Country Queries

These are example N1QL queries that may can performed to retrieve country related data.

Country By ID

The following query will get a Country by its Document ID.

Query

country_by_document_id.n1ql

```
SELECT countries
FROM `flight-data` AS countries
USE KEYS 'country_FI'
```

Result

```
[
  {
    "countries": {
      "_id": "country_FI",
      "continent_code": "EU",
      "country_code": "FI",
      "country_name": "Finland",
      "doc_type": "country"
    }
  }
]
```

The `USE KEYS` statement can accept a single document id or an array of document ids. Lets say we wanted to retrieve the country information for the United States (US), Canada (CA) and Mexico (MX).

Query

countries_by_document_id.n1ql

```
SELECT countries.*
FROM `flight-data` AS countries
USE KEYS ['country_US', 'country_CA', 'country_MX']
```

Result

```json
[
  {
    "_id": "country_US",
    "continent_code": "NA",
    "country_code": "US",
    "country_name": "United States",
    "doc_type": "country"
  },
  {
    "_id": "country_CA",
    "continent_code": "NA",
    "country_code": "CA",
    "country_name": "Canada",
    "doc_type": "country"
  },
  {
    "_id": "country_MX",
    "continent_code": "NA",
    "country_code": "MX",
    "country_name": "Mexico",
    "doc_type": "country"
  }
]
```

Notice the difference in our `SELECT` statements between these two queries. The first query we used `SELECT countries`, this return us an array of objects with a single attribute `"countries"`, which contained each of the documents attributes.

The second query we used `SELECT countries.*`, this returns every attribute on the document, and now we are returned results as an array of objects.

Country By Code

Using a well defined key pattern, we can derive a documents key, this is by far the fastest way to retrieve documents. However, this can sometimes be limiting, requiring us to be creative with key patterns and look up documents.

As our country documents are keyed by the `country_code`, generally we will retrieve those documents directly through their document id. We can retrieve these same documents by querying on the `country_code` value within the document.

Query

country_by_code.n1ql

```
SELECT countries.*
FROM `flight-data` AS countries
WHERE countries.country_code = 'US'
    AND countries.doc_type = 'country'
```

Results:

```
[
  {
    "_id": "country_US",
    "continent_code": "NA",
    "country_code": "US",
    "country_name": "United States",
    "doc_type": "country"
  }
]
```

Now if we run this query, we will get results but it will be slow, this is because we are performing a PrimaryIndex scan. We can see how our query will perform by using the `EXPLAIN` keyword.

Query

country_by_code_explain.n1ql

```
EXPLAIN
SELECT countries.*
FROM `flight-data` AS countries
WHERE countries.country_code = 'US'
    AND countries.doc_type = 'country'
```

Results

```
[
  {
    "plan": {
      "#operator": "Sequence",
      "~children": [
        {
          "#operator": "PrimaryScan",
          "index": "idx_primary",
          "keyspace": "flight-data",
          "namespace": "default",
     "using": "gsi"
    },
    {
     "#operator": "Parallel",
     "~child": {
      "#operator": "Sequence",
      "~children": [
       {
        "#operator": "Fetch",
        "as": "countries",
        "keyspace": "flight-data",
        "namespace": "default"
       },
       {
        "#operator": "Filter",
        "condition": "(((`countries`.`country_code`) = \"US\") and ((`countries`
       },
       {
        "#operator": "InitialProject",
        "result_terms": [
         {
          "expr": "`countries`",
          "star": true
         }
        ]
       },
       {
        "#operator": "FinalProject"
       }
      ]
     }
    }
   ]
  },
  "text": "\nSELECT countries.*\nFROM `flight-data` AS countries\nWHERE countries.coun
 }
]
```

We can create an index on the country code by executing the following statement:

Index

```
CREATE INDEX idx_countries_country_code ON `flight-data`( country_code )
WHERE doc_type = 'country'
    AND country_code IS NOT MISSING
USING GSI
```

This will create an index for all documents that have a `country_code` attribute and their `doc_type` is "country". Now by executing our previous query we will get results a faster.

Because we have created an index on the `country_code` attribute, we can now get all of the available country codes and names. To be sure our index is being used we can use the `EXPLAIN` keyword

Query

country_by_code_explain.n1ql

```
EXPLAIN
SELECT countries.*
FROM `flight-data` AS countries
WHERE countries.country_code = 'US'
    AND countries.doc_type = 'country'
```

