

Route Queries

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

////////////////////////////////////

Routes by Originating Airport

If we want to find all of the routes originating from a given airport.

Index

[idxroutesourceairportcodes.n1ql](#)

1	CREATE INDEX idx_routes_source_airport_codes ON `flight-data` (source_airport_code, dest
2	WHERE doc_type = 'route'
3	AND source_airport_code IS NOT NULL
4	AND destination_airport_code IS NOT NULL
5	USING GSI

Query

[routesoriginatingfrom ICT.n1ql](#)

```

1  SELECT
2      {
3          "airline": {
4              "airline_code": IFNULL(
5                  airlines.airline_iata,
6                  airlines.airline_icao
7              ),
8              "airline_name": airlines.airline_name
9          },
10         "destination_airport": {
11             "airport_name": destination_airports.airport_name,
12             "iso_country": destination_airports.iso_country,
13             "iso_region": destination_airports.iso_region,
14             "airport_code": IFNULL(
15                 destination_airports.airport_iata,
16                 destination_airports.airport_icao,
17                 destination_airports.airport_ident
18             )
19         }
20     } AS route
21 FROM `flight-data` AS routes
22 INNER JOIN `flight-data` AS destination_codes
23     ON KEYS 'airport_code_' || routes.destination_airport_code
24 INNER JOIN `flight-data` AS destination_airports
25     ON KEYS 'airport_' || TOSTRING( destination_codes.id )
26 INNER JOIN `flight-data` AS airline_codes
27     ON KEYS 'airline_code_' || routes.airline_code
28 INNER JOIN `flight-data` AS airlines
29     ON KEYS 'airline_' || TOSTRING( airline_codes.id )
30 WHERE routes.source_airport_code = 'ICT'
31     AND routes.destination_airport_code IS NOT NULL
32     AND routes.doc_type = 'route'
33     AND routes.active = true
34 ORDER BY destination_airports.name ASC

```

Results

```

1  [
2    {
3      "route": {
4        "airline": {
5          "airline_code": "FL",
6          "airline_name": "AirTran Airways"
7        },
8        "destination_airport": {
9          "airport_code": "MDW",
10         "airport_name": "Chicago Midway Intl",
11         "iso_country": "US",
12         "iso_region": "US-IL"
13       }
14     },
15     {
16       "route": {
17         "airline": {
18           "airline_code": "WN",
19           "airline_name": "Southwest Airlines"
20         },
21         "destination_airport": {
22           "airport_code": "MDW",
23           "airport_name": "Chicago Midway Intl",
24           "iso_country": "US",
25           "iso_region": "US-IL"
26         }
27       }
28     },
29     ...
30   ]

```

We can retrieve an aggregate count of the number of routes originating from a given airport.

Query

[totalroutesoriginatingfromlCT.n1ql](#)

```

1  SELECT COUNT(1) AS total_routes
2  FROM `flight-data` AS routes
3  WHERE routes.source_airport_code = 'ATL'
4         AND routes.destination_airport_code IS NOT NULL
5         AND routes.doc_type = 'route'
6         AND routes.active = true

```

Result

```
1  [
2    {
3      "airports": 915
4    }
5  ]
```

Our routes model has many documents, to increase performance we are going to create 2 partitioned range indexes. The first will be for Airport Codes starting with the letters "A-M" and the second with letters "N-Z".

Index

Drop the previously created index.

[idxroutesourceairportcodes_drop.n1ql](#)

```
1  DROP INDEX `flight-data`.idx_routes_source_airport_codes
```

[idxroutesourceairportcodes_AtoM.n1ql](#)

```
1  CREATE INDEX idx_routes_source_airport_codes_AtoM ON `flight-data` (source_airport_code,
2  WHERE doc_type = 'route'
3        AND source_airport_code > 'A'
4        AND source_airport_code < 'N'
5        AND destination_airport_code IS NOT NULL
6  USING GSI
```

[idxroutesourceairportcodes_NtoZ.n1ql](#)

```
1  CREATE INDEX idx_routes_source_airport_codes_NtoZ ON `flight-data` (source_airport_code,
2  WHERE doc_type = 'route'
3        AND source_airport_code > 'N'
4        AND destination_airport_code IS NOT NULL
5  USING GSI
```

Using the `EXPLAIN` keyword on the following queries we can see that they are using separate indexes.

