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

idx*routes*source*airport*codes.n1ql

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

Query

routes*originating*from_ICT.n1ql

```
SELECT
    {
        "airline": {
            "airline_code": IFNULL(
                airlines.airline_iata,
                airlines.airline_icao
            ),
            "airline_name": airlines.airline_name
        },
        "destination_airport": {
            "airport_name": destination_airports.airport_name,
            "iso_country": destination_airports.iso_country,
            "iso_region": destination_airports.iso_region,
            "airport_code": IFNULL(
                destination_airports.airport_iata,
                destination_airports.airport_icao,
                destination_airports.airport_ident
            )
        }
    } AS route
FROM `flight-data` AS routes
INNER JOIN `flight-data` AS destination_codes
    ON KEYS 'airport_code_' || routes.destination_airport_code
INNER JOIN `flight-data` AS destination_airports
    ON KEYS 'airport_' || TOSTRING( destination_codes.id )
INNER JOIN `flight-data` AS airline_codes
    ON KEYS 'airline_code_' || routes.airline_code
INNER JOIN `flight-data` AS airlines
    ON KEYS 'airline_' || TOSTRING( airline_codes.id )
WHERE routes.source_airport_code = 'ICT'
    AND routes.destination_airport_code IS NOT NULL
    AND routes.doc_type = 'route'
    AND routes.active = true
ORDER BY destination_airports.name ASC
```

Results

```
[
  {
    "route": {
      "airline": {
        "airline_code": "FL",
        "airline_name": "AirTran Airways"
      },
      "destination_airport": {
        "airport_code": "MDW",
        "airport_name": "Chicago Midway Intl",
        "iso_country": "US",
        "iso_region": "US-IL"
      }
    }
  },
  {
    "route": {
      "airline": {
        "airline_code": "WN",
        "airline_name": "Southwest Airlines"
      },
      "destination_airport": {
        "airport_code": "MDW",
        "airport_name": "Chicago Midway Intl",
        "iso_country": "US",
        "iso_region": "US-IL"
      }
    }
  },
  ...
]
```

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

Query

totalroutesoriginatingfromICT.n1ql

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

Result

```
[
  {
    "airports": 915
  }
]
```

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.

idx*routes*source*airport*codes_drop.n1ql

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

idx*routes*source*airport*codes_AtoM.n1ql

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

idx*routes*source*airport*codes_NtoZ.n1ql

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

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

total*routes*originating*from*ATL_explain.n1ql

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

```
[
  {
    "plan": {
      "#operator": "Sequence",
      "~children": [
        {
          "#operator": "IndexScan",
          "index": "idx_routes_source_airport_codes_AtoM",
          "index_id": "31d1f9b502d8dabd",
          "keyspace": "flight-data",
          "namespace": "default",
```

total*routes*originating*from*SFO_explain.n1ql

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

```
[
  {
    "plan": {
      "#operator": "Sequence",
      "~children": [
        {
          "#operator": "IndexScan",
          "index": "idx_routes_source_airport_codes_NtoZ",
          "index_id": "cd7e02bdb0ccd1d4",
          "keyspace": "flight-data",
          "namespace": "default",
```

Airlines flying from Airport

Query

airlines*flying*from_airport.n1ql

```
SELECT results.airline_code, results.airline_name, COUNT(1) AS routes
FROM (
    SELECT IFNULL( airlines.airline_iata, airlines.airline_icao ) AS airline_code,
        airlines.airline_name
    FROM `flight-data` AS routes
    INNER JOIN `flight-data` AS airline_codes
        ON KEYS 'airline_code_' || routes.airline_code
    INNER JOIN `flight-data` AS airlines
        ON KEYS 'airline_' || TOSTRING( airline_codes.id )
    WHERE routes.source_airport_code = 'MRY'
        AND routes.destination_airport_code IS NOT NULL
        AND routes.doc_type = 'route'
        AND routes.active = true
) AS results
GROUP BY results.airline_code, results.airline_name
ORDER BY results.airline_code ASC
```

Results

```
[
  {
    "airline_code": "AA",
    "airline_name": "American Airlines",
    "routes": 2
  },
  {
    "airline_code": "AS",
    "airline_name": "Alaska Airlines",
    "routes": 2
  },
  {
    "airline_code": "G4",
    "airline_name": "Allegiant Air",
    "routes": 1
  },
  {
    "airline_code": "UA",
    "airline_name": "United Airlines",
    "routes": 3
  },
  {
    "airline_code": "US",
    "airline_name": "US Airways",
    "routes": 2
  }
]
```


