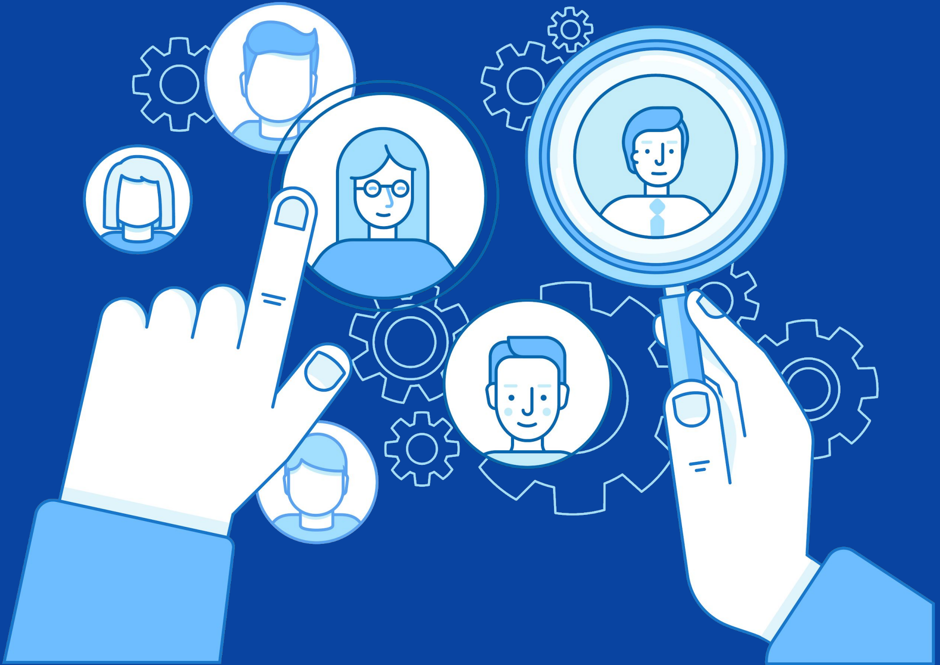
Digital Career Institute

Python Course - Database - Basic Usage



Digital Career Institute

Data Query Language

Column Distinct Values

Digital Career Institute

```
SELECT DISTINCT <columns>
FROM <table>;
```

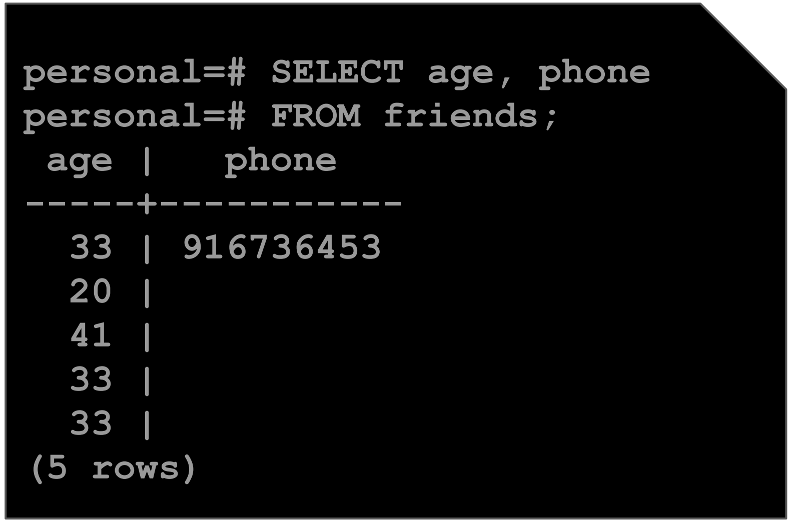
The **`DISTINCT`** clause of the **`SELECT`** command returns only the values that are different.

Column Distinct Values

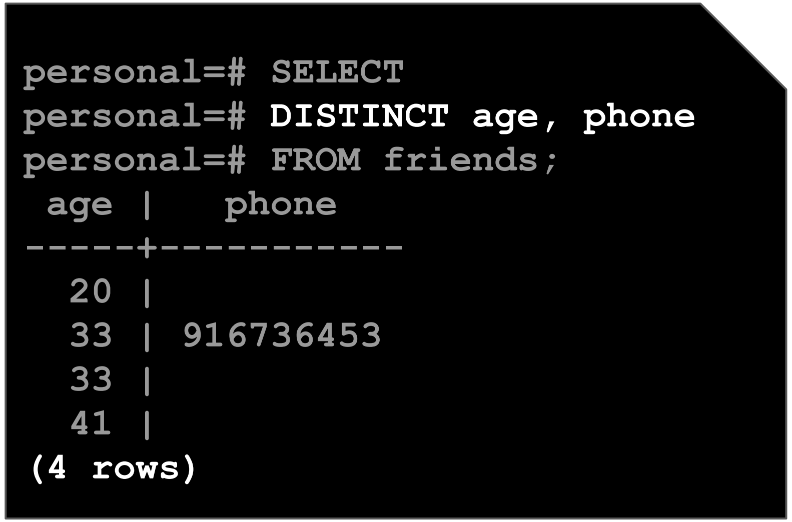
```
personal=# SELECT age
personal=# FROM friends;
 age
-----
  33
  20
  41
  33
  33
(5 rows)
```

```
personal=# SELECT DISTINCT age
personal=# FROM friends;
 age
-----
  20
  33
  41
(3 rows)
```

Column Distinct Values



```
personal=# SELECT age, phone
personal=# FROM friends;
 age |   phone
-----+-----------
   33 | 916736453
   20 |
   41 |
   33 |
   33 |
(5 rows)
```



```
personal=# SELECT
personal=# DISTINCT age, phone
personal=# FROM friends;
 age |   phone
-----+-----------
   20 |
   33 | 916736453
   33 |
   41 |
(4 rows)
```

If multiple columns are used, the result shows the records with different values in both columns.

Column Aliases

```
SELECT <column1> AS <alias1>
FROM <table>;
```

A column name can be retrieved with a different name, using an alias.

An alias is just a change on what the user sees, the table column name remains the same.

Column Aliases

```
personal=# SELECT first_name AS "Name", last_name AS "Family name"
personal-# FROM friends;
 Name        | Family name
-------------+------------
 Lisa        | Klepp
(1 row)
```

Limit the Results

Digital Career Institute

```
SELECT <columns> FROM <table>
LIMIT <number>;
```

The `LIMIT` clause can be used to limit the amount of results returned, to the indicated `<number>`.

Limit the Results

```
personal=# SELECT first_name
personal=# FROM friends;
 first_name
------------
 Lisa
 Maria
 Lidia
 James
 Karen
(5 rows)
```

```
personal=# SELECT first_name
personal=# FROM friends
personal=# LIMIT 3;
 first_name
------------
 Lisa
 Maria
 Lidia
(3 rows)
```

Limit the Results

```
SELECT <columns> FROM <table>
OFFSET <number>;
```

The `OFFSET` clause will omit the first `<number>` of rows in the output.

Limit the Results

```
personal=# SELECT first_name
personal=# FROM friends;
 first_name
------------
 Lisa
 Maria
 Lidia
 James
 Karen
(5 rows)
```

```
personal=# SELECT first_name
personal=# FROM friends
personal=# OFFSET 3;
 first_name
------------
 James
 Karen
(2 rows)
```

Sort the Results

```
SELECT <columns> FROM <table>
ORDER BY <column1> [ASC|DESC];
```

The **`ORDER BY`** clause can be used to sort the results.

An additional clause can be used to define the direction of the sorting: **`ASC`**ending or **`DESC`**ending.

If this clause is not define, it will be sorted ascendingly.

Sort the Results

```
personal=# SELECT age
personal=# FROM friends;
 age
-----
  33
  20
  41
  33
  33
(5 rows)
```

```
personal=# SELECT age
personal=# FROM friends
personal=# ORDER BY age;
 age
-----
  20
  33
  33
  33
  41
(5 rows)
```

Sort the Results

```
SELECT <columns> FROM <table>
ORDER BY
    <column1> [ASC|DESC], <column2> [ASC|DESC];
```

The output can be sorted using multiple criteria.

It will be sorted first using the first criteria.

Those records with identical value in the first column
will be sorted using the second criteria.

Sort the Results

```
personal=# SELECT age, phone
personal=# FROM friends
personal=# ORDER BY age;
 age |   phone
-----+-----------
  20 |
  33 | 916736453
  33 |
  33 |
  41 |
(5 rows)
```

```
personal=# SELECT age
personal=# FROM friends
personal=# ORDER BY age, phone;
 age |   phone
-----+-----------
  20 |
  33 |
  33 |
  33 | 916736453
  41 |
(5 rows)
```

Combining Clauses: Paginating

Digital Career Institute

```
SELECT <columns> FROM <table>
OFFSET (<page> - 1) * <size>
LIMIT <size>;
```

The **`OFFSET`** and **`LIMIT`** clauses are often used together to provide a pagination feature.

For a page size of 10 rows:

<page>	**OFFSET**	**LIMIT**
1	0	10
2	10	10
...	...	...

Combining Clauses: Rankings

Digital Career Institute

```
SELECT <columns> FROM <table>
ORDER BY <column>
LIMIT <size>;
```

The `ORDER BY` and `LIMIT` clauses are often used together to retrieve the top `<size>` records based on `<column>`.

Combining Clauses: Rankings

```
personal=# SELECT first_name, age
personal=# FROM friends
personal=# ORDER BY age DESC
personal=# LIMIT 3;
 first_name | age
------------+-----
 Karen      |  41
 Lisa       |  31
 Lidia      |  32
(5 rows)
```

The **three oldest** friends in the database.

```
personal=# SELECT first_name, age
personal=# FROM friends
personal=# ORDER BY age
personal=# LIMIT 1;
 first_name | age
------------+-----
 Maria      |  20
(5 rows)
```

The **youngest** friend in the database.

Combining Clauses: Rankings

```
SELECT <columns> FROM <table>
ORDER BY <column>
LIMIT 1
OFFSET <rank>;
```

Together with the `OFFSET` clause, the combination can be used to retrieve a rank (the Nth position in a ranking).

Combining Clauses: Rankings

```
personal=# SELECT first_name, age
personal=# FROM friends
personal=# ORDER BY age
personal=# LIMIT 1;
personal=# OFFSET 1;
 first_name | age
------------+-----
 Lisa       |  31
(5 rows)
```

The **second youngest** friend in the database.

```
personal=# SELECT first_name, age
personal=# FROM friends
personal=# ORDER BY age
personal=# LIMIT 2;
personal=# OFFSET 2;
 first_name | age
------------+-----
 Lidia      |  32
 James      |  33
(5 rows)
```

The **third and fourth youngest** friends in the database.

Combining Clauses: Rankings

```
personal=# SELECT DISTINCT age
personal=# FROM friends
personal=# ORDER BY age
personal=# LIMIT 2;
 age
-----
  20
  33
(2 rows)
```

The two youngest **ages** among the friends in the database.

```
personal=# SELECT DISTINCT age
personal=# FROM friends
personal=# ORDER BY age
personal=# LIMIT 3;
 age
-----
  20
  33
  41
(3 rows)
```

The three youngest **ages** among the friends in the database.

SQL Logical Expressions

Digital Career Institute

Logical Expressions

```
SELECT <columns> FROM <table>
WHERE <logical expression>;
```

Logical expressions can be used with various commands (`SELECT`, `UPDATE`, `DELETE`), often in the `WHERE` clause.

They behave similarly to Python logical expressions.

Logical Operators

Like Python, it has the basic operators implemented.

AND

OR

NOT

Comparison Operators

To compare a value with another one we can use:

<	>	=
<=	>=	<>

The operator `IN` can be used to match the equality in a list:

```
<column_name> IN ('value1', 'value2')
```

Comparison Operators

The operator **`IN`** can be used to compare the value in the column with a list of valid matches:

```
SELECT <columns> FROM <table>
WHERE <column> IN (<value1>, <value2>,...);
```

Comparison Operators

The operator `BETWEEN` can be used to compare the value with a range:

```
SELECT <columns> FROM <table>
WHERE <column> BETWEEN <value1> AND <value2>;
```

Equivalent to:

```
SELECT <columns> FROM <table>
WHERE <column> >= <value1> AND column <= <value2>;
```

Comparison Operators

Text fields have an additional operator named `LIKE`, that is used to match against patterns.

The `LIKE` operator uses the `%` symbol that matches against any number of characters.

```
SELECT * FROM friends
WHERE last_name LIKE 'O%';
```

This example returns a list of friends whose last name starts with the letter O.

THANK YOU

Contact Details
DCI Digital Career Institute gGmbH

Digital Career Institute
DCI