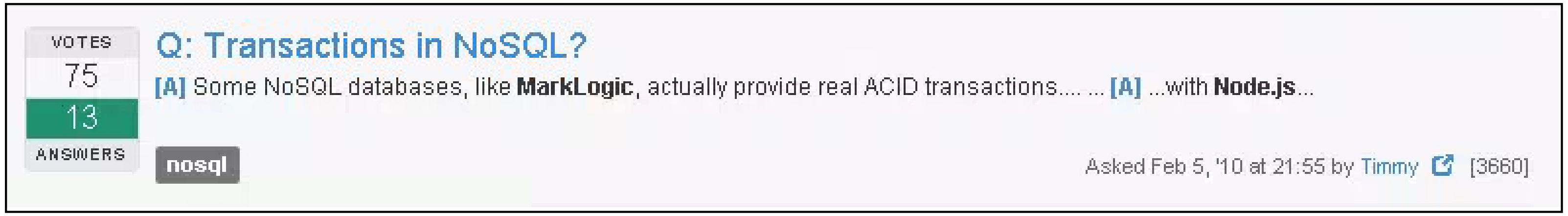
Unit 10: Implementing Snippets, Highlighting, Sorting, and Pagination

Learning Objectives

- Describe the implementation of:
  - Snippets
  - Highlights
  - Sorting
  - Pagination

Snippets

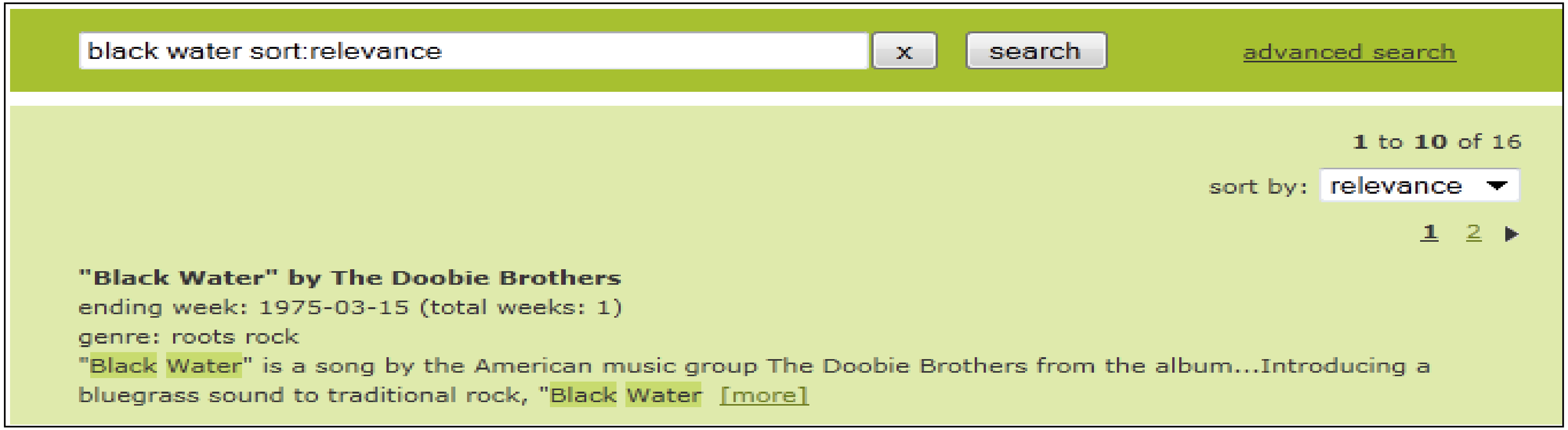
- Selected pieces of text from a document that provide context around the matching query term.



Q: Transactions in NoSQL?

[A] Some NoSQL databases, like **MarkLogic**, actually provide real ACID transactions.... ... [A] ...with **Node.js**...

Asked Feb 5, '10 at 21:55 by Timmy [3660]