[totalroutesoriginatingfromATL_explain.n1ql](#)

```

1 EXPLAIN
2 SELECT COUNT(1) AS total_routes
3 FROM `flight-data` AS routes
4 WHERE routes.source_airport_code = 'ATL'
5       AND routes.destination_airport_code IS NOT NULL
6       AND routes.doc_type = 'route'
7       AND routes.active = true

```

```

1 [
2   {
3     "plan": {
4       "#operator": "Sequence",
5       "~children": [
6         {
7           "#operator": "IndexScan",
8           "index": "idx_routes_source_airport_codes_AtoM",
9           "index_id": "31d1f9b502d8dabd",
10          "keyspace": "flight-data",
11          "namespace": "default",

```

[totalroutesoriginatingfromSFO_explain.n1ql](#)

```

1 EXPLAIN
2 SELECT COUNT(1) AS total_routes
3 FROM `flight-data` AS routes
4 WHERE routes.source_airport_code = 'SFO'
5       AND routes.destination_airport_code IS NOT NULL
6       AND routes.doc_type = 'route'
7       AND routes.active = true

```

```

1 [
2   {
3     "plan": {
4       "#operator": "Sequence",
5       "~children": [
6         {
7           "#operator": "IndexScan",
8           "index": "idx_routes_source_airport_codes_NtoZ",
9           "index_id": "cd7e02bdb0ccd1d4",
10          "keyspace": "flight-data",
11          "namespace": "default",

```

Airlines flying from Airport

Query

[airlines flying from airport.n1ql](#)

```
1  SELECT results.airline_code, results.airline_name, COUNT(1) AS routes
2  FROM (
3      SELECT IFNULL( airlines.airline_iata, airlines.airline_icao ) AS airline_code,
4             airlines.airline_name
5  FROM `flight-data` AS routes
6  INNER JOIN `flight-data` AS airline_codes
7      ON KEYS 'airline_code_' || routes.airline_code
8  INNER JOIN `flight-data` AS airlines
9      ON KEYS 'airline_' || TOSTRING( airline_codes.id )
10 WHERE routes.source_airport_code = 'MRY'
11       AND routes.destination_airport_code IS NOT NULL
12       AND routes.doc_type = 'route'
13       AND routes.active = true
14 ) AS results
15 GROUP BY results.airline_code, results.airline_name
16 ORDER BY results.airline_code ASC
```

Results

```
1  [
2    {
3      "airline_code": "AA",
4      "airline_name": "American Airlines",
5      "routes": 2
6    },
7    {
8      "airline_code": "AS",
9      "airline_name": "Alaska Airlines",
10     "routes": 2
11   },
12   {
13     "airline_code": "G4",
14     "airline_name": "Allegiant Air",
15     "routes": 1
16   },
17   {
18     "airline_code": "UA",
19     "airline_name": "United Airlines",
20     "routes": 3
21   },
22   {
23     "airline_code": "US",
24     "airline_name": "US Airways",
25     "routes": 2
26   }
27 ]
```

Airlines flying more than 1 route from an Airport, ordered by the the # of routes highest to lowest.

Query

[*airlinesflyingfromairportwith2plusroutes.n1ql*](#)

```

1 SELECT results.airline_code, results.airline_name, COUNT(1) AS routes
2 FROM (
3     SELECT IFNULL( airlines.airline_iata, airlines.airline_icao ) AS airline_code,
4             airlines.airline_name
5     FROM `flight-data` AS routes
6     INNER JOIN `flight-data` AS airline_codes
7         ON KEYS 'airline_code_' || routes.airline_code
8     INNER JOIN `flight-data` AS airlines
9         ON KEYS 'airline_' || TOSTRING( airline_codes.id )
10    WHERE routes.source_airport_code = 'MRY'
11          AND routes.destination_airport_code IS NOT NULL
12          AND routes.doc_type = 'route'
13          AND routes.active = true
14 ) AS results
15 GROUP BY results.airline_code, results.airline_name
16 HAVING COUNT(1) > 1
17 ORDER BY COUNT(1) DESC, results.airline_code ASC

```

Results

```

1 [
2   {
3     "airline_code": "UA",
4     "airline_name": "United Airlines",
5     "routes": 3
6   },
7   {
8     "airline_code": "AA",
9     "airline_name": "American Airlines",
10    "routes": 2
11  },
12  {
13    "airline_code": "AS",
14    "airline_name": "Alaska Airlines",
15    "routes": 2
16  },
17  {
18    "airline_code": "US",
19    "airline_name": "US Airways",
20    "routes": 2
21  }
22 ]

```


Routes by Destination Airport

If we want to find all of the routes arriving at a given airport.

