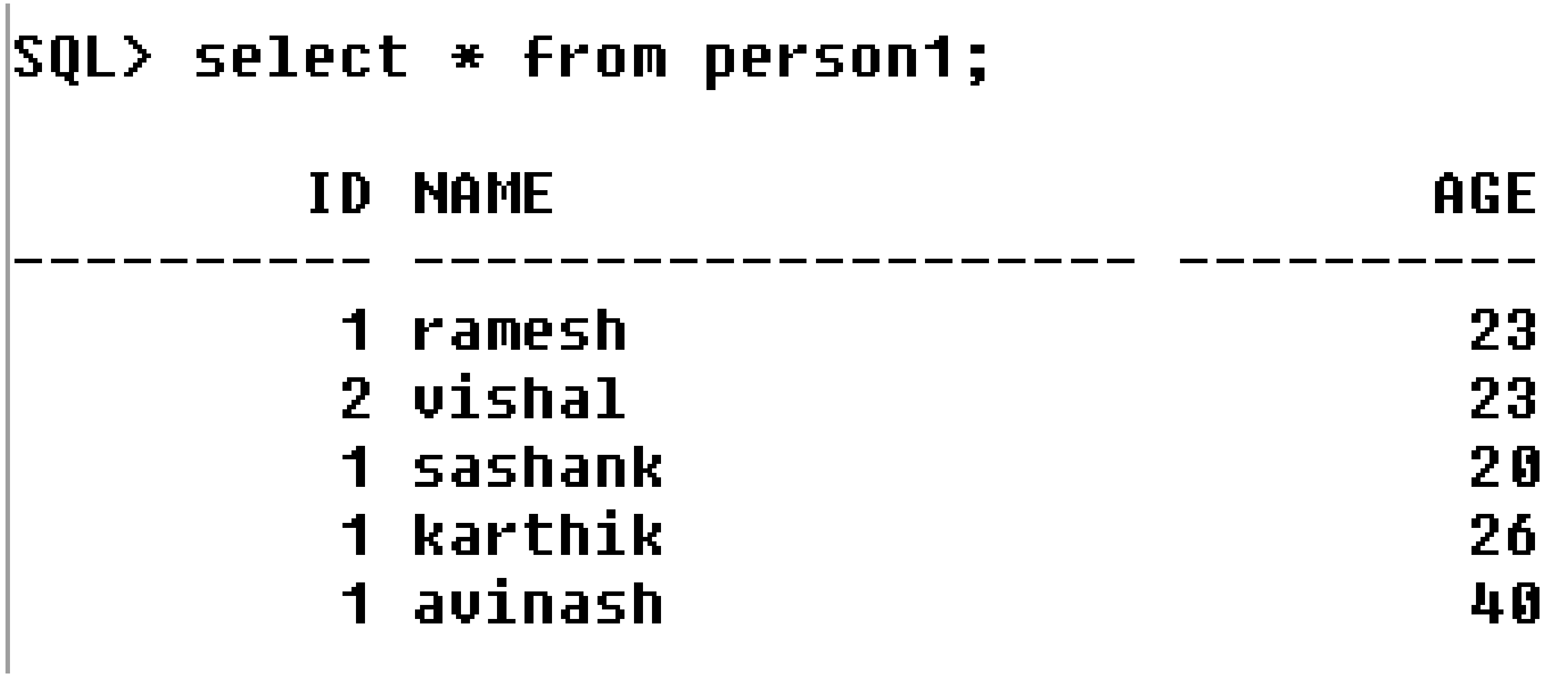
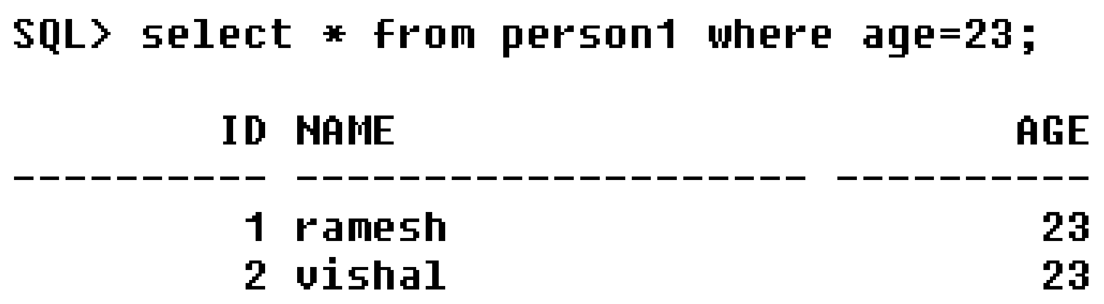
Operators and Aggregate Functions



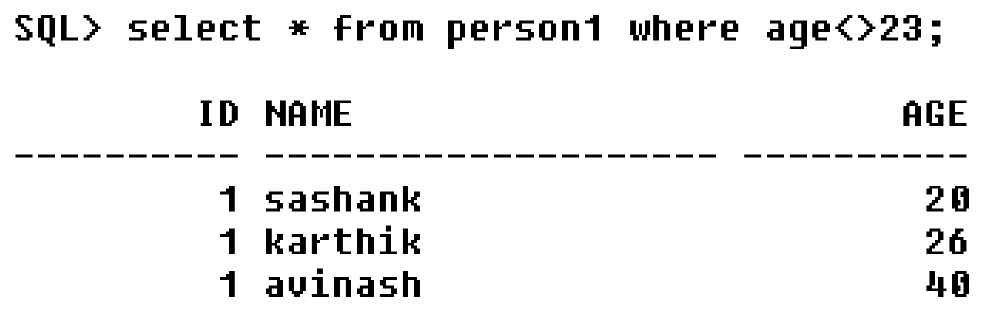
```
SQL> select * from person1;

        ID NAME                        AGE
---------- -------------------- ----------
         1 ramesh                       23
         2 vishal                       23
         1 sashank                      20
         1 karthik                      26
         1 avinash                      40
```

Comparison operator

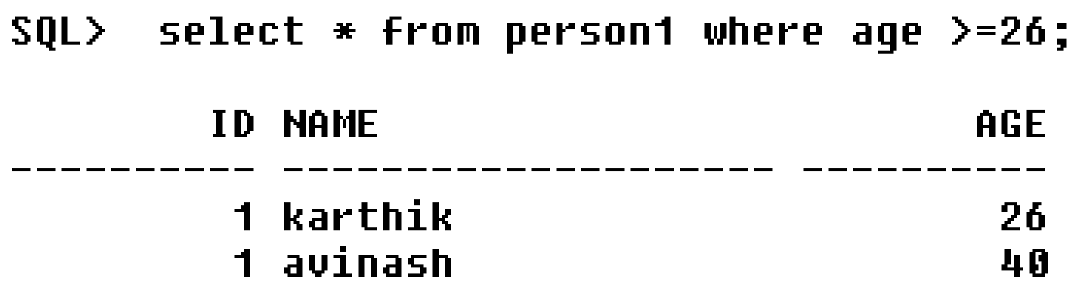
Comparison operator	Description
=	Equal
<>	Not Equal
!=	Not equal
>	Greater than
>=	Greater than or equal
<	Less than
<=	Less than or equal



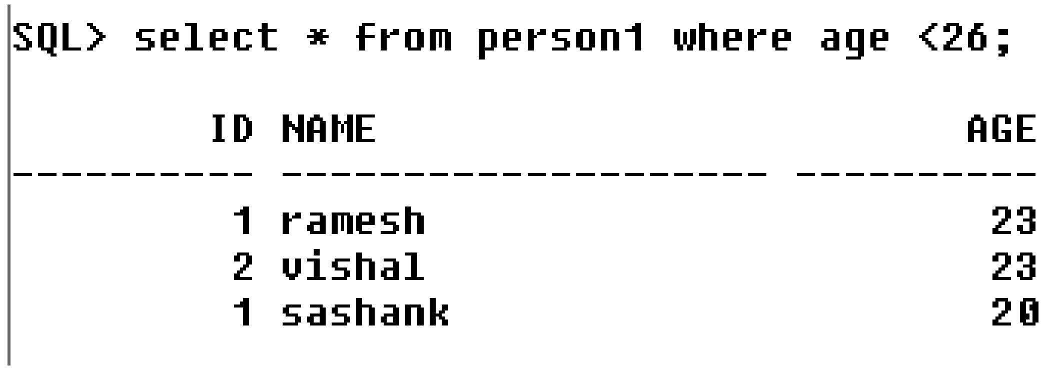
```
SQL> select * from person1 where age=23;

        ID NAME                        AGE
---------- -------------------- ----------
         1 ramesh                       23
         2 vishal                       23
```



```
SQL> select * from person1 where age<>23;

        ID NAME                        AGE
---------- -------------------- ----------
         1 sashank                      20
         1 karthik                      26
         1 avinash                      40
```



```
SQL>  select * from person1 where age >=26;

        ID NAME                        AGE
---------- -------------------- ----------
         1 karthik                      26
         1 avinash                      40
```



```
SQL> select * from person1 where age <26;

        ID NAME                        AGE
---------- -------------------- ----------
         1 ramesh                       23
         2 vishal                       23
         1 sashank                      20
```

Logical operator: AND, OR and NOT Operators

- The AND operator displays a record if all the conditions separated by AND are TRUE.

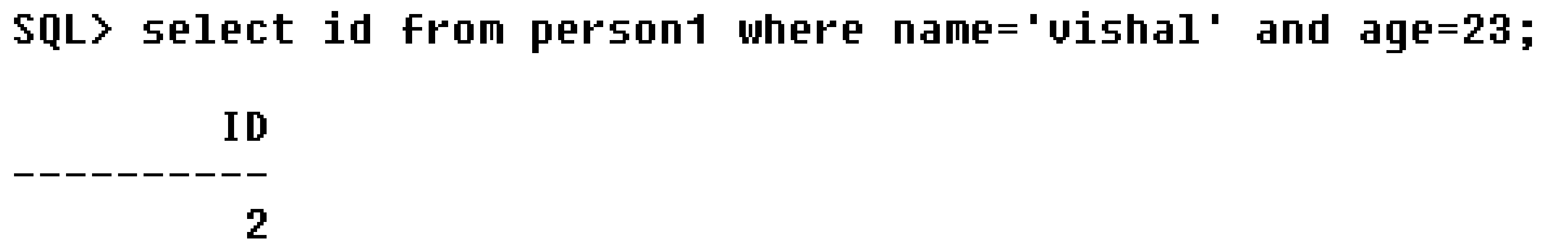
SELECT *column1*, *column2*,...FROM *table_name*
WHERE *condition1* AND *condition2* AND *condition3* ...;

- The OR operator displays a record if any of the conditions separated by OR is TRUE.

