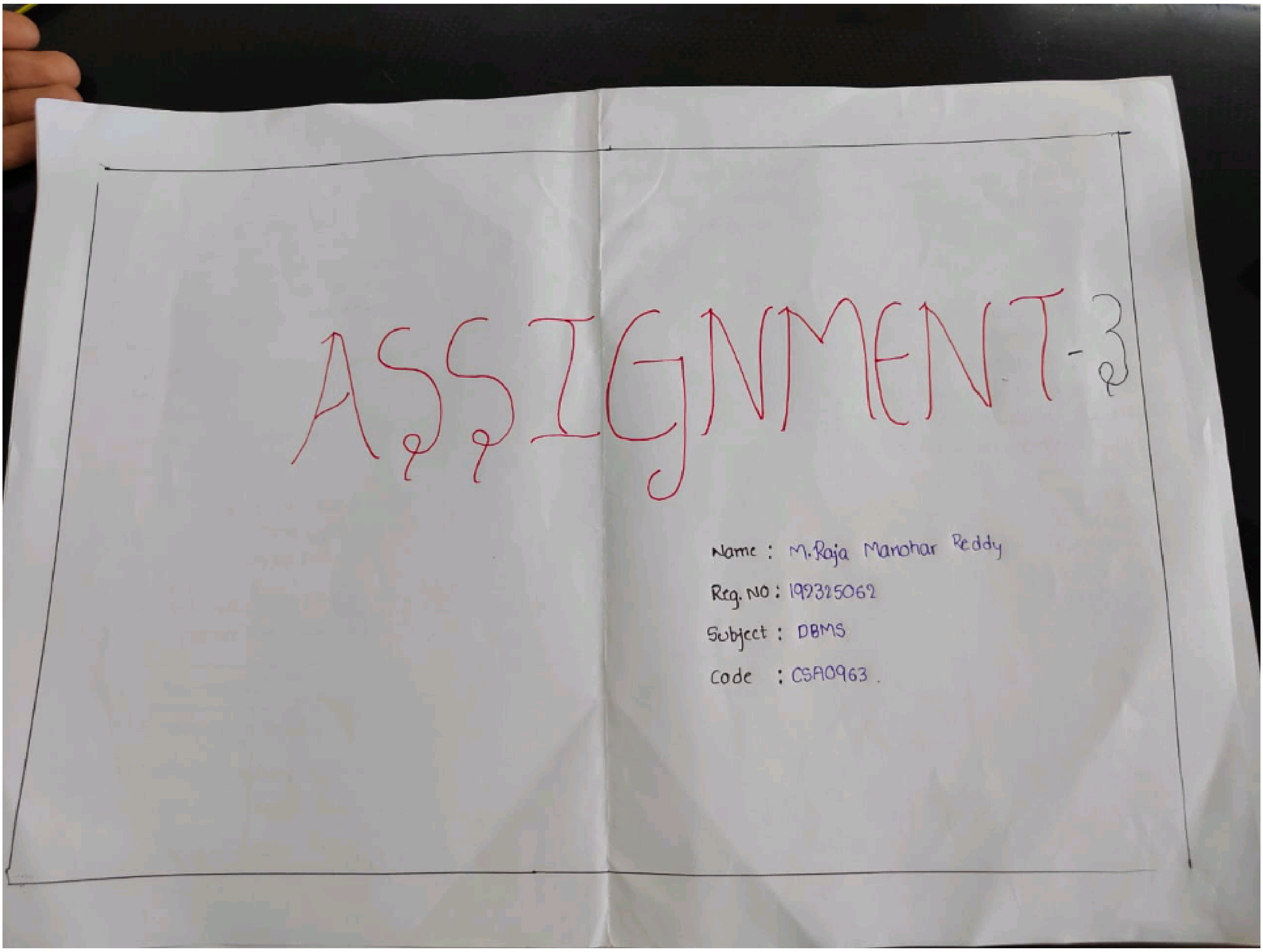


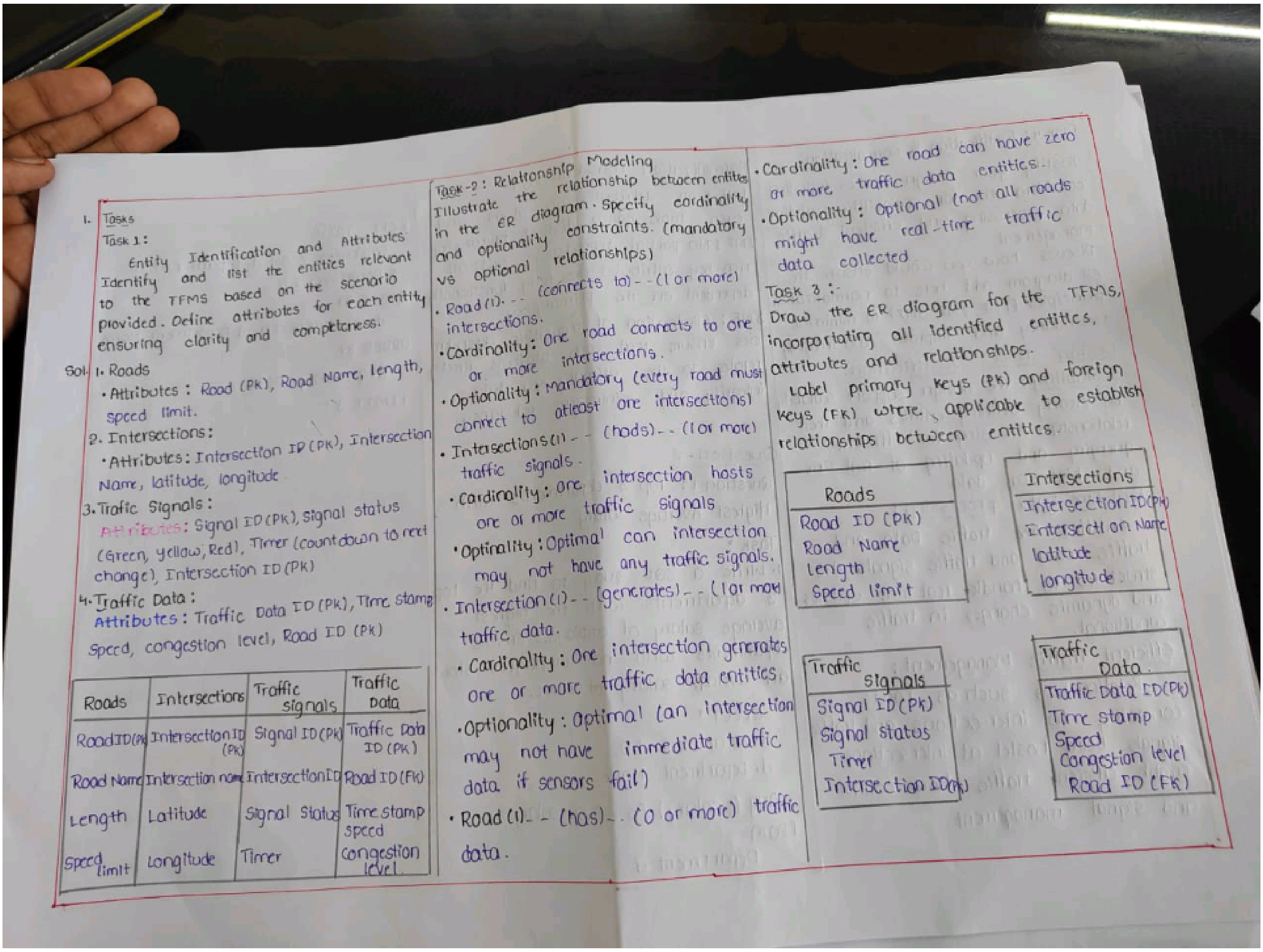
ASSIGNMENT-3

Name : M.Raja Manohar Reddy

Reg. NO : 192325062

Subject : DBMS

Code : CSA0963.



1. Tasks

Task 1:

Entity Identification and Attributes

Identify and list the entities relevant to the TFMS based on the scenario provided. Define attributes for each entity ensuring clarity and completeness.

Sol. 1. Roads

- Attributes : Road (PK), Road Name, length, speed limit.

