SQL TASK – 4 - StudentManagement Database

1. Create Database and Table

SQL Query:

```
CREATE DATABASE Students_New;
USE Students_New;

CREATE TABLE Student_Mark(
                StudentID INT AUTO_INCREMENT PRIMARY KEY,
                Name VARCHAR(250),
                MathScore INT,
                TotalScore INT);
```

***Purpose**: Creates a new database and a table for storing student academic data.*

***Observation**: Establishes the structure to store detailed student information.*

2. Insert Sample Records

SQL Query:

```
INSERT INTO Students (Name,MathScore,TotalScore) VALUES
('Abhishek', 71, 200),
('Alice', 85, 250),
('Anirudh', 69, 199),
('Basil', 88, 240),
('Binu', 60, 185),
('Cyna', 75, 184),
('Danniel', 88, 200),
('Diya', 93, 273),
('Fadalu', 95, 275),
('Fathima', 68, 184);
```

***Purpose**: Populates the Students table with 10 records for analysis.*

***Observation**: Data is varied by math and total scores.*

3. Display All Students

SQL Query:

```
SELECT * FROM Student_Mark;
```

StudentID	Name	MathScore	TotalScore
1	Abhishek	71	200
2	Alice	85	250
3	Anirudh	69	199
4	Basil	88	240
5	Binu	60	185
6	Cyna	75	184
7	Danniel	88	200
8	Diya	93	273
9	Fadalu	95	275
10	Fathima	68	184

***Purpose**: Displays all student records.*

***Observation**: Useful for reviewing the inserted data.*

4. Rank Students by Total Scores

SQL Query:

```
SELECT
  student_id, name, MathScore, TotalScore, RANK() OVER (ORDER BY TotalScore DESC AS
Student_Ranking
  FROM Student_Mark;
```

***Purpose**: Rank students based on their total scores.*

***Observation**: Ranks students by academic performance based on total scores. Effectively highlights top scorers and handles ties fairly by skipping ranks where necessary.*

***Query Explanation**: Uses RANK() OVER (ORDER BY TotalScore DESC) to assign a rank to each student based on their total score. The ORDER BY clause sorts students from highest to lowest score. If two students have the same score, they receive the same rank, and the next rank is skipped.*

StudentID	Name	MathScore	TotalScore	Student_Ranking
9	Fadalu	95	275	1
8	Diya	93	273	2
2	Alice	85	250	3
4	Basil	88	240	4
1	Abhishek	71	200	5
7	Danniel	88	200	5
3	Anirudh	69	199	7
5	Binu	60	185	8
6	Cyna	75	184	9
10	Fathima	68	184	9

5. Running Totals for Math Scores

SQL Query:

```
SELECT StudentID,Name, MathScore, TotalScore, RANK() OVER (ORDER BY TotalScore
DESC) AS Student_Ranking FROM Student_Mark;
```

***Purpose**: Calculates the cumulative sum of Math scores in the order of StudentID to track progressive performance.*

***Observation**: Shows how Math scores accumulate student by student, making it easier to track performance trends over time.*

***Query Explanation**: Uses SUM(MathScore) OVER (ORDER BY StudentID) to calculate a running (cumulative) total of Math scores. The OVER() clause defines the order in which rows are processed, ensuring that each student's score is added to the total in StudentID order.*

StudentID	Name	MathScore	RunningTotal_MathScore
1	Abhishek	71	71
2	Alice	85	156
3	Anirudh	69	225
4	Basil	88	313
5	Binu	60	373
6	Cyna	75	448
7	Danniel	88	536
8	Diya	93	629
9	Fadalu	95	724
10	Fathima	68	792