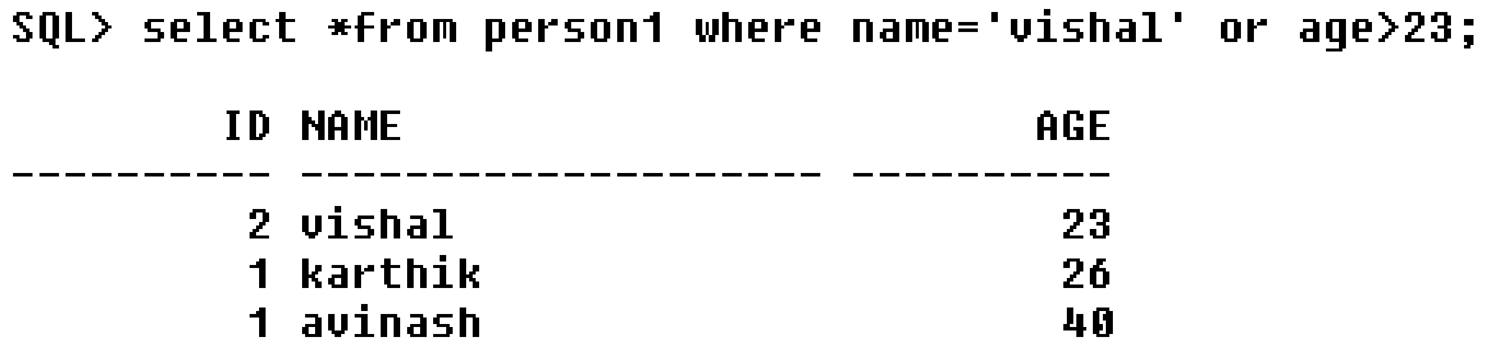
SELECT *column1*, *column2*,
...FROM *table_name*WHERE *condition1* OR *condition2* OR *condition3*;

- The NOT operator displays a record if the condition(s) is NOT TRUE.

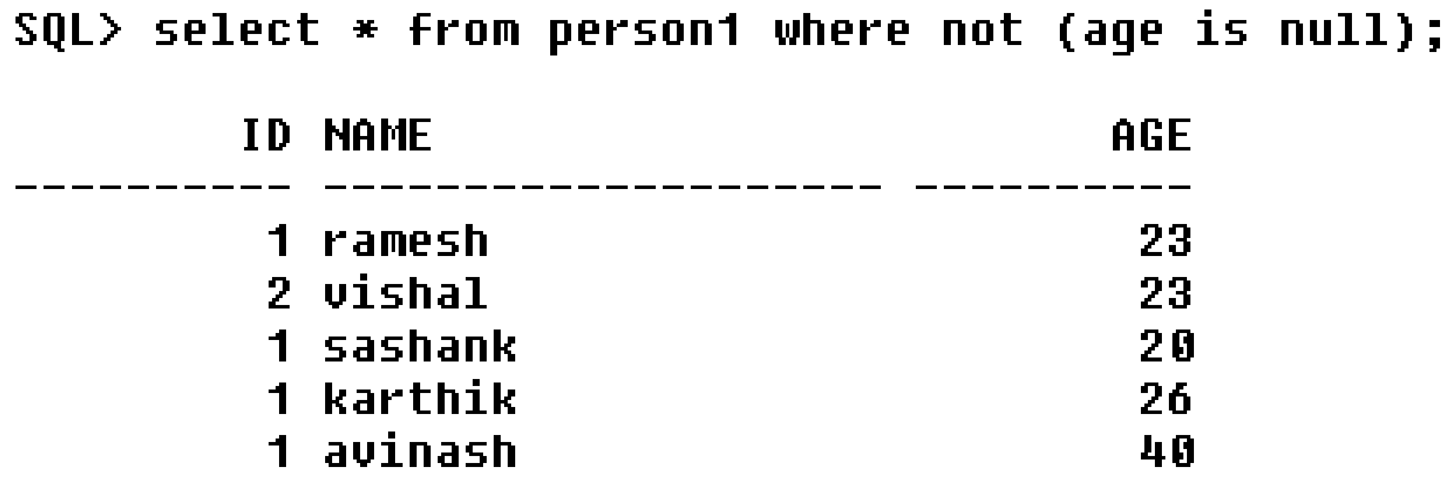
SELECT *column1*, *column2*, ...FROM *table_name*
WHERE NOT *condition*;



```
SQL> select id from person1 where name='vishal' and age=23;

        ID
----------
         2
```



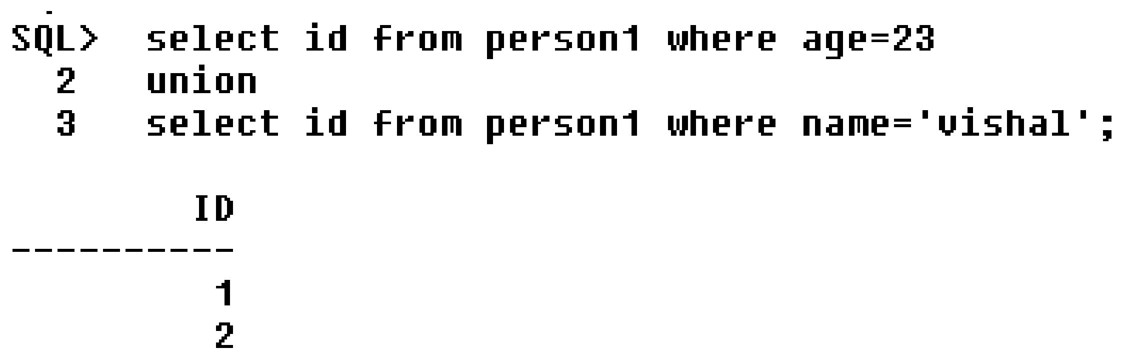
```
SQL> select *from person1 where name='vishal' or age>23;

        ID NAME                                  AGE
---------- -------------------- ----------
         2 vishal                                 23
         1 karthik                                26
         1 avinash                                40
```



```
SQL> select * from person1 where not (age is null);

        ID NAME                                  AGE
---------- -------------------- ----------
         1 ramesh                                 23
         2 vishal                                 23
         1 sashank                                20
         1 karthik                                26
         1 avinash                                40
```

SET operator: Union, Union All, Intersect, Minus

combine the result sets of two or more Oracle SELECT statements.
 It combines the both SELECT statement and removes duplicate rows between them
Each SELECT statement within the UNION operator must have the same number of fields in the result sets with similar data types.
Return distinct rows



```
SQL>  select id from person1 where age=23
  2   union
  3   select id from person1 where name='vishal';

        ID
----------
         1
         2
```

1. **SELECT** expression1, expression2, ... expression_n
2. **FROM** table1
3. **WHERE** conditions
4. **UNION**
5. **SELECT** expression1, expression2, ... expression_n
6. **FROM** table2
7. **WHERE** conditions;

UNION operator removes duplicate rows while UNION ALL operator does not remove duplicate rows

- .

- INTERSECT Operator is used to return the results of 2 or more SELECT statement.

```
SELECT expression1, expression2, ... expression_n
FROM table1
WHERE conditions
INTERSECT
SELECT expression1, expression2, ... expression_n
FROM table2
WHERE conditions;
```

MINUS operator

- return all rows in the first SELECT statement that are not returned by the second SELECT statement

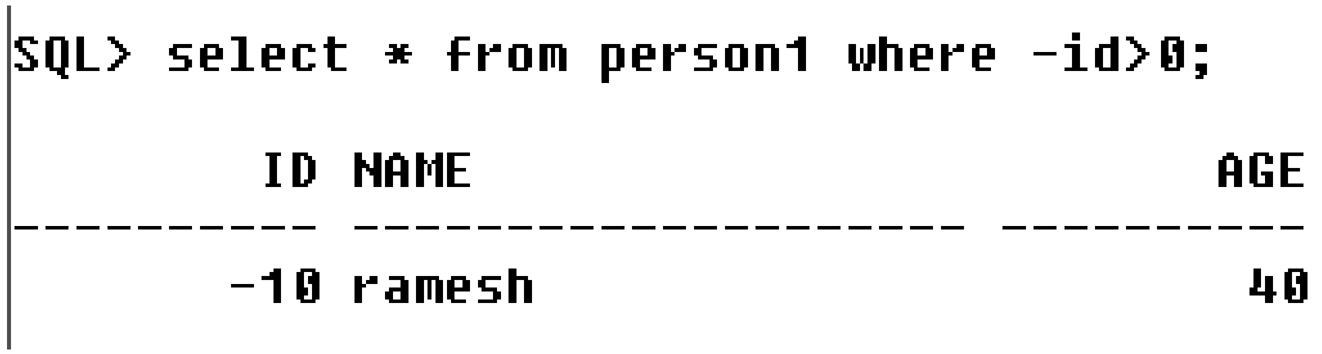
-
  ```
  SELECT expression1, expression2, ... expression_n
  FROM table1
  WHERE conditions
  MINUS
  SELECT expression1, expression2, ... expression_n
  FROM table2
  WHERE conditions;
  ```

Arithmetic operator

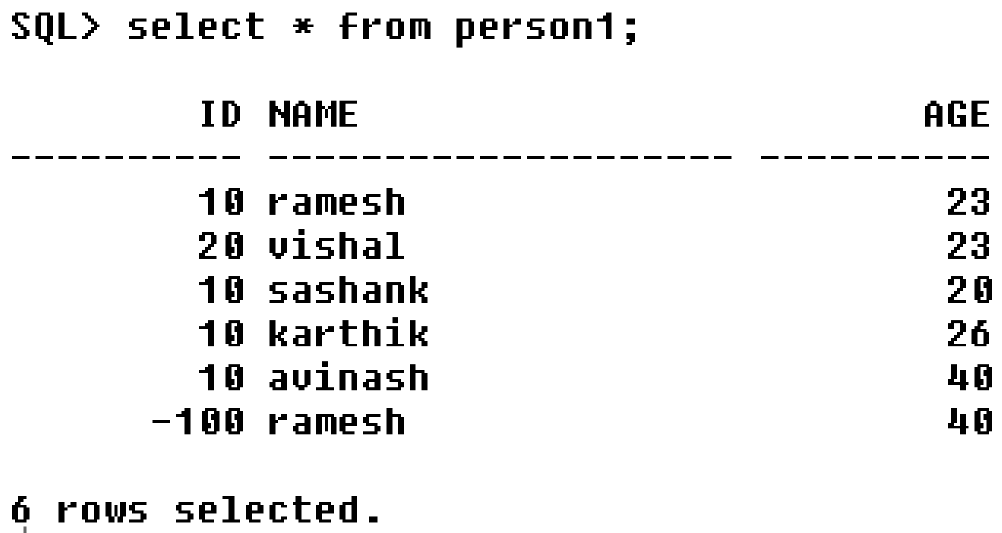
+ Denotes a positive
- or negative expression. These are unary operators.