2. Intersections:

- Attributes: Intersection ID (PK), Intersection Name, latitude, longitude

3. Trafic Signals:

Attributes: Signal ID (PK), signal status (Green, yellow, Red), Timer (countdown to next change), Intersection ID (PK)

4. Traffic Data:

Attributes: Traffic Data ID (PK), Time stamp Speed, congestion level, Road ID (PK)

Roads	Intersections	Traffic signals	Traffic Data
RoadID(PK)	Intersection ID (PK)	Signal ID (PK)	Traffic Data ID (PK)
Road Name	Intersection name	IntersectionID	Road ID (FK)
Length	Latitude	Signal Status	Time stamp speed
Speed limit	Longitude	Timer	congestion level

Task-2 : Relationship Modeling

Illustrate the relationship between entities in the ER diagram. Specify cordinality and optionality constraints. (mandatory vs optional relationships)

- Road (1) - - (connects to) - - (1 or more) intersections.
- Cardinality: One road connects to one or more intersections.
- Optionality : Mandatory (every road must connect to atleast one intersections)
- Intersections (1) - - (hads) - - (1 or more) traffic signals.
- Cardinality : one intersection hosts one or more traffic signals
- Optinality : Optimal can intersection may not have any traffic signals.
- Intersection (1) - - (generates) - - (1 or more traffic data.
- Cardinality : One intersection generates one or more traffic data entities.
- Optionality : optimal (an intersection may not have immediate traffic data if sensors fail)
- Road (1) - - (has) - - (0 or more) traffic data.
- Cardinality : One road can have zero or more traffic data entities.
- Optionality : Optional (not all roads might have real-time traffic data collected

Task 3 :-

Draw the ER diagram for the TFMS, incorportating all identified entities, attributes and relationships.

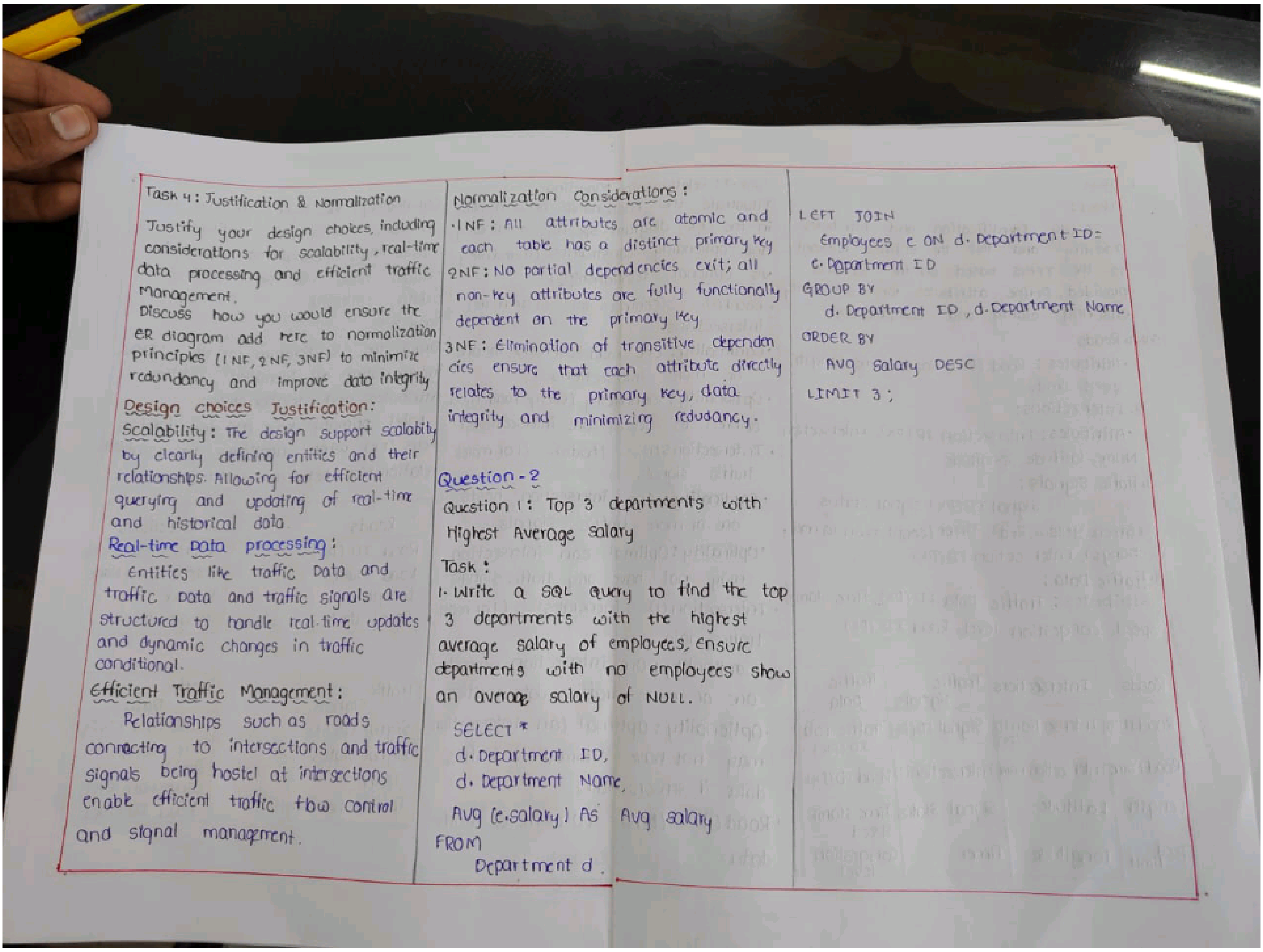
Label primary keys (PK) and foreign keys (FK) where applicable to establish relationships between entities.