Index

[idxroutesdestinationairportcodes_AtoM.n1ql](#)

```
1 CREATE INDEX idx_routes_destination_airport_codes_AtoM ON `flight-data` ( destination_air
2 WHERE doc_type = 'route'
3     AND destination_airport_code > 'A'
4     AND destination_airport_code < 'N'
5     AND source_airport_code IS NOT NULL
6 USING GSI
```

[idxroutesdestinationairportcodes_NtoZ.n1ql](#)

```
1 CREATE INDEX idx_routes_destination_airport_codes_NtoZ ON `flight-data` ( destination_air
2 WHERE doc_type = 'route'
3     AND destination_airport_code > 'N'
4     AND source_airport_code IS NOT NULL
5 USING GSI
```

Query

[routes_toairport.n1ql](#)

```
1 SELECT airlines.airline_name,
2     IFNULL(
3         airlines.airline_iata,
4         airlines.airline_icao
5     ) AS airline_code,
6     source_airports.airport_name,
7     source_airports.iso_country,
8     source_airports.iso_region,
9     IFNULL(
10        source_airports.airport_iata,
11        source_airports.airport_icao,
12        source_airports.airport_ident
13    ) AS airport_code
14 FROM `flight-data` AS routes
15 INNER JOIN `flight-data` AS airport_codes
16     ON KEYS 'airport_code_' || routes.source_airport_code
17 INNER JOIN `flight-data` AS source_airports
18     ON KEYS 'airport_' || TOSTRING( airport_codes.id )
19 INNER JOIN `flight-data` AS airline_codes
20     ON KEYS 'airline_code_' || routes.airline_code
21 INNER JOIN `flight-data` AS airlines
22     ON KEYS 'airline_' || TOSTRING( airline_codes.id )
23 WHERE routes.destination_airport_code = 'MRY'
24     AND routes.source_airport_code IS NOT NULL
25     AND routes.doc_type = 'route'
26     AND routes.active = true
27 ORDER BY source_airports.airport_name ASC
```

Results

```
1  [  
2    {  
3      "airline_code": "UA",  
4      "airline_name": "United Airlines",  
5      "airport_code": "DEN",  
6      "airport_name": "Denver Intl",  
7      "iso_country": "US",  
8      "iso_region": "US-CO"  
9    },  
10   {  
11     "airline_code": "UA",  
12     "airline_name": "United Airlines",  
13     "airport_code": "LAX",  
14     "airport_name": "Los Angeles Intl",  
15     "iso_country": "US",  
16     "iso_region": "US-CA"  
17   },  
18   ...  
19 ]
```

What if we wanted to customize the output into a nested object / collection?

Query

[routes_toairport_formatted.n1ql](#)

```

1  SELECT
2      {
3          "airline": {
4              "airline_code": IFNULL(
5                  airlines.airline_iata,
6                  airlines.airline_icao
7              ),
8              "airline_name": airlines.airline_name
9          },
10         "source_airport": {
11             "airport_name": source_airports.airport_name,
12             "iso_country": source_airports.iso_country,
13             "iso_region": source_airports.iso_region,
14             "airport_code": IFNULL(
15                 source_airports.airport_iata,
16                 source_airports.airport_icao,
17                 source_airports.airport_ident
18             )
19         }
20     } AS route
21 FROM `flight-data` AS routes
22 INNER JOIN `flight-data` AS airport_codes
23     ON KEYS 'airport_code_' || routes.source_airport_code
24 INNER JOIN `flight-data` AS source_airports
25     ON KEYS 'airport_' || TOSTRING( airport_codes.id )
26 INNER JOIN `flight-data` AS airline_codes
27     ON KEYS 'airline_code_' || routes.airline_code
28 INNER JOIN `flight-data` AS airlines
29     ON KEYS 'airline_' || TOSTRING( airline_codes.id )
30 WHERE routes.destination_airport_code = 'MRY'
31     AND routes.source_airport_code IS NOT NULL
32     AND routes.doc_type = 'route'
33     AND routes.active = true
34 ORDER BY source_airports.airport_name ASC