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

Query

airlines*flying*from*airport*with*2plus*routes.n1ql

```
SELECT results.airline_code, results.airline_name, COUNT(1) AS routes
FROM (
    SELECT IFNULL( airlines.airline_iata, airlines.airline_icao ) AS airline_code,
        airlines.airline_name
    FROM `flight-data` AS routes
    INNER JOIN `flight-data` AS airline_codes
        ON KEYS 'airline_code_' || routes.airline_code
    INNER JOIN `flight-data` AS airlines
        ON KEYS 'airline_' || TOSTRING( airline_codes.id )
    WHERE routes.source_airport_code = 'MRY'
        AND routes.destination_airport_code IS NOT NULL
        AND routes.doc_type = 'route'
        AND routes.active = true
) AS results
GROUP BY results.airline_code, results.airline_name
HAVING COUNT(1) > 1
ORDER BY COUNT(1) DESC, results.airline_code ASC
```

Results


```
[
  {
    "airline_code": "UA",
    "airline_name": "United Airlines",
    "routes": 3
  },
  {
    "airline_code": "AA",
    "airline_name": "American Airlines",
    "routes": 2
  },
  {
    "airline_code": "AS",
    "airline_name": "Alaska Airlines",
    "routes": 2
  },
  {
    "airline_code": "US",
    "airline_name": "US Airways",
    "routes": 2
  }
]
```

Routes by Destination Airport

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

Index

idx*routes*destination*airport*codes_AtoM.n1ql

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

idx*routes*destination*airport*codes_NtoZ.n1ql

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

Query

routes*to*airport.n1ql

```
SELECT airlines.airline_name,
    IFNULL(
        airlines.airline_iata,
        airlines.airline_icao
    ) AS airline_code,
    source_airports.airport_name,
    source_airports.iso_country,
    source_airports.iso_region,
    IFNULL(
        source_airports.airport_iata,
        source_airports.airport_icao,
        source_airports.airport_ident
    ) AS airport_code
FROM `flight-data` AS routes
INNER JOIN `flight-data` AS airport_codes
    ON KEYS 'airport_code_' || routes.source_airport_code
INNER JOIN `flight-data` AS source_airports
    ON KEYS 'airport_' || TOSTRING( airport_codes.id )
INNER JOIN `flight-data` AS airline_codes
    ON KEYS 'airline_code_' || routes.airline_code
INNER JOIN `flight-data` AS airlines
    ON KEYS 'airline_' || TOSTRING( airline_codes.id )
WHERE routes.destination_airport_code = 'MRY'
    AND routes.source_airport_code IS NOT NULL
    AND routes.doc_type = 'route'
    AND routes.active = true
ORDER BY source_airports.airport_name ASC
```

Results

```
[
  {
    "airline_code": "UA",
    "airline_name": "United Airlines",
    "airport_code": "DEN",
    "airport_name": "Denver Intl",
    "iso_country": "US",
    "iso_region": "US-CO"
  },
  {
    "airline_code": "UA",
    "airline_name": "United Airlines",
    "airport_code": "LAX",
    "airport_name": "Los Angeles Intl",
    "iso_country": "US",
    "iso_region": "US-CA"
  },
  ...
]
```


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

Query

routes*to*airport_formatted.n1ql

```
SELECT
    {
      "airline": {
          "airline_code": IFNULL(
              airlines.airline_iata,
              airlines.airline_icao
          ),
          "airline_name": airlines.airline_name
      },
      "source_airport": {
          "airport_name": source_airports.airport_name,
          "iso_country": source_airports.iso_country,
          "iso_region": source_airports.iso_region,
          "airport_code": IFNULL(
              source_airports.airport_iata,
              source_airports.airport_icao,
              source_airports.airport_ident
          )
      }
    } AS route
FROM `flight-data` AS routes
INNER JOIN `flight-data` AS airport_codes
    ON KEYS 'airport_code_' || routes.source_airport_code
INNER JOIN `flight-data` AS source_airports
    ON KEYS 'airport_' || TOSTRING( airport_codes.id )
INNER JOIN `flight-data` AS airline_codes
    ON KEYS 'airline_code_' || routes.airline_code
INNER JOIN `flight-data` AS airlines
    ON KEYS 'airline_' || TOSTRING( airline_codes.id )
WHERE routes.destination_airport_code = 'MRY'
    AND routes.source_airport_code IS NOT NULL
    AND routes.doc_type = 'route'
    AND routes.active = true
ORDER BY source_airports.airport_name ASC
```