Roads
Road ID (PK)
Road Name
length
Speed limit

Intersections
Intersection ID(PK)
Intersection Name
latitude
longitude

Traffic signals
Signal ID (PK)
Signal status
Timer
Intersection ID(PK)

Traffic Data.
Traffic Data ID(PK)
Time stamp
Speed
Congestion level
Road ID (FK)



Task 4 : Justification & Normalization

Justify your design choices, including considerations for scalability, real-time data processing and efficient traffic management.

Discuss how you would ensure the ER diagram add here to normalization principles (1 NF, 2 NF, 3NF) to minimize redundancy and improve data integrity

Design choices Justification:

Scalability : The design support scalabity by clearly defining entities and their relationships. Allowing for efficient querying and updating of real-time and historical data

Real-time Data processing :

Entities like traffic Data and traffic Data and traffic signals are structured to handle real-time updates and dynamic changes in traffic conditional.

Efficient Traffic Management :

Relationships such as roads connecting to intersections and traffic signals being hostel at intersections enable efficient traffic flow control and signal management.

Normalization Considerations :

- 1NF : All attributes are atomic and each table has a distinct primary key

2NF : No partial dependencies exit; all non-key attributes are fully functionally dependent on the primary key

3NF : Elimination of transitive dependencies ensure that each attribute directly relates to the primary key, data integrity and minimizing redudancy.