* Multiplies, divides.
/ These are binary operators.

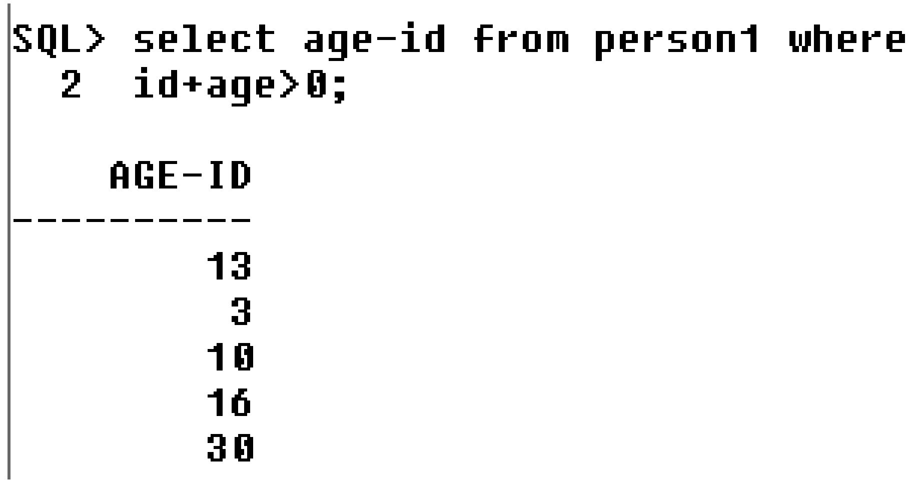
+ Adds, subtracts.
- These are binary operators.



```
SQL> select * from person1 where -id>0;

        ID NAME                                  AGE
---------- -------------------- ----------
       -10 ramesh                                 40
```

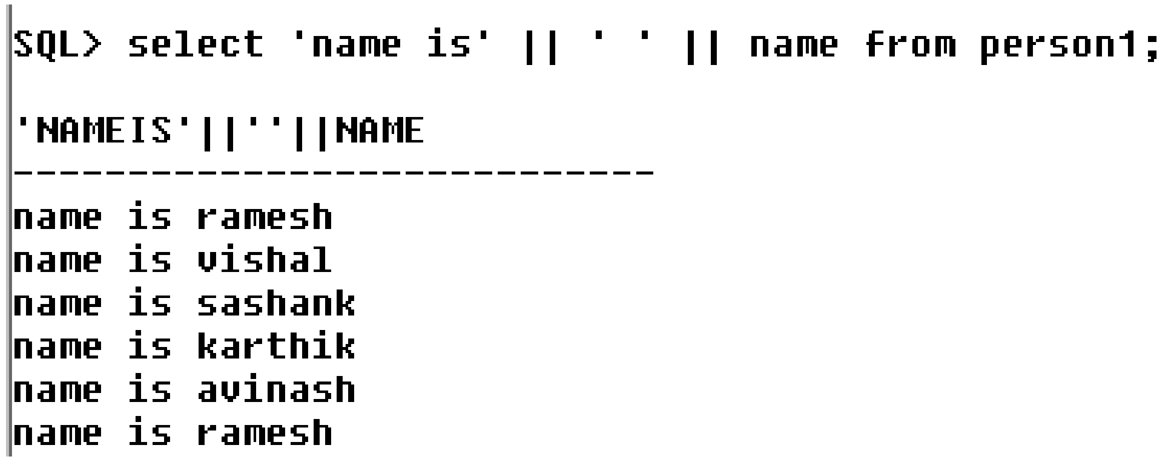


```
SQL> select * from person1;

        ID NAME                        AGE
---------- -------------------- ----------
        10 ramesh                       23
        20 vishal                       23
        10 sashank                      20
        10 karthik                      26
        10 avinash                      40
      -100 ramesh                       40

6 rows selected.
```

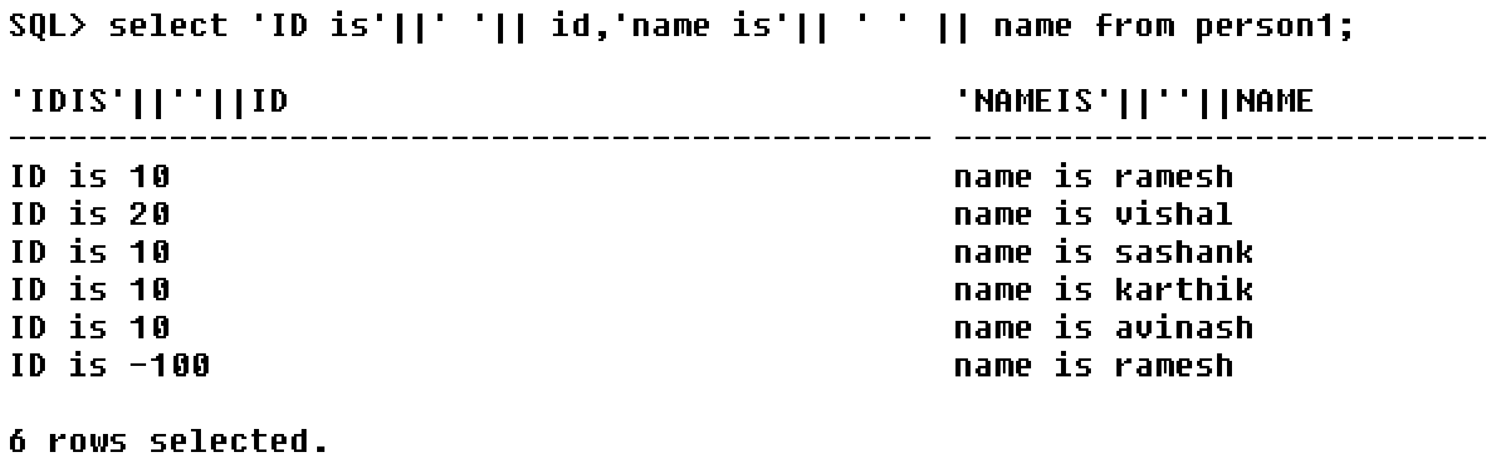


```
SQL> select age-id from person1 where
  2  id+age>0;

    AGE-ID
----------
        13
         3
        10
        16
        30
```

Concatenation



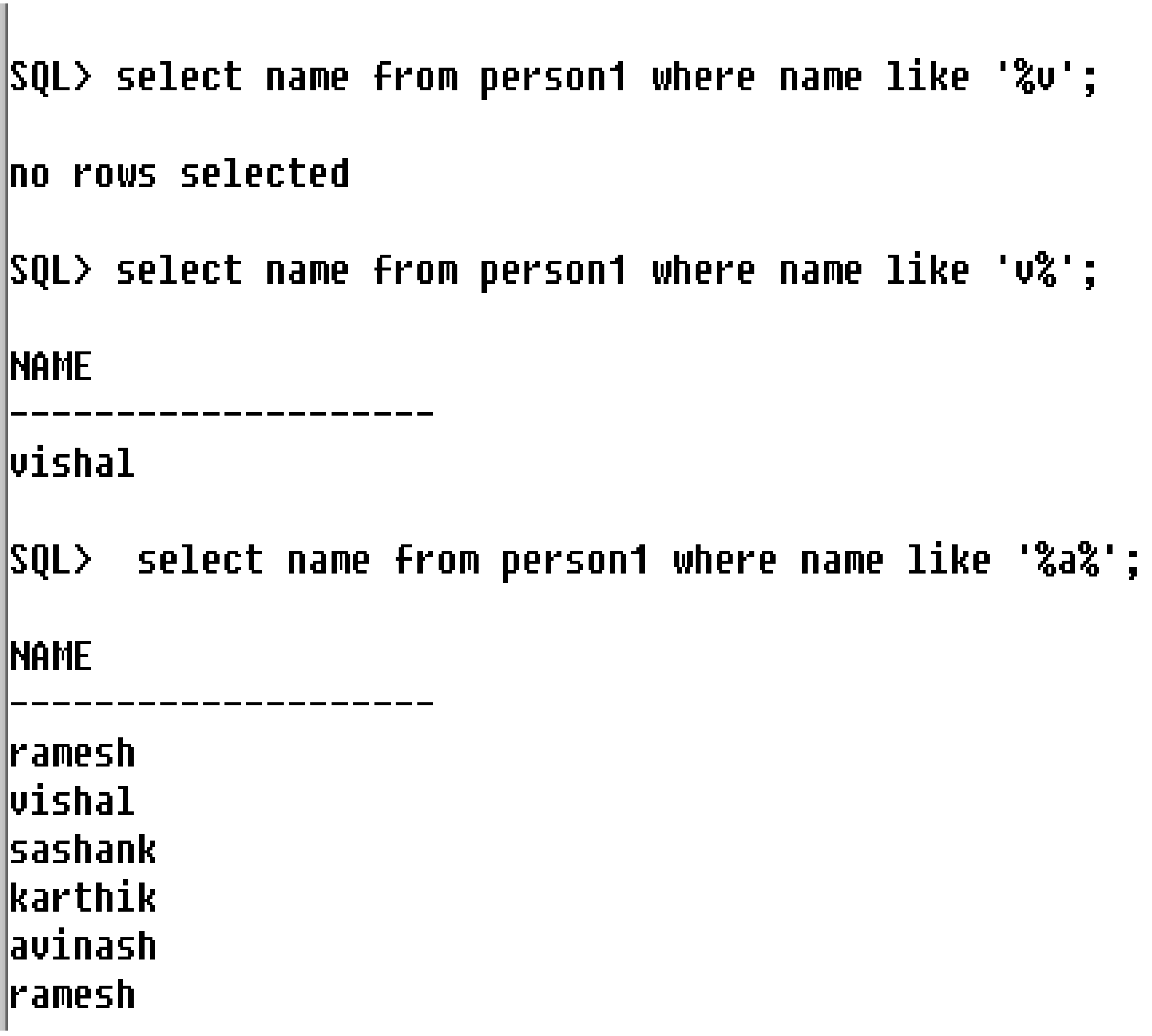
```
SQL> select 'name is' || ' ' || name from person1;

'NAMEIS'||''||NAME
----------------------------
name is ramesh
name is vishal
name is sashank
name is karthik
name is avinash
name is ramesh
```