black water sort:relevance

1 to **10** of 16

sort by: relevance

"Black Water" by The Doobie Brothers

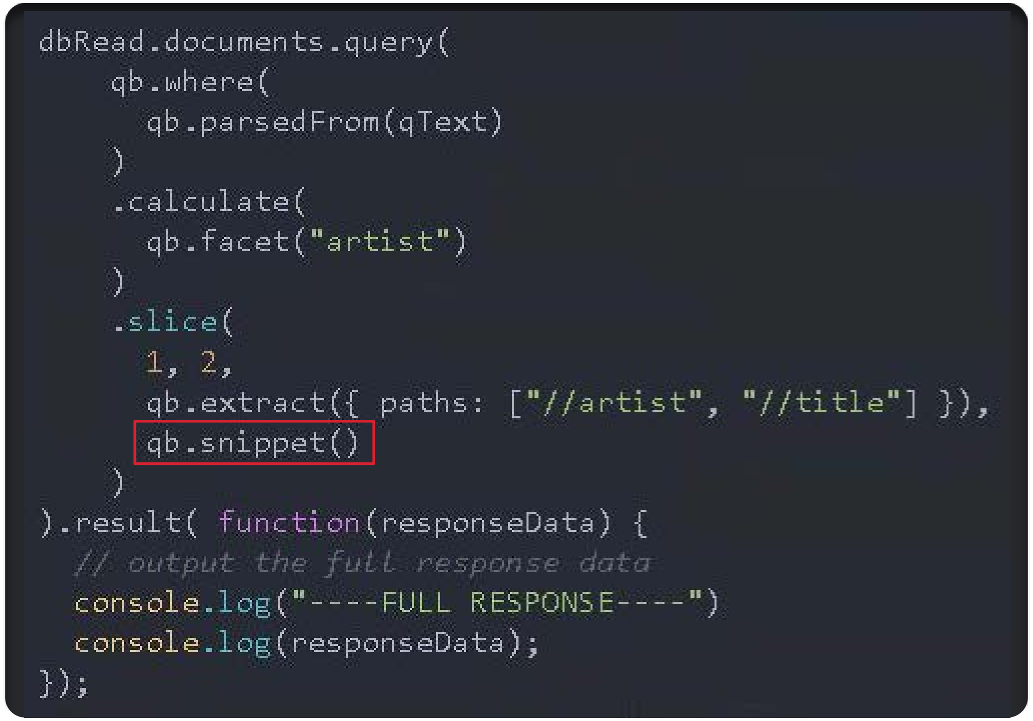
ending week: 1975-03-15 (total weeks: 1)

genre: roots rock

"Black Water" is a song by the American music group The Doobie Brothers from the album...Introducing a bluegrass sound to traditional rock, "Black Water [more]

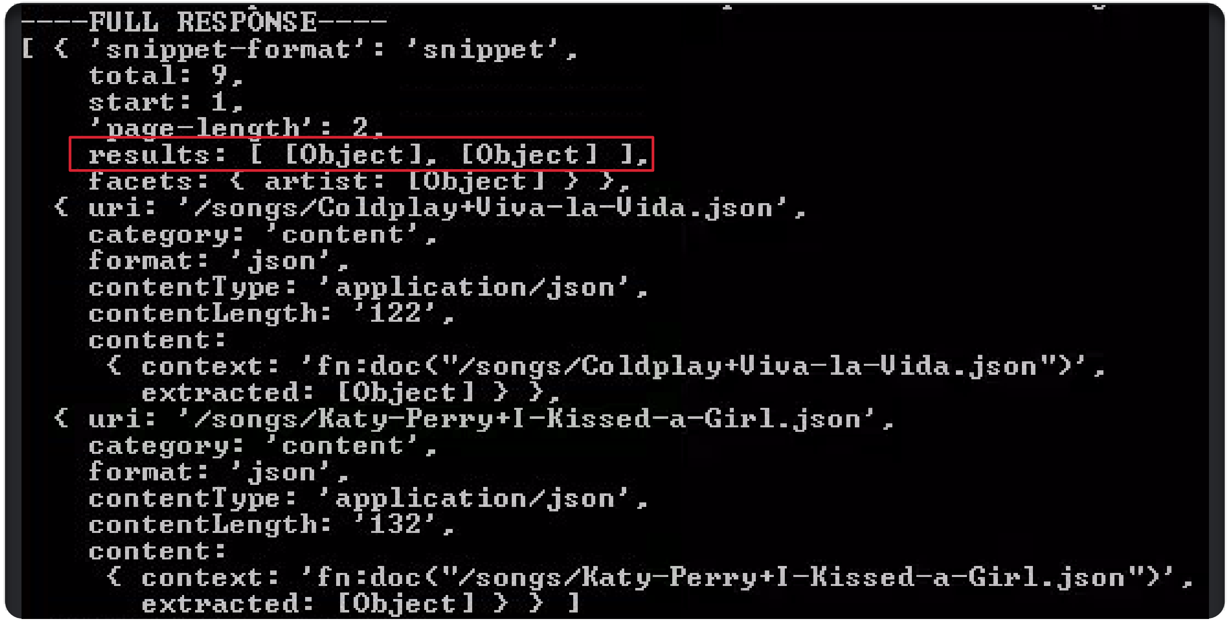
Get Snippet Data

- Query



```
dbRead.documents.query(
    qb.where(
      qb.parsedFrom(qText)
    )
    .calculate(
      qb.facet("artist")
    )
    .slice(
      1, 2,
      qb.extract({ paths: ["//artist", "//title"] }),
      qb.snippet()
    )
).result( function(responseData) {
  // output the full response data
  console.log("----FULL RESPONSE----")
  console.log(responseData);
});
```

