Region Queries

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

Country By ID

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

Query

region_by_document_id.n1ql

```
SELECT regions.*
FROM `flight-data` AS regions
USE KEYS 'region_US-KS'
```

Result

```
[
  {
    "_id": "region_US-KS",
    "continent_code": "NA",
    "doc_type": "region",
    "iso_country": "US",
    "local_code": "KS",
    "region_code": "US-KS",
    "region_id": 306092,
    "region_name": "Kansas"
  }
]
```

The following query will retrieve multiple Regions by their Document ID.

Query

regions_by_document_id.n1ql

```
SELECT regions.*
FROM `flight-data` AS regions
USE KEYS ['region_US-KS', 'region_US-NC', 'region_US-MN']
```

Result

```
[
  {
    "_id": "region_US-KS",
    "continent_code": "NA",
    "doc_type": "region",
    "iso_country": "US",
    "local_code": "KS",
    "region_code": "US-KS",
    "region_id": 306092,
    "region_name": "Kansas"
  },
  {
    "_id": "region_US-NC",
    "continent_code": "NA",
    "doc_type": "region",
    "iso_country": "US",
    "local_code": "NC",
    "region_code": "US-NC",
    "region_id": 306103,
    "region_name": "North Carolina"
  },
  {
    "_id": "region_US-MN",
    "continent_code": "NA",
    "doc_type": "region",
    "iso_country": "US",
    "local_code": "MN",
    "region_code": "US-MN",
    "region_id": 306099,
    "region_name": "Minnesota"
  }
]
```

Region By Code

As our region documents are keyed by the `region_code`, generally we will retrieve these documents directly through their document id. However, we can retrieve these same documents by querying on the `region_code` value within the document.

Index

idx_regions_region_code.n1ql

```
CREATE INDEX idx_regions_region_code ON `flight-data`( iso_country, region_code )
WHERE doc_type = 'region'
    AND iso_country IS NOT MISSING
    AND region_code IS NOT MISSING
USING GSI
```

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

Query

region_by_code.n1ql

```
SELECT regions.region_code, regions.region_name
FROM `flight-data` AS regions
WHERE regions.iso_country = 'US'
    AND regions.region_code = 'US-KS'
    AND regions.doc_type = 'region'
```

Result

```
[
  {
    "region_code": "US-KS",
    "region_name": "Kansas"
  }
]
```

The following query will retrieve multiple Regions by their `region_code`

Query

regions_by_code_or.n1ql

```
SELECT regions.region_code, regions.region_name
FROM `flight-data` AS regions
WHERE regions.iso_country = 'US'
    AND (
        regions.region_code = 'US-KS'
        OR regions.region_code = 'US-NC'
        OR regions.region_code = 'US-MN'
    )
    AND regions.doc_type = 'region'
```

Result

```
[
  {
    "region_code": "US-KS",
    "region_name": "Kansas"
  },
  {
    "region_code": "US-MN",
    "region_name": "Minnesota"
  },
  {
    "region_code": "US-NC",
    "region_name": "North Carolina"
  }
]
```

While the previous query will return the correct results, it is not efficient or concise. We can update the query to use the `IN` statement defining our codes as an array.

Query

regions_by_code_in.n1ql

```
SELECT regions.region_code, regions.region_name
FROM `flight-data` AS regions
WHERE regions.iso_country = 'US'
    AND regions.region_code IN ['US-KS', 'US-NC', 'US-MN']
    AND regions.doc_type = 'region'
```

Result

```
[
  {
    "region_code": "US-KS",
    "region_name": "Kansas"
  },
  {
    "region_code": "US-MN",
    "region_name": "Minnesota"
  },
  {
    "region_code": "US-NC",
    "region_name": "North Carolina"
  }
]
```

All Regions for a Country

Next we want to retrieve all region codes and names for a given country.

Query

all_regions_by_country_code.n1ql

```
SELECT regions.region_code, regions.region_name
FROM `flight-data` AS regions
WHERE regions.iso_country = 'US'
    AND regions.region_code IS NOT MISSING
    AND regions.doc_type = 'region'
ORDER BY regions.region_name ASC
```

Results

```
[
  {
    "region_code": "US-AL",
    "region_name": "Alabama"
  },
  {
    "region_code": "US-AK",
    "region_name": "Alaska"
  },
  {
    "region_code": "US-AZ",
    "region_name": "Arizona"
  },
  {
    "region_code": "US-AR",
    "region_name": "Arkansas"
  },
  {
    "region_code": "US-CA",
    "region_name": "California"
  },
  ...
]
```