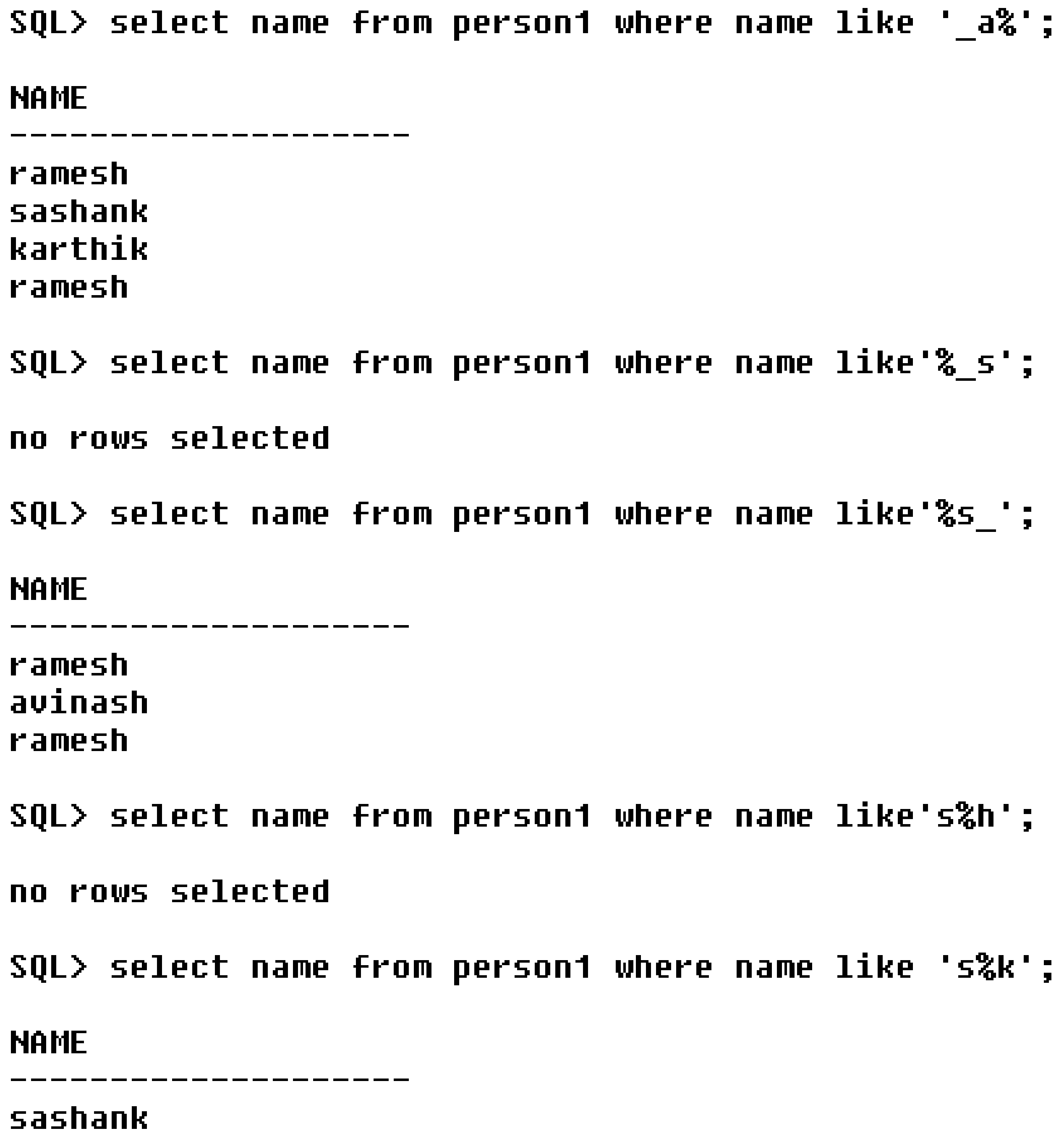
```
SQL> select 'ID is'||' '|| id,'name is'|| ' ' || name from person1;

'IDIS'||''||ID                                 'NAMEIS'||''||NAME
---------------------------------------------- ----------------------------
ID is 10                                       name is ramesh
ID is 20                                       name is vishal
ID is 10                                       name is sashank
ID is 10                                       name is karthik
ID is 10                                       name is avinash
ID is -100                                     name is ramesh

6 rows selected.
```

LIKE Operator : used in a WHERE clause to search for a specified pattern in a column

LIKE Operator	Description
WHERE Address LIKE 'a%'	Finds any values that starts with "a"
WHERE Address LIKE '%a'	Finds any values that ends with "a"
WHERE Address LIKE '%or%'	Finds any values that have "or" in any position
WHERE Address LIKE '_r%'	Finds any values that have "r" in the second position
WHERE AddressLIKE 'a_%_%'	Finds any values that starts with "a" and are at least 3 characters in length
WHERE Address LIKE 'a%o'	Finds any values that starts with "a" and ends with "o"



```
SQL> select name from person1 where name like '%v';

no rows selected

SQL> select name from person1 where name like 'v%';

NAME
--------------------
vishal

SQL>  select name from person1 where name like '%a%';

NAME
--------------------
ramesh
vishal
sashank
karthik
avinash
ramesh
```



```
SQL> select name from person1 where name like '_a%';

NAME
--------------------
ramesh
sashank
karthik
ramesh

SQL> select name from person1 where name like'%_s';

no rows selected

SQL> select name from person1 where name like'%s_';

NAME
--------------------
ramesh
avinash
ramesh

SQL> select name from person1 where name like's%h';

no rows selected

SQL> select name from person1 where name like 's%k';

NAME
--------------------
sashank
```

ORACLE ALIASES

Syntax for column:

Column_name **AS** alias_name

Syntax for table:

