Hello, my name is Karina Marlenne Rodriguez Miranda. Thank you for taking the time to review my work. Below, you will find the results of my technical assessment

TASK 1:

To create DepartmentTable:

```
CREATE TABLE DepartmentTable(
    Id int NOT NULL UNIQUE,
    DeptName varchar(255) NOT NULL,
    Location varchar(255),
    PRIMARY KEY (Id)
);
```

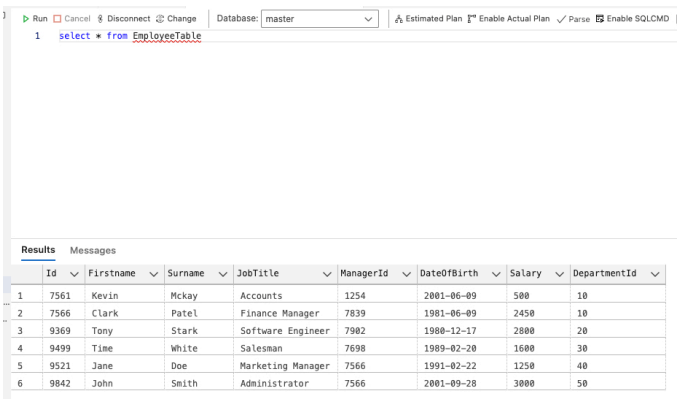
To create EmployeeTable:

```
CREATE TABLE EmployeeTable(
    Id int NOT NULL UNIQUE,
    Firstname VARCHAR(255) NOT NULL,
    Surname varchar(255),
    JobTitle varchar(255),
    ManagerId int,
    DateOfBirth DATE ,
    Salary int,
    DepartmentId int,
    PRIMARY KEY (Id),
    CONSTRAINT FK_DepartmentId FOREIGN KEY (DepartmentId)
    REFERENCES DepartmentTable (Id)
);
```

Please write separate T-SQL statements for the following requirements (one for each requirement)

1. Return all columns and rows of the EmployeeTable

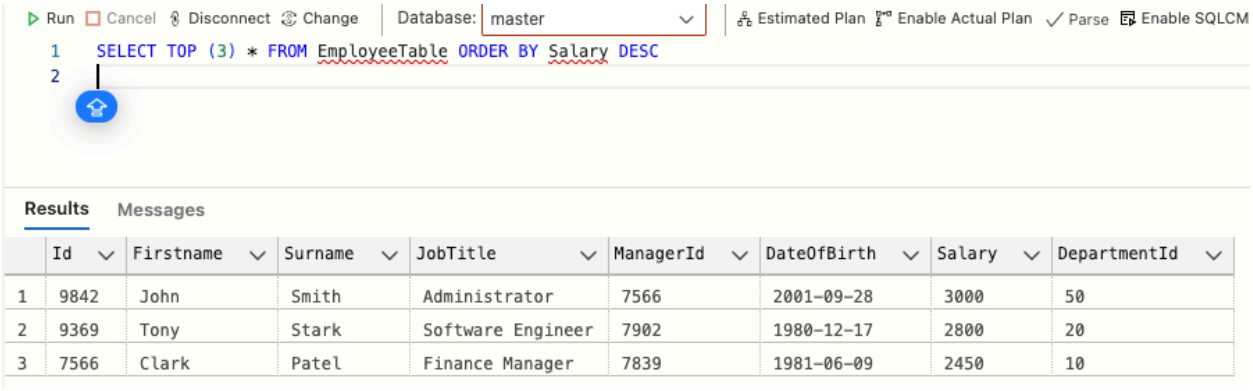
```
SELECT * FROM EmployeeTable
```



	Id	Firstname	Surname	JobTitle	ManagerId	DateOfBirth	Salary	DepartmentId
1	7561	Kevin	Mckay	Accounts	1254	2001-06-09	500	10
2	7566	Clark	Patel	Finance Manager	7839	1981-06-09	2450	10
3	9369	Tony	Stark	Software Engineer	7902	1980-12-17	2800	20
4	9499	Time	White	Salesman	7698	1989-02-20	1600	30
5	9521	Jane	Doe	Marketing Manager	7566	1991-02-22	1250	40
6	9842	John	Smith	Administrator	7566	2001-09-28	3000	50

2. Return all columns of the3 highest paid employees

```
SELECT TOP 3 * FROM EmployeeTable ORDER BY Salary DESC
```



Run Cancel Disconnect Change Database: master Estimated Plan Enable Actual Plan Parse Enable SQLCM