```

Results

```

1  [
2    {
3      "route": {
4        "airline": {
5          "airline_code": "UA",
6          "airline_name": "United Airlines"
7        },
8        "source_airport": {
9          "airport_code": "DEN",
10         "airport_name": "Denver Intl",
11         "iso_country": "US",
12         "iso_region": "US-CO"
13       }
14     },
15   },
16   {
17     "route": {
18       "airline": {
19         "airline_code": "UA",
20         "airline_name": "United Airlines"
21       },
22       "source_airport": {
23         "airport_code": "LAX",
24         "airport_name": "Los Angeles Intl",
25         "iso_country": "US",
26         "iso_region": "US-CA"
27       }
28     }
29   },
30   ...
31 ]

```

This works, however our `airline` and `source_airport` attributes are now nested under a `route` attribute, we can remove this by using a `.*` at the end of the objects construction.

Query

[routes to airport formatted flattened.n1ql](#)

```

1  SELECT
2      {
3          "airline": {
4              "airline_code": IFNULL(
5                  airlines.airline_iata,
6                  airlines.airline_icao
7              ),
8              "airline_name": airlines.airline_name
9          },
10         "source_airport": {
11             "airport_name": source_airports.airport_name,
12             "iso_country": source_airports.iso_country,
13             "iso_region": source_airports.iso_region,
14             "airport_code": IFNULL(
15                 source_airports.airport_iata,
16                 source_airports.airport_icao,
17                 source_airports.airport_ident
18             )
19         }
20     }.*
21 FROM `flight-data` AS routes
22 INNER JOIN `flight-data` AS airport_codes
23     ON KEYS 'airport_code_' || routes.source_airport_code
24 INNER JOIN `flight-data` AS source_airports
25     ON KEYS 'airport_' || TOSTRING( airport_codes.id )
26 INNER JOIN `flight-data` AS airline_codes
27     ON KEYS 'airline_code_' || routes.airline_code
28 INNER JOIN `flight-data` AS airlines
29     ON KEYS 'airline_' || TOSTRING( airline_codes.id )
30 WHERE routes.destination_airport_code = 'MRY'
31     AND routes.source_airport_code IS NOT NULL
32     AND routes.doc_type = 'route'
33     AND routes.active = true
34 ORDER BY source_airports.airport_name ASC

```

Results

```

1  [
2    {
3      "airline": {
4        "airline_code": "UA",
5        "airline_name": "United Airlines"
6      },
7      "source_airport": {
8        "airport_code": "DEN",
9        "airport_name": "Denver Intl",
10       "iso_country": "US",
11       "iso_region": "US-CO"
12     }
13   },
14   {
15     "airline": {
16       "airline_code": "UA",
17       "airline_name": "United Airlines"
18     },
19     "source_airport": {
20       "airport_code": "LAX",
21       "airport_name": "Los Angeles Intl",
22       "iso_country": "US",
23       "iso_region": "US-CA"
24     }
25   },
26   ...
27 ]

```

Now that we have a nicely formatted object, what if we wanted to return the route distance in miles as well? Note that `69` is used for miles and `111.045` is used for kilometers. This is the distance between degrees (latitude / longitude).