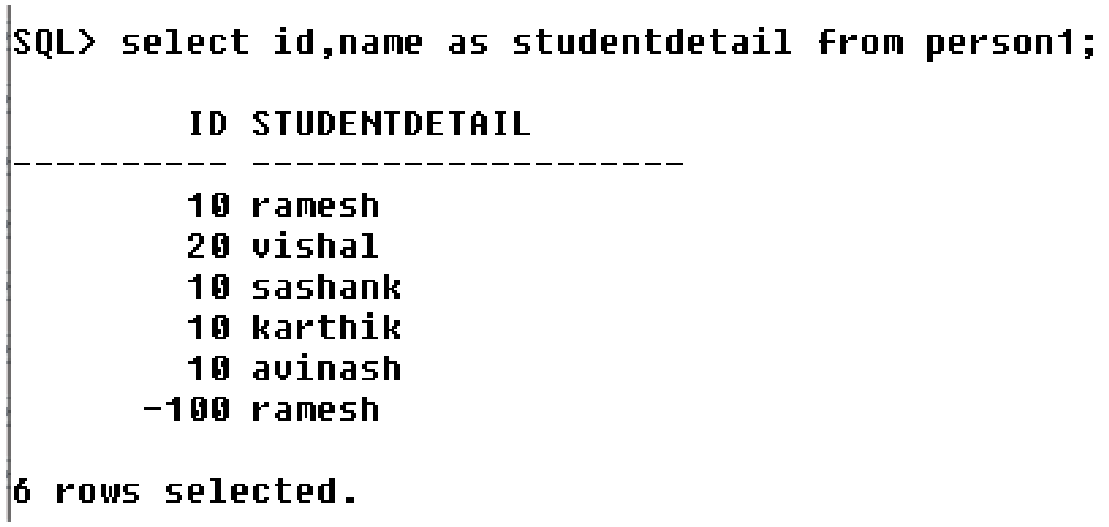
Table_name alias_name

Parameters

column_name: original name of the column

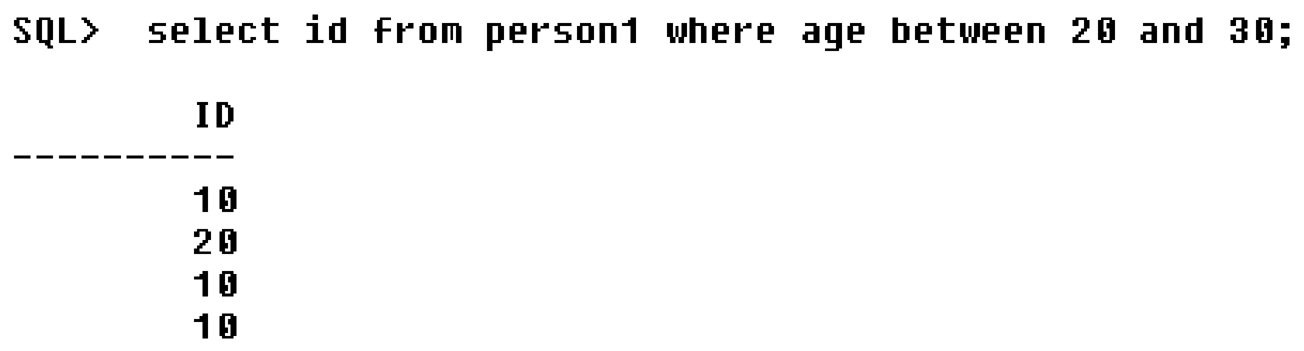
table_name: original name of the table

alias_name: temporary name



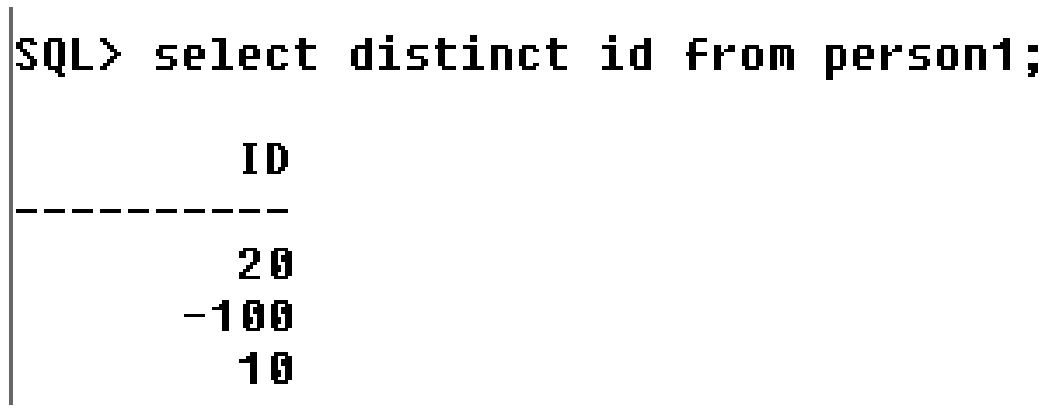
```
SQL> select id,name as studentdetail from person1;

        ID STUDENTDETAIL
---------- --------------------
        10 ramesh
        20 vishal
        10 sashank
        10 karthik
        10 avinash
      -100 ramesh

6 rows selected.
```

Between



```
SQL>  select id from person1 where age between 20 and 30;

        ID
----------
        10
        20
        10
        10
```

1. **SELECT DISTINCT** expressions
2. **FROM** tables
3. **WHERE** conditions

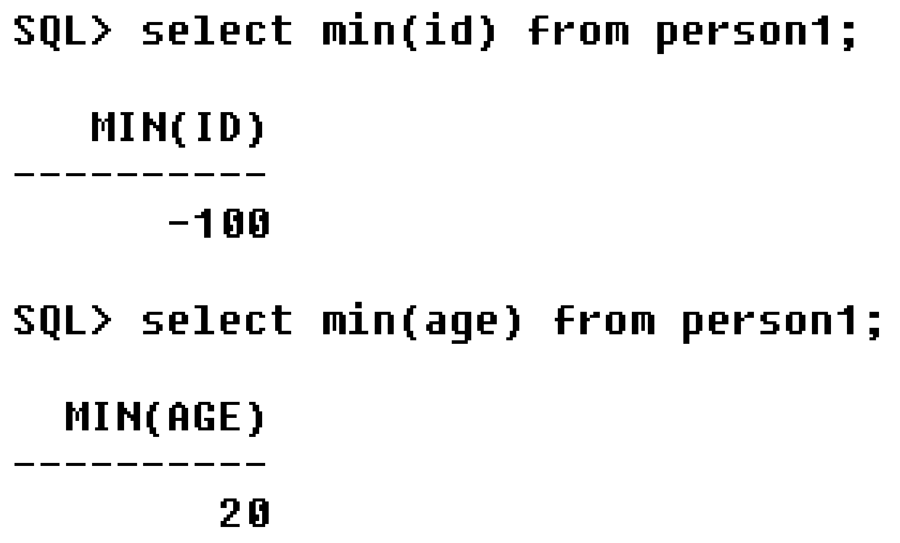


```
SQL> select distinct id from person1;

        ID
----------
        20
      -100
        10
```

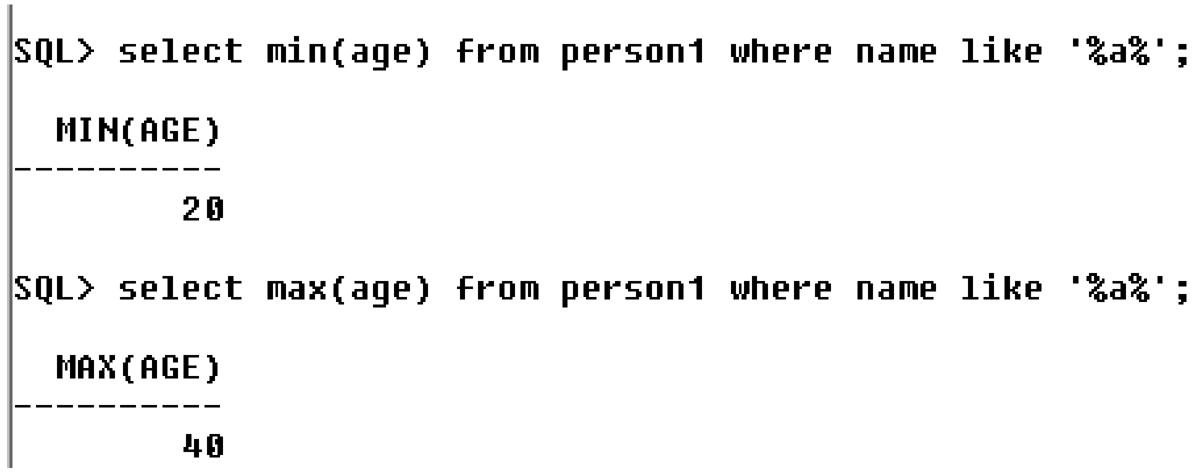
Minimum, Maximum

SELECT MIN(column_name)
FROM table_name
WHERE condition;

SELECT max(column_name)
FROM table_name
WHERE condition;



```
SQL> select min(id) from person1;

   MIN(ID)
----------
      -100

SQL> select min(age) from person1;

  MIN(AGE)
----------
        20
```

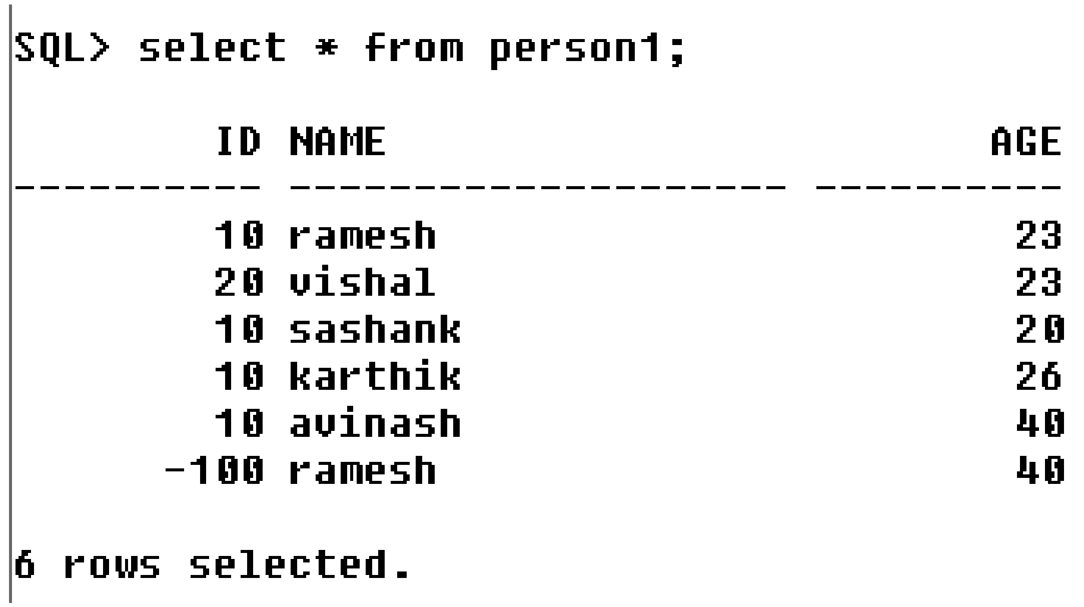


```
SQL> select min(age) from person1 where name like '%a%';

  MIN(AGE)
----------
        20

SQL> select max(age) from person1 where name like '%a%';

  MAX(AGE)
----------
        40
```

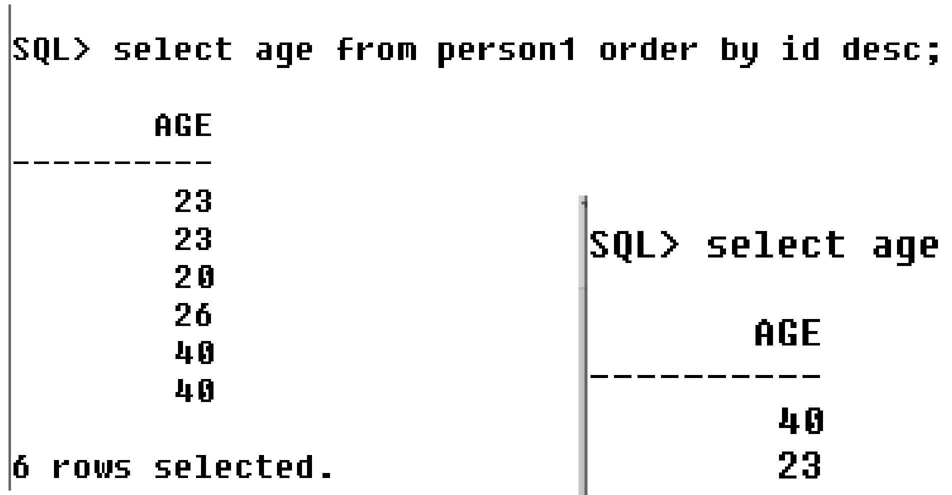
ORDER BY: sort the result-set in ascending or descending order

sorts the records in ascending order by default.
To sort the records in descending order, use the DESC keyword