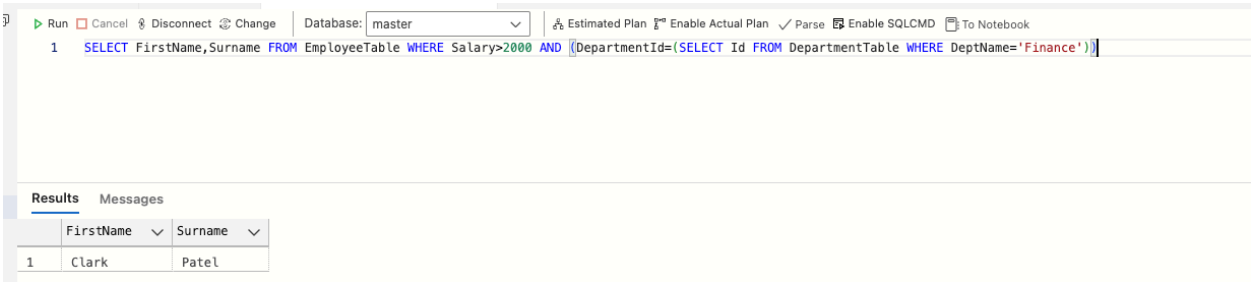
```
1  SELECT TOP (3) * FROM EmployeeTable ORDER BY Salary DESC
2
```

Results Messages

	Id	Firstname	Surname	JobTitle	ManagerId	DateOfBirth	Salary	DepartmentId
1	9842	John	Smith	Administrator	7566	2001-09-28	3000	50
2	9369	Tony	Stark	Software Engineer	7902	1980-12-17	2800	20
3	7566	Clark	Patel	Finance Manager	7839	1981-06-09	2450	10

3. Return [Firstname] and [Surname] of the person(s) with salary greater than 2000

```
SELECT FirstName,Surname FROM EmployeeTable WHERE
Salary>2000 AND (DepartmentId=(SELECT Id FROM
DepartmentTable WHERE DeptName='Finance'))
```



Run Cancel Disconnect Change Database: master Estimated Plan Enable Actual Plan Parse Enable SQLCMD To Notebook

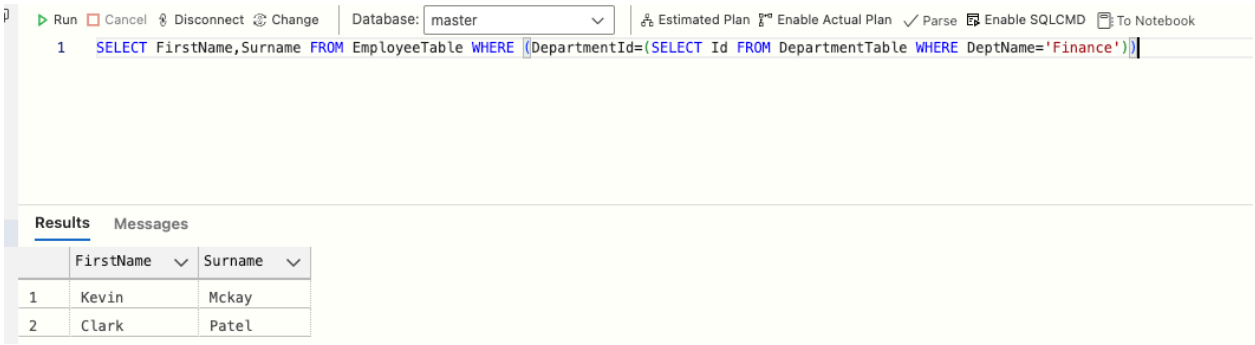
```
1  SELECT FirstName,Surname FROM EmployeeTable WHERE Salary>2000 AND (DepartmentId=(SELECT Id FROM DepartmentTable WHERE DeptName='Finance'))
```

Results Messages

	FirstName	Surname
1	Clark	Patel

4. [Firstname] and [Surname] of the person(s) in the Finance department

```
SELECT FirstName,Surname FROM EmployeeTable WHERE
(DepartmentId=(SELECT Id FROM DepartmentTable WHERE
DeptName='Finance'))
```



Run Cancel Disconnect Change Database: master Estimated Plan Enable Actual Plan Parse Enable SQLCMD To Notebook