- Response

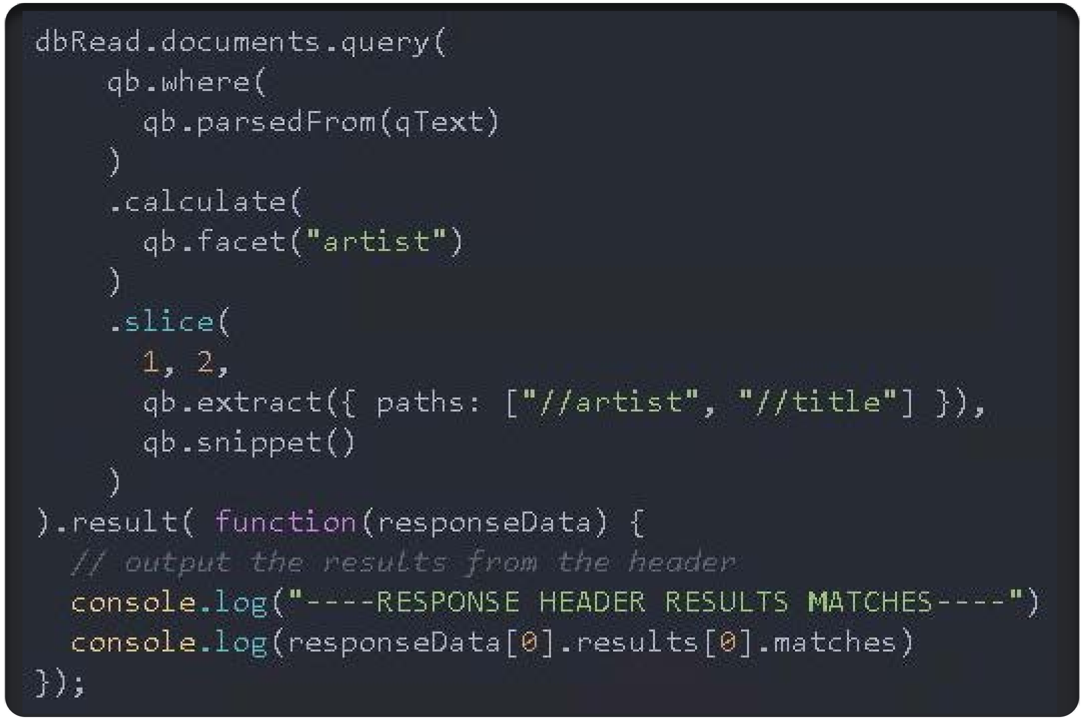


```
----FULL RESPONSE----
[ { 'snippet-format': 'snippet',
    total: 9,
    start: 1,
    'page-length': 2,
    results: [ [Object], [Object] ],
    facets: { artist: [Object] } },
  { uri: '/songs/Coldplay+Viva-la-Vida.json',
    category: 'content',
    format: 'json',
    contentType: 'application/json',
    contentLength: '122',
    content:
     { context: 'fn:doc("/songs/Coldplay+Viva-la-Vida.json")',
       extracted: [Object] } },
  { uri: '/songs/Katy-Perry+I-Kissed-a-Girl.json',
    category: 'content',
    format: 'json',
    contentType: 'application/json',
    contentLength: '132',
    content:
     { context: 'fn:doc("/songs/Katy-Perry+I-Kissed-a-Girl.json")',
       extracted: [Object] } } ]
```

- Snippet data is in the first item of the response array in **results**.

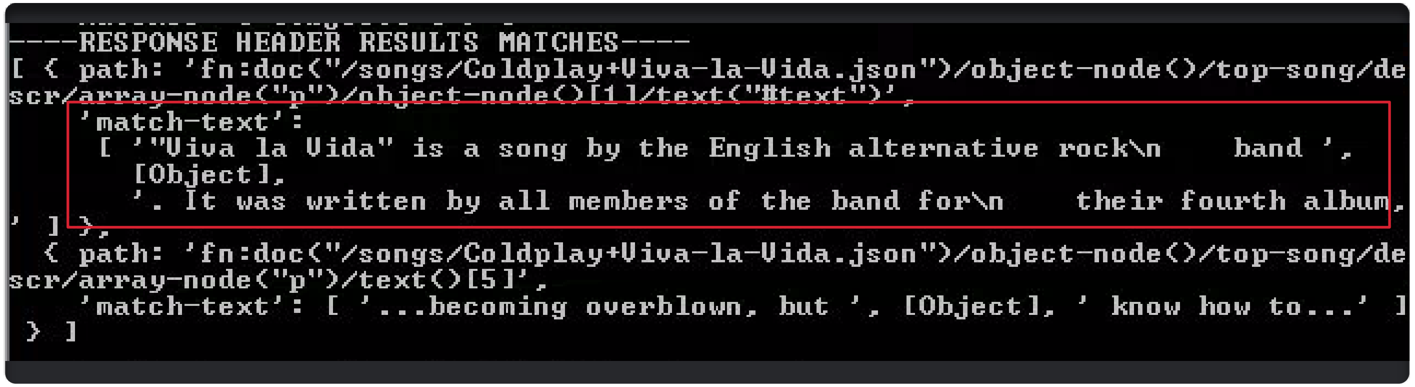
Get Snippet Data

- Query



```
dbRead.documents.query(
    qb.where(
      qb.parsedFrom(qText)
    )
    .calculate(
      qb.facet("artist")
    )
    .slice(
      1, 2,
      qb.extract({ paths: ["//artist", "//title"] }),
      qb.snippet()
    )
).result( function(responseData) {
  // output the results from the header
  console.log("----RESPONSE HEADER RESULTS MATCHES----")
  console.log(responseData[0].results[0].matches)
});
```