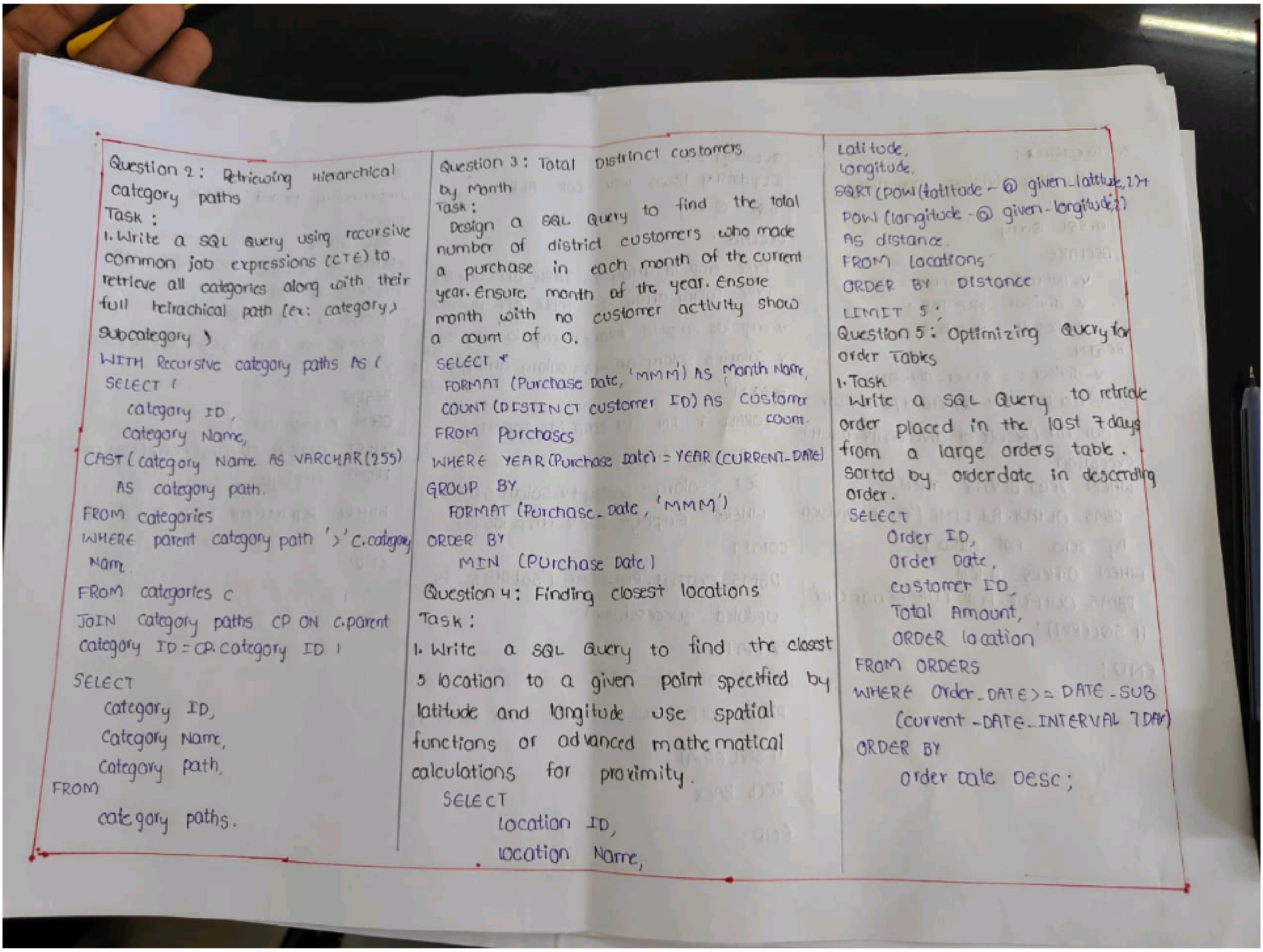
Question - 2

Question 1 : Top 3 departments with Highest Average salary

Task :

1. Write a SQL query to find the top 3 departments with the highest average salary of employees, ensure departments with no employees show an average salary of NULL.

```
SELECT *
d. Department ID,
d. Department Name,
Avg (e.salary) As Avg salary
FROM
    Department d.
LEFT JOIN
  Employees e ON d. Department ID=
 e. Department ID
GROUP BY
   d. Department ID , d. Department Name
ORDER BY
  Avg salary DESC
LIMIT 3 ;
```



Question 2 : Retrieving Hierarchical category paths

Task :

1. Write a SQL Query using recursive common job expressions (CTE) to retrieve all categories along with their full heirachical path (ex: category > Subcategory)

```
WITH Recursive category paths AS (
 SELECT (
   category ID ,
   Category Name,
CAST ( category Name AS VARCHAR(255)
   AS category path.
FROM categories
WHERE parent category path '>' C.category
 Name
FROM categories c
JOIN category paths CP ON c.parent
category ID = CP.category ID )
SELECT
   category ID,
   Category Name,
   Category path,
FROM
   category paths.
```

Question 3 : Total Distrinct customers by Month

Task :

Design a SQL Query to find the total number of district customers who made a purchase in each month of the current year. Ensure month of the year. Ensure month with no customer activity show a count of 0.

```
SELECT *
 FORMAT (Purchase Date, 'MMM') AS Month Name,
 COUNT (DISTINCT customer ID) AS Customer
                                   count.
FROM Purchases
WHERE YEAR (Purchase Date) = YEAR (CURRENT_DATE)
GROUP BY
  FORMAT (Purchase_Date, 'MMM')
ORDER BY
   MIN (Purchase Date)
```

Question 4 : Finding closest locations

Task :

1. Write a SQL Query to find the closest 5 location to a given point specified by latitude and longitude use spatial functions or advanced mathematical calculations for proximity.