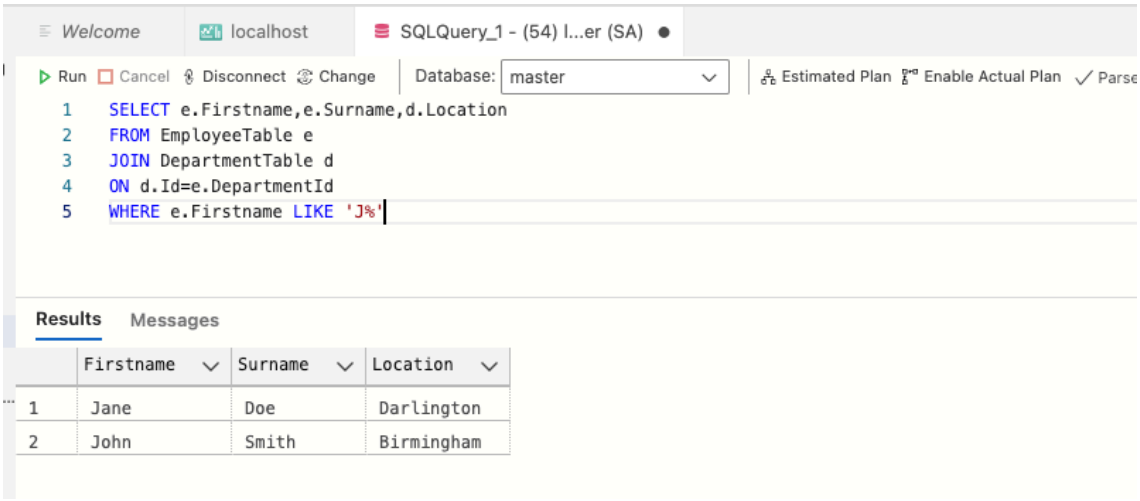
```
1  SELECT FirstName,Surname FROM EmployeeTable WHERE (DepartmentId=(SELECT Id FROM DepartmentTable WHERE DeptName='Finance'))
```

Results Messages

	FirstName	Surname
1	Kevin	Mckay
2	Clark	Patel

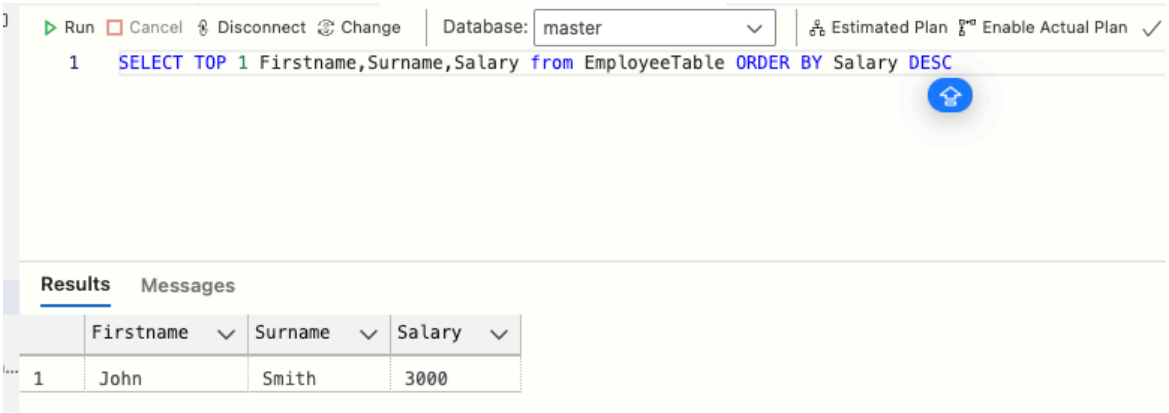
5. Return [Firstname], [Surname] and [Location] of all the employees with first names beginning with “J"

```
SELECT e.Firstname,e.Surname,d.Location
FROM EmployeeTable e
JOIN DepartmentTable d
ON d.Id=e.DepartmentId
WHERE e.Firstname LIKE 'J%'
```



