Airport Queries

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

Airport By ID

The following query will get an Airline by its Document ID.

Query

airport_by_document_id.n1ql

```
SELECT airports.*
FROM `flight-data` AS airports
USE KEYS 'airport_3605'
```

Result

```
[
  {
    "_id": "airport_3605",
    "airport_gps_code": "KICT",
    "airport_iata": "ICT",
    "airport_icao": "KICT",
    "airport_id": 3605,
    "airport_ident": "KICT",
    "airport_local_code": "ICT",
    "airport_name": "Wichita Dwight D. Eisenhower National Airport",
    "airport_type": "large_airport",
    "doc_type": "airport",
    "dst": "A",
    "elevation": 1333,
    "geo": {
      "latitude": 37.64989853,
      "longitude": -97.43309784
    },
    "iso_continent": "NA",
    "iso_country": "US",
    "iso_region": "US-KS",
    "municipality": "Wichita",
    "timezone": "America/Chicago",
    "timezone_offset": -6
  }
]
```


The following query will retrieve many Airlines by their ID.

Query

airports_by_document_id.n1ql

```
SELECT airports.*
FROM `flight-data` AS airports
USE KEYS ['airport_3605', 'airport_3568']
```

Result


```
[
  {
    "_id": "airport_3605",
    "airport_gps_code": "KICT",
    "airport_iata": "ICT",
    "airport_icao": "KICT",
    "airport_id": 3605,
```

```
    "airport_ident": "KICT",
    "airport_local_code": "ICT",
    "airport_name": "Wichita Dwight D. Eisenhower National Airport",
    "airport_type": "large_airport",
    "doc_type": "airport",
    "dst": "A",
    "elevation": 1333,
    "geo": {
      "latitude": 37.64989853,
      "longitude": -97.43309784
    },
    "iso_continent": "NA",
    "iso_country": "US",
    "iso_region": "US-KS",
    "municipality": "Wichita",
    "timezone": "America/Chicago",
    "timezone_offset": -6
  },
  {
    "_id": "airport_3568",
    "airport_gps_code": "KGSO",
    "airport_iata": "GSO",
    "airport_icao": "KGSO",
    "airport_id": 3568,
    "airport_ident": "KGSO",
    "airport_local_code": "GSO",
    "airport_name": "Piedmont Triad",
    "airport_type": "large_airport",
    "doc_type": "airport",
    "dst": "A",
    "elevation": 925,
    "geo": {
      "latitude": 36.09780121,
      "longitude": -79.93730164
    },
    "iso_continent": "NA",
    "iso_country": "US",
    "iso_region": "US-NC",
    "municipality": "Greensboro",
    "timezone": "America/New_York",
    "timezone_offset": -5
  }
]
```

Airports in a Country

The following index and queries allows for finding airports based in a given country by creating an index on the `iso_country` where the `doc_type` is `airport`

Index

idx_airports_iso_country.n1ql

```
CREATE INDEX idx_airports_iso_country ON `flight-data`( iso_country )
WHERE doc_type = 'airport'
    AND iso_country IS NOT NULL
USING GSI
```

Query

airport_by_country.n1ql

```
SELECT airports.*
FROM `flight-data` AS airports
WHERE airports.iso_country = 'FI'
    AND airports.doc_type = 'airport'
LIMIT 1
```

Result

```
[
  {
    "_id": "airport_2330",
    "airport_gps_code": "EFMA",
    "airport_iata": "MHQ",
    "airport_icao": "EFMA",
    "airport_id": 2330,
    "airport_ident": "EFMA",
    "airport_local_code": null,
    "airport_name": "Mariehamn",
    "airport_type": "medium_airport",
    "doc_type": "airport",
    "dst": "E",
    "elevation": 17,
    "geo": {
      "latitude": 60.12220001,
      "longitude": 19.89819908
    },
    "iso_continent": "EU",
    "iso_country": "FI",
    "iso_region": "FI-AL",
    "municipality": "Mariehamn",
    "timezone": "Europe/Mariehamn",
    "timezone_offset": 2
  }
]
```


Now that we know we can retrieve an airport in a country, lets retrieve all airports in a given country by querying on the `iso_country`, sorted by the `airport_name`