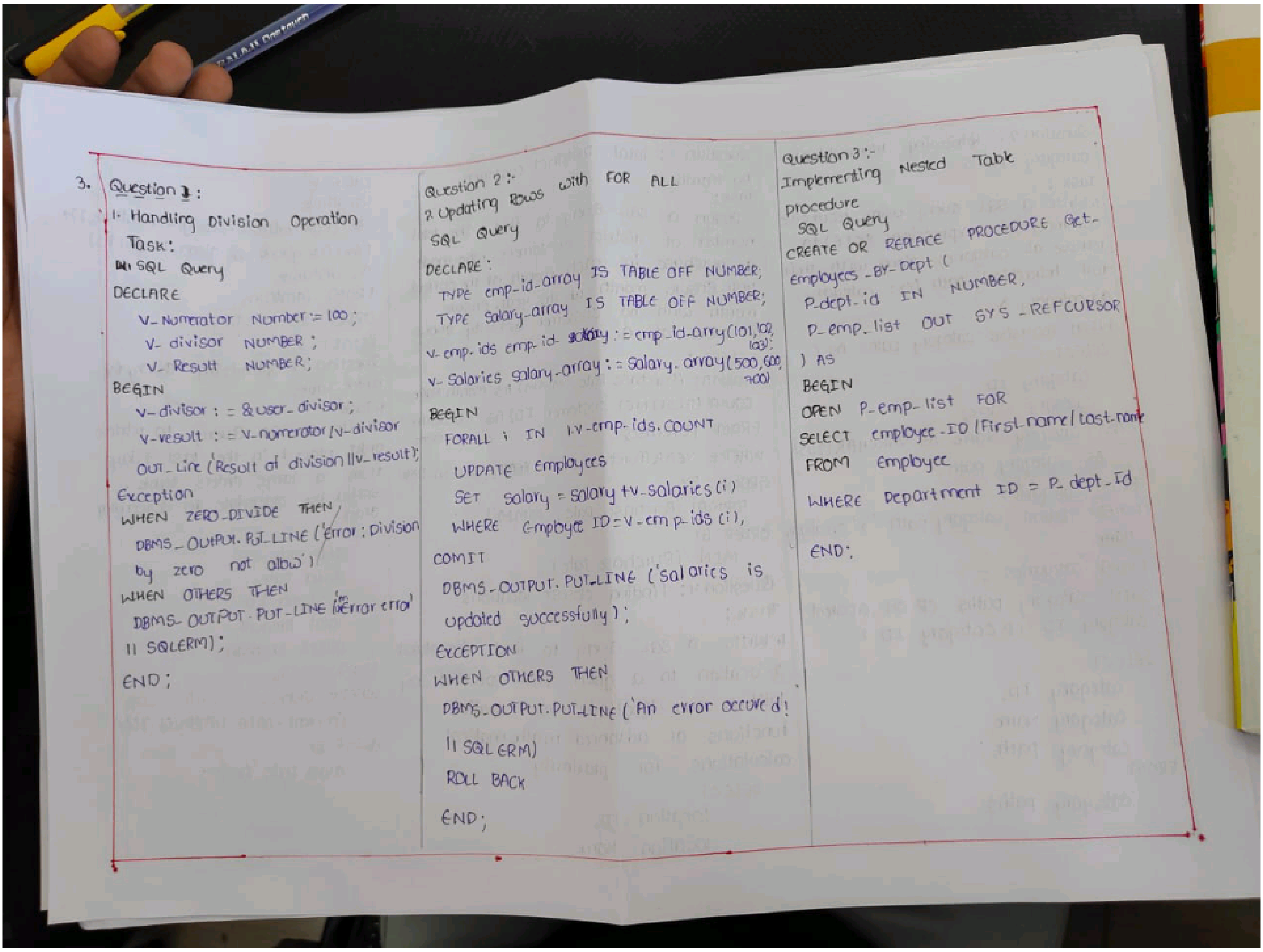
```
 SELECT
      location ID,
      location Name,
Latitude,
longitude,
SQRT (POW (latitude - @ given_latitude, 2) +
 POW (longitude - @ given-longitude, 2)
 AS distance.
 FROM Locations
 ORDER BY Distance
 LIMIT 5 ;
```