- Response

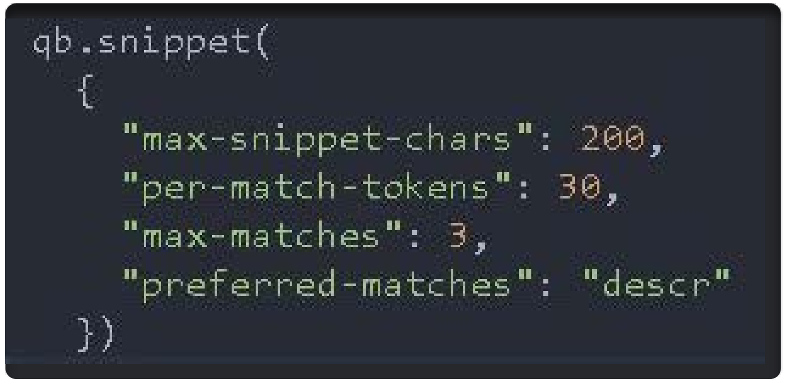


```
----RESPONSE HEADER RESULTS MATCHES----
[ { path: 'fn:doc("/songs/Coldplay+Viva-la-Vida.json")/object-node()/top-song/de
scr/array-node("p")/object-node()[1]/text("#text")',
    'match-text':
     [ '"Viva la Vida" is a song by the English alternative rock\n    band ',
       [Object],
       '. It was written by all members of the band for\n    their fourth album,
' ] },
  { path: 'fn:doc("/songs/Coldplay+Viva-la-Vida.json")/object-node()/top-song/de
scr/array-node("p")/text()[5]',
    'match-text': [ '...becoming overblown, but ', [Object], ' know how to...' ]
 } ]
```

- **match-text** is the snippet.

Customize Snippet Data

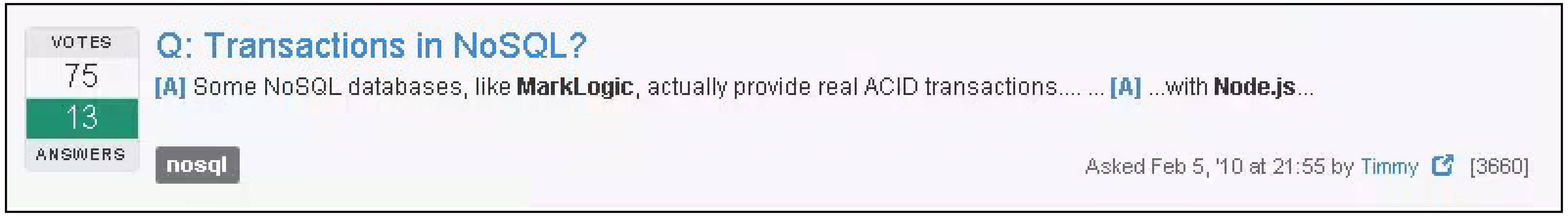
- By default snippet is output where the first match occurs
- Customize snippet response with options



```
qb.snippet(
  {
    "max-snippet-chars": 200,
    "per-match-tokens": 30,
    "max-matches": 3,
    "preferred-matches": "descr"
  })
```

Highlighting

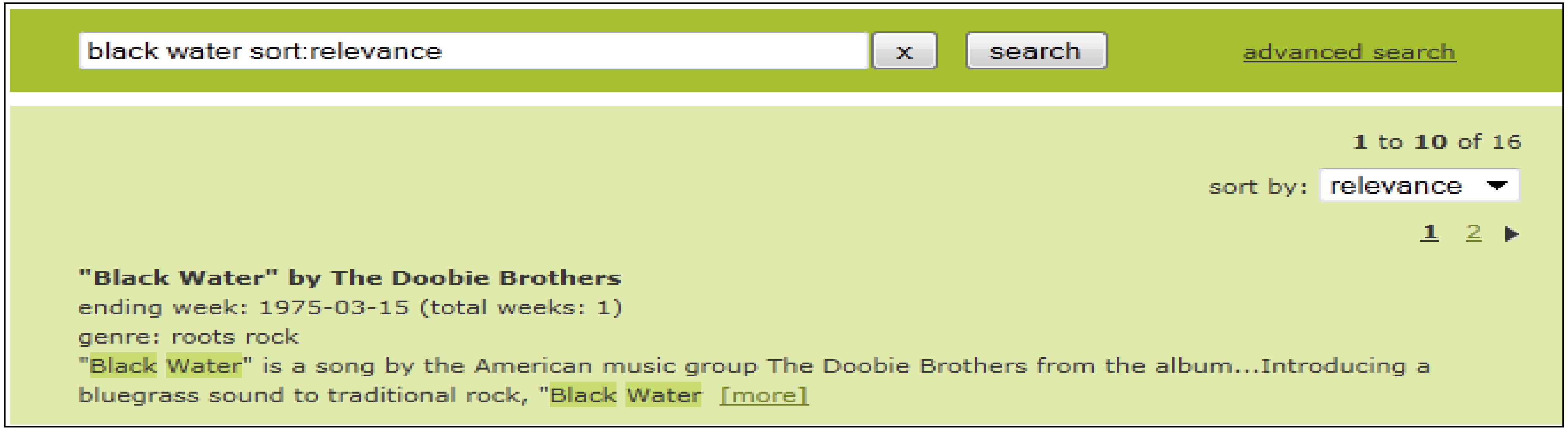
- Draw attention to the matching terms in the snippet.