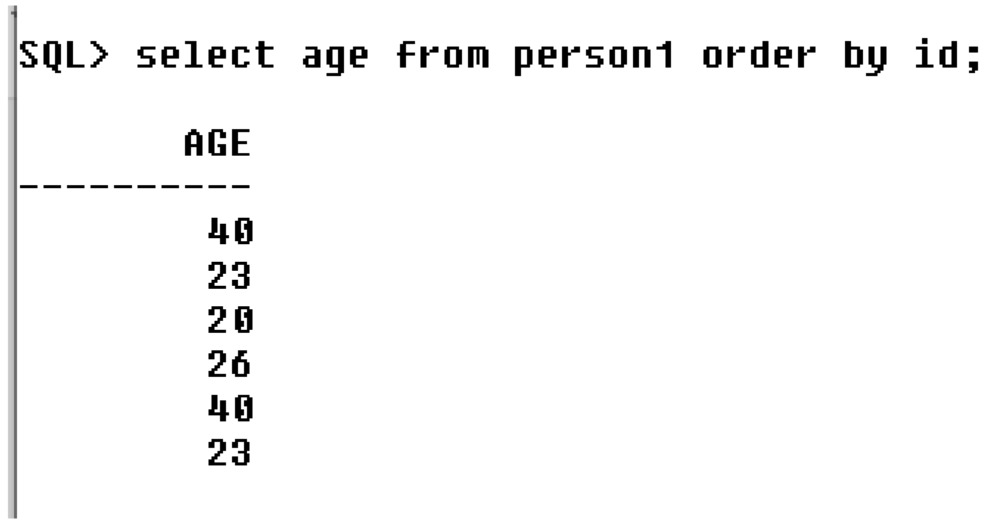
SELECT column1, column2, ...
FROM table_name ORDER BY column1, column2, ... ASC|DESC;



```
SQL> select * from person1;

        ID NAME                                   AGE
---------- -------------------- ----------
        10 ramesh                                  23
        20 vishal                                  23
        10 sashank                                 20
        10 karthik                                 26
        10 avinash                                 40
      -100 ramesh                                  40

6 rows selected.
```



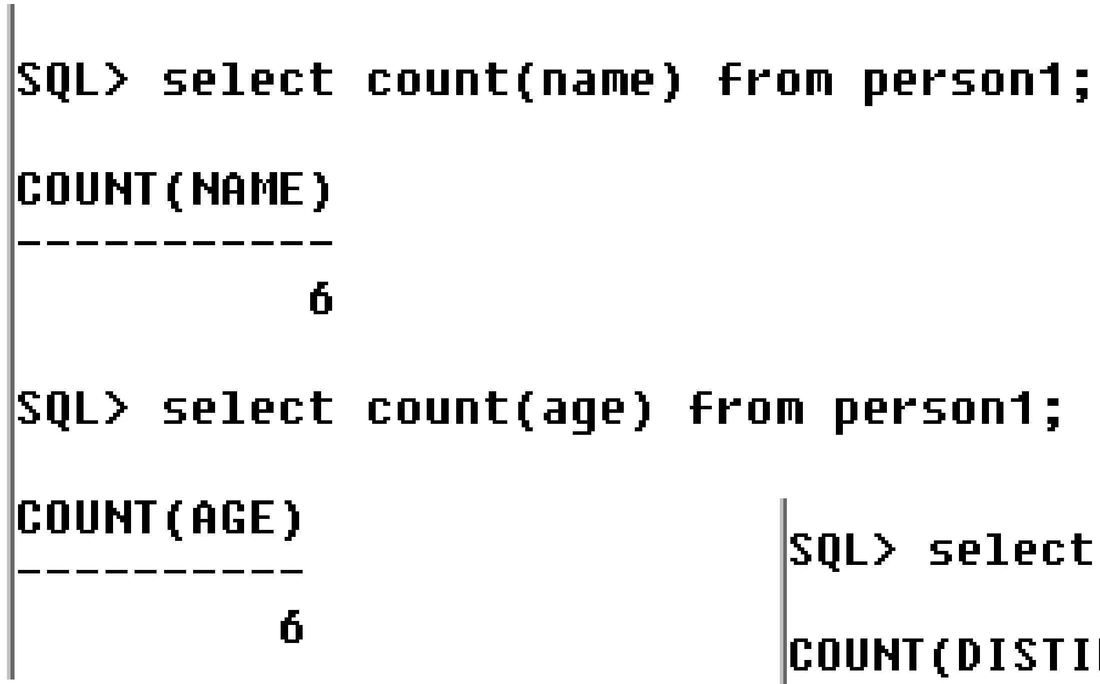
```
SQL> select age from person1 order by id desc;

       AGE
----------
        23
        23
        20
        26
        40
        40

6 rows selected.
```



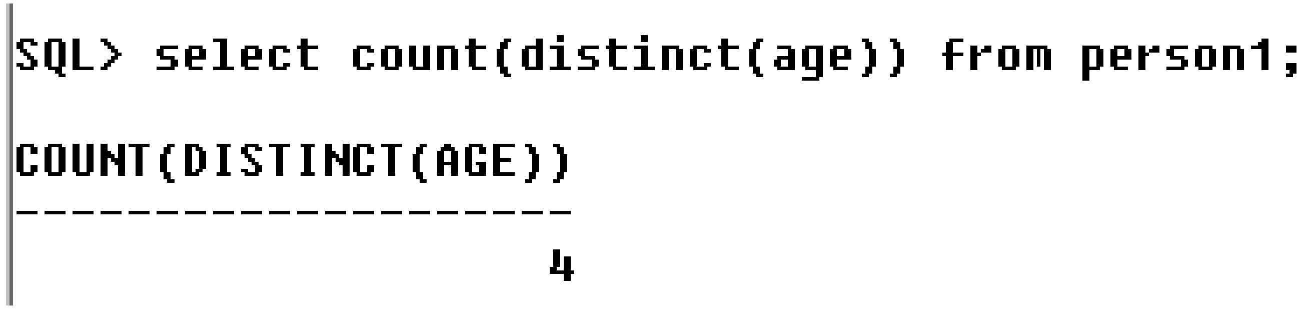
```
SQL> select age from person1 order by id;

       AGE
----------
        40
        23
        20
        26
        40
        23
```

COUNT() :function returns the number of rows that matches a specified criterion

SELECT COUNT(column_name) FROM table_name WHERE condition;



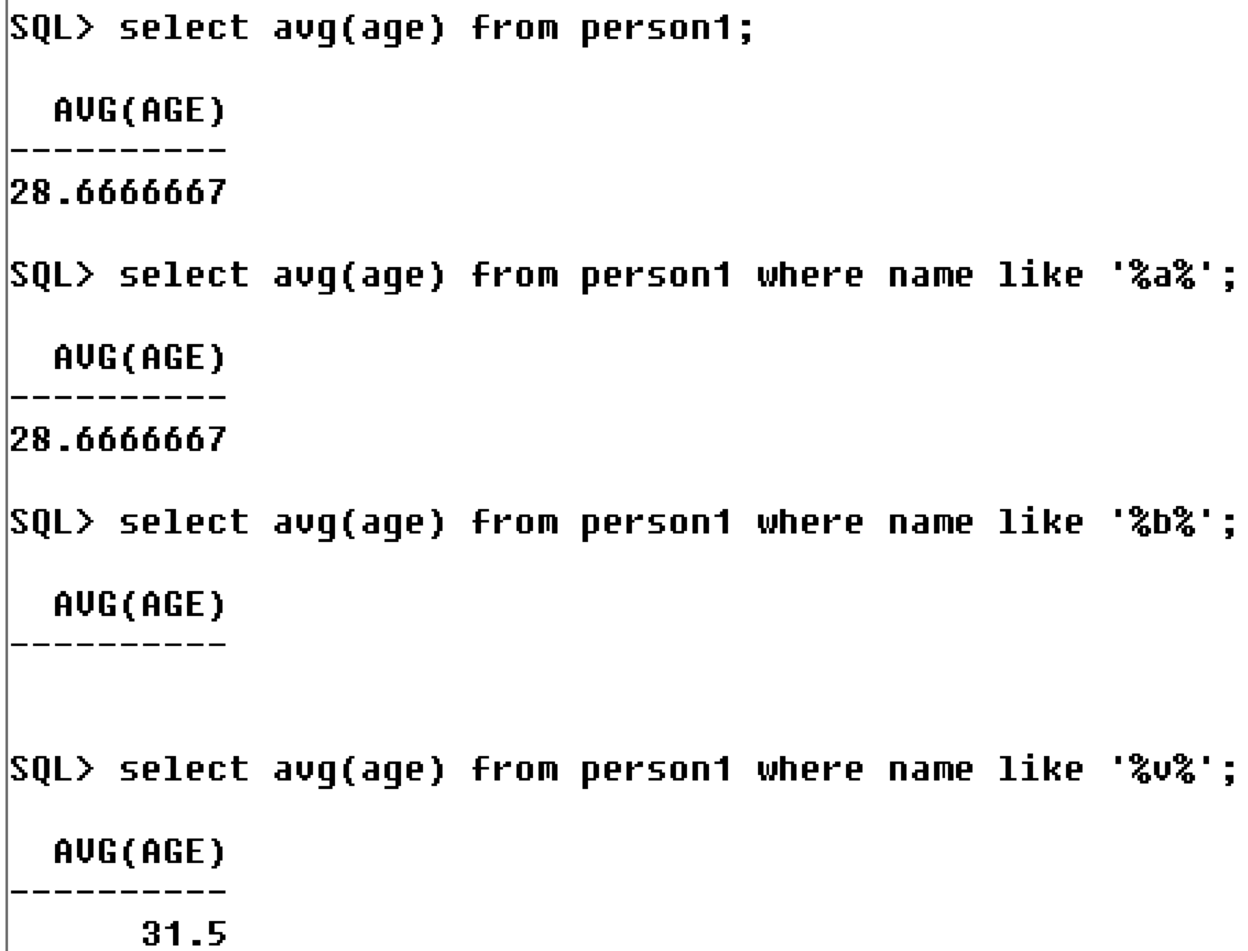
```
SQL> select count(name) from person1;

COUNT(NAME)
-----------
          6

SQL> select count(age) from person1;

COUNT(AGE)
----------
         6
```



```
SQL> select count(distinct(age)) from person1;

COUNT(DISTINCT(AGE))
--------------------
                   4
```