Query

[routes to airport with distance.n1ql](#)

```

1  SELECT
2      {
3          "airline": {
4              "airline_code": IFNULL(
5                  airlines.airline_iata,
6                  airlines.airline_icao
7              ),
8              "airline_name": airlines.airline_name
9          },
10         "source_airport": {
11             "airport_name": source_airports.airport_name,
12             "iso_country": source_airports.iso_country,
13             "iso_region": source_airports.iso_region,
14             "airport_code": IFNULL(
15                 source_airports.airport_iata,
16                 source_airports.airport_icao,
17                 source_airports.airport_ident
18             )
19         },
20         "distance": ROUND(69 * DEGREES(ACOS(COS(RADIANS( source_airports.geo.latitude ))
21             * COS(RADIANS( destination_airports.geo.latitude ))
22             * COS(RADIANS( source_airports.geo.longitude ) - RADIANS( destination_airports.geo
23             + SIN(RADIANS( source_airports.geo.latitude ))
24             * SIN(RADIANS( destination_airports.geo.latitude )))), 2)
25     }.*
26 FROM `flight-data` AS routes
27 INNER JOIN `flight-data` AS airport_codes
28     ON KEYS 'airport_code_' || routes.source_airport_code
29 INNER JOIN `flight-data` AS source_airports
30     ON KEYS 'airport_' || TOSTRING( airport_codes.id )
31 INNER JOIN `flight-data` AS destination_airport_codes
32     ON KEYS 'airport_code_' || routes.destination_airport_code
33 INNER JOIN `flight-data` AS destination_airports
34     ON KEYS 'airport_' || TOSTRING( destination_airport_codes.id )
35 INNER JOIN `flight-data` AS airline_codes
36     ON KEYS 'airline_code_' || routes.airline_code
37 INNER JOIN `flight-data` AS airlines
38     ON KEYS 'airline_' || TOSTRING( airline_codes.id )
39 WHERE routes.destination_airport_code = 'MRY'
40     AND routes.source_airport_code IS NOT NULL
41     AND routes.doc_type = 'route'
42     AND routes.active = true
43 ORDER BY source_airports.airport_name ASC

```

Results


```
1  [
2    {
3      "airline": {
4        "airline_code": "UA",
5        "airline_name": "United Airlines"
6      },
7      "distance": 956.1,
8      "source_airport": {
9        "airport_code": "DEN",
10       "airport_name": "Denver Intl",
11       "iso_country": "US",
12       "iso_region": "US-CO"
13     }
14   },
15   {
16     "airline": {
17       "airline_code": "UA",
18       "airline_name": "United Airlines"
19     },
20     "distance": 265.94,
21     "source_airport": {
22       "airport_code": "LAX",
23       "airport_name": "Los Angeles Intl",
24       "iso_country": "US",
25       "iso_region": "US-CA"
26     }
27   },
28   ...
29 ]
```

Airlines flying into Airport

Query

[totalairlinesflyingtodestination.n1ql](#)

```

1 SELECT results.airline_code, results.airline_name, COUNT(1) AS routes
2 FROM (
3     SELECT IFNULL( airlines.airline_iata, airlines.airline_icao ) AS airline_code,
4             airlines.airline_name
5     FROM `flight-data` AS routes
6     INNER JOIN `flight-data` AS airline_codes
7         ON KEYS 'airline_code_' || routes.airline_code
8     INNER JOIN `flight-data` AS airlines
9         ON KEYS 'airline_' || TOSTRING( airline_codes.id )
10    WHERE routes.destination_airport_code = 'GSO'
11          AND routes.source_airport_code IS NOT NULL
12          AND routes.doc_type = 'route'
13          AND routes.active = true
14 ) AS results
15 GROUP BY results.airline_code, results.airline_name
16 ORDER BY results.airline_code ASC

```

Results

```

1 [
2   {
3     "airline_code": "9E",
4     "airline_name": "Pinnacle Airlines",
5     "routes": 1
6   },
7   {
8     "airline_code": "AA",
9     "airline_name": "American Airlines",
10    "routes": 6
11  },
12  {
13    "airline_code": "AF",
14    "airline_name": "Air France",
15    "routes": 1
16  },
17  {
18    "airline_code": "AS",
19    "airline_name": "Alaska Airlines",
20    "routes": 1
21  },
22  {
23    "airline_code": "AZ",
24    "airline_name": "Alitalia",
25    "routes": 1
26  },
27  {

```

```

28     "airline_code": "DL",
29     "airline_name": "Delta Air Lines",
30     "routes": 3
31 },
32 {
33     "airline_code": "F9",
34     "airline_name": "Frontier Airlines",
35     "routes": 1
36 },
37 {
38     "airline_code": "G4",
39     "airline_name": "Allegiant Air",
40     "routes": 2
41 },
42 {
43     "airline_code": "KL",
44     "airline_name": "KLM Royal Dutch Airlines",
45     "routes": 1
46 },
47 {
48     "airline_code": "UA",
49     "airline_name": "United Airlines",
50     "routes": 3
51 },
52 {
53     "airline_code": "US",
54     "airline_name": "US Airways",
55     "routes": 6
56 }
57 ]

```

Airlines flying more than 2 routes into an Airport, ordered by the the # of routes highest to lowest.