Results

```
[
  {
    "route": {
      "airline": {
        "airline_code": "UA",
        "airline_name": "United Airlines"
      },
      "source_airport": {
        "airport_code": "DEN",
        "airport_name": "Denver Intl",
        "iso_country": "US",
        "iso_region": "US-CO"
      }
    }
  },
  {
    "route": {
      "airline": {
        "airline_code": "UA",
        "airline_name": "United Airlines"
      },
      "source_airport": {
        "airport_code": "LAX",
        "airport_name": "Los Angeles Intl",
        "iso_country": "US",
        "iso_region": "US-CA"
      }
    }
  },
  ...
]
```


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

```
SELECT
    {
      "airline": {
          "airline_code": IFNULL(
              airlines.airline_iata,
              airlines.airline_icao
          ),
          "airline_name": airlines.airline_name
      },
      "source_airport": {
          "airport_name": source_airports.airport_name,
          "iso_country": source_airports.iso_country,
          "iso_region": source_airports.iso_region,
          "airport_code": IFNULL(
              source_airports.airport_iata,
              source_airports.airport_icao,
              source_airports.airport_ident
          )
      }
    }.*
FROM `flight-data` AS routes
INNER JOIN `flight-data` AS airport_codes
    ON KEYS 'airport_code_' || routes.source_airport_code
INNER JOIN `flight-data` AS source_airports
    ON KEYS 'airport_' || TOSTRING( airport_codes.id )
INNER JOIN `flight-data` AS airline_codes
    ON KEYS 'airline_code_' || routes.airline_code
INNER JOIN `flight-data` AS airlines
    ON KEYS 'airline_' || TOSTRING( airline_codes.id )
WHERE routes.destination_airport_code = 'MRY'
    AND routes.source_airport_code IS NOT NULL
    AND routes.doc_type = 'route'
    AND routes.active = true
ORDER BY source_airports.airport_name ASC
```

Results

```
[
  {
    "airline": {
      "airline_code": "UA",
      "airline_name": "United Airlines"
    },
    "source_airport": {
      "airport_code": "DEN",
      "airport_name": "Denver Intl",
      "iso_country": "US",
      "iso_region": "US-CO"
    }
  },
  {
    "airline": {
      "airline_code": "UA",
      "airline_name": "United Airlines"
    },
    "source_airport": {
      "airport_code": "LAX",
      "airport_name": "Los Angeles Intl",
      "iso_country": "US",
      "iso_region": "US-CA"
    }
  },
  ...
]
```


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

```
SELECT
    {
      "airline": {
          "airline_code": IFNULL(
              airlines.airline_iata,
              airlines.airline_icao
          ),
          "airline_name": airlines.airline_name
      },
      "source_airport": {
          "airport_name": source_airports.airport_name,
          "iso_country": source_airports.iso_country,
          "iso_region": source_airports.iso_region,
          "airport_code": IFNULL(
              source_airports.airport_iata,
              source_airports.airport_icao,
              source_airports.airport_ident
          )
      },
      "distance": ROUND(69 * DEGREES(ACOS(COS(RADIANS( source_airports.geo.latitude ))
      * COS(RADIANS( destination_airports.geo.latitude ))
      * COS(RADIANS( source_airports.geo.longitude ) - RADIANS( destination_airports.geo
      + SIN(RADIANS( source_airports.geo.latitude ))
      * SIN(RADIANS( destination_airports.geo.latitude )))), 2)
    }.*
FROM `flight-data` AS routes
INNER JOIN `flight-data` AS airport_codes
    ON KEYS 'airport_code_' || routes.source_airport_code
INNER JOIN `flight-data` AS source_airports
    ON KEYS 'airport_' || TOSTRING( airport_codes.id )
INNER JOIN `flight-data` AS destination_airport_codes
    ON KEYS 'airport_code_' || routes.destination_airport_code
INNER JOIN `flight-data` AS destination_airports
    ON KEYS 'airport_' || TOSTRING(  destination_airport_codes.id  )
INNER JOIN `flight-data` AS airline_codes
    ON KEYS 'airline_code_' || routes.airline_code
INNER JOIN `flight-data` AS airlines
    ON KEYS 'airline_' || TOSTRING( airline_codes.id )
WHERE routes.destination_airport_code = 'MRY'
    AND routes.source_airport_code IS NOT NULL
    AND routes.doc_type = 'route'
    AND routes.active = true
ORDER BY source_airports.airport_name ASC
```