Result

```
[
  {
    "plan": {
      "#operator": "Sequence",
      "~children": [
        {
          "#operator": "IndexScan",
          "index": "idx_countries_country_code",
          "index_id": "da6f8a9fa0f32767",
          "keyspace": "flight-data",
          "namespace": "default",
          "spans": [
            {
              "Range": {
                "High": [
                  "\"US\""
                ],
                "Inclusion": 3,
                "Low": [
                  "\"US\""
                ]
              }
      }
     ],
     "using": "gsi"
    },
    {
     "#operator": "Parallel",
     "~child": {
      "#operator": "Sequence",
      "~children": [
       {
        "#operator": "Fetch",
        "as": "countries",
        "keyspace": "flight-data",
        "namespace": "default"
       },
       {
        "#operator": "Filter",
        "condition": "(((`countries`.`country_code`) = \"US\") and ((`countries`
       },
       {
        "#operator": "InitialProject",
        "result_terms": [
         {
          "expr": "`countries`",
          "star": true
         }
        ]
       },
       {
        "#operator": "FinalProject"
       }
      ]
     }
    }
   ]
  },
  "text": "\nSELECT countries.*\nFROM `flight-data` AS countries\nWHERE countries.coun
 }
]
```

We see that our query is now using our `idx_countries_country_code` index. Remove the `EXPLAIN` keyword and execute the query, the results will be returned much faster.

Query

countrybycode.n1ql

```
SELECT countries.*
FROM `flight-data` AS countries
WHERE countries.country_code = 'US'
    AND countries.doc_type = 'country'
```

Results

```
[
  {
    "_id": "country_US",
    "continent_code": "NA",
    "country_code": "US",
    "country_name": "United States",
    "doc_type": "country"
  }
]
```

All Countries

Next we want to retrieve all country codes and names. To ensure our index will be used, we need to specify both the `country_code` AND `doc_type` attributes in the `WHERE` clause. Since we want all countries regardless of value we will use the `IS NOT MISSING` condition.

Query

all_countries.n1ql

```
SELECT countries.country_code, countries.country_name
FROM `flight-data` AS countries
WHERE countries.country_code IS NOT MISSING
    AND countries.doc_type = 'country'
```

Results

```
[
  {
    "country_code": "AD",
    "country_name": "Andorra"
  },
  {
    "country_code": "AE",
    "country_name": "United Arab Emirates"
  },
  {
    "country_code": "AF",
    "country_name": "Afghanistan"
  },
  {
    "country_code": "AG",
    "country_name": "Antigua and Barbuda"
  },
  ...
]
```

Our results are returned in order by the value in the index, which is `country_code`, we want the results ordered ascending by the `country_name` attribute.

Query

all_countries_ordered.n1ql

```
SELECT countries.country_code, countries.country_name
FROM `flight-data` AS countries
WHERE countries.country_code IS NOT MISSING
    AND countries.doc_type = 'country'
ORDER BY countries.country_name ASC
```

Results

```
[
  {
    "country_code": "AF",
    "country_name": "Afghanistan"
  },
  {
    "country_code": "AL",
    "country_name": "Albania"
  },
  {
    "country_code": "DZ",
    "country_name": "Algeria"
  },
  {
    "country_code": "AS",
    "country_name": "American Samoa"
  },
  ...
]
```

Countries with Continent

Now lets say we also want to return the continent that the country is in as part of the query. Building on the previous query, we can add the `continent_code` attribute

Query

all_countries_continent.n1ql

```
SELECT countries.country_code, countries.country_name,
    countries.continent
FROM `flight-data` AS countries
WHERE countries.country_code IS NOT MISSING
    AND countries.doc_type = 'country'
ORDER BY countries.country_name ASC
```

Results

```json
[
  {
    "country_code": "AF",
    "country_name": "Afghanistan"
  },
  {
    "country_code": "AL",
    "country_name": "Albania"
  },
  {
    "country_code": "DZ",
    "country_name": "Algeria"
  },
  {
    "country_code": "AS",
    "country_name": "American Samoa"
  },
  {
    "country_code": "AD",
    "country_name": "Andorra"
  },
  ...
]
```

Notice that there is no `continent` attribute, even though it was specified in the query, each result contains the `country_code` and `country_name`. This is because each of our country documents do not contain and attribute for `continent`, it is actually named `continent_code`. This is a missing attribute that will not be returned as part of your query and will not cause an error.

Query

all_countries_continent_code.n1ql

```sql
SELECT countries.country_code, countries.country_name,
    countries.continent_code
FROM `flight-data` AS countries
WHERE countries.country_code IS NOT MISSING
    AND countries.doc_type = 'country'
ORDER BY countries.country_name ASC
```

Results

```json
[
  {
    "continent_code": "AS",
    "country_code": "AF",
    "country_name": "Afghanistan"
  },
  {
    "continent_code": "EU",
    "country_code": "AL",
    "country_name": "Albania"
  },
  {
    "continent_code": "AF",
    "country_code": "DZ",
    "country_name": "Algeria"
  },
  {
    "continent_code": "OC",
    "country_code": "AS",
    "country_name": "American Samoa"
  },
  {
    "continent_code": "EU",
    "country_code": "AD",
    "country_name": "Andorra"
  },
  ...
]
```