Question 5 : Optimizing Query for order Tables

1. Task

Write a SQL Query to retrieve order placed in the last 7 days from a large orders table. Sorted by orderdate in descending order.

```
SELECT
     Order ID,
     Order Date,
     customer ID,
     Total Amount,
     ORDER location
FROM ORDERS
WHERE Order_DATE >= DATE_SUB
    (current -DATE_INTERVAL 7 DAY)
ORDER BY
     order Date DESC ;
```



3. Question 1 :

1. Handling Division Operation Task:

SQL Query

```
DECLARE
   V_Numerator  Number := 100 ;
    V_ divisor  NUMBER ;
    V_ Result  NUMBER;
BEGIN
  v_divisor : = &user_divisor ;
  v-result : = V_numerator /v-divisor
  OUT_Line (Result of division ||v_result);
Exception
 WHEN ZERO_DIVIDE THEN
  DBMS_OUtPUt. PUT_LINE ('Error : Division
  by zero not allow')
 WHEN OTHERS THEN
  DBMS_ OUTPUT . PUT-LINE ('in Error erra'
 || SQLERM);
 END;
```

Question 2 :

2. Updating Rows with FOR ALL

SQL Query

```
DECLARE :
  TYPE emp-id-array IS TABLE OFF NUMBER;
  TYPE Salary-array IS TABLE OFF NUMBER;
v-emp. ids emp-id ~~salary~~ := emp_id-arry(101,102,
                                          103);
v- Salaries salary-array : = Salary. array(500,600,
                                          700)
BEGIN
  FORALL i IN 1.v-emp-ids. COUNT
   UPDATE employees
   SET salary = salary +v_salaries (i)
   WHERE Employee ID = v - em p_ ids (i),
 COMIT
  DBMS_OUTPUT. PUT_LINE ('Salaries is
  updated successfully );
 EXCEPTION
 WHEN OTHERS THEN
  DBMS_OUTPUT.PUT_LINE ('An error occured!
  || SQL ERM)
  ROLL BACK
  END;
```