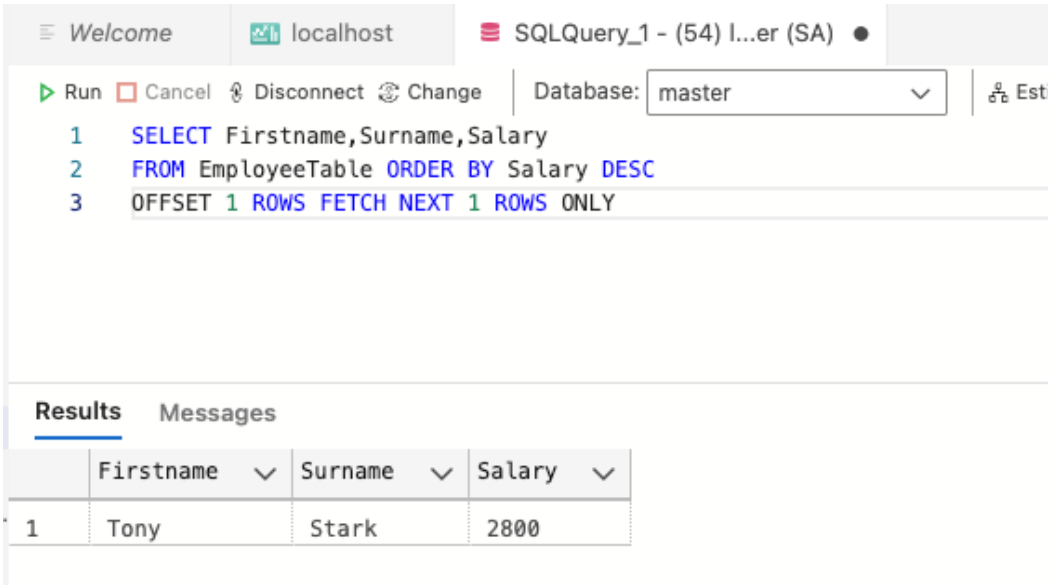
6. [Firstname], [Surname] and [Salary] of the highest earner

```
SELECT TOP 1 Firstname,Surname,Salary from EmployeeTable
ORDER By Salary DESC
```



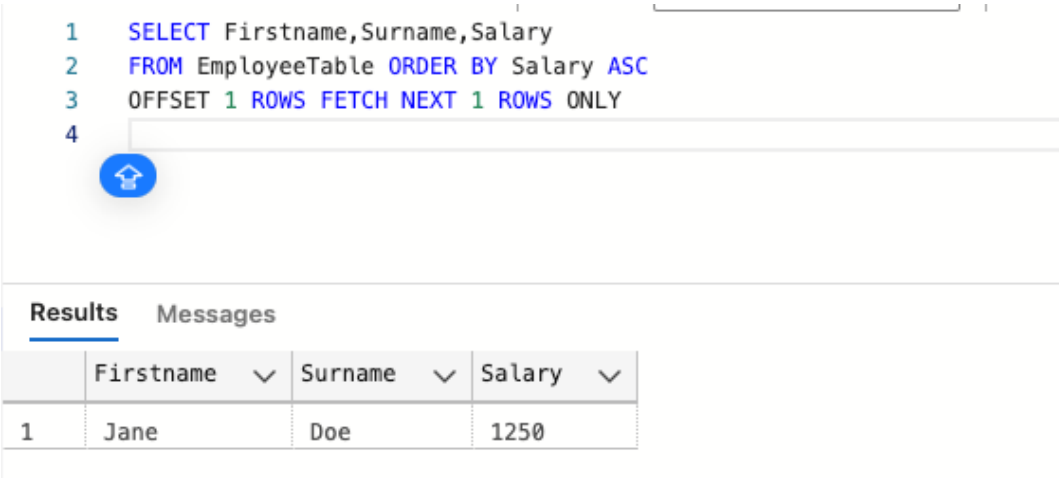
7. Return [Firstname], [Surname] and [Salary] of the 2nd highest earner

```
SELECT  Firstname,Surname,Salary
FROM EmployeeTable ORDER BY Salary DESC
OFFSET 1 ROWS FETCH NEXT 1 ROWS ONLY
```



8. Return [Firstname], [Surname] and [Salary] of the 2nd lowest earner

```
SELECT  Firstname,Surname,Salary
FROM EmployeeTable ORDER BY Salary ASC
OFFSET 1 ROWS FETCH NEXT 1 ROWS ONLY
```



9. Return[Firstname], [Surname] and [Salary] of the person with the highest [Salary] in Edinburgh

```
SELECT TOP 1 Firstname,Surname,Salary
FROM EmployeeTable
WHERE DepartmentId=(SELECT Id FROM DepartmentTable WHERE
Location='Edinburgh')
ORDER BY Salary DESC
```

```
1  SELECT TOP 1 Firstname,Surname,Salary
2  FROM EmployeeTable
3  WHERE DepartmentId=(SELECT Id FROM DepartmentTable WHERE Location='Edinburgh')
4  ORDER BY Salary DESC
5
6
```

