TATA CONSULTANCY SERVICES

SQL Functions – Analytic Functions

Version 1.0

3.2 Analytic Functions

Analytic functions compute an aggregate value based on a group of rows. The general syntax of analytic functions are:

Syntax :

analytic_function([arguments]) OVER (analytic_clause)

The analytic_clause breaks down into the following optional elements. [query_partition_clause] [order_by_clause [windowing_clause]]

Following are the analytic functions described in this document

- Row_number()
- Rank()
- Dense_rank()
- Decode()
- case

3.2.1 ROW_NUMBER()

ROW_NUMBER( ) gives a running serial number to a partition of records. It is very useful in reporting, especially in places where different partitions have their own serial numbers.

Syntax :

Row_Number( ) Over([Partition By Expression1 , ... [N]]
Order_By Expression1 , ... [N])

Consider the following table

Example :

```
SQL> select * from Student_Marks;

STUDENT_ID STUDENT_NAME                TOTAL_MARKS BATCH
---------- ------------------------- ----------- ---------
      1001 Aarathi Sharma                      381 A1001
      1003 Lakshman K                          327 A1001
      1006 Silpa Sukul                         307 B2001
      1002 Zenith Sam                          307 A1001
      1005 Marithi Gunja                       355 B2001
      1007 Priya Mayi                          266 B2001
      1004 Jiyah Jigar                         171 A1001
      1008 Gopika Nakul                        355 B2001
      1009 Veena Vanji                         367 C3001
      1010 Mundhan Hubba                       302 C3001
      1011 Rohit Jayan                         302 C3001

STUDENT_ID STUDENT_NAME                TOTAL_MARKS BATCH
---------- ------------------------- ----------- ---------
      1012 Meena Kayak                         285 C3001

12 rows selected.
```

Provide serial numbers to the students based on their total marks

select m.student_name, m.total_marks, m.batch,
row_number() over (order by m.total_marks) Serial_Number
from Student_Marks m;

```
SQL> select m.student_name, m.total_marks, m.batch,
  2  row_number() over (order by m.total_marks) Serial_Number
  3  from Student_Marks m;

STUDENT_NAME              TOTAL_MARKS BATCH           SERIAL_NUMBER
------------------------- ----------- --------------- -------------
Jiyah Jigar                       171 A1001                       1
Priya Mayi                        266 B2001                       2
Meena Kayak                       285 C3001                       3
Mundhan Hubba                     302 C3001                       4
Rohit Jayan                       302 C3001                       5
Silpa Sukul                       307 B2001                       6
Zenith Sam                        307 A1001                       7
Lakshman K                        327 A1001                       8
Marithi Gunja                     355 B2001                       9
Gopika Nakul                      355 B2001                      10
Veena Vanji                       367 C3001                      11

STUDENT_NAME              TOTAL_MARKS BATCH           SERIAL_NUMBER
------------------------- ----------- --------------- -------------
Aarathi Sharma                    381 A1001                      12

12 rows selected.
```

Provide serial numbers to the students based on their batch

select m.student_name, m.total_marks, m.batch,
row_number() over (order by m.batch) Serial_Number
from Student_Marks m;

```
SQL> select m.student_name, m.total_marks, m.batch,
  2  row_number() over (order by m.batch) Serial_Number
  3  from Student_Marks m;

STUDENT_NAME              TOTAL_MARKS BATCH           SERIAL_NUMBER
------------------------- ----------- --------------- -------------
Jiyah Jigar                       171 A1001                     1
Lakshman K                        327 A1001                     2
Aarathi Sharma                    381 A1001                     3
Zenith Sam                        307 A1001                     4
Marithi Gunja                     355 B2001                     5
Silpa Sukul                       307 B2001                     6
Priya Mayi                        266 B2001                     7
Gopika Nakul                      355 B2001                     8
Veena Vanji                       367 C3001                     9
Mundhan Hubba                     302 C3001                    10
Rohit Jayan                       302 C3001                    11

STUDENT_NAME              TOTAL_MARKS BATCH           SERIAL_NUMBER
------------------------- ----------- --------------- -------------
Meena Kayak                       285 C3001                    12

12 rows selected.
```

