Problem : File System

File System Size Aggregation with Recursive SQL

Problem Statement

We are given a hierarchical file system where each file or folder is represented in a table called FileSystem. Each item has:

- NodeID: Unique identifier
- NodeName: Name of the file or folder
- ParentID: The ID of the folder it is contained in
- SizeBytes: Size in bytes (only for files; folders have NULL)

Goal:
Calculate the total size of each folder including the sizes of all files and subfolders inside it.

Sample Input Table

NodeID NodeName ParentID SizeBytes
1 Documents NULL NULL
2 Pictures NULL NULL
3 File1.txt 1 500
4 Folder1 NULL NULL
5 Image.jpg 2 1200
6 Subfolder1 4 NULL
7 File2.txt 4 750
8 File3.txt 3 300
9 Folder2 2 NULL
10 File4.txt 9 250

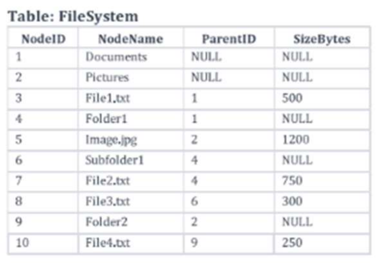
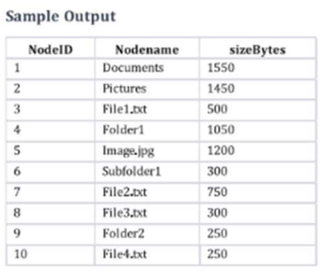


Table: FileSystem

NodeID	NodeName	ParentID	SizeBytes
1	Documents	NULL	NULL
2	Pictures	NULL	NULL
3	File1.txt	1	500
4	Folder1	1	NULL
5	Image.jpg	2	1200
6	Subfolder1	4	NULL
7	File2.txt	4	750
8	File3.txt	6	300
9	Folder2	2	NULL
10	File4.txt	9	250

Expected Output

```
NodeID NodeName SizeBytes
1 Documents 1550
2 Pictures 1450
3 File1.txt 500
4 Folder1 1050
5 Image.jpg 1200
6 Subfolder1 750
7 File2.txt 750
8 File3.txt 300
9 Folder2 250
10 File4.txt 250
```