Results Messages

	Firstname	Surname	Salary
1	Clark	Patel	2450

10. Return the [Firstname], [Surname] and [Age] of the person(s) with the two highest salaries using the date '2024-06-30'to work out their age.

```
SELECT TOP 2 Firstname,Surname,
DATEDIFF(YEAR,DateOfBirth,'2024-06-30') AS Age
FROM EmployeeTable
ORDER BY Salary DESC
```

```
1  SELECT TOP 2 Firstname,Surname,
2  DATEDIFF(YEAR,DateOfBirth,'2024-06-30') AS Age
3  FROM EmployeeTable
4  ORDER BY Salary DESC
```

Results Messages

	Firstname	Surname	Age
1	John	Smith	23
2	Tony	Stark	44

11. Return the [Firstname], [Surname], [Salary] and [Age] of the person(s) who are older than 30 years using the date '2024-06-30' to work out their age.

```
SELECT Firstname,Surname,Salary,
DATEDIFF(YEAR,DateOfBirth,'2024-06-30') AS Age
FROM EmployeeTable
WHERE DATEDIFF(YEAR,DateOfBirth,'2024-06-30') > 30
```

```
1   SELECT  Firstname,Surname,Salary,
2   DATEDIFF(YEAR,DateOfBirth,'2024-06-30') AS Age
3   FROM EmployeeTable
4   WHERE DATEDIFF(YEAR,DateOfBirth,'2024-06-30') > 30
```

Results Messages

	Firstname	Surname	Salary	Age
1	Clark	Patel	2450	43
2	Tony	Stark	2800	44
3	Time	White	1600	35
4	Jane	Doe	1250	33

TASK 2:
For this task I had problems to understood the request. I don't sure if the output is the expected.

To create SalesTable:

```
CREATE TABLE SalesTable (
    Year INT NOT NULL,
    Month INT NOT NULL,
    Sales DECIMAL(10, 2) NOT NULL )
```

1. Using the lag function show the [Sales] value for the year end and the previous months

```
SELECT Year,Month,Sales,
LAG(Sales,1,0) OVER (PARTITION BY Year ORDER BY Month) AS
PreviousSale
FROM SalesTable
```

```
1  SELECT Year,Month,Sales,
2  LAG(Sales,1,0) OVER (PARTITION BY Year ORDER BY Month) AS PreviousSale
3  FROM SalesTable
```

Results Messages

	Year	Month	Sales	PreviousSale
1	2023	1	10000.00	0.00
2	2023	2	15000.00	10000.00
3	2023	3	12000.00	15000.00
4	2023	4	18000.00	12000.00
5	2023	5	17000.00	18000.00
6	2023	6	16000.00	17000.00
7	2023	7	20000.00	16000.00
8	2023	8	21000.00	20000.00
9	2023	9	19000.00	21000.00
10	2023	10	22000.00	19000.00
11	2023	11	23000.00	22000.00
12	2023	12	24000.00	23000.00

2. Using the lag function show the [Sales] value for the year end and the sales 2 months prior to the year end.

```
WITH SalesWithLag AS (
    SELECT
        Year,
        Month,
        Sales,
        LAG(Sales, 1) OVER (PARTITION BY Year ORDER BY
Month) AS PreviousSale,
        LAG(Sales, 2) OVER (PARTITION BY Year ORDER BY
Month) AS TwoMonthsPriorSales
    FROM SalesTable
)
```

```
SELECT *
FROM SalesWithLag
WHERE Month = 12;
```

```
1   WITH SalesWithLag AS (
2       SELECT
3           Year,
4           Month,
5           Sales,
6           LAG(Sales, 1) OVER (PARTITION BY Year ORDER BY Month) AS PreviousSale,
7           LAG(Sales, 2) OVER (PARTITION BY Year ORDER BY Month) AS TwoMonthsPriorSales
8       FROM SalesTable
9   )
10  SELECT *
11  FROM SalesWithLag
12  WHERE Month = 12;
```

Results Messages

	Year	Month	Sales	PreviousSale	TwoMonthsPriorSales
1	2023	12	24000.00	23000.00	22000.00

TASK 3:

To create MonthEnds:

```
CREATE TABLE MonthEnds (AsAtDate Date NOT NULL)
```

To create Salary:

```
CREATE TABLE Salary
    (EmployeeId int NOT NULL UNIQUE,
    EmployeeName varchar(255),
    Salary decimal
    PRIMARY KEY (EmployeeId))
```