Provide serial numbers to the students based on their total marks. Each batch should have their own serial numbers

select m.student_name, m.total_marks, m.batch,
row_number() over (partition by m.batch order by m.total_marks) Serial_Number from Student_Marks m;

```
SQL> select m.student_name, m.total_marks, m.batch,
  2  row_number() over
  3  (partition by m.batch order by m.total_marks) Serial_Number
  4  from Student_Marks m;

STUDENT_NAME              TOTAL_MARKS BATCH           SERIAL_NUMBER
------------------------- ----------- --------------- -------------
Jiyah Jigar                       171 A1001                     1
Zenith Sam                        307 A1001                     2
Lakshman K                        327 A1001                     3
Aarathi Sharma                    381 A1001                     4
Priya Mayi                        266 B2001                     1
Silpa Sukul                       307 B2001                     2
Gopika Nakul                      355 B2001                     3
Marithi Gunja                     355 B2001                     4
Meena Kayak                       285 C3001                     1
Mundhan Hubba                     302 C3001                     2
Rohit Jayan                       302 C3001                     3

STUDENT_NAME              TOTAL_MARKS BATCH           SERIAL_NUMBER
------------------------- ----------- --------------- -------------
Veena Vanji                       367 C3001                     4

12 rows selected.
```

3.2.2 Rank ()

Rank function is used to provide rank to the records based on some column value or expression. In case of a tie of 2 records at position N, RANK declares 2 positions N and skips position N+1 and gives position N+2 to the next record.

Syntax :

```
Rank( ) Over( [ Partition By Expression1 , ... [ N ] ]
                    Order_By Expression1 , ... [ N ] )
```

Example :

Rank the students based on total marks.

```
select m.student_name, m.total_marks, m.batch,
rank() over (order by m.total_marks) Rank
from Student_Marks m;
```

```
SQL> select m.student_name, m.total_marks, m.batch,
  2  rank() over (order by m.total_marks) Rank
  3  from Student_Marks m;

STUDENT_NAME              TOTAL_MARKS BATCH                RANK
------------------------- ----------- --------------- ----------
Jiyah Jigar                       171 A1001                    1
Priya Mayi                        266 B2001                    2
Meena Kayak                       285 C3001                    3
Mundhan Hubba                     302 C3001                    4
Rohit Jayan                       302 C3001                    4
Silpa Sukul                       307 B2001                    6
Zenith Sam                        307 A1001                    6
Lakshman K                        327 A1001                    8
Marithi Gunja                     355 B2001                    9
Gopika Nakul                      355 B2001                    9
Veena Vanji                       367 C3001                   11

STUDENT_NAME              TOTAL_MARKS BATCH                RANK
------------------------- ----------- --------------- ----------
Aarathi Sharma                    381 A1001                   12

12 rows selected.
```

```
select m.student_name, m.total_marks, m.batch,
```

rank() over (order by m.total_marks desc) Rank
from Student_Marks m;

```
SQL> select m.student_name, m.total_marks, m.batch,
  2  rank() over (order by m.total_marks desc) Rank
  3  from Student_Marks m;

STUDENT_NAME           TOTAL_MARKS BATCH           RANK
---------------------- ----------- --------------- ----------
Aarathi Sharma                 381 A1001                    1
Veena Vanji                    367 C3001                    2
Marithi Gunja                  355 B2001                    3
Gopika Nakul                   355 B2001                    3
Lakshman K                     327 A1001                    5
Zenith Sam                     307 A1001                    6
Silpa Sukul                    307 B2001                    6
Mundhan Hubba                  302 C3001                    8
Rohit Jayan                    302 C3001                    8
Meena Kayak                    285 C3001                   10
Priya Mayi                     266 B2001                   11

STUDENT_NAME           TOTAL_MARKS BATCH           RANK
---------------------- ----------- --------------- ----------
Jiyah Jigar                    171 A1001                   12

12 rows selected.
```

Rank the students in each batch based on total marks.