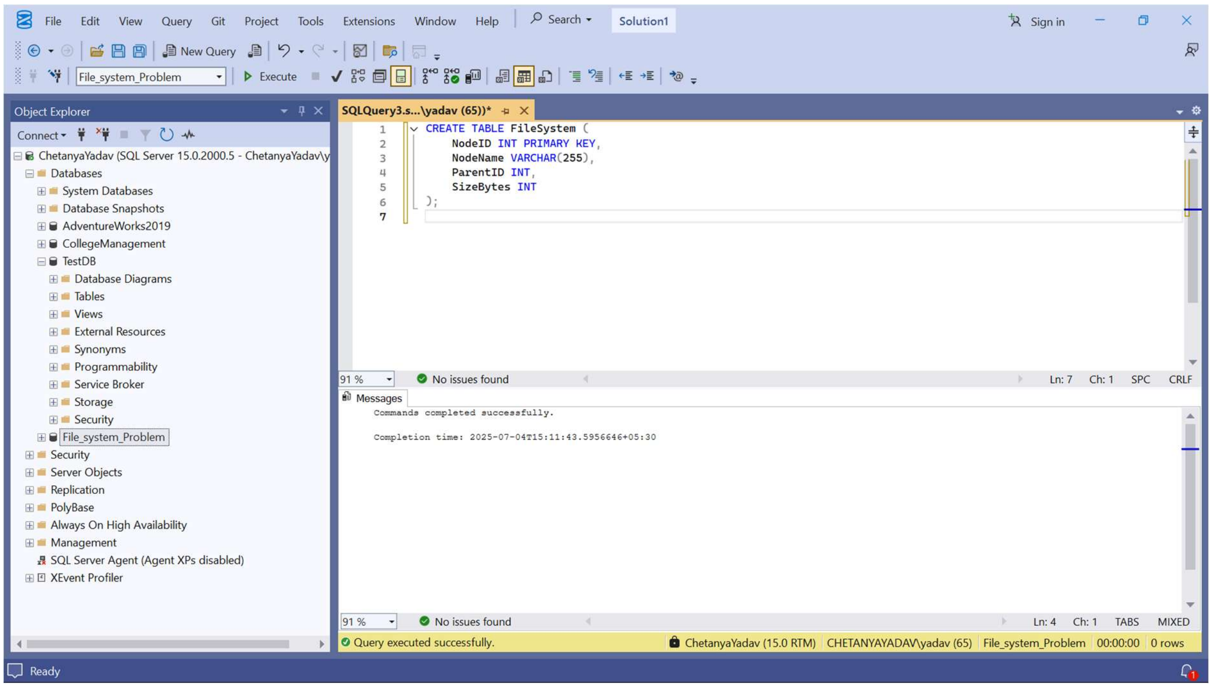
Sample Output

NodeID	Nodename	sizeBytes
1	Documents	1550
2	Pictures	1450
3	File1.txt	500
4	Folder1	1050
5	Image.jpg	1200
6	Subfolder1	300
7	File2.txt	750
8	File3.txt	300
9	Folder2	250
10	File4.txt	250

SQL Server Solution

Create Table

```
CREATE TABLE FileSystem (
NodeID INT PRIMARY KEY,
NodeName VARCHAR(100),
ParentID INT,
SizeBytes INT
);
```



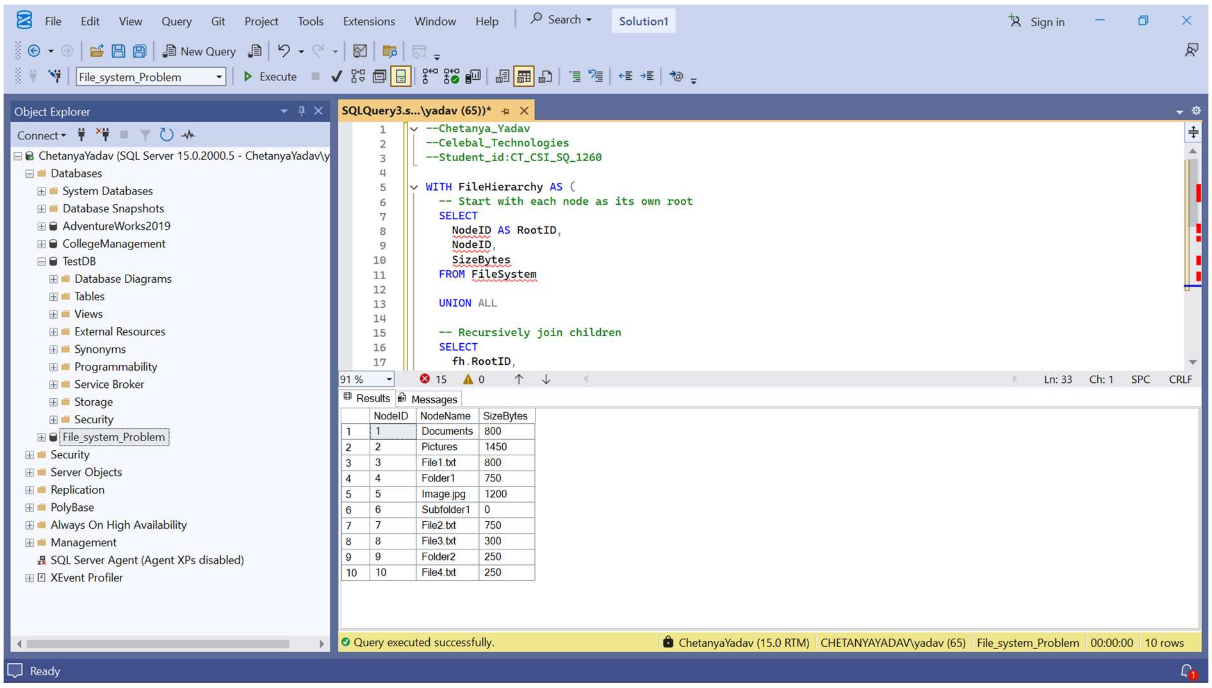
```
CREATE TABLE FileSystem (
    NodeID INT PRIMARY KEY,
    NodeName VARCHAR(255),
    ParentID INT,
    SizeBytes INT
);
```

```
Commands completed successfully.

Completion time: 2025-07-04T15:11:43.5956646+05:30
```

