**Ex. No: 3 LOGICAL OPERATORS, AGGREGATE and GROUP BY**

**AIM**

To study basic SQL Aggregate and GROUP BY commands.

**PROCEDURE**

1. **LOGICAL OPERATORS**

* **OR:** The OR Operator in SQL displays the records where any one condition is true, i.e. either condition1 or condition2 is True.

Syntax: SELECT \* FROM <Relation name> WHERE condition1 OR condition2 OR… conditionN;

* **AND:** The AND operator allows you to filter data based on multiple conditions, all of which must be true for the record to be included in the result set.

Syntax: SELECT \* FROM <Relation name> WHERE condition1 AND condition2 AND …conditionN;

* **IN:** The IN operator allows you to specify multiple values in a WHERE clause. The IN operator is a shorthand for multiple OR conditions.

Syntax: SELECT <Attribute name>(s) FROM <Relation name> WHERE <Attribute name> IN (value1, value2, ...);

* **BETWEEN:** The BETWEEN operator selects values within a given range. The values can be numbers, text, or dates. The BETWEEN operator is inclusive: begin and end values are included.

Syntax: SELECT <Attribute name>(s) FROM <Relation name> WHERE <Attribute name> BETWEEN value1 AND value2;

* **LIKE:** The LIKE operator is used in a WHERE clause to search for a specified pattern in a column. The percent sign ***%*** represents zero, one, or multiple characters. The underscore sign ***\_*** represents one, single character.

Syntax: SELECT <Attribute name1>,<Attribute name2>, ... FROM <Relation name> WHERE <Attribute nameN> LIKE pattern;

1. **AGGREGATE FUNCTIONS**

* AVERAGE: *avg*
* MINIMUM: *min*
* MAXIMUM: *max*
* TOTAL: *sum*
* COUNT: *count*

1. **GROUP BY | GROUP BY HAVING | ORDER BY**

* **GROUP BY:** This clause is extension of query statement which is used to group set of tuples based on attribute or attributes.
* **HAVING:** This clause is used to apply condition to groups rather than just tuples.
* **ORDER BY:** This clause causes the tuples in the result of a query to appear in sorted order. To specify the sort order **desc** is used for descending order and **asc** is used for ascending order. By default it list items by ascending order.
* *Syntax*: SELECT (<Attribute name1>,<Attribute name2>,…,<Attribute namen>)