select m.student_name, m.total_marks, m.batch,
rank() over (partition by m.batch order by m.total_marks) Rank
from Student_Marks m;

```
SQL> select m.student_name, m.total_marks, m.batch,
  2  rank() over
  3  (partition by m.batch order by m.total_marks) Rank
  4  from Student_Marks m;

STUDENT_NAME           TOTAL_MARKS BATCH           RANK
---------------------- ----------- --------------- ----------
Jiyah Jigar                    171 A1001                    1
Zenith Sam                     307 A1001                    2
Lakshman K                     327 A1001                    3
Aarathi Sharma                 381 A1001                    4
Priya Mayi                     266 B2001                    1
Silpa Sukul                    307 B2001                    2
Gopika Nakul                   355 B2001                    3
Marithi Gunja                  355 B2001                    3
Meena Kayak                    285 C3001                    1
Mundhan Hubba                  302 C3001                    2
Rohit Jayan                    302 C3001                    2

STUDENT_NAME           TOTAL_MARKS BATCH           RANK
---------------------- ----------- --------------- ----------
Veena Vanji                    367 C3001                    4

12 rows selected.
```

```
select m.student_name, m.total_marks, m.batch,
rank() over
(partition by m.batch order by m.total_marks desc) Rank
from Student_Marks m;
```

```
SQL> select m.student_name, m.total_marks, m.batch,
  2  rank() over
  3  (partition by m.batch order by m.total_marks desc) Rank
  4  from Student_Marks m;

STUDENT_NAME              TOTAL_MARKS BATCH               RANK
------------------------- ----------- --------------- ----------
Aarathi Sharma                    381 A1001                    1
Lakshman K                        327 A1001                    2
Zenith Sam                        307 A1001                    3
Jiyah Jigar                       171 A1001                    4
Marithi Gunja                     355 B2001                    1
Gopika Nakul                      355 B2001                    1
Silpa Sukul                       307 B2001                    3
Priya Mayi                        266 B2001                    4
Veena Vanji                       367 C3001                    1
Mundhan Hubba                     302 C3001                    2
Rohit Jayan                       302 C3001                    2

STUDENT_NAME              TOTAL_MARKS BATCH               RANK
------------------------- ----------- --------------- ----------
Meena Kayak                       285 C3001                    4

12 rows selected.
```

3.2.3 Dense_Rank()

Dense_Rank function is also used to provide rank to the records. In case of a tie of 2 records at position N, dense rank declares 2 positions N and gives position N+1 to the next record. Dense_Rank t does not skip any positions.

Syntax :

```
Dense_Rank( ) Over( [ Partition By Expression1 , ... [ N ] ]
                    Order_By Expression1 , ... [ N ] )
```

Example :

Rank the students based on total marks.

```
select m.student_name, m.total_marks, m.batch,
dense_rank() over (order by m.total_marks desc) dense_Rank
```

```
from Student_Marks m;
```

```
SQL> select m.student_name, m.total_marks, m.batch,
  2  dense_rank() over (order by m.total_marks desc) dense_rank
  3  from Student_Marks m;

STUDENT_NAME              TOTAL_MARKS BATCH           DENSE_RANK
------------------------- ----------- --------------- ----------
Aarathi Sharma                    381 A1001                   1
Veena Vanji                       367 C3001                   2
Marithi Gunja                     355 B2001                   3
Gopika Nakul                      355 B2001                   3
Lakshman K                        327 A1001                   4
Zenith Sam                        307 A1001                   5
Silpa Sukul                       307 B2001                   5
Mundhan Hubba                     302 C3001                   6
Rohit Jayan                       302 C3001                   6
Meena Kayak                       285 C3001                   7
Priya Mayi                        266 B2001                   8

STUDENT_NAME              TOTAL_MARKS BATCH           DENSE_RANK
------------------------- ----------- --------------- ----------
Jiyah Jigar                       171 A1001                   9

12 rows selected.
```

Rank the students in each batch based on total marks.

```
select m.student_name, m.total_marks, m.batch,
dense_rank() over
(partition by m.batch order by m.total_marks desc) dense_Rank
from Student_Marks m;
```

```
SQL> select m.student_name, m.total_marks, m.batch,
  2  dense_rank() over
  3  (partition by m.batch order by m.total_marks desc) dense_rank
  4  from Student_Marks m;

STUDENT_NAME              TOTAL_MARKS BATCH           DENSE_RANK
------------------------- ----------- --------------- ----------
Aarathi Sharma                    381 A1001                   1
Lakshman K                        327 A1001                   2
Zenith Sam                        307 A1001                   3
Jiyah Jigar                       171 A1001                   4
Marithi Gunja                     355 B2001                   1
Gopika Nakul                      355 B2001                   1
Silpa Sukul                       307 B2001                   2
Priya Mayi                        266 B2001                   3
Veena Vanji                       367 C3001                   1
Mundhan Hubba                     302 C3001                   2
Rohit Jayan                       302 C3001                   2

STUDENT_NAME              TOTAL_MARKS BATCH           DENSE_RANK
------------------------- ----------- --------------- ----------
Meena Kayak                       285 C3001                   3

12 rows selected.
```