1. Please describe and show the out put of what this TSQL code is doing

```
SELECT
    Monthend.AsAtDate,
    Salary.EmployeeId,
    Salary.EmployeeName,
    Salary.Salary
FROM
    MonthEnds Monthend
CROSS JOIN
    Salary Salary
ORDER BY
    Monthend.AsAtDate, Salary.EmployeeId
```

This request is merging two tables Salary and MonthEnds. To explain better I am splitting the query:

- SELECT: Here is selecting the columns which the user needs, as we are joining two tables it's necessary create an alias for every table. In this example the alias is a similar name that the tables; "Monthend" corresponds to MonthEnds' table and "Salary" to Salary's table. So before the name of each column is necessary the alias's name.
- FROM: We are specifying the table's name base with the respective alias.
- CROSS JOIN: For SQL CROSS JOIN is an operation which joins each row of a tables with the row of another table, with this we can get all the combinations of records. Here we can specify the name of the second table in this case is Salary and its identifier.

  The CROSS JOIN produces a Cartesian product, which means it returns all possible combinations of rows between the two tables.This is useful when you want to view the data of each employee for every month-end date (assuming there is no direct relationship between the MonthEnds and Salary tables).

- ORDER BY: Finally the query sorts the records with two constrainst; first, according the date and second according the EmployeeId. So we can see the salary of all employees in the same month.

```sql
SELECT
    Monthend.AsAtDate,
    Salary.EmployeeId,
    Salary.EmployeeName,
    Salary.Salary
FROM
    MonthEnds Monthend
CROSS JOIN
    Salary Salary
ORDER BY
    Monthend.AsAtDate, Salary.EmployeeId
```

Results Messages

	AsAtDate	EmployeeId	EmployeeName	Salary
1	2023-01-31	1	John Doe	60000
2	2023-01-31	2	Jane Smith	80000
3	2023-02-28	1	John Doe	60000
4	2023-02-28	2	Jane Smith	80000
5	2023-03-31	1	John Doe	60000
6	2023-03-31	2	Jane Smith	80000
7	2023-04-30	1	John Doe	60000
8	2023-04-30	2	Jane Smith	80000
9	2023-05-31	1	John Doe	60000
10	2023-05-31	2	Jane Smith	80000
11	2023-06-30	1	John Doe	60000
12	2023-06-30	2	Jane Smith	80000
13	2023-07-31	1	John Doe	60000
14	2023-07-31	2	Jane Smith	80000
15	2023-08-31	1	John Doe	60000
16	2023-08-31	2	Jane Smith	80000
17	2023-09-30	1	John Doe	60000
18	2023-09-30	2	Jane Smith	80000
19	2023-10-31	1	John Doe	60000
20	2023-10-31	2	Jane Smith	80000
21	2023-11-30	1	John Doe	60000
22	2023-11-30	2	Jane Smith	80000
23	2023-12-31	1	John Doe	60000
24	2023-12-31	2	Jane Smith	80000

TASK 4:

To products:

```
 CREATE TABLE Products (
    product_id int IDENTITY(101,1) PRIMARY KEY,
    product_name varchar(255),
    category varchar(255),
    price decimal
)
```

To Sales;

```
 CREATE TABLE Sales (
    sale_id int IDENTITY(1,1) PRIMARY KEY,
    product_id int NOT NULL,
    sale_date date,
    quantity int,
    total_amount decimal,
    CONSTRAINT FK_Product_id FOREIGN KEY (product_id)
    REFERENCES Products(product_id)
)
```

1. Write a SQL query using a CTE to calculate the total sales amount for each product category. a. The query should return the category, total sales amount, and the number of products sold in that category b. The results should be ordered by the total sales amount in descending order.

```
WITH Sales_CTE AS(
    SELECT * FROM Sales),
Product_CTE AS (
    SELECT * FROM Products)
SELECT
    p.category,
    s.quantity,
    s.total_amount
FROM
    Sales_CTE s
JOIN
    Product_CTE p ON s.product_id=p.product_id
ORDER BY s.total_amount DESC
```

```sql
WITH Sales_CTE AS(
    SELECT * FROM Sales),
Product_CTE AS (
    SELECT * FROM Products)
SELECT
    p.category,
    s.quantity,
    s.total_amount
FROM
    Sales_CTE s
JOIN
    Product_CTE p ON s.product_id=p.product_id
ORDER BY s.total_amount DESC
```

Results Messages

	category	quantity	total_amount
1	Category 3	2	400
2	Category 1	3	300
3	Category 1	2	200
4	Category 2	1	150
5	Category 1	1	100