Results

```
[
  {
    "airline": {
      "airline_code": "UA",
      "airline_name": "United Airlines"
    },
    "distance": 956.1,
    "source_airport": {
      "airport_code": "DEN",
      "airport_name": "Denver Intl",
      "iso_country": "US",
      "iso_region": "US-CO"
    }
  },
  {
    "airline": {
      "airline_code": "UA",
      "airline_name": "United Airlines"
    },
    "distance": 265.94,
    "source_airport": {
      "airport_code": "LAX",
      "airport_name": "Los Angeles Intl",
      "iso_country": "US",
      "iso_region": "US-CA"
    }
  },
  ...
]
```


Airlines flying into Airport

Query

total*airlines*flying*to*destination.n1ql

```
SELECT results.airline_code, results.airline_name, COUNT(1) AS routes
FROM (
    SELECT IFNULL( airlines.airline_iata, airlines.airline_icao ) AS airline_code,
        airlines.airline_name
    FROM `flight-data` AS routes
    INNER JOIN `flight-data` AS airline_codes
        ON KEYS 'airline_code_' || routes.airline_code
    INNER JOIN `flight-data` AS airlines
        ON KEYS 'airline_' || TOSTRING( airline_codes.id )
    WHERE routes.destination_airport_code = 'GSO'
        AND routes.source_airport_code IS NOT NULL
        AND routes.doc_type = 'route'
        AND routes.active = true
) AS results
GROUP BY results.airline_code, results.airline_name
ORDER BY results.airline_code ASC
```

Results


```
[
  {
    "airline_code": "9E",
    "airline_name": "Pinnacle Airlines",
    "routes": 1
  },
  {
    "airline_code": "AA",
    "airline_name": "American Airlines",
    "routes": 6
  },
  {
    "airline_code": "AF",
    "airline_name": "Air France",
    "routes": 1
  },
  {
    "airline_code": "AS",
    "airline_name": "Alaska Airlines",
    "routes": 1
  },
  {
    "airline_code": "AZ",
    "airline_name": "Alitalia",
    "routes": 1
  },
  {
```

```
    "airline_code": "DL",
    "airline_name": "Delta Air Lines",
    "routes": 3
  },
  {
    "airline_code": "F9",
    "airline_name": "Frontier Airlines",
    "routes": 1
  },
  {
    "airline_code": "G4",
    "airline_name": "Allegiant Air",
    "routes": 2
  },
  {
    "airline_code": "KL",
    "airline_name": "KLM Royal Dutch Airlines",
    "routes": 1
  },
  {
    "airline_code": "UA",
    "airline_name": "United Airlines",
    "routes": 3
  },
  {
    "airline_code": "US",
    "airline_name": "US Airways",
    "routes": 6
  }
]
```


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

Query

airlines*flying*to*destination*with*3plus*routes.n1ql

```
SELECT results.airline_code, results.airline_name, COUNT(1) AS routes
FROM (
    SELECT IFNULL( airlines.airline_iata, airlines.airline_icao ) AS airline_code,
        airlines.airline_name
    FROM `flight-data` AS routes
    INNER JOIN `flight-data` AS airline_codes
        ON KEYS 'airline_code_' || routes.airline_code
    INNER JOIN `flight-data` AS airlines
        ON KEYS 'airline_' || TOSTRING( airline_codes.id )
    WHERE routes.destination_airport_code = 'GSO'
        AND routes.source_airport_code IS NOT NULL
        AND routes.doc_type = 'route'
        AND routes.active = true
) AS results
GROUP BY results.airline_code, results.airline_name
HAVING COUNT(1) > 2
ORDER BY COUNT(1) DESC, results.airline_code ASC
```

Results


```
[
  {
    "airline_code": "AA",
    "airline_name": "American Airlines",
    "routes": 6
  },
  {
    "airline_code": "US",
    "airline_name": "US Airways",
    "routes": 6
  },
  {
    "airline_code": "DL",
    "airline_name": "Delta Air Lines",
    "routes": 3
  },
  {
    "airline_code": "UA",
    "airline_name": "United Airlines",
    "routes": 3
  }
]
```