3.2.3.1 Row_number(), Rank() and Dense_Rank() - A Comparison

With out Partition

```
select
m.batch,
m.student_name,
m.total_marks,
rank() over (order by m.total_marks desc) rank ,
dense_rank() over (order by m.total_marks desc) dense_rank ,
row_number() over (order by m.total_marks desc) rank
from Student_Marks m;
```

```
SQL> select
  2  m.batch,
  3  m.student_name,
  4  m.total_marks,
  5  rank() over (order by m.total_marks desc) rank ,
  6  dense_rank() over (order by m.total_marks desc) dense_rank ,
  7  row_number() over (order by m.total_marks desc) row_no
  8  from Student_Marks m;

BATCH           STUDENT_NAME               TOTAL_MARKS       RANK DENSE_RANK     ROW_NO
--------------- ------------------------- ----------- ---------- ---------- ----------
A1001           Aarathi Sharma                    381          1          1          1
C3001           Veena Vanji                       367          2          2          2
B2001           Marithi Gunja                     355          3          3          3
B2001           Gopika Nakul                      355          3          3          4
A1001           Lakshman K                        327          5          4          5
A1001           Zenith Sam                        307          6          5          6
B2001           Silpa Sukul                       307          6          5          7
C3001           Mundhan Hubba                     302          8          6          8
C3001           Rohit Jayan                       302          8          6          9
C3001           Meena Kayak                       285         10          7         10
B2001           Priya Mayi                        266         11          8         11
A1001           Jiyah Jigar                       171         12          9         12

12 rows selected
```

With Partition

```
select
m.batch,
m.student_name,
m.total_marks,
rank() over (partition by m.batch order by m.total_marks desc) rank ,
dense_rank() over (partition by m.batch order by
```

```
m.total_marks desc) dense_rank ,
row_number() over (partition by m.batch order by
m.total_marks desc) rank
from Student_Marks m;
```

```
SQL> select
  2  m.batch,
  3  m.student_name,
  4  m.total_marks,
  5  rank() over (partition by m.batch order by m.total_marks desc) rank ,
  6  dense_rank() over (partition by m.batch order by m.total_marks desc) dense_rank ,
  7  row_number() over (partition by m.batch order by m.total_marks desc) row_no
  8  from Student_Marks m;

BATCH           STUDENT_NAME              TOTAL_MARKS       RANK DENSE_RANK     ROW_NO
--------------- ------------------------- ----------- ---------- ---------- ----------
A1001           Aarathi Sharma                    381          1          1          1
A1001           Lakshman K                        327          2          2          2
A1001           Zenith Sam                        307          3          3          3
A1001           Jiyah Jigar                       171          4          4          4
B2001           Marithi Gunja                     355          1          1          1
B2001           Gopika Nakul                      355          1          1          2
B2001           Silpa Sukul                       307          3          2          3
B2001           Priya Mayi                        266          4          3          4
C3001           Veena Vanji                       367          1          1          1
C3001           Mundhan Hubba                     302          2          2          2
C3001           Rohit Jayan                       302          2          2          3
C3001           Meena Kayak                       285          4          3          4

12 rows selected
```

3.2.4 Ratio_to_Report

It computes the ratio of a value to the sum of a set of values. If expr evaluates to null, then the ratio-to-report value also evaluates to null. The set of values is determined by the query_partition_clause. If you omit that clause, then the ratio-to-report is computed over all rows returned by the query.

Syntax:

Ratio_to_report(expression) over (query_partition_clause)

Example:

Calculate the ratio of each student's mark to the total of all students' marks.

```
select m.student_name, m.total_marks, m.batch ,
ratio_to_report(m.total_marks) over() Ratio
```

```
from student_marks m;
```

```
SQL> select m.student_name, m.total_marks, m.batch ,
  2  ratio_to_report(m.total_marks) over() Ratio
  3  from student_marks m;

STUDENT_NAME           TOTAL_MARKS BATCH           RATIO
---------------------- ----------- --------------- ----------
Aarathi Sharma                 381 A1001           .102281879
Lakshman K                     327 A1001           .087785235
Silpa Sukul                    307 B2001           .082416107
Zenith Sam                     307 A1001           .082416107
Marithi Gunja                  355 B2001           .095302013
Priya Mayi                     266 B2001           .071409396
Jiyah Jigar                    171 A1001            .04590604
Gopika Nakul                   355 B2001           .095302013
Veena Vanji                    367 C3001            .09852349
Mundhan Hubba                  302 C3001           .081073826
Rohit Jayan                    302 C3001           .081073826

STUDENT_NAME           TOTAL_MARKS BATCH           RATIO
---------------------- ----------- --------------- ----------
Meena Kayak                    285 C3001           .076510067

12 rows selected.
```