Q: Transactions in NoSQL?

[A] Some NoSQL databases, like **MarkLogic**, actually provide real ACID transactions.... ... [A] ...with **Node.js**...

Asked Feb 5, '10 at 21:55 by Timmy [3660]



"Black Water" by The Doobie Brothers

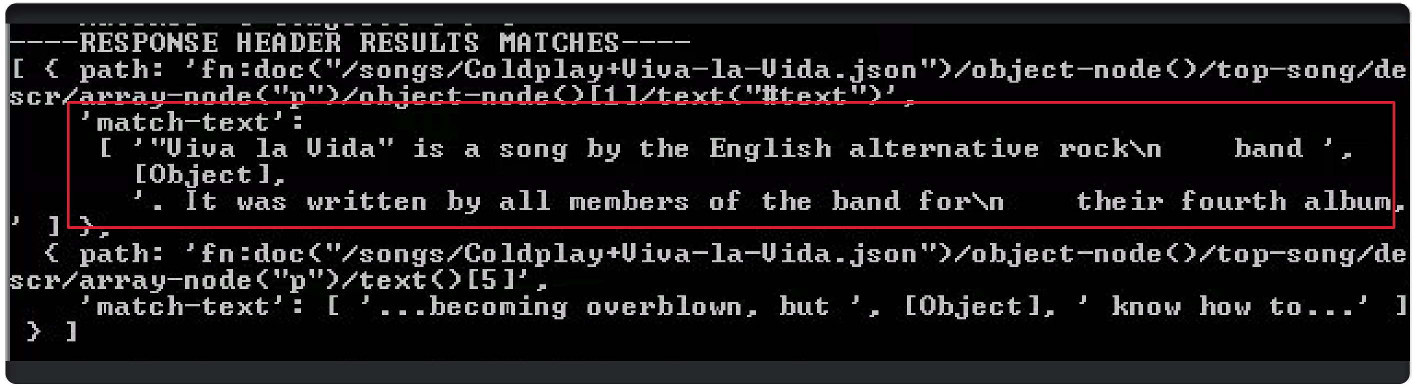
ending week: 1975-03-15 (total weeks: 1)

genre: roots rock

"Black Water" is a song by the American music group The Doobie Brothers from the album...Introducing a bluegrass sound to traditional rock, "Black Water [more]

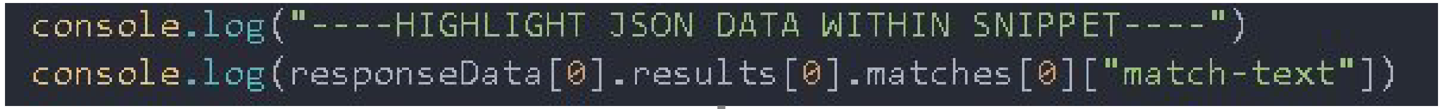
Highlighting

- The JSON objects in the **match-text** indicate a highlighted term:

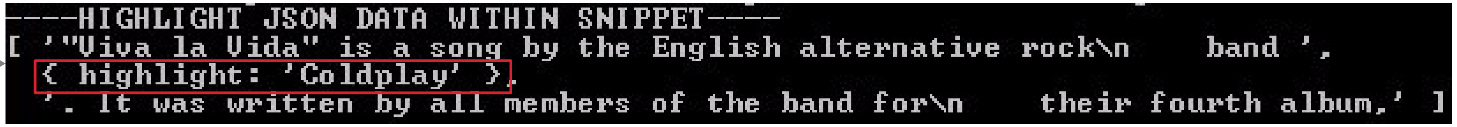


```
----RESPONSE HEADER RESULTS MATCHES----
[ { path: 'fn:doc("/songs/Coldplay+Viva-la-Vida.json")/object-node()/top-song/de
scr/array-node("p")/object-node()[1]/text("#text")',
    'match-text':
     [ '"Viva la Vida" is a song by the English alternative rock\n    band ',
       [Object],
       '. It was written by all members of the band for\n    their fourth album,
' ] },
  { path: 'fn:doc("/songs/Coldplay+Viva-la-Vida.json")/object-node()/top-song/de
scr/array-node("p")/text()[5]',
    'match-text': [ '...becoming overblown, but ', [Object], ' know how to...' ]
 } ]
```

- Access using:



```
console.log("----HIGHLIGHT JSON DATA WITHIN SNIPPET----")
console.log(responseData[0].results[0].matches[0]["match-text"])
```

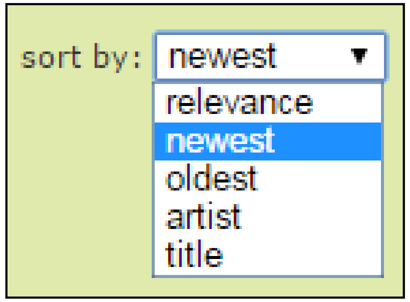


```
----HIGHLIGHT JSON DATA WITHIN SNIPPET----
[ '"Viva la Vida" is a song by the English alternative rock\n    band ',
  { highlight: 'Coldplay' },
  '. It was written by all members of the band for\n    their fourth album,' ]
```