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

```
SELECT results.route.airline, results.route.source_airport,
    ROUND( results.route.distance, 2 ) AS distance
FROM (
    SELECT
        {
            "airline": {
                "airline_code": IFNULL( airlines.airline_iata, airlines.airline_icao ),
                "airline_name": airlines.airline_name
            },
            "source_airport": {
                "airport_name": source_airports.airport_name,
                "iso_country": source_airports.iso_country,
                "iso_region": source_airports.iso_region,
                "airport_code": IFNULL(
                    source_airports.airport_iata,
                    source_airports.airport_icao,
                    source_airports.airport_ident )
            },
            "distance": {{distance_unit}} * DEGREES(ACOS(COS(RADIANS( source_airports.ge
            * COS(RADIANS( destination_airports.geo.latitude ))
            * COS(RADIANS( source_airports.geo.longitude ) - RADIANS( destination_airpor
            + SIN(RADIANS( source_airports.geo.latitude ))
            * SIN(RADIANS( destination_airports.geo.latitude ))))
        } AS route
    FROM `flight-data` AS routes
    INNER JOIN `flight-data` AS source_airport_codes
        ON KEYS 'airport_code_' || routes.source_airport_code
    INNER JOIN `flight-data` AS source_airports
        ON KEYS 'airport_' || TOSTRING(  source_airport_codes.id  )
    INNER JOIN `flight-data` AS destination_airport_codes
        ON KEYS 'airport_code_' || routes.destination_airport_code
    INNER JOIN `flight-data` AS destination_airports
        ON KEYS 'airport_' || TOSTRING(  destination_airport_codes.id  )
    INNER JOIN `flight-data` AS airline_codes
        ON KEYS 'airline_code_' || routes.airline_code
    INNER JOIN `flight-data` AS airlines
        ON KEYS 'airline_' || TOSTRING(  airline_codes.id  )
    WHERE routes.destination_airport_code = '{{airport_code}}'
        AND routes.source_airport_code IS NOT NULL
        AND routes.doc_type = 'route'
        AND routes.active = true
) AS results
WHERE results.route.distance <= {{max_distance}}
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`.

routes*within*300*miles*of_MCI.n1ql

```
SELECT results.route.airline, results.route.source_airport,
    ROUND( results.route.distance, 2 ) AS distance
FROM (
    SELECT
        {
            "airline": {
                "airline_code": IFNULL( airlines.airline_iata, airlines.airline_icao ),
                "airline_name": airlines.airline_name
            },
            "source_airport": {
                "airport_name": source_airports.airport_name,
                "iso_country": source_airports.iso_country,
                "iso_region": source_airports.iso_region,
                "airport_code": IFNULL(
                    source_airports.airport_iata,
                    source_airports.airport_icao,
                    source_airports.airport_ident )
            },
            "distance": 69 * DEGREES(ACOS(COS(RADIANS( source_airports.geo.latitude ))
            * COS(RADIANS( destination_airports.geo.latitude ))
            * COS(RADIANS( source_airports.geo.longitude ) - RADIANS( destination_airpor
            + SIN(RADIANS( source_airports.geo.latitude ))
            * SIN(RADIANS( destination_airports.geo.latitude ))))
        } AS route
    FROM `flight-data` AS routes
    INNER JOIN `flight-data` AS source_airport_codes
        ON KEYS 'airport_code_' || routes.source_airport_code
    INNER JOIN `flight-data` AS source_airports
        ON KEYS 'airport_' || TOSTRING( source_airport_codes.id )
    INNER JOIN `flight-data` AS destination_airport_codes
        ON KEYS 'airport_code_' || routes.destination_airport_code
    INNER JOIN `flight-data` AS destination_airports
        ON KEYS 'airport_' || TOSTRING( destination_airport_codes.id )
    INNER JOIN `flight-data` AS airline_codes
        ON KEYS 'airline_code_' || routes.airline_code
    INNER JOIN `flight-data` AS airlines
        ON KEYS 'airline_' || TOSTRING( airline_codes.id )
    WHERE routes.destination_airport_code = 'MCI'
        AND routes.source_airport_code IS NOT NULL
        AND routes.doc_type = 'route'
        AND routes.active = true
) AS results
WHERE results.route.distance <= 300
ORDER BY results.route.distance ASC
```

Airports Routes within a given distance in Miles Results