Calculate the ratio of each student's mark to the total of all students' marks in a batch.

```
select m.student_name, m.total_marks, m.batch ,
ratio_to_report(m.total_marks) over(partition by m.batch) Ratio
from student_marks m;
```

```
SQL> select m.student_name, m.total_marks, m.batch ,
  2  ratio_to_report(m.total_marks) over(partition by m.batch) Ratio
  3  from student_marks m;

STUDENT_NAME           TOTAL_MARKS BATCH           RATIO
---------------------- ----------- --------------- ----------
Jiyah Jigar                    171 A1001           .144182125
Lakshman K                     327 A1001           .275716695
Aarathi Sharma                 381 A1001           .321247892
Zenith Sam                     307 A1001           .258853288
Marithi Gunja                  355 B2001           .276695246
Silpa Sukul                    307 B2001           .239282931
Priya Mayi                     266 B2001           .207326578
Gopika Nakul                   355 B2001           .276695246
Veena Vanji                    367 C3001           .292197452
Mundhan Hubba                  302 C3001            .24044586
Rohit Jayan                    302 C3001            .24044586

STUDENT_NAME           TOTAL_MARKS BATCH           RATIO
---------------------- ----------- --------------- ----------
Meena Kayak                    285 C3001           .226910828

12 rows selected.
```

3.2.5 Analytic Functions and Aggregate functions – A comparison

Analytic functions and Aggregate Functions compute an aggregate value based on a group of rows. Analytic functions differ from aggregate functions in that they return multiple rows for each group while aggregate functions return only single row per group.

Group	Source data	Output for Aggregate funtions (sum)	Output for analytic functions sum() over ()
Group 1	1	10	10
	3		10
	1		10
	5		10
Group 2	1	7	7
	2		7
	3		7
	1		7
Group 3	1	4	4
	1		4
	1		4
	1		4
Group 4	2	12	12
	3		12
	7		12

Note: For an aggregate function , there will be only one output record per group. But for an analytic function, there will be a one output record per one input record.

Example:

Consider the table

```
SQL> select * from TC_Marks;

SUBJECT                   STUDENT_ID      MARKS
------------------------- ---------- ----------
Maths                           1001         98
Language                        1001         94
Science                         1001         97
Env Science                     1001         92
Maths                           1002         63
Language                        1002         89
Science                         1002         95
Env Science                     1002         60
Maths                           1003         85
Language                        1003         79
Science                         1003         63

SUBJECT                   STUDENT_ID      MARKS
------------------------- ---------- ----------
Env Science                     1003        100
Maths                           1004         71
Language                        1004         25
Science                         1004         46
Env Science                     1004         29
Maths                           1005         52
Language                        1005         68
Science                         1005         26
Env Science                     1005         83
Maths                           1006         91
Language                        1006         96

SUBJECT                   STUDENT_ID      MARKS
------------------------- ---------- ----------
Science                         1006         73
Env Science                     1006         47
Maths                           1007         48
Language                        1007         39
Science                         1007         80
Env Science                     1007         99

28 rows selected.
```

Query 1:

```
select
sum(m.marks)SUM,
avg(m.marks)AVERAGE,
max(m.marks)HIGHEST
from tc_marks m;
```

```
SQL> select
  2  sum(m.marks)SUM,
  3  avg(m.marks)AVERAGE,
  4  max(m.marks)HIGHEST
  5  from tc_marks m;

       SUM    AVERAGE    HIGHEST
---------- ---------- ----------
      1988         71        100
```

Here the query has aggregate functions and they calculate the overall sum,

average and highest value for all the records and displays the result. Even though there are 28 input records, there is only one output record.