Query

[airlinesflyingtodestinationwith3plusroutes.n1ql](#)

```

1 SELECT results.airline_code, results.airline_name, COUNT(1) AS routes
2 FROM (
3     SELECT IFNULL( airlines.airline_iata, airlines.airline_icao ) AS airline_code,
4         airlines.airline_name
5     FROM `flight-data` AS routes
6     INNER JOIN `flight-data` AS airline_codes
7         ON KEYS 'airline_code_' || routes.airline_code
8     INNER JOIN `flight-data` AS airlines
9         ON KEYS 'airline_' || TOSTRING( airline_codes.id )
10    WHERE routes.destination_airport_code = 'GSO'
11        AND routes.source_airport_code IS NOT NULL
12        AND routes.doc_type = 'route'
13        AND routes.active = true
14 ) AS results
15 GROUP BY results.airline_code, results.airline_name
16 HAVING COUNT(1) > 2
17 ORDER BY COUNT(1) DESC, results.airline_code ASC

```

Results

```

1 [
2   {
3     "airline_code": "AA",
4     "airline_name": "American Airlines",
5     "routes": 6
6   },
7   {
8     "airline_code": "US",
9     "airline_name": "US Airways",
10    "routes": 6
11  },
12  {
13    "airline_code": "DL",
14    "airline_name": "Delta Air Lines",
15    "routes": 3
16  },
17  {
18    "airline_code": "UA",
19    "airline_name": "United Airlines",
20    "routes": 3
21  }
22 ]

```

Routes within certain distance

If we only want to return routes originating from a given airport that are with a certain distance in miles we would perform the following queries.

For this query we need to provide it 3 pieces of information which are represented by `{{tokens}}` .

- The Airport IATA, ICAO or Ident Code i.e. MCI
- A `distance_unit`
 - Kilometers: 111.045
 - Miles: 69
- A `max_distance` in which to contain results in, i.e. `300`

Base Query

```

1  SELECT results.route.airline, results.route.source_airport,
2      ROUND( results.route.distance, 2 ) AS distance
3  FROM (
4      SELECT
5          {
6              "airline": {
7                  "airline_code": IFNULL( airlines.airline_iata, airlines.airline_icao ),
8                  "airline_name": airlines.airline_name
9              },
10             "source_airport": {
11                 "airport_name": source_airports.airport_name,
12                 "iso_country": source_airports.iso_country,
13                 "iso_region": source_airports.iso_region,
14                 "airport_code": IFNULL(
15                     source_airports.airport_iata,
16                     source_airports.airport_icao,
17                     source_airports.airport_ident )
18             },
19             "distance": {{{distance_unit}}} * DEGREES(ACOS(COS(RADIANS( source_airports.geo
20 * COS(RADIANS( destination_airports.geo.latitude ))
21 * COS(RADIANS( source_airports.geo.longitude ) - RADIANS( destination_airpor
22 + SIN(RADIANS( source_airports.geo.latitude ))
23 * SIN(RADIANS( destination_airports.geo.latitude )))))
24     } AS route
25 FROM `flight-data` AS routes
26 INNER JOIN `flight-data` AS source_airport_codes
27     ON KEYS 'airport_code_' || routes.source_airport_code
28 INNER JOIN `flight-data` AS source_airports
29     ON KEYS 'airport_' || TOSTRING( source_airport_codes.id )
30 INNER JOIN `flight-data` AS destination_airport_codes
31     ON KEYS 'airport_code_' || routes.destination_airport_code
32 INNER JOIN `flight-data` AS destination_airports
33     ON KEYS 'airport_' || TOSTRING( destination_airport_codes.id )
34 INNER JOIN `flight-data` AS airline_codes
35     ON KEYS 'airline_code_' || routes.airline_code
36 INNER JOIN `flight-data` AS airlines
37     ON KEYS 'airline_' || TOSTRING( airline_codes.id )
38 WHERE routes.destination_airport_code = '{{airport_code}}'
39     AND routes.source_airport_code IS NOT NULL
40     AND routes.doc_type = 'route'
41     AND routes.active = true
42 ) AS results
43 WHERE results.route.distance <= {{{max_distance}}}
44 ORDER BY results.route.distance ASC

```

Airports Routes within a given distance in Miles Query

For our example we want to find any routes within 300 miles of "MCI". Our `{{distance_unit}}` is miles, this value needs to be `69` and our `{{max_distance}}` is `300`.