```
[
  {
    "airline": {
      "airline_code": "K5",
      "airline_name": "SeaPort Airlines"
    },
    "destination_airport": {
      "airport_code": "SLN",
      "airport_name": "Salina Municipal Airport",
      "iso_country": "US",
      "iso_region": "US-KS"
    },
    "distance": 161.29
  },
  {
    "airline": {
      "airline_code": "K5",
      "airline_name": "SeaPort Airlines"
    },
    "destination_airport": {
      "airport_code": "HRO",
      "airport_name": "Boone Co",
      "iso_country": "US",
      "iso_region": "US-AR"
    },
    "distance": 226.08
  },
  {
    "airline": {
      "airline_code": "FL",
      "airline_name": "AirTran Airways"
    },
    "destination_airport": {
      "airport_code": "STL",
      "airport_name": "Lambert St Louis Intl",
      "iso_country": "US",
      "iso_region": "US-MO"
    },
    "distance": 235.89
  },
  {
    "airline": {
      "airline_code": "WN",
      "airline_name": "Southwest Airlines"
```

```
    },
    "destination_airport": {
      "airport_code": "STL",
      "airport_name": "Lambert St Louis Intl",
      "iso_country": "US",
      "iso_region": "US-MO"
    },
    "distance": 235.89
  }
]
```


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`.

routes*within*300*kilometers*of_CDG.n1ql

```
SELECT results.route.airline, results.route.source_airport,
    ROUND( results.route.distance, 2 ) AS distance
FROM (
    SELECT
        {
            "airline": {
                "airline_code": IFNULL( airlines.airline_iata, airlines.airline_icao ),
                "airline_name": airlines.airline_name
            },
            "source_airport": {
                "airport_name": source_airports.airport_name,
                "iso_country": source_airports.iso_country,
                "iso_region": source_airports.iso_region,
                "airport_code": IFNULL(
                    source_airports.airport_iata,
                    source_airports.airport_icao,
                    source_airports.airport_ident )
            },
            "distance": 111.045 * DEGREES(ACOS(COS(RADIANS( source_airports.geo.latitude
            * COS(RADIANS( destination_airports.geo.latitude ))
            * COS(RADIANS( source_airports.geo.longitude ) - RADIANS( destination_airpor
            + SIN(RADIANS( source_airports.geo.latitude ))
            * SIN(RADIANS( destination_airports.geo.latitude ))))
        } AS route
    FROM `flight-data` AS routes
    INNER JOIN `flight-data` AS source_airport_codes
        ON KEYS 'airport_code_' || routes.source_airport_code
    INNER JOIN `flight-data` AS source_airports
        ON KEYS 'airport_' || TOSTRING(  source_airport_codes.id  )
    INNER JOIN `flight-data` AS destination_airport_codes
        ON KEYS 'airport_code_' || routes.destination_airport_code
    INNER JOIN `flight-data` AS destination_airports
        ON KEYS 'airport_' || TOSTRING(  destination_airport_codes.id  )
    INNER JOIN `flight-data` AS airline_codes
        ON KEYS 'airline_code_' || routes.airline_code
    INNER JOIN `flight-data` AS airlines
        ON KEYS 'airline_' || TOSTRING(  airline_codes.id  )
    WHERE routes.destination_airport_code = 'CDG'
        AND routes.source_airport_code IS NOT NULL
        AND routes.doc_type = 'route'
        AND routes.active = true
) AS results
WHERE results.route.distance <= 300
ORDER BY results.route.distance ASC
```

Airports Routes within a given distance in Kilometers Results


```
[
  {
    "airline": {
      "airline_code": "SN",
      "airline_name": "Brussels Airlines"
    },
    "distance": 251.14,
    "source_airport": {
      "airport_code": "BRU",
      "airport_name": "Brussels Natl",
      "iso_country": "BE",
      "iso_region": "BE-BRU"
    }
  },
  {
    "airline": {
      "airline_code": "ET",
      "airline_name": "Ethiopian Airlines"
    },
    "distance": 251.14,
    "source_airport": {
      "airport_code": "BRU",
      "airport_name": "Brussels Natl",
      "iso_country": "BE",
      "iso_region": "BE-BRU"
    }
  },
  {
    "airline": {
      "airline_code": "AF",
      "airline_name": "Air France"
    },
    "distance": 273.63,
    "source_airport": {
      "airport_code": "LUX",
      "airport_name": "Luxembourg",
      "iso_country": "LU",
      "iso_region": "LU-L"
    }
  },
  {
    "airline": {
      "airline_code": "LG",
      "airline_name": "Luxair"
```

```
    },
    "distance": 273.63,
    "source_airport": {
      "airport_code": "LUX",
      "airport_name": "Luxembourg",
      "iso_country": "LU",
      "iso_region": "LU-L"
    }
  }
]
```