AVG(): function returns the average value of a numeric column

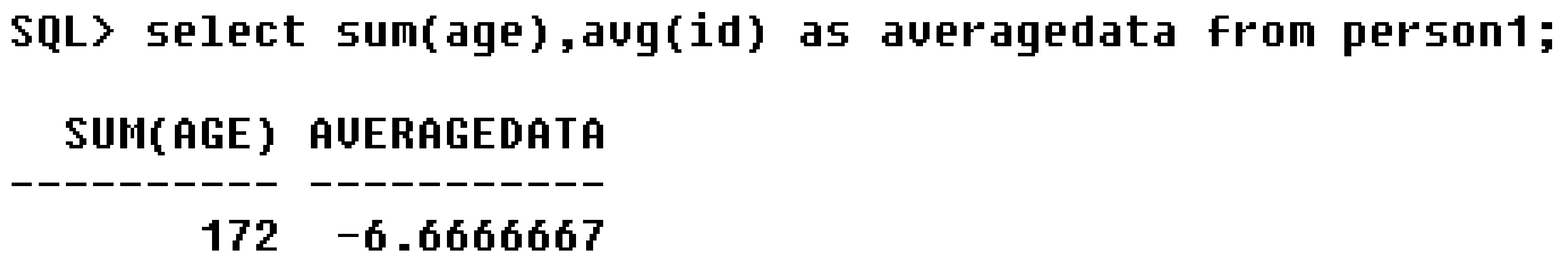
SELECT AVG(column_name) FROM table_name
WHERE condition;



```
SQL> select avg(age) from person1;

  AVG(AGE)
----------
28.6666667

SQL> select avg(age) from person1 where name like '%a%';

  AVG(AGE)
----------
28.6666667

SQL> select avg(age) from person1 where name like '%b%';

  AVG(AGE)
----------


SQL> select avg(age) from person1 where name like '%v%';

  AVG(AGE)
----------
      31.5
```

SUM() function returns the total sum of a numeric column

SELECT SUM(*column_name*) FROM *table_name*
WHERE *condition*;

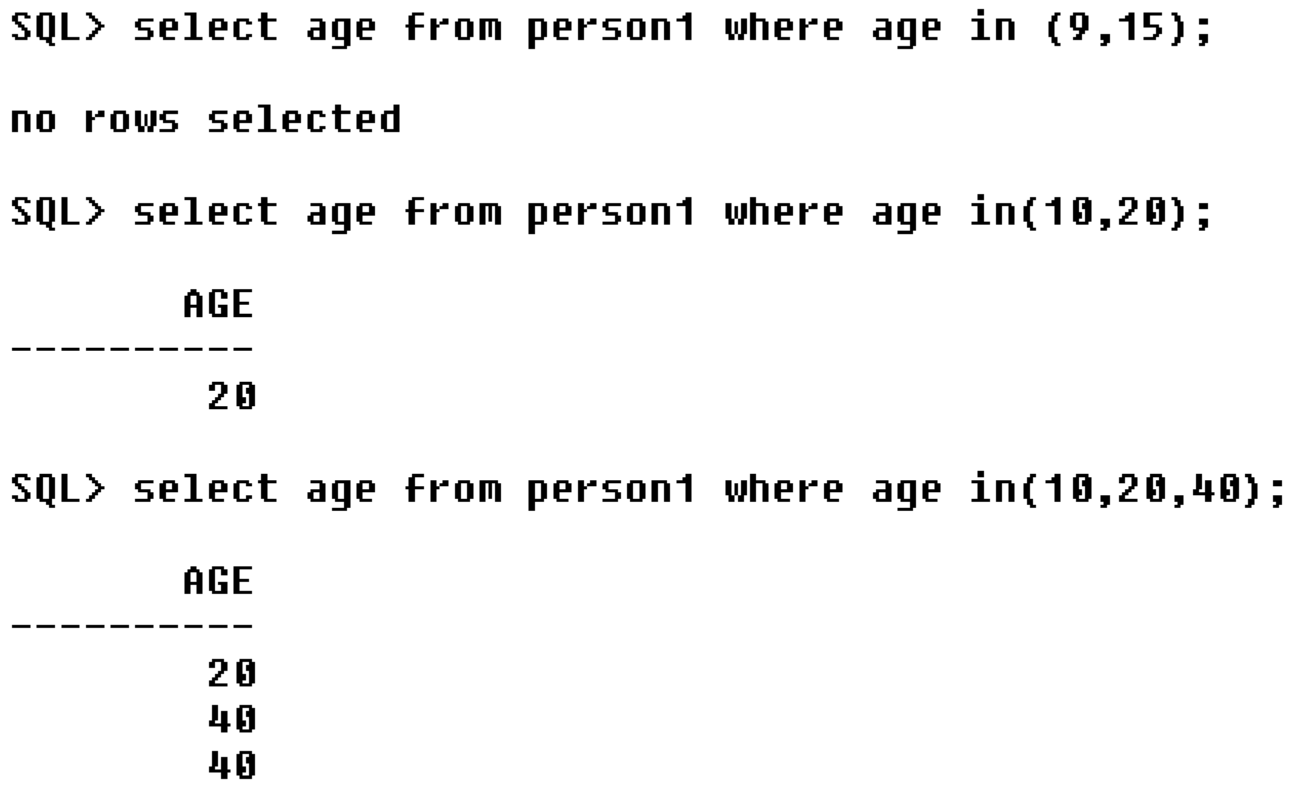


```
SQL> select sum(age),avg(id) as averagedata from person1;

  SUM(AGE) AVERAGEDATA
---------- -----------
       172  -6.6666667
```

IN Operator: allows to specify multiple values in a WHERE clause

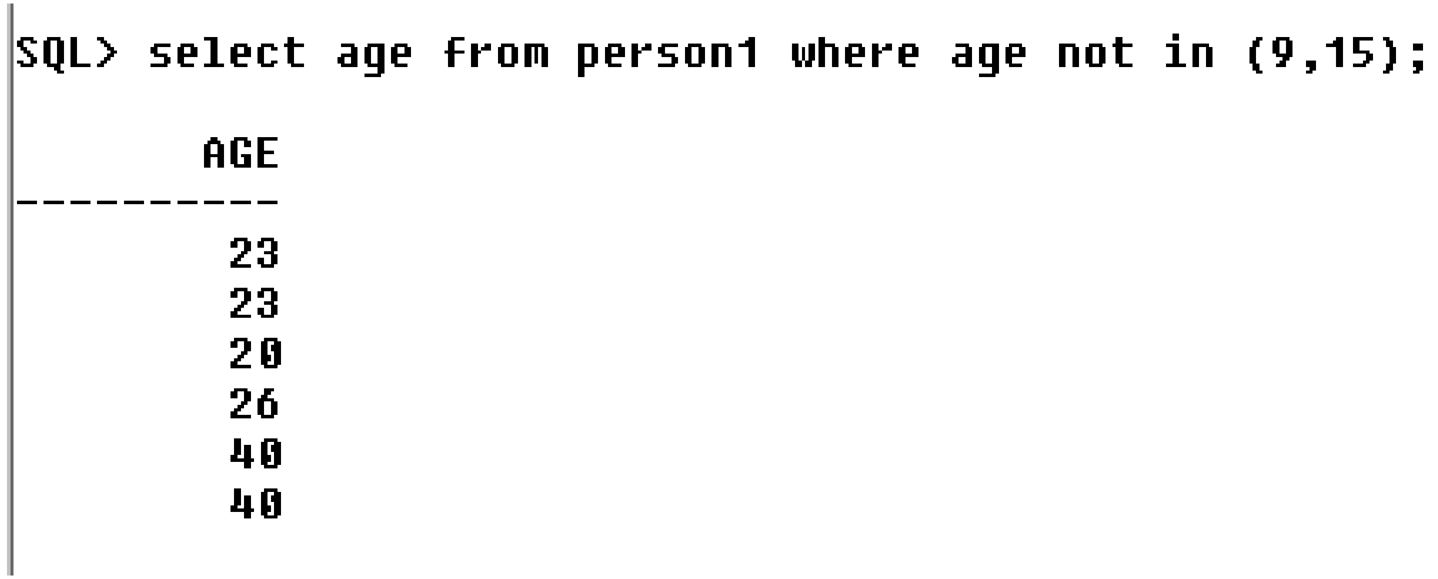
SELECT column_name(s) FROM table_name WHERE column_name IN (value1, value2, ...);

SELECT column_name(s)FROM table_nameWHERE column_name IN (SELECT STATEMENT);



```
SQL> select age from person1 where age in (9,15);

no rows selected

SQL> select age from person1 where age in(10,20);

       AGE
----------
        20

SQL> select age from person1 where age in(10,20,40);

       AGE
----------
        20
        40
        40
```