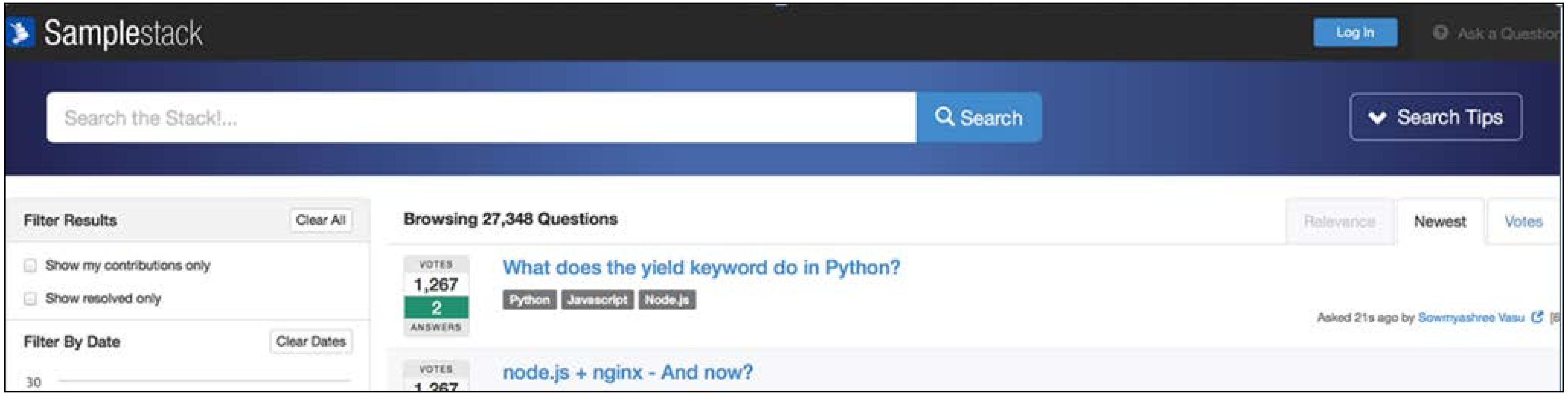
MarkLogic®

Sorting

- Return results in various orders.
- Must be backed by a range index.



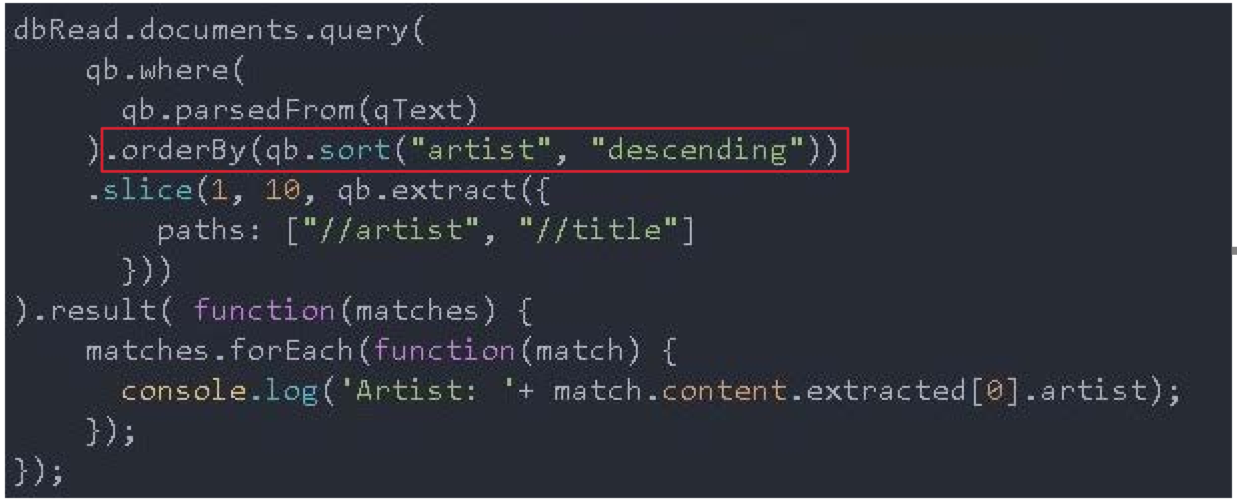
sort by: newest



MarkLogic

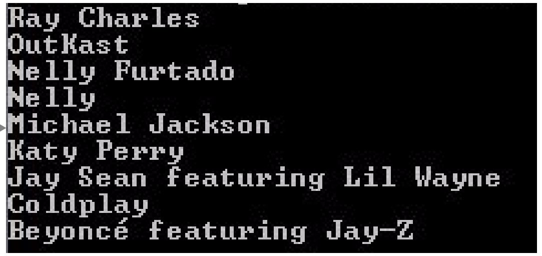
Sorting

- Query

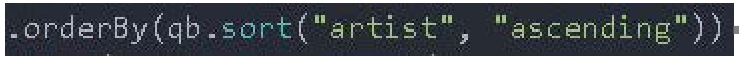


```
dbRead.documents.query(
    qb.where(
      qb.parsedFrom(qText)
    ).orderBy(qb.sort("artist", "descending"))
    .slice(1, 10, qb.extract({
        paths: ["//artist", "//title"]
      }))
).result( function(matches) {
    matches.forEach(function(match) {
      console.log('Artist: '+ match.content.extracted[0].artist);
    });
});
```

