Aggregate Queries in Oracle

Aggregate Queries in Oracle work on multiple rows unlike normal functions which work on single rows.

There are 5 aggregate functions in SQL. They are:-

1) Max
   Max works on all data types for which ordering is defined.For numerical types value based ordering is done, for character types dictionary ordering is use, for Date & Time the instance that later is considered to be greater.
2) Min --- Same as above
3) Sum Totals the data. Only works for numerical types.
4) Avg finds the average. Only works for numerical types.
5) Count .Counts the number of records. Works for all data types.

We create the following table to illustrate the concepts.

create table Cricket_Scores (batsman varchar(100),InningsNo int,MatchType varchar(10),score int)

```
desc Cricket_Scores
```

Results Explain **Describe** Saved SQL History

Object Type **TABLE** Object **CRICKET_SCORES**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
CRICKET_SCORES	BATSMAN	Varchar2	100	-	-	-	✓	-	-
	INNINGSNO	Number	-	-	0	-	✓	-	-
	MATCHTYPE	Varchar2	10	-	-	-	✓	-	-
	SCORE	Number	-	-	0	-	✓	-	-

1 - 4

```
insert into cricket_scores values('Champak',1,'Test',111)
```

Results Explain Describe Saved SQL History

```
1 row(s) inserted.
```

0.00 seconds

Finally we have the table with the following records.

BATSMAN	INNINGSNO	MATCHTYPE	SCORE
Champak	1	Test	111
Champak	1	One Day	112
Gaurav	1	One Day	0
Gaurav	2	One Day	10
Champak	2	One Day	113
Pappu	1	One Day	3
Pappu	2	One Day	2
Pappu	3	One Day	1
Pappu	4	One Day	0
Pappu	5	One Day	0

We try the Queries one by one:-

```
select max(score) from Cricket_Scores
```

Results Explain Describe Saved SQL History

MAX(SCORE)
113

1 rows returned in 0.00 seconds CSV Export

```
select min(score) from Cricket_Scores
```

Results Explain Describe Saved SQL History

MIN(SCORE)
0

1 rows returned in 0.00 seconds CSV Export

```
select sum(score) from Cricket_Scores
```

Results Explain Describe Saved SQL History

SUM(SCORE)
352

1 rows returned in 0.00 seconds CSV Export

```
select avg(score) from Cricket_Scores
```

Results Explain Describe Saved SQL History

AVG(SCORE)
35.2

1 rows returned in 0.00 seconds CSV Export

```
select count(score) from Cricket_Scores
```

Results | Explain | Describe | Saved SQL | History

COUNT(SCORE)
10

1 rows returned in 0.00 seconds CSV Export

Using a aggregate query with other fields.

Let us try the following query:-

```
select batsman,max(score) from Cricket_Scores
```

Results Explain Describe Saved SQL History

ORA-00937: not a single-group group function

In such cases we need to specify the group by option

```
select batsman,max(score) from Cricket_Scores group by batsman
```

Results | Explain | Describe | Saved SQL | History

BATSMAN	MAX(SCORE)
Gaurav	10
Pappu	3
Champak	113

3 rows returned in 0.03 seconds CSV Export

No Oracle creates groups based on batsman and calculates max based on those groups separately.

```
select batsman,max(score),min(score),avg(score),sum(score),count(score) from Cricket_Scores group by batsman
```

Results Explain Describe Saved SQL History

BATSMAN	MAX(SCORE)	MIN(SCORE)	AVG(SCORE)	SUM(SCORE)	COUNT(SCORE)
Gaurav	10	0	5	10	2
Pappu	3	0	1.2	6	5
Champak	113	111	112	336	3

3 rows returned in 0.02 seconds CSV Export

This gives the batsman wise statistics.

Multiple group by clauses can be used.

```
select batsman,matchtype,max(score),min(score),avg(score),sum(score),count(score) from Cricket_Scores group by batsman,matchtype
```

Results Explain Describe Saved SQL History

BATSMAN	MATCHTYPE	MAX(SCORE)	MIN(SCORE)	AVG(SCORE)	SUM(SCORE)	COUNT(SCORE)
Champak	Test	111	111	111	111	1
Champak	One Day	113	112	112.5	225	2
Pappu	One Day	3	0	1.2	6	5
Gaurav	One Day	10	0	5	10	2

4 rows returned in 0.00 seconds CSV Export

The order of group by clauses will not affect the output.

```
select batsman,matchtype,max(score),min(score),avg(score),sum(score),count(score) from Cricket_Scores group by matchtype,batsman
```

Results Explain Describe Saved SQL History

BATSMAN	MATCHTYPE	MAX(SCORE)	MIN(SCORE)	AVG(SCORE)	SUM(SCORE)	COUNT(SCORE)
Gaurav	One Day	10	0	5	10	2
Champak	One Day	113	112	112.5	225	2
Pappu	One Day	3	0	1.2	6	5
Champak	Test	111	111	111	111	1

4 rows returned in 0.01 seconds CSV Export

It does affect the processing time. So, we should specify the groups such that sizes are reduced in the beginning.

Using the having clause.

The having clause is used for providing conditions based on aggregate queries.

Suppose we wish to find the player with max score greater than equal to 100.

```
select batsman,max(score) from cricket_scores  group by batsman having max(score) >=100
```

Results Explain Describe Saved SQL History

BATSMAN	MAX(SCORE)
Champak	113

1 rows returned in 0.01 seconds CSV Export

A query containing where, group by,and having clauses.

```
select batsman,max(score) from cricket_scores where inningsno=1 and matchtype='Test'  group by batsman having max(score) >=100
```

Results Explain Describe Saved SQL History

BATSMAN	MAX(SCORE)
Champak	111

1 rows returned in 0.00 seconds CSV Export

Some questions:-

a) Find the highest scores in Test Matches and One Days

```
select MatchType,max(score) from cricket_scores  group by MatchType
```

Results Explain Describe Saved SQL History

MATCHTYPE	MAX(SCORE)
Test	111
One Day	113

2 rows returned in 0.00 seconds CSV Export

b) Find the highest score in the 1st innings of a batsman:-

```
select Batsman,InningsNo,max(score) from cricket_scores where InningsNo=1  group by Batsman,InningsNo
```

Results Explain Describe Saved SQL History

BATSMAN	INNINGSNO	MAX(SCORE)
Gaurav	1	0
Pappu	1	3
Champak	1	112

3 rows returned in 0.00 seconds CSV Export

Removing the group by on batsman we get.

```
select InningsNo,max(score) from cricket_scores where InningsNo=1  group by InningsNo
```

Results Explain Describe Saved SQL History

INNINGSNO	MAX(SCORE)
1	112

1 rows returned in 0.01 seconds CSV Export

c) Find the batsman with highest average in tests.

```
select batsman,avg(score) from cricket_scores having avg(score)=  (select max(average) from (select batsman,avg(score) as average from cricket_scores  group by batsman))
group by batsman
```

Results Explain Describe Saved SQL History

BATSMAN	AVG(SCORE)
Champak	112

1 rows returned in 0.00 seconds CSV Export

d)