[routeswithin300miles_of_MCI.n1ql](#)

```

1  SELECT results.route.airline, results.route.source_airport,
2      ROUND( results.route.distance, 2 ) AS distance
3  FROM (
4      SELECT
5          {
6              "airline": {
7                  "airline_code": IFNULL( airlines.airline_iata, airlines.airline_icao ),
8                  "airline_name": airlines.airline_name
9              },
10             "source_airport": {
11                 "airport_name": source_airports.airport_name,
12                 "iso_country": source_airports.iso_country,
13                 "iso_region": source_airports.iso_region,
14                 "airport_code": IFNULL(
15                     source_airports.airport_iata,
16                     source_airports.airport_icao,
17                     source_airports.airport_ident )
18             },
19             "distance": 69 * DEGREES(ACOS(COS(RADIANS( source_airports.geo.latitude ))
20             * COS(RADIANS( destination_airports.geo.latitude ))
21             * COS(RADIANS( source_airports.geo.longitude ) - RADIANS( destination_airpor
22             + SIN(RADIANS( source_airports.geo.latitude ))
23             * SIN(RADIANS( destination_airports.geo.latitude ))))
24         } AS route
25  FROM `flight-data` AS routes
26  INNER JOIN `flight-data` AS source_airport_codes
27      ON KEYS 'airport_code_' || routes.source_airport_code
28  INNER JOIN `flight-data` AS source_airports
29      ON KEYS 'airport_' || TOSTRING( source_airport_codes.id )
30  INNER JOIN `flight-data` AS destination_airport_codes
31      ON KEYS 'airport_code_' || routes.destination_airport_code
32  INNER JOIN `flight-data` AS destination_airports
33      ON KEYS 'airport_' || TOSTRING( destination_airport_codes.id )
34  INNER JOIN `flight-data` AS airline_codes
35      ON KEYS 'airline_code_' || routes.airline_code
36  INNER JOIN `flight-data` AS airlines
37      ON KEYS 'airline_' || TOSTRING( airline_codes.id )
38  WHERE routes.destination_airport_code = 'MCI'
39      AND routes.source_airport_code IS NOT NULL
40      AND routes.doc_type = 'route'
41      AND routes.active = true
42  ) AS results
43  WHERE results.route.distance <= 300
44  ORDER BY results.route.distance ASC

```


Airports Routes within a given distance in Miles Results

```
1  [
2    {
3      "airline": {
4        "airline_code": "K5",
5        "airline_name": "SeaPort Airlines"
6      },
7      "destination_airport": {
8        "airport_code": "SLN",
9        "airport_name": "Salina Municipal Airport",
10       "iso_country": "US",
11       "iso_region": "US-KS"
12     },
13     "distance": 161.29
14   },
15   {
16     "airline": {
17       "airline_code": "K5",
18       "airline_name": "SeaPort Airlines"
19     },
20     "destination_airport": {
21       "airport_code": "HRO",
22       "airport_name": "Boone Co",
23       "iso_country": "US",
24       "iso_region": "US-AR"
25     },
26     "distance": 226.08
27   },
28   {
29     "airline": {
30       "airline_code": "FL",
31       "airline_name": "AirTran Airways"
32     },
33     "destination_airport": {
34       "airport_code": "STL",
35       "airport_name": "Lambert St Louis Intl",
36       "iso_country": "US",
37       "iso_region": "US-MO"
38     },
39     "distance": 235.89
40   },
41   {
42     "airline": {
43       "airline_code": "WN",
44       "airline_name": "Southwest Airlines"
```

```
45     },
46     "destination_airport": {
47         "airport_code": "STL",
48         "airport_name": "Lambert St Louis Intl",
49         "iso_country": "US",
50         "iso_region": "US-MO"
51     },
52     "distance": 235.89
53 }
54 ]
```

Airports Routes within a given distance in Kilometers Query

For our example we want to find any routes within 300 kilometers of "CDG". Our `{{distance_unit}}` is kilometers, this value needs to be `111.045` and our `{{max_distance}}` is `300`.