Not in

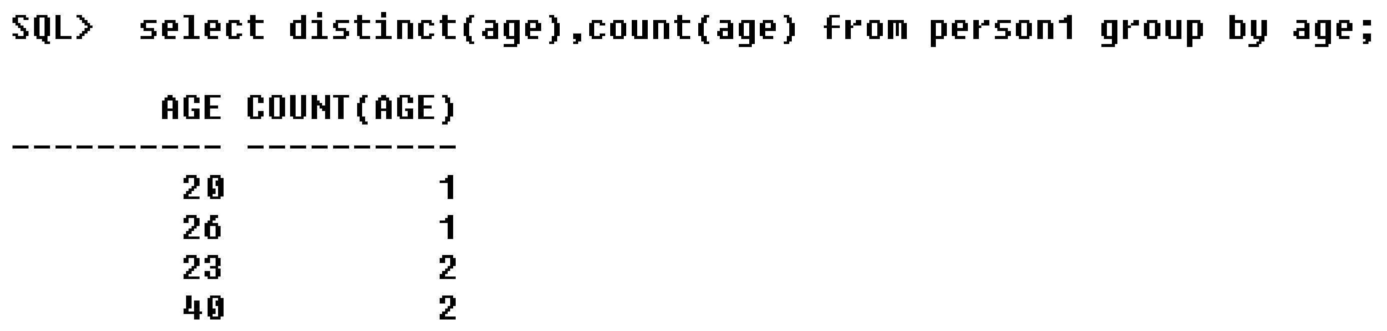


```
SQL> select age from person1 where age not in (9,15);

       AGE
----------
        23
        23
        20
        26
        40
        40
```

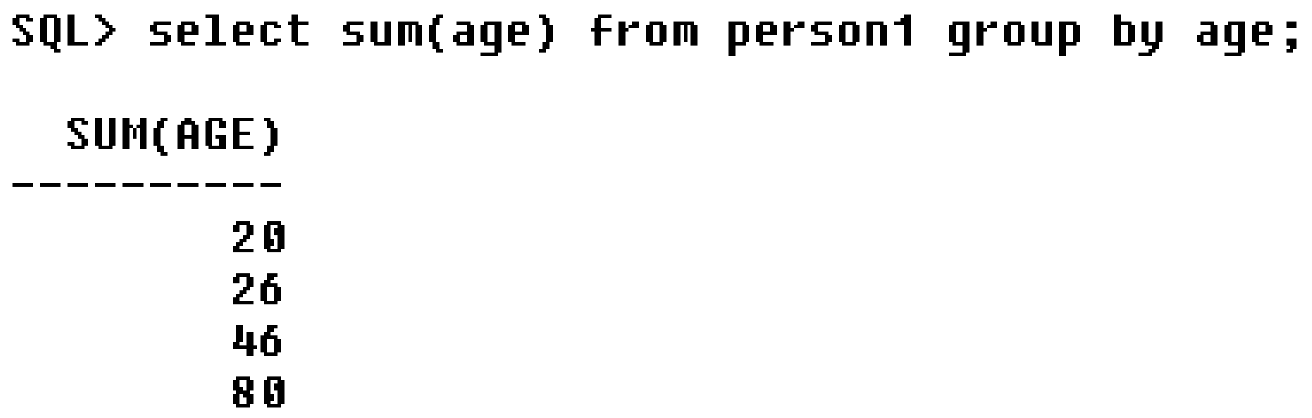
GROUP BY

groups rows that have the same values into summary rows

SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
[ORDER BY column_name(s)];



```
SQL>  select distinct(age),count(age) from person1 group by age;

       AGE COUNT(AGE)
---------- ----------
        20          1
        26          1
        23          2
        40          2
```



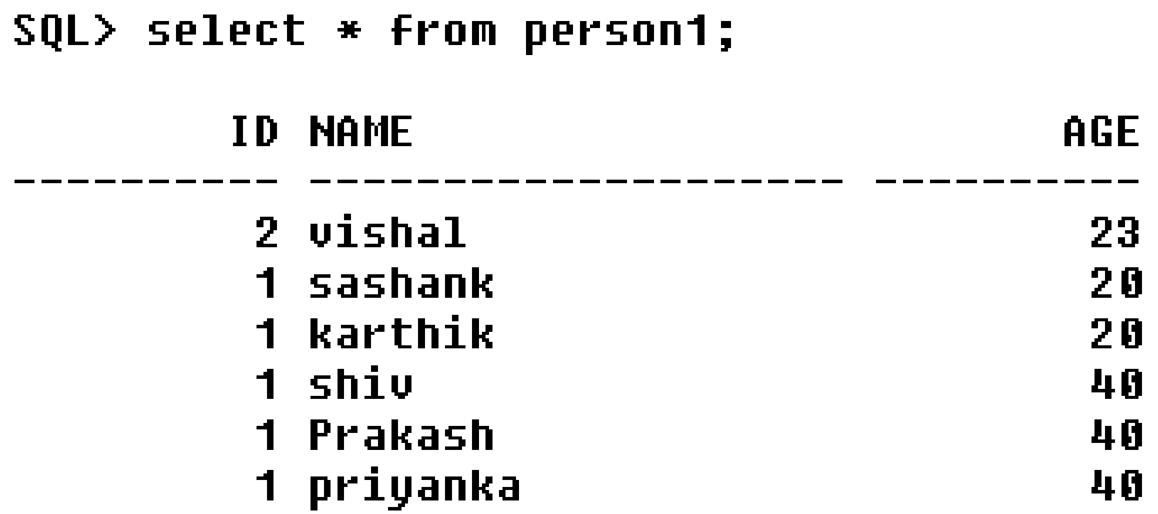
```
SQL> select sum(age) from person1 group by age;

  SUM(AGE)
----------
        20
        26
        46
        80
```

Having clause

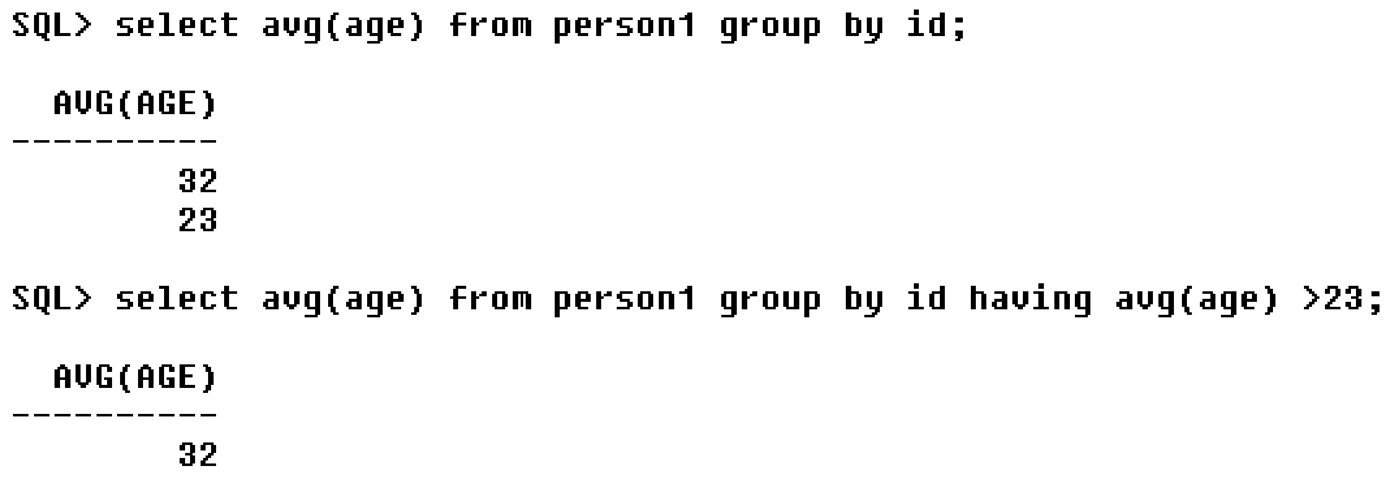
```
SELECT expression1, expression2, ... expres
sion_n,
 aggregate_function (aggregate_expression
)
FROM tables
WHERE conditions
GROUP BY expression1, expression2, ... exp
ression_n
HAVING having_condition;
```

- A HAVING clause restricts the results of a GROUP BY in a select expression.
- The HAVING clause is applied to each group of the grouped table,
  - much as a WHERE clause is applied to a select list.
- If there is no GROUP BY clause, the HAVING clause is applied to the entire result as a single group.

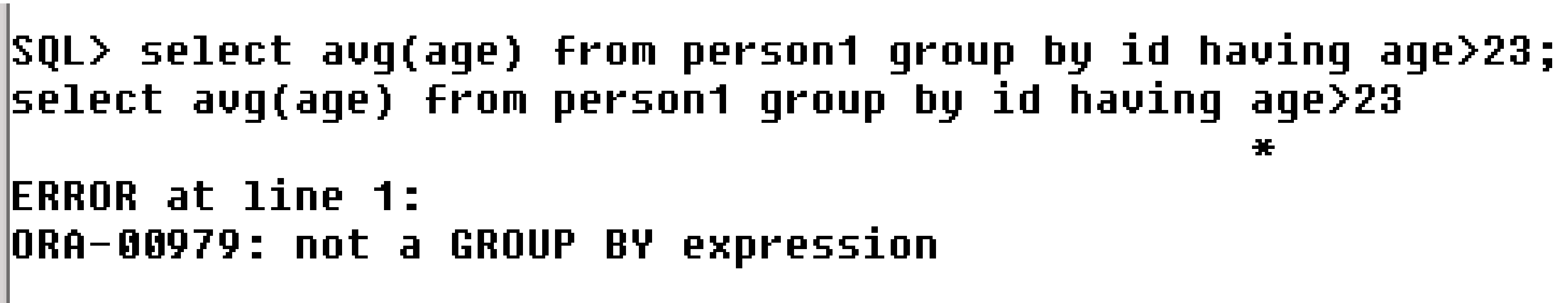
having_conditions: It specifies the conditions that are applied only to the aggregated results to restrict the groups of returned rows.



```
SQL> select * from person1;

        ID NAME                                  AGE
---------- -------------------- ----------
         2 vishal                                 23
         1 sashank                                20
         1 karthik                                20
         1 shiv                                   40
         1 Prakash                                40
         1 priyanka                               40
```



```
SQL> select avg(age) from person1 group by id;

  AVG(AGE)
----------
        32
        23

SQL> select avg(age) from person1 group by id having avg(age) >23;

  AVG(AGE)
----------
        32
```



```
SQL> select avg(age) from person1 group by id having age>23;
select avg(age) from person1 group by id having age>23
                                                *
ERROR at line 1:
ORA-00979: not a GROUP BY expression
```