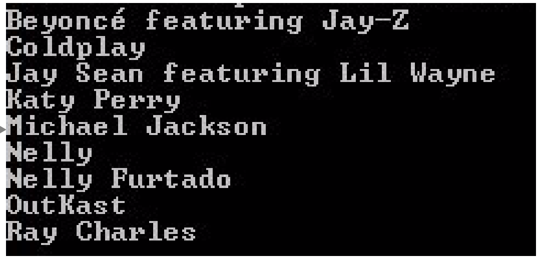
- Response



```
Ray Charles
OutKast
Nelly Furtado
Nelly
Michael Jackson
Katy Perry
Jay Sean featuring Lil Wayne
Coldplay
Beyoncé featuring Jay-Z
```



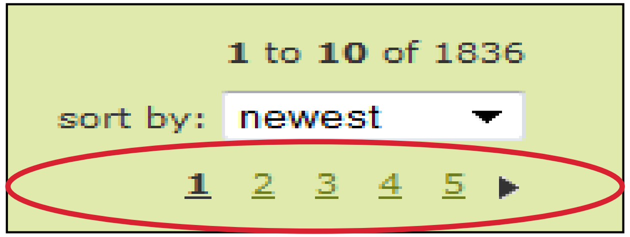
```
.orderBy(qb.sort("artist", "ascending"))
```

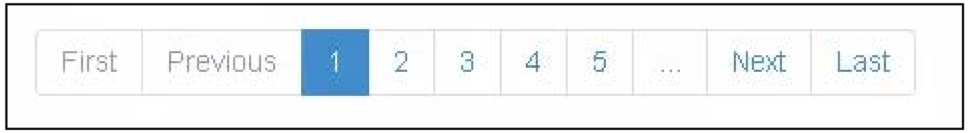


```
Beyoncé featuring Jay-Z
Coldplay
Jay Sean featuring Lil Wayne
Katy Perry
Michael Jackson
Nelly
Nelly Furtado
OutKast
Ray Charles
```

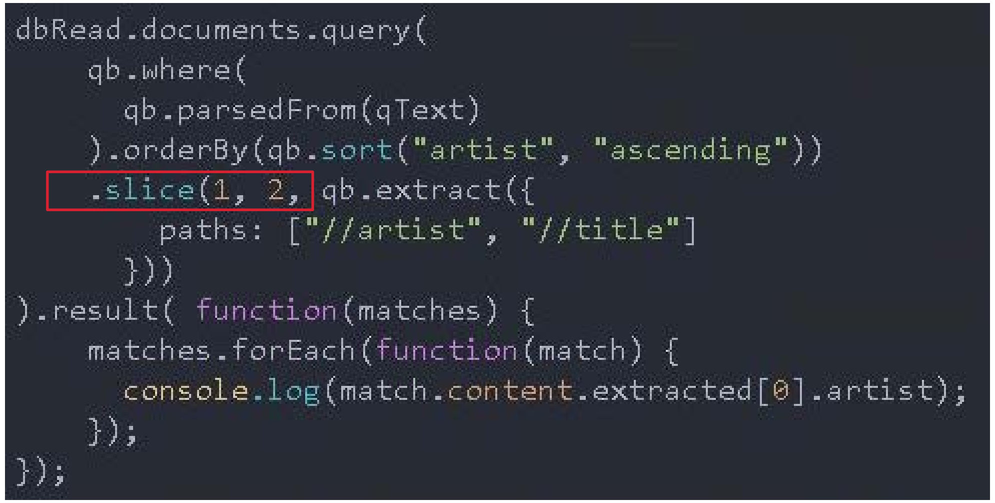
Pagination

- Break up larger result sets, optimize performance:

