To address the given requirements using MapReduce, we need to assume that the input data consists of student records, each having the following fields:

- `roll\_no`: The roll number of the student.

- `name`: The name of the student.

- `class`: The class of the student (e.g., "TE", "SE", "BE").

- `dept`: The department of the student (e.g., "Computer Science", "Mechanical").

- `aggregate\_marks`: The aggregate marks of the student (an integer).

### Input Format:

Each record represents a student and looks like this:

```

(roll\_no, name, class, dept, aggregate\_marks)

```

### Task Breakdown:

### 1. \*\*Finding the total marks of students of "TE" class department-wise\*\*

We need to calculate the total marks for students in the "TE" class, grouped by their department.

#### Map Function:

- Emit the department as the key and the aggregate\_marks as the value, only if the student's class is "TE".

#### Reduce Function:

- The reducer will sum the marks for each department, returning the department with the total marks of students in the "TE" class.

\*\*Map Function:\*\*

```python

def map\_function(student):

roll\_no, name, class\_, dept, aggregate\_marks = student

# Emit the department as key and aggregate\_marks as value for class "TE"

if class\_ == "TE":

emit(dept, aggregate\_marks)

```

\*\*Reduce Function:\*\*

```python

def reduce\_function(dept, marks):

# Sum the aggregate marks for each department

total\_marks = sum(marks)

return (dept, total\_marks)

```

### 2. \*\*Finding the highest marks of students of "SE" class department-wise\*\*

We need to find the highest marks for students in the "SE" class, grouped by their department.

#### Map Function:

- Emit the department as the key and the aggregate\_marks as the value, only if the student's class is "SE".

#### Reduce Function:

- The reducer will find the maximum marks for each department.

\*\*Map Function:\*\*

```python

def map\_function(student):

roll\_no, name, class\_, dept, aggregate\_marks = student

# Emit the department as key and aggregate\_marks as value for class "SE"

if class\_ == "SE":

emit(dept, aggregate\_marks)

```

\*\*Reduce Function:\*\*

```python

def reduce\_function(dept, marks):

# Find the maximum marks for each department

highest\_marks = max(marks)

return (dept, highest\_marks)

```

### 3. \*\*Find Average marks of students of "BE" class department-wise\*\*

We need to calculate the average marks for students in the "BE" class, grouped by their department.

#### Map Function:

- Emit the department as the key and the aggregate\_marks as the value, only if the student's class is "BE".

#### Reduce Function:

- The reducer will calculate the average marks for each department.

\*\*Map Function:\*\*

```python

def map\_function(student):

roll\_no, name, class\_, dept, aggregate\_marks = student

# Emit the department as key and aggregate\_marks as value for class "BE"

if class\_ == "BE":

emit(dept, aggregate\_marks)

```

\*\*Reduce Function:\*\*

```python

def reduce\_function(dept, marks):

# Calculate the average marks for each department

avg\_marks = sum(marks) / len(marks) if marks else 0

return (dept, avg\_marks)

```

### Example:

#### Input Data:

```python

[

(101, "John", "TE", "Computer Science", 85),

(102, "Alice", "TE", "Mechanical", 78),

(103, "Bob", "SE", "Computer Science", 90),

(104, "Charlie", "BE", "Mechanical", 88),

(105, "David", "SE", "Computer Science", 92),

(106, "Eve", "BE", "Computer Science", 95),

(107, "Frank", "TE", "Mechanical", 70),

(108, "Grace", "BE", "Mechanical", 85)

]

```

#### Task 1 Output (Total Marks of "TE" Class Students Department-wise):

```python

[

("Computer Science", 85),

("Mechanical", 148)

]

```

#### Task 2 Output (Highest Marks of "SE" Class Students Department-wise):

```python

[

("Computer Science", 92)

]

```

#### Task 3 Output (Average Marks of "BE" Class Students Department-wise):

```python

[

("Mechanical", 86.5),

("Computer Science", 90)

]

```

### Explanation:

1. \*\*Task 1 - Total Marks of Students of "TE" Class Department-wise\*\*:

- The mapper filters out students who are not in the "TE" class and emits the department and the marks.

- The reducer sums the marks for each department, giving the total marks for all "TE" class students in that department.

2. \*\*Task 2 - Highest Marks of Students of "SE" Class Department-wise\*\*:

- The mapper filters out students who are not in the "SE" class and emits the department and the marks.

- The reducer finds the highest marks for each department, giving the highest marks for "SE" class students in that department.

3. \*\*Task 3 - Average Marks of Students of "BE" Class Department-wise\*\*:

- The mapper filters out students who are not in the "BE" class and emits the department and the marks.

- The reducer calculates the average marks for each department, which is the sum of marks divided by the number of students in the department.

### Conclusion:

By using the MapReduce paradigm, we efficiently process large datasets in a parallel manner. The map functions filter the data based on the class (TE, SE, or BE), and the reduce functions perform aggregations (sum, max, or average) for each department. This enables us to solve the problem of calculating total, highest, and average marks for students in various departments across different classes.