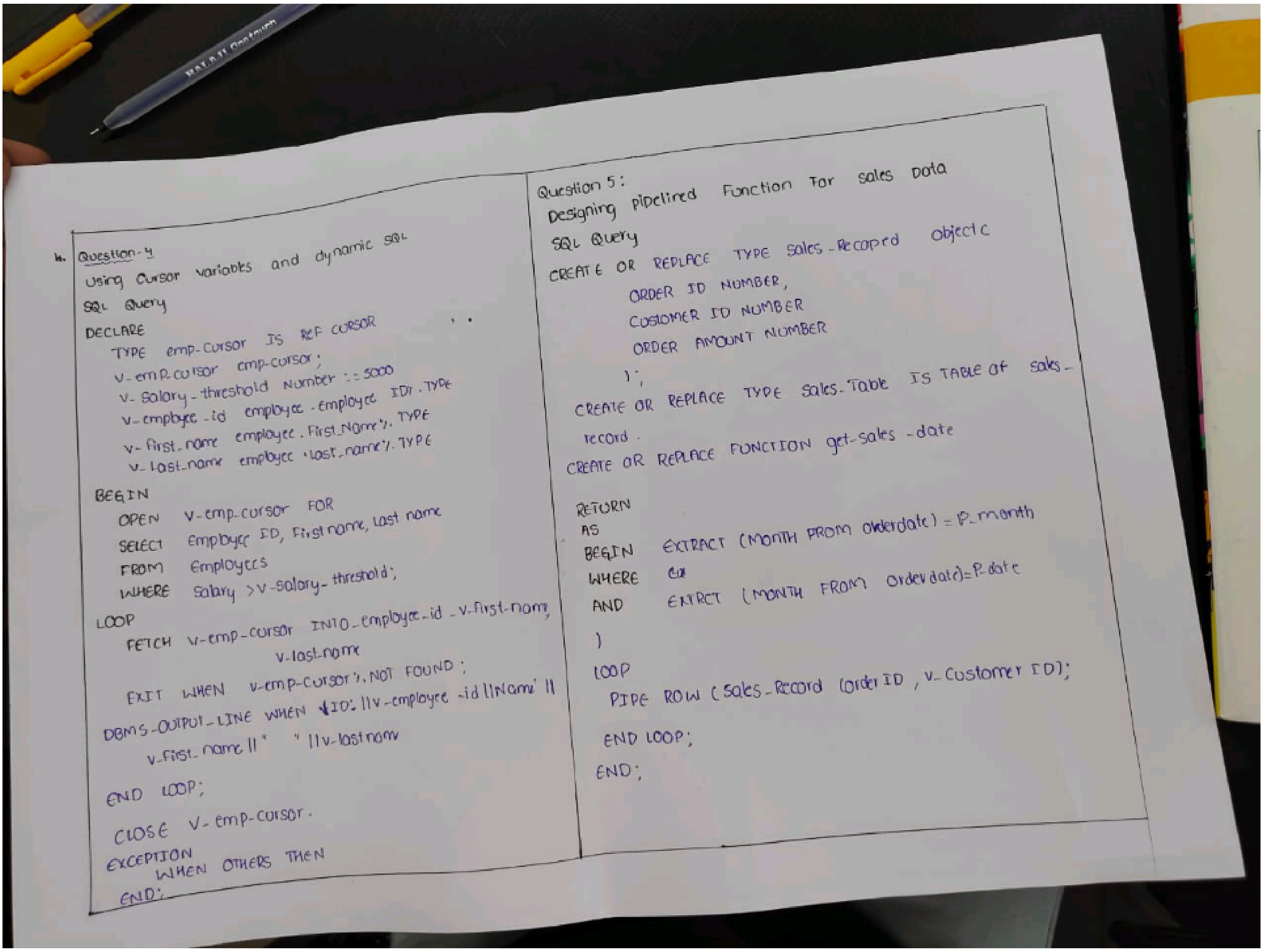
Question 3 :-

Implementing Nested Table procedure

SQL Query

```
CREATE OR REPLACE PROCEDURE Get-
employees _BY_ Dept (
  P_dept_id  IN  NUMBER,
  P_emp_list  OUT  SYS _REFCURSOR
 ) AS
 BEGIN
 OPEN P_emp_list FOR
 SELECT employee_ID /First_name/ Last_name
 FROM employee
 WHERE Department ID = P_ dept _Id

 END;
```



4. Question-4

Using Cursor variables and dynamic SQL

SQL Query

```
DECLARE
  TYPE emp-Cursor IS REF CURSOR
  v-emp.cursor emp-cursor;
  v- Salary - threshold Number := 5000
  v-employee -id employee . employee ID% . TYPE
  v- first_name employee . First_Name%. TYPE
  v- last_name employee .last_name%. TYPE

BEGIN
  OPEN v-emp_cursor FOR
  SELECT Employee ID, First name, Last name
  FROM Employees
  WHERE Salary >v-salary- threshold;
LOOP
  FETCH v-emp_cursor INTO_employee_id _v-First-name,
                        v-last-name

  EXIT WHEN v-emp-Cursor%.NOT FOUND;
DBMS-OUTPUT_LINE WHEN 'ID: ||v-employee -id ||Name' ||
    v_First_name ||" " ||v-lastname

END LOOP;

CLOSE V- emp-cursor.
EXCEPTION
    WHEN OTHERS THEN
END;
```

Question 5:

Designing pipelined Function For sales Data

SQL Query

```
CREATE OR REPLACE TYPE Sales _Recopred objectc
        ORDER ID NUMBER,
        CUSTOMER ID NUMBER
        ORDER AMOUNT NUMBER
       );
  CREATE OR REPLACE TYPE Sales_Table IS TABLE of sales_
   record .
 CREATE OR REPLACE FUNCTION get-sales -date

  RETURN
  AS
  BEGIN  EXTRACT (MONTH FROM orderdate) = P_month
   WHERE  ~~ca~~
   AND    EXTRCT (MONTH FROM Orderdate)=P_date

   )
   LOOP
    PIPE ROW (Sales_Record (order ID , v_Customer ID);

   END LOOP;
  END;
```