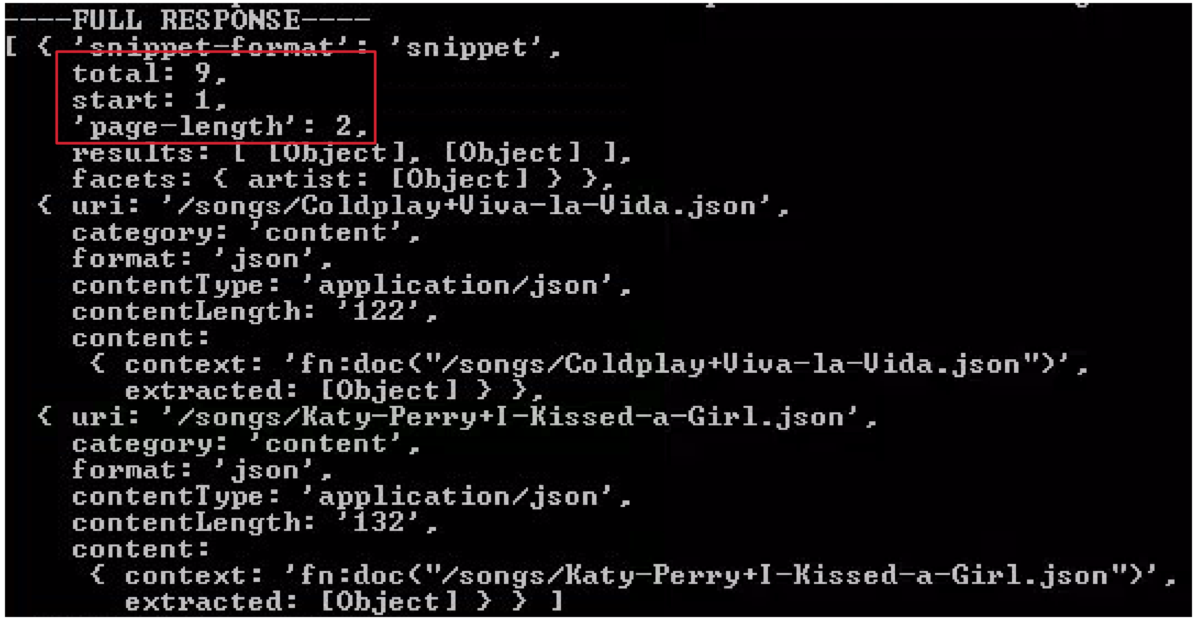




- Control through slice and response:



```
dbRead.documents.query(
    qb.where(
      qb.parsedFrom(qText)
    ).orderBy(qb.sort("artist", "ascending"))
    .slice(1, 2, qb.extract({
        paths: ["//artist", "//title"]
      }))
).result( function(matches) {
    matches.forEach(function(match) {
      console.log(match.content.extracted[0].artist);
    });
});
```



```
----FULL RESPONSE----
[ { 'snippet-format': 'snippet',
    total: 9,
    start: 1,
    'page-length': 2,
    results: [ [Object], [Object] ],
    facets: { artist: [Object] } },
  { uri: '/songs/Coldplay+Viva-la-Vida.json',
    category: 'content',
    format: 'json',
    contentType: 'application/json',
    contentLength: '122',
    content:
     { context: 'fn:doc("/songs/Coldplay+Viva-la-Vida.json")',
       extracted: [Object] } },
  { uri: '/songs/Katy-Perry+I-Kissed-a-Girl.json',
    category: 'content',
    format: 'json',
    contentType: 'application/json',
    contentLength: '132',
    content:
     { context: 'fn:doc("/songs/Katy-Perry+I-Kissed-a-Girl.json")',
       extracted: [Object] } } ]
```

Labs: Unit 10

Exercise 1: Implement Search Snippets
Exercise 2: Customize Search Snippets
Exercise 3: Implement Sorting

Unit Review Question 1:

Implementing a sort on a specific JSON property requires:

1. Positional indexes turned on
2. A field defined which includes the property
3. A trigger
4. A range index

Unit Review Question 1:

Implementing a sort on a specific JSON property requires:

1. Positional indexes turned on
2. A field defined which includes the property
3. A trigger
4. **A range index**

Unit Review Question 2:

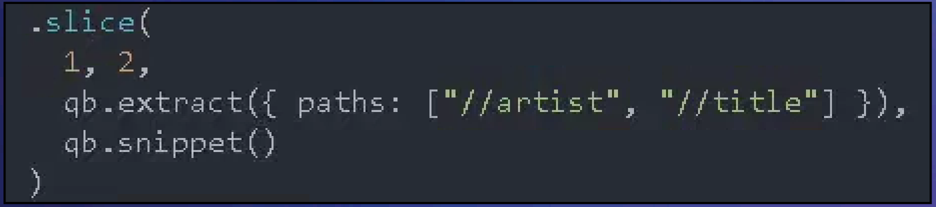
When executing a search, you receive all the results for that search in the response.

1. True
2. False

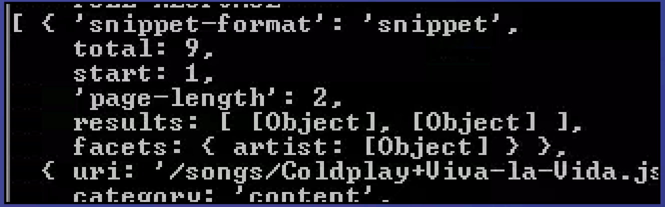
Unit Review Question 2:

When executing a search, you receive all the results for that search in the response.

1. True
2. **False – results are paginated. You receive results from the starting position and length defined in the slice:**



```
.slice(
  1, 2,
  qb.extract({ paths: ["//artist", "//title"] }),
  qb.snippet()
)
```



```
[ { 'snippet-format': 'snippet',
    total: 9,
    start: 1,
    'page-length': 2,
    results: [ [Object], [Object] ],
    facets: { artist: [Object] } },
  { uri: '/songs/Coldplay+Viva-la-Vida.js
    category: 'content',
```