Query 2:

```
select
m.student_id,
m.marks,
sum(m.marks) over() SUM,
avg(m.marks) over() AVERAGE ,
max(m.marks) over() HIGHEST
from tc_marks m;
```

```
SQL> select
  2  m.student_id,
  3  m.marks,
  4  sum(m.marks) over() SUM,
  5  avg(m.marks) over() AVERAGE ,
  6  max(m.marks) over() HIGHEST
  7  from tc_marks m;

STUDENT_ID      MARKS        SUM    AVERAGE    HIGHEST
---------- ---------- ---------- ---------- ----------
      1001         98       1988         71        100
      1001         94       1988         71        100
      1001         97       1988         71        100
      1001         92       1988         71        100
      1002         63       1988         71        100
      1002         89       1988         71        100
      1002         95       1988         71        100
      1002         60       1988         71        100
      1003         85       1988         71        100
      1003         79       1988         71        100
      1003         63       1988         71        100

STUDENT_ID      MARKS        SUM    AVERAGE    HIGHEST
---------- ---------- ---------- ---------- ----------
      1003        100       1988         71        100
      1004         71       1988         71        100
      1004         25       1988         71        100
      1004         46       1988         71        100
      1004         29       1988         71        100
      1005         52       1988         71        100
      1005         68       1988         71        100
      1005         26       1988         71        100
      1005         83       1988         71        100
      1006         91       1988         71        100
      1006         96       1988         71        100

STUDENT_ID      MARKS        SUM    AVERAGE    HIGHEST
---------- ---------- ---------- ---------- ----------
      1006         73       1988         71        100
      1006         47       1988         71        100
      1007         48       1988         71        100
      1007         39       1988         71        100
      1007         80       1988         71        100
      1007         99       1988         71        100

28 rows selected.
```

Here the query has analytic functions and they calculate the overall sum,

average and highest value for all the records and the values are appended to each and every record. Here there are 28 output records also

Query 3:

```
select
m.student_id,
sum(m.marks)SUM,
avg(m.marks)AVERAGE,
max(m.marks)HIGHEST
from tc_marks m
group by m.student_id;
```

```
SQL> select
  2  m.student_id,
  3  sum(m.marks)SUM,
  4  avg(m.marks)AVERAGE,
  5  max(m.marks)HIGHEST
  6  from tc_marks m
  7  group by m.student_id;

STUDENT_ID        SUM    AVERAGE    HIGHEST
---------- ---------- ---------- ----------
      1003        327      81.75        100
      1006        307      76.75         96
      1001        381      95.25         98
      1002        307      76.75         95
      1007        266       66.5         99
      1004        171      42.75         71
      1005        229      57.25         83

7 rows selected.
```

Here the aggregate functions calculate the sum, average and highest value for all the groups (student_id) and displays the result.

Query 4:

```
select
m.student_id,
m.marks,
sum(m.marks) over(partition by m.student_id) SUM,
avg(m.marks) over(partition by m.student_id) AVERAGE ,
max(m.marks) over(partition by m.student_id) HIGHEST
from tc_marks m;
```

```
SQL> select
  2  m.student_id,
  3  m.marks,
  4  sum(m.marks) over(partition by m.student_id) SUM,
  5  avg(m.marks) over(partition by m.student_id) AVERAGE ,
  6  max(m.marks) over(partition by m.student_id) HIGHEST
  7  from tc_marks m;

STUDENT_ID      MARKS        SUM    AVERAGE    HIGHEST
---------- ---------- ---------- ---------- ----------
      1001         98        381      95.25         98
      1001         94        381      95.25         98
      1001         97        381      95.25         98
      1001         92        381      95.25         98
      1002         63        307      76.75         95
      1002         89        307      76.75         95
      1002         60        307      76.75         95
      1002         95        307      76.75         95
      1003         85        327      81.75        100
      1003         63        327      81.75        100
      1003         79        327      81.75        100

STUDENT_ID      MARKS        SUM    AVERAGE    HIGHEST
---------- ---------- ---------- ---------- ----------
      1003        100        327      81.75        100
      1004         71        171      42.75         71
      1004         25        171      42.75         71
      1004         29        171      42.75         71
      1004         46        171      42.75         71
      1005         52        229      57.25         83
      1005         68        229      57.25         83
      1005         26        229      57.25         83
      1005         83        229      57.25         83
      1006         91        307      76.75         96
      1006         96        307      76.75         96

STUDENT_ID      MARKS        SUM    AVERAGE    HIGHEST
---------- ---------- ---------- ---------- ----------
      1006         47        307      76.75         96
      1006         73        307      76.75         96
      1007         48        266       66.5         99
      1007         39        266       66.5         99
      1007         80        266       66.5         99
      1007         99        266       66.5         99

28 rows selected.
```

Here analytic functions calculate the sum, average and highest value in a group (student_id) and the values are appended to each and every record. Here also there are 28 output records.