Query

airports*by*country.n1ql

```
SELECT airports.airport_id, airports.airport_name, airports.airport_type,
    airports.iso_region, airports.municipality,
    IFNULL( airports.airport_iata, airports.airport_icao, airports.airport_ident ) AS ai
FROM `flight-data` AS airports
WHERE airports.iso_country = 'AE'
    AND airports.doc_type = 'airport'
ORDER BY airports.airport_name ASC
```

Results

```
[
  {
    "airport_code": "AUH",
    "airport_id": 5226,
    "airport_name": "Abu Dhabi Intl",
    "airport_type": "large_airport",
    "iso_region": "AE-AZ",
    "municipality": "Abu Dhabi"
  },
  {
    "airport_code": "AAN",
    "airport_id": 5230,
    "airport_name": "Al Ain International Airport",
    "airport_type": "medium_airport",
    "iso_region": "AE-AZ",
    "municipality": "Al Ain"
  },
  {
    "airport_code": "DHF",
    "airport_id": 5231,
    "airport_name": "Al Dhafra",
    "airport_type": "medium_airport",
    "iso_region": "AE-AZ",
    "municipality": "Abu Dhabi"
  },
  ...
]
```


Additionally we can retrieve an aggregate count of the number of airports in a given country.

Query

total*airports*in_country.n1ql

```
SELECT COUNT(1) AS total_airports
FROM `flight-data` AS airports
WHERE airports.iso_country = 'FI'
    AND airports.doc_type = 'airport'
```

Result

```
[
  {
    "total_airports": 50
  }
]
```

Airports by Country and Region

Now that we know we can retrieve all airports in a given country by querying on the `iso_country` we can add the `iso_region` to further narrow the results.

Index

First we need to delete our previously created index and then create a new one. The new index will index both `iso_country` and `iso_region`

idx_airports_iso_country_drop.n1ql

```
DROP INDEX `flight-data`.idx_airports_iso_country
```

idx_airports_iso_country_region.n1ql

```
CREATE INDEX idx_airports_iso_country_region ON `flight-data`( iso_country, iso_region )
WHERE doc_type = 'airport'
    AND iso_country IS NOT NULL
    AND iso_region IS NOT NULL
USING GSI
```

Query

airports_by_country_region.n1ql

```
SELECT airports.airport_id, airports.airport_name, airports.airport_type,
    airports.iso_region, airports.municipality,
    IFNULL( airports.airport_iata, airports.airport_icao, airports.airport_ident ) AS ai
FROM `flight-data` AS airports
WHERE airports.iso_country = 'US'
    AND airports.iso_region = 'US-VT'
    AND airports.doc_type = 'airport'
ORDER BY airports.airport_name ASC
```

Results

```
[
  {
    "airport_code": "BTV",
    "airport_id": 3430,
    "airport_name": "Burlington Intl",
    "airport_type": "medium_airport",
    "iso_region": "US-VT",
    "municipality": "Burlington"
  },
  {
    "airport_code": "MPV",
    "airport_id": 20551,
    "airport_name": "Edward F Knapp State",
    "airport_type": "medium_airport",
    "iso_region": "US-VT",
    "municipality": "Montpelier"
  },
  {
    "airport_code": "VSF",
    "airport_id": 21336,
    "airport_name": "Hartness State",
    "airport_type": "small_airport",
    "iso_region": "US-VT",
    "municipality": "Springfield VT"
  },
  {
    "airport_code": "MVL",
    "airport_id": 20575,
    "airport_name": "Morrisville Stowe State Airport",
    "airport_type": "small_airport",
    "iso_region": "US-VT",
    "municipality": "Morrisville"
  },
  {
    "airport_code": "RUT",
    "airport_id": 3859,
    "airport_name": "Rutland State Airport",
    "airport_type": "medium_airport",
    "iso_region": "US-VT",
    "municipality": "Rutland"
  }
]
```


Additionally we can retrieve an aggregate count of the number of airports in a given country and region.

Query

total_airports_in_country_region.n1ql

```
SELECT COUNT(1) AS total_airports
FROM `flight-data` AS airports
WHERE airports.iso_country = 'US'
    AND airports.iso_region = 'US-VT'
    AND airports.doc_type = 'airport'
```

Result

```
[
  {
    "total_airports": 5
  }
]
```

Airport Codes

The following queries allows for finding airports by their IATA, ICAO or Ident Codes.

Just like Airlines, our Codes model is keyed by `{{designation}}_code_{{code}}` i.e. `airport_code_ICT`. Because of how these documents are keyed, we do not even need an index. Using this predictive key pattern we use the code as part of the key name on