FROM <Relation name1>, <Relation name2>,…, <Relation namen>

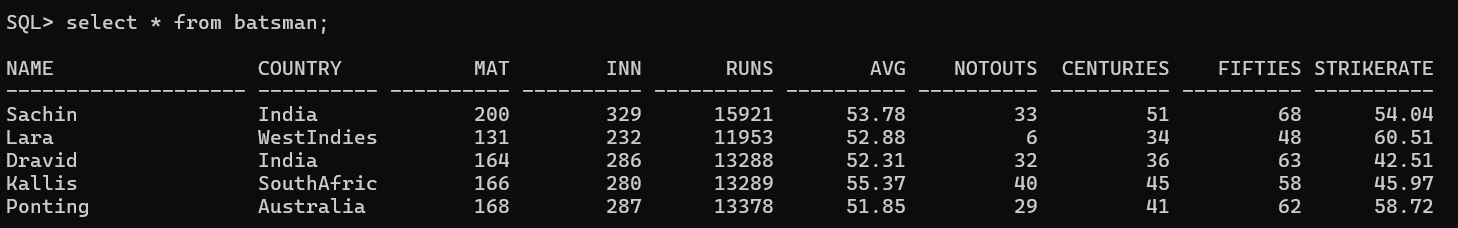
WHERE <Condition>

GROUP BY <Attribute name>

HAVING <AGGREGATE FUNCTION (Attribute name)>

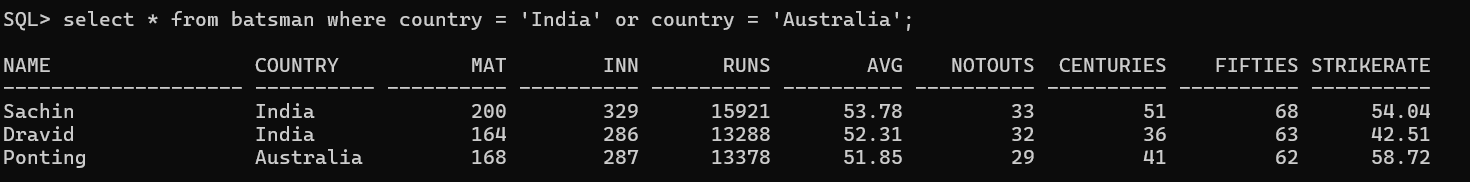
ORDER BY <Attribute name DESC | ASC>

**EXECUTION**

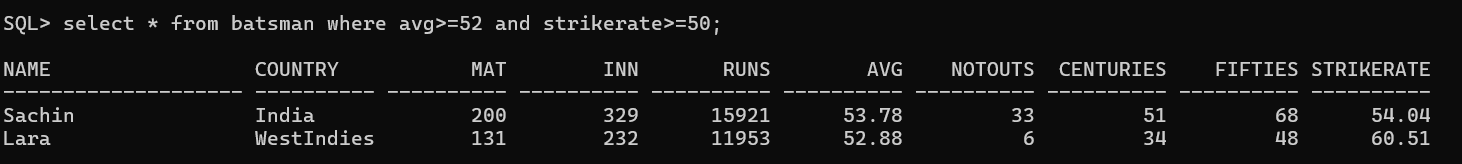


* **SELECT USING OR & AND**

SQL> **select \* from batsman where country = 'India' or country = 'Australia';**

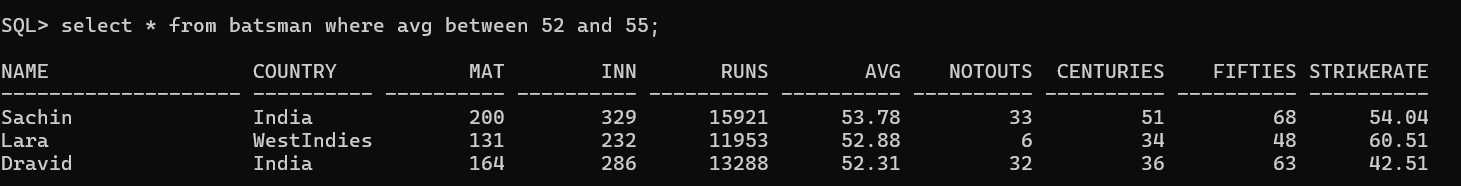


SQL> **select \* from batsman where avg>=52 and strikerate>=50;**



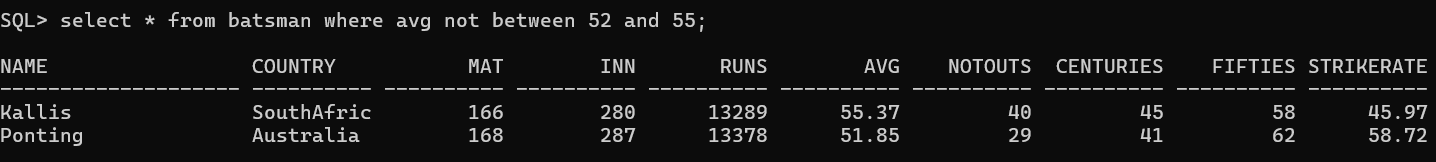
* **SELECT USING BETWEEN**

SQL> **select \* from batsman where avg between 52 and 55;**



* **SELECT USING NOT BETWEEN**

SQL> **select \* from batsman where avg not between 52 and 55;**



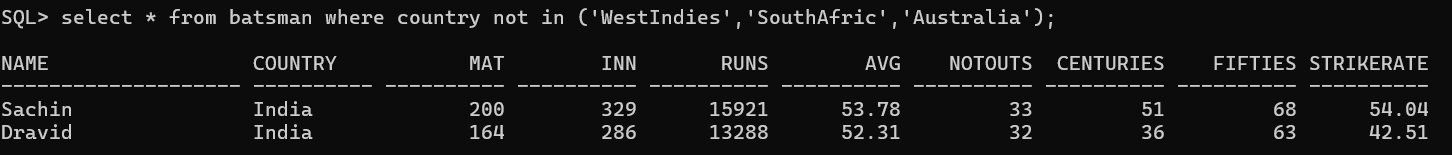
* **SELECT USING IN**

SQL> **select \* from batsman where country in ('WestIndies','SouthAfric','Australia');**