Query executed successfully. ChetanyaYadav (15.0 RTM) CHETANYAYADAV\yadav (65) File_system_Problem 00:00:00 0 rows

Insert Sample Data

INSERT INTO FileSystem (NodeID, NodeName, ParentID, SizeBytes) VALUES
(1, 'Documents', NULL, NULL),
(2, 'Pictures', NULL, NULL),
(3, 'File1.txt', 1, 500),
(4, 'Folder1', NULL, NULL),
(5, 'Image.jpg', 2, 1200),
(6, 'Subfolder1', 4, NULL),
(7, 'File2.txt', 4, 750),
(8, 'File3.txt', 3, 300),
(9, 'Folder2', 2, NULL),
(10, 'File4.txt', 9, 250);



```sql
--Chetanya_Yadav
--Celebal_Technologies
--Student_id:CT_CSI_SQ_1260

WITH FileHierarchy AS (
  -- Start with each node as its own root
  SELECT
    NodeID AS RootID,
    NodeID,
    SizeBytes
  FROM FileSystem

  UNION ALL

  -- Recursively join children
  SELECT
    fh.RootID,
```

	NodeID	NodeName	SizeBytes
1	1	Documents	800
2	2	Pictures	1450
3	3	File1.txt	800
4	4	Folder1	750
5	5	Image.jpg	1200
6	6	Subfolder1	0
7	7	File2.txt	750
8	8	File3.txt	300
9	9	Folder2	250
10	10	File4.txt	250

Query executed successfully. ChetanyaYadav (15.0 RTM) CHETANYAYADAV\yadav (65) File_system_Problem 00:00:00 10 rows

Recursive Query (for SQL Server)

WITH FileHierarchy AS (
SELECT
NodeID AS RootID,
NodeID,
SizeBytes
FROM FileSystem

UNION ALL

SELECT
fh.RootID,
fs.NodeID,
fs.SizeBytes

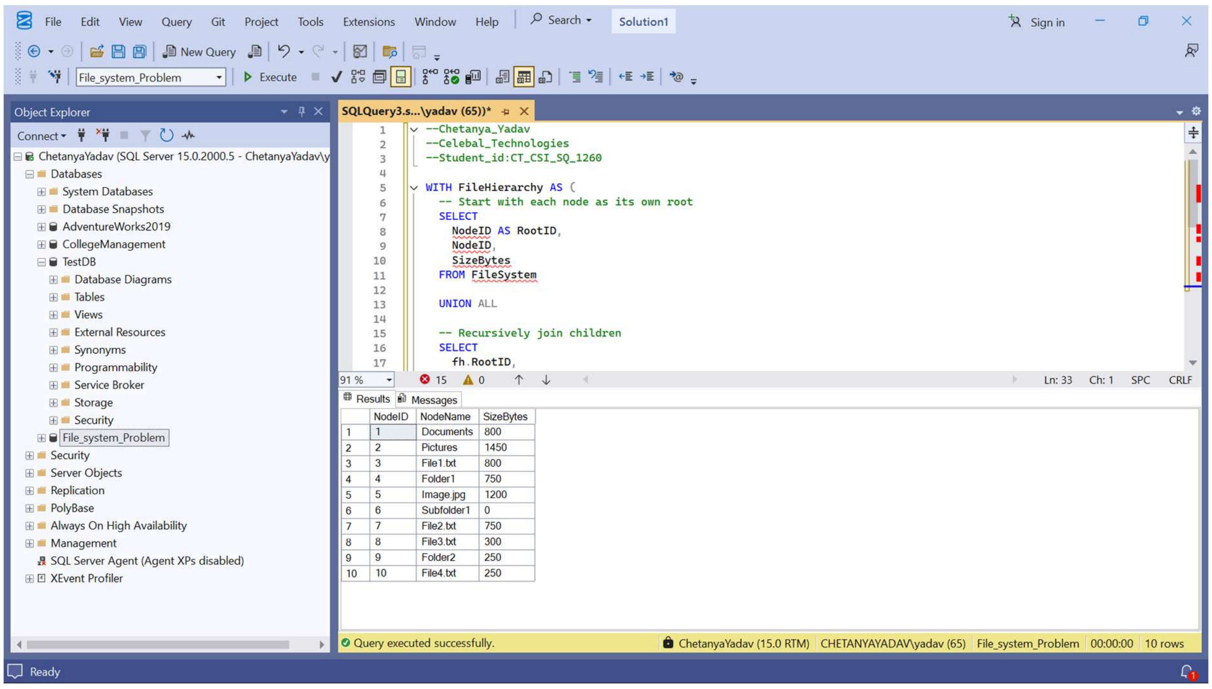
```
FROM FileHierarchy fh
JOIN FileSystem fs ON fs.ParentID = fh.NodeID
)

SELECT
fs.NodeID,
fs.NodeName,
SUM(COALESCE(fh.SizeBytes, 0)) AS SizeBytes
FROM FileHierarchy fh
JOIN FileSystem fs ON fs.NodeID = fh.RootID
GROUP BY fs.NodeID, fs.NodeName
ORDER BY fs.NodeID;
```