Regions Country with Continent

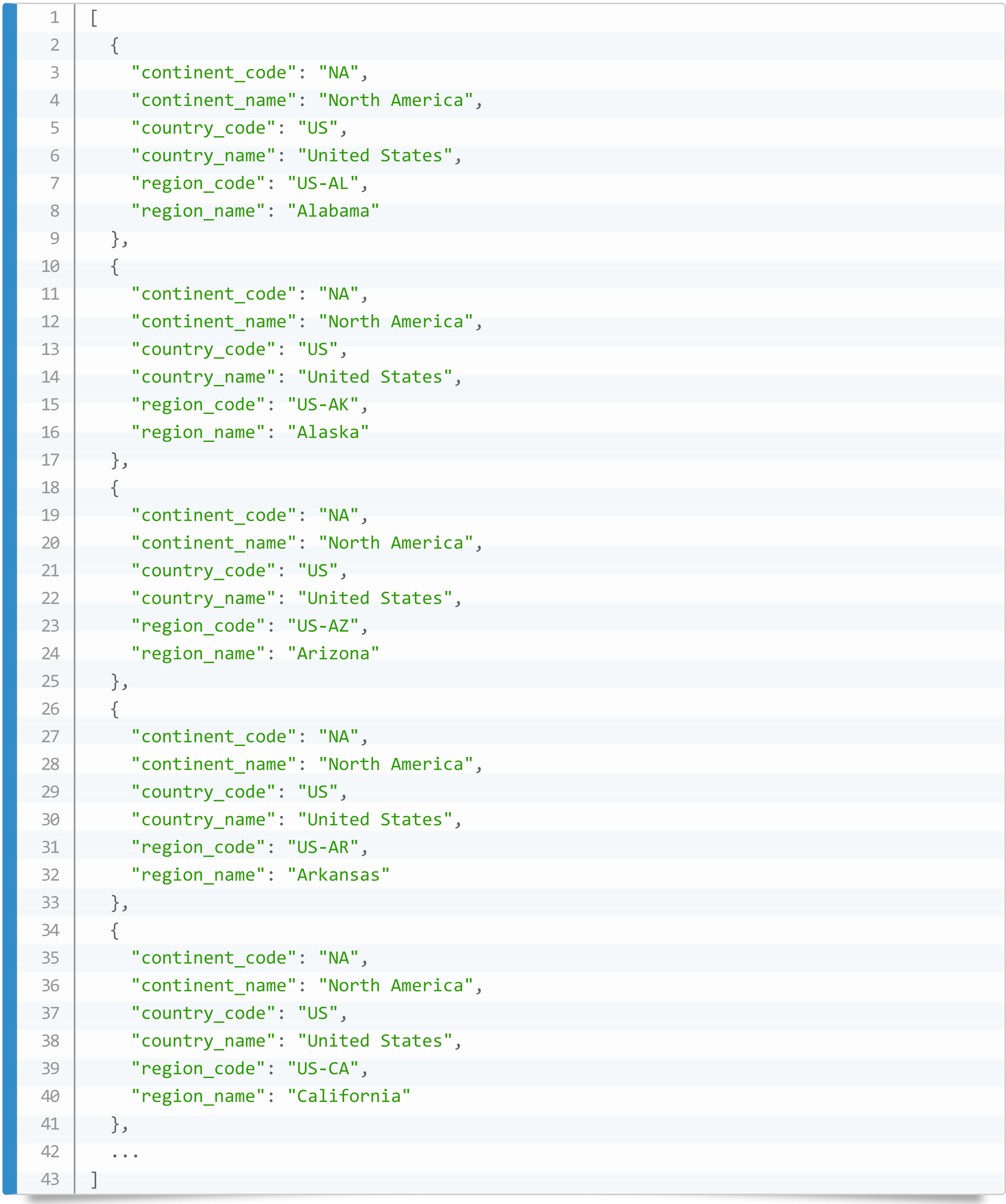
Now lets say we also want to return both the continent and country codes and names for each region.

Query

all_regions_by_country_code_continent.n1ql

```
SELECT regions.region_code, regions.region_name,
    countries.country_code, countries.country_name,
    continents.continent_code, continents.continent_name
FROM `flight-data` AS regions
INNER JOIN `flight-data` AS countries
    ON KEYS 'country_' || regions.iso_country
INNER JOIN `flight-data` AS continents
    ON KEYS 'continent_' || regions.continent_code
WHERE regions.iso_country = 'US'
    AND regions.region_code IS NOT MISSING
    AND regions.doc_type = 'region'
ORDER BY regions.region_name ASC
```

Results



```
[
  {
    "continent_code": "NA",
    "continent_name": "North America",
    "country_code": "US",
    "country_name": "United States",
    "region_code": "US-AL",
    "region_name": "Alabama"
  },
  {
    "continent_code": "NA",
    "continent_name": "North America",
    "country_code": "US",
    "country_name": "United States",
    "region_code": "US-AK",
    "region_name": "Alaska"
  },
  {
    "continent_code": "NA",
    "continent_name": "North America",
    "country_code": "US",
    "country_name": "United States",
    "region_code": "US-AZ",
    "region_name": "Arizona"
  },
  {
    "continent_code": "NA",
    "continent_name": "North America",
    "country_code": "US",
    "country_name": "United States",
    "region_code": "US-AR",
    "region_name": "Arkansas"
  },
  {
    "continent_code": "NA",
    "continent_name": "North America",
    "country_code": "US",
    "country_name": "United States",
    "region_code": "US-CA",
    "region_name": "California"
  },
  ...
]
```