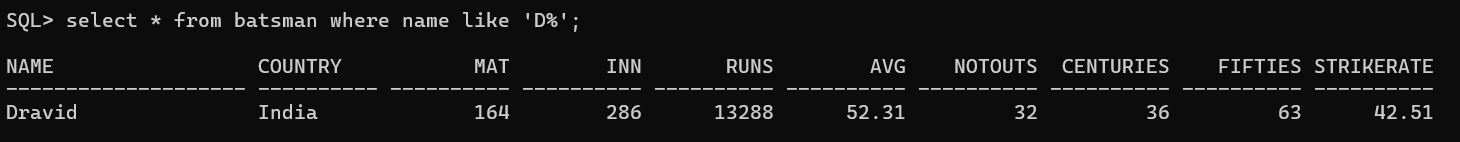
* **SELECT USING NOT IN**

SQL> **select \* from batsman where country not in ('WestIndies','SouthAfric','Australia');**



* **SELECT USING LIKE**

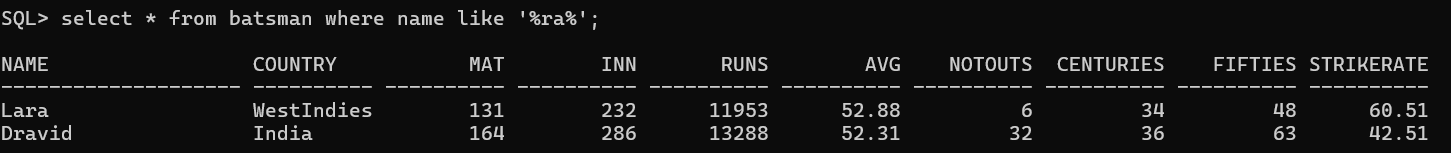
SQL> **select \* from batsman where name like 'D%';**

****

SQL> **select \* from batsman where name like '%a';**

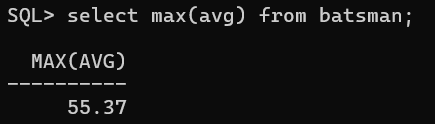


SQL> **select \* from batsman where name like '%ra%';**

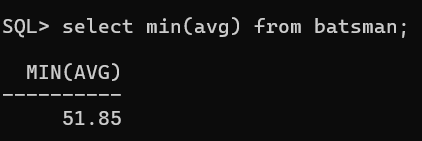
****

* **AGGREGATE FUNCTIONS**

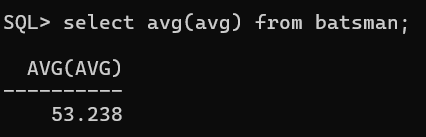
SQL> select max(avg) from batsman;



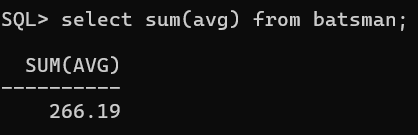
SQL> select min(avg) from batsman;



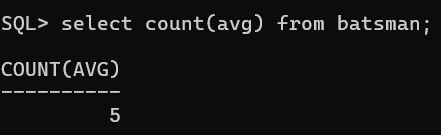
SQL> select avg(avg) from batsman;



SQL> select sum(avg) from batsman;

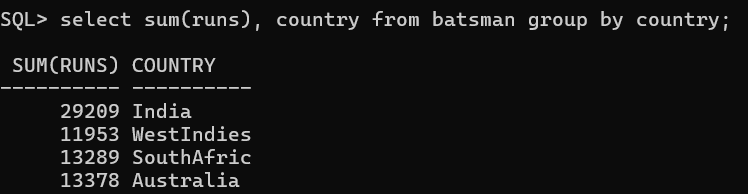


SQL> select count(avg) from batsman;