Explanation

- We use a recursive CTE to get all children and sub-children under each node.
- For each root node, we calculate the total size by summing up all SizeBytes of its descendant files.
- COALESCE(SizeBytes, 0) handles folders (which have NULL size).

Tools Used

- SQL Server
- Recursive CTE (Common Table Expressions)



SQLQuery3.s...\yadav (65))*

```
--Chetanya_Yadav
--Celebal_Technologies
--Student_id:CT_CSI_SQ_1260

WITH FileHierarchy AS (
  -- Start with each node as its own root
  SELECT
    NodeID AS RootID,
    NodeID,
    SizeBytes
  FROM FileSystem

  UNION ALL

  -- Recursively join children
  SELECT
    fh.RootID,
```

	NodeID	NodeName	SizeBytes
1	1	Documents	800
2	2	Pictures	1450
3	3	File1.txt	800
4	4	Folder1	750
5	5	Image.jpg	1200
6	6	Subfolder1	0
7	7	File2.txt	750
8	8	File3.txt	300
9	9	Folder2	250
10	10	File4.txt	250

Query executed successfully. ChetanyaYadav (15.0 RTM) CHETANYAYADAV\yadav (65) File_system_Problem 00:00:00 10 rows

How to Use

1. **Clone the Repository**

git clone: https://github.com/ChetanyaYadav/Celebal-Internship/tree/main/Week6_Assignment_Celebal_Technology/Leetcode_Performed_Questions

cd LeetCode_SQL_Problems

2. **Open in Your SQL Editor** Use any SQL engine (MySQL, PostgreSQL, SQLite, etc.) and open the files from each category.

3. **Practice by Modifying Queries** Try tweaking the queries to understand better, run explain plans, or adapt them to your own datasets.

4. **Cross-reference with LeetCode** Each SQL file corresponds directly to a problem on LeetCode, so you can test solutions live on the platform.

Technologies Used

- **SQL** – Core language for all queries
- **Git** – Version control and collaboration
- **Markdown** – For documentation

Contributing

If you find errors, improvements, or want to add more problems:

1. Fork this repo
2. Create a new branch
3. Submit a Pull Request

Your contributions are welcome!

Contact:

Author: Chetanya Yadav
Email: yadavchetanya111@gmail.com
LinkedIn: https://www.linkedin.com/in/chetanya-yadav-07a048207/

ID: CT_CSI_SQ_1260
GitHub: https://github.com/ChetanyaYadav