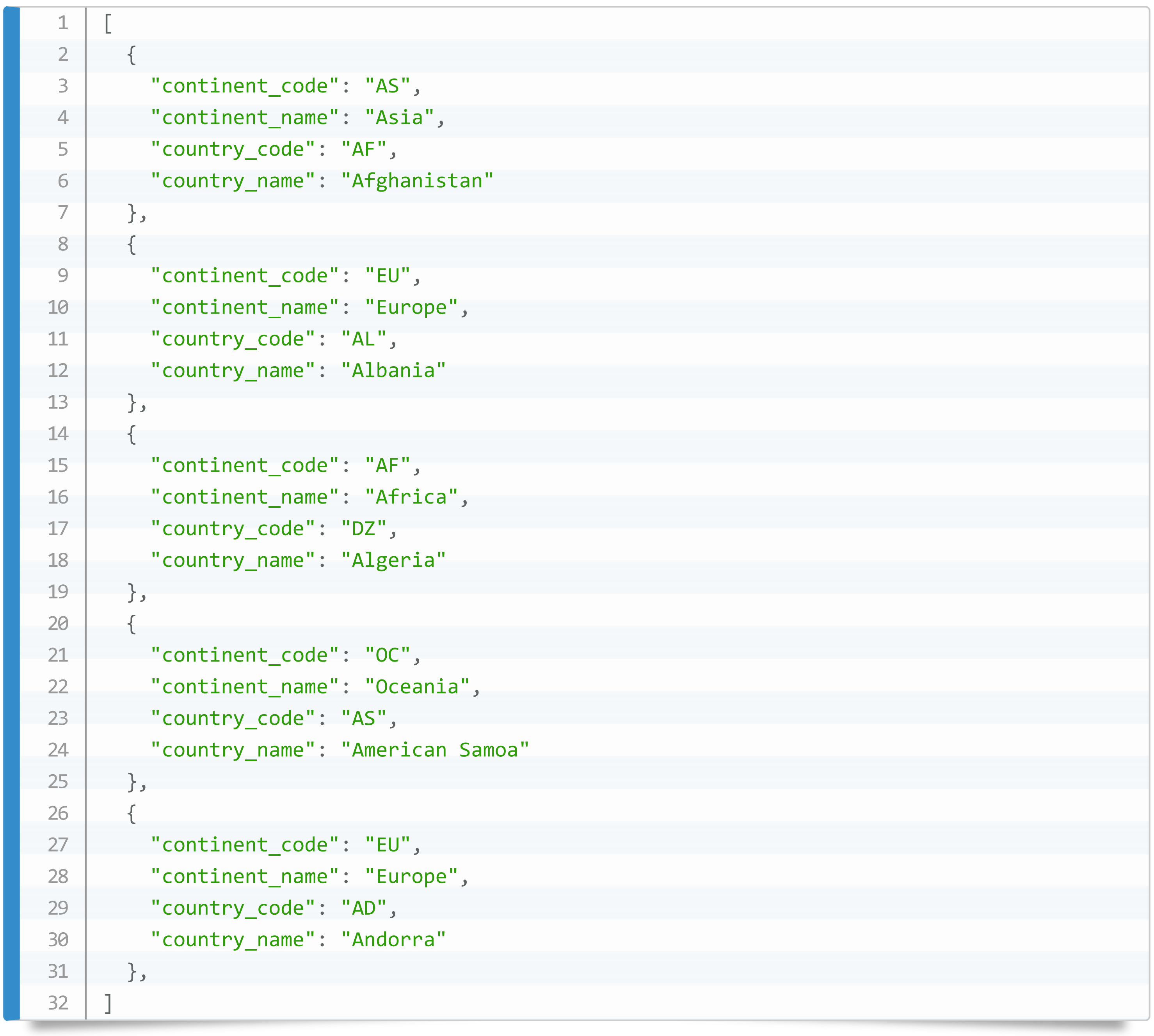
Now the `continent_code` is returned with our results. What if we wanted the name of the continent as well? To do this we need to use a `JOIN` statement. Our continent documents use the key pattern `continent_{{code}}` knowing this we can join on those document ids using the `USE KEYS` statement.

Query

all_countries_continent_names.n1ql

```sql
SELECT countries.country_code, countries.country_name,
    countries.continent_code, continents.continent_name
FROM `flight-data` AS countries
INNER JOIN `flight-data` AS continents
    ON KEYS 'continent_' || countries.continent_code
WHERE countries.country_code IS NOT MISSING
    AND countries.doc_type = 'country'
ORDER BY countries.country_name ASC
```

Results



```
[
  {
    "continent_code": "AS",
    "continent_name": "Asia",
    "country_code": "AF",
    "country_name": "Afghanistan"
  },
  {
    "continent_code": "EU",
    "continent_name": "Europe",
    "country_code": "AL",
    "country_name": "Albania"
  },
  {
    "continent_code": "AF",
    "continent_name": "Africa",
    "country_code": "DZ",
    "country_name": "Algeria"
  },
  {
    "continent_code": "OC",
    "continent_name": "Oceania",
    "country_code": "AS",
    "country_name": "American Samoa"
  },
  {
    "continent_code": "EU",
    "continent_name": "Europe",
    "country_code": "AD",
    "country_name": "Andorra"
  },
]
```