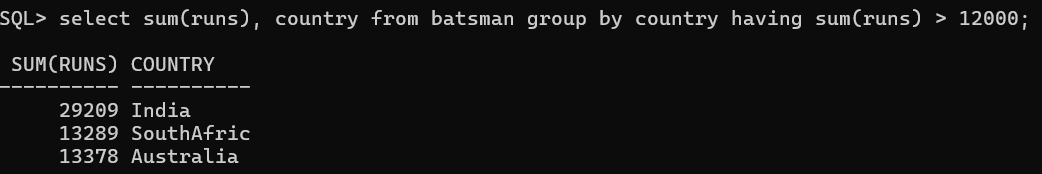


* **GROUP BY | HAVING | ORDER BY**

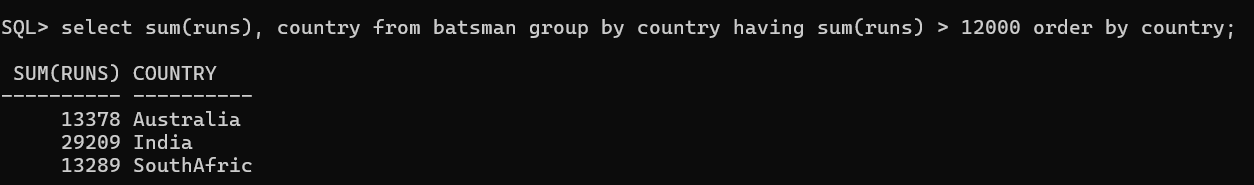
SQL> **select sum(runs), country from batsman group by country;**

****

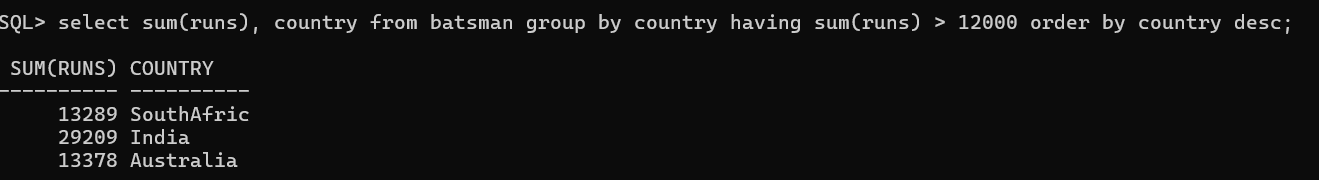
SQL> **select sum(runs), country from batsman group by country having sum(runs) > 12000;**

****

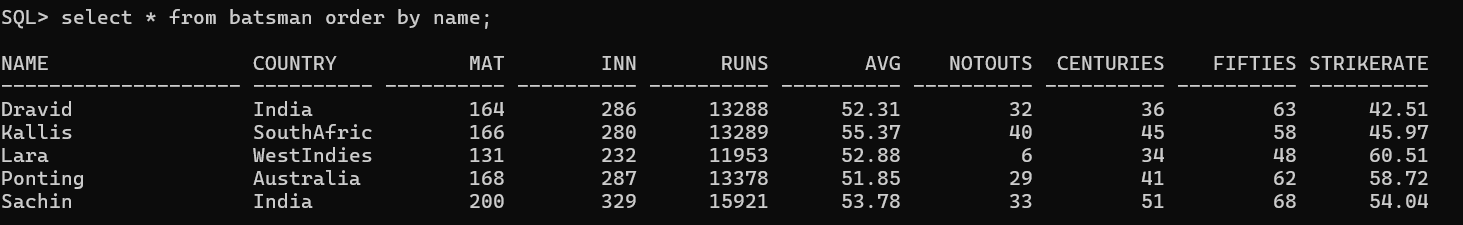
SQL> **select sum(runs), country from batsman group by country having sum(runs) > 12000 order by country;**

****

SQL> **select sum(runs), country from batsman group by country having sum(runs) > 12000 order by country desc;**

****

SQL> **select \* from batsman order by name;**

****