[routeswithin300kilometersof_CDG.n1ql](#)

```

1  SELECT results.route.airline, results.route.source_airport,
2      ROUND( results.route.distance, 2 ) AS distance
3  FROM (
4      SELECT
5          {
6              "airline": {
7                  "airline_code": IFNULL( airlines.airline_iata, airlines.airline_icao ),
8                  "airline_name": airlines.airline_name
9              },
10             "source_airport": {
11                 "airport_name": source_airports.airport_name,
12                 "iso_country": source_airports.iso_country,
13                 "iso_region": source_airports.iso_region,
14                 "airport_code": IFNULL(
15                     source_airports.airport_iata,
16                     source_airports.airport_icao,
17                     source_airports.airport_ident )
18             },
19             "distance": 111.045 * DEGREES(ACOS(COS(RADIANS( source_airports.geo.latitude
20                 * COS(RADIANS( destination_airports.geo.latitude ))
21                 * COS(RADIANS( source_airports.geo.longitude ) - RADIANS( destination_airpor
22                 + SIN(RADIANS( source_airports.geo.latitude ))
23                 * SIN(RADIANS( destination_airports.geo.latitude )))))
24         } AS route
25  FROM `flight-data` AS routes
26  INNER JOIN `flight-data` AS source_airport_codes
27      ON KEYS 'airport_code_' || routes.source_airport_code
28  INNER JOIN `flight-data` AS source_airports
29      ON KEYS 'airport_' || TOSTRING( source_airport_codes.id )
30  INNER JOIN `flight-data` AS destination_airport_codes
31      ON KEYS 'airport_code_' || routes.destination_airport_code
32  INNER JOIN `flight-data` AS destination_airports
33      ON KEYS 'airport_' || TOSTRING( destination_airport_codes.id )
34  INNER JOIN `flight-data` AS airline_codes
35      ON KEYS 'airline_code_' || routes.airline_code
36  INNER JOIN `flight-data` AS airlines
37      ON KEYS 'airline_' || TOSTRING( airline_codes.id )
38  WHERE routes.destination_airport_code = 'CDG'
39      AND routes.source_airport_code IS NOT NULL
40      AND routes.doc_type = 'route'
41      AND routes.active = true
42  ) AS results
43  WHERE results.route.distance <= 300
44  ORDER BY results.route.distance ASC

```

Airports Routes within a given distance in Kilometers Results

```
1  [
2    {
3      "airline": {
4        "airline_code": "SN",
5        "airline_name": "Brussels Airlines"
6      },
7      "distance": 251.14,
8      "source_airport": {
9        "airport_code": "BRU",
10       "airport_name": "Brussels Natl",
11       "iso_country": "BE",
12       "iso_region": "BE-BRU"
13     }
14   },
15   {
16     "airline": {
17       "airline_code": "ET",
18       "airline_name": "Ethiopian Airlines"
19     },
20     "distance": 251.14,
21     "source_airport": {
22       "airport_code": "BRU",
23       "airport_name": "Brussels Natl",
24       "iso_country": "BE",
25       "iso_region": "BE-BRU"
26     }
27   },
28   {
29     "airline": {
30       "airline_code": "AF",
31       "airline_name": "Air France"
32     },
33     "distance": 273.63,
34     "source_airport": {
35       "airport_code": "LUX",
36       "airport_name": "Luxembourg",
37       "iso_country": "LU",
38       "iso_region": "LU-L"
39     }
40   },
41   {
42     "airline": {
43       "airline_code": "LG",
44       "airline_name": "Luxair"
```

```
45     },
46     "distance": 273.63,
47     "source_airport": {
48         "airport_code": "LUX",
49         "airport_name": "Luxembourg",
50         "iso_country": "LU",
51         "iso_region": "LU-L"
52     }
53 }
54 ]
```