Name	Departments	Salary
Krish	Data Science	10000
Krish	IOT	5000
Mahesh	Big Data	4000
Krish	Big Data	4000
Mahesh	Data Science	3000
Sudhansu	Data Science	20000
Sudhansu	IOT	10000
Sudhansu	Big Data	5000
Sunny	Data Science	10000
Sunny	Big Data	2000

Sample Table

GroupBy & Aggregate Functionality:-

groupBy() method is used for this functionality.

GroupBy and aggregate function work together, i.e, first we apply groupBy(), and then apply aggregate functions.

Some of the aggregate functions are :-
avg(), count(), min(), max(), sum(), cogroup(), ...

Suppose, in the above sample table, if we want to find out the person with max salary (where salary of person is the sum of salaries he is

getting from all the departments he belongs to), then the code will be as follow :-

```
df_spark.groupBy('Name').sum().show()
```

we want to group by the "Name" column, as of each person, we want sum of all his salaries

aggregate function.

4 decimals

sum() works only on integers, and not on strings. So, the only integer column here is "Salary", so that will be summed up.

In the result, the Name (groupBy applied column) and Salary column will be shown.

⇒ Salary is shown as it is the column in which aggregate function worked.

⇒ Department is not shown as it is un-affected by aggregate function.

Output of above code :-

Name	sum(salary)
Sudhanshu	35000
Sunny	12000
Krish	19000
Mahesh	7000

⊛ GroupBy Departments which gives max salary :-

```
df_spark.groupBy("Departments").sum().show()
```

o/p →

Departments	sum(salary)
IOT	15000
Big Data	15000
Data Science	43000

⊛ Mean salary of each department :-

```
df_spark.groupBy("Departments").mean().show()
```

o/p →

Departments	sum(salary)
IOT	7500.0
Big Data	3750.0
Data Science	10750.0

⊛ No. of employees in each department :-

```
df_spark.groupBy("Departments").count().show()
```

o/p →

Departments	sum(salary)
IOT	2
Big Data	4
Data Science	4

Using aggregate function directly on entire column :-

If we want to apply aggregation on an entire column, we use the agg() method.

Ex:-

Suppose, we want the total sum of all the employees salary. It is done as :-

```
df_spark.agg({'Salary': 'sum'}).show()
```

'Salary' : column name on which aggregation is to be applied.

'sum' : aggregate function name

o/p →

sum(Salary)
73000

agg() method takes key-value pair / dictionary as the argument.

The key is the column name

The value is the aggregate function name