Nested

Nested In SELECT Clause

The nested function is used in the `SELECT` clause to unnest nested object type collections. The nested collection is flattened and a cartesian product is returned when querying against two or more nested collections.

Syntax

The `field_expression` parameter is required and the `path_expression` parameter is optional. Dot notation is used to show nesting level for both `field_expression` and `path_expression` parameters. For example `nestedObj.innerFieldName` denotes a field nested one level. If the user does not provide the `path_expression` parameter, the value of the path will be generated dynamically. For example the field `user.office.cubicle` would dynamically generate the path `user.office`.

```
nested(field_expression | field_expression, path_expression)
```

Using `*` With Nested In The SELECT Clause

The `*` character can be used in the `nested` function `field_expression` parameter in the `SELECT` clause to select all inner fields to a nested object. For example a user could have a `field_expression` parameter of `nestedObj.*` to denote all inner fields under `nestedObj`.

Flattening

Flattening is the process of changing the response format from OpenSearch by making the full path of an object the key, and the object it refers to the value.

Sample Input:

```
{
  "comment": {
    "data": "abc"
  }
}
```

Sample Output:

```
[
  { "comment.data": "abc" }
]
```

Query Example

The following example uses a nested query in the `SELECT` clause:

```
SELECT nested(comment.data), nested(message.info) FROM nested_objects;
```

Dataset:

```
{
  "comment": {
    "data": "abc"
  },
  "message": [
    { "info": "letter1" },
    { "info": "letter2" }
  ]
}
```

The results contain documents that match the nested query:

nested(comment.data)	nested(message.info)
abc	letter1
abc	letter2

Nested In WHERE Clause

Nested object type documents can be filtered in a query by using the nested function in the `WHERE` clause of an SQL query.

Syntax

There are two syntax options supported for the nested function when used in the `WHERE` clause of an SQL query. Both syntax options accomplish the same result of filtering a nested field with a literal value.

OPTION #1

The first option specifies the boolean condition inside the nested function with the `condition_expression` parameter.

```
nested(path_expression, condition_expression)
```

OPTION #2 (SUPPORTED IN V2 ENGINE)

The second syntax option uses the nested function on the left side of a predicate expression, and a `literal_expression` on the right. The `path_expression` parameter is optional and will be determined dynamically by the SQL plugin if not supplied. See Nested Select Clause for a more in-depth description about the `path_expression` parameter.

```
nested(field_expression | field_expression, path_expression) Operator Lite
```

Query Example

The following example uses nested queries in the `WHERE` clause and return the same result:

```
SELECT nested(message.info) FROM nested_objects WHERE nested(message.info)
SELECT nested(message.info) FROM nested_objects WHERE nested(message, mess
```

Dataset:

```
{
  "message": {
    "message": [
      { "info": "letter1" },
      { "info": "letter2" }
    ]
  }
}
```

The results contain documents that match the nested query:

nested(message.info)
letter2

Nested In ORDER BY Clause

Sorting based on nested fields across documents can be accomplished by using the nested function in the `ORDER BY` clause of an SQL query.

Syntax

By default the `ORDER BY` will be in `ASC` order. The user can specify `ASC` or `DESC` after the nested function to specify an order in the query.

```
nested(field_expression | field_expression, path_expression)
```

Query Example

The following example uses nested queries in the `ORDER BY` clause:

```
SELECT nested(message.info) FROM nested_objects ORDER BY nested(message.in
```

Dataset:

```
{
  "message": {
    "message": [
      { "info": "letter1" },
      { "info": "letter2" }
    ]
  }
}
```

The results contain documents that match the nested query:

nested(message.info)
letter2
letter1

Nested In Aggregation Queries

Nested fields can be aggregated by using the nested function in the GROUP BY clause and filtered in the HAVING clause of an SQL query.

Syntax

```
nested(field_expression | field_expression, path_expression)
```

Query Example

The following example uses nested queries in the `GROUP BY` and `HAVING` clauses:

```
SELECT count(*) FROM nested_objects GROUP BY nested(message.info) HAV
```

Dataset:

```
{
  {
    "message": [
      {"info": "letter1"},
      {"info": "letter2"}
    ]
  },
  {
    "message": [
      {"info": "letter1"},
      {"info": "letter3"}
    ]
  }
}
```

The results contain documents that match the nested query:

count(*)
2

Limitations

The nested function is supported in the V2 engine in the `SELECT` and `WHERE` (Syntax Option 2) clauses, see query-processing-engines for more details.

Resources

FAQ

Testimonials

Trademark and Brand Policy

OpenSearch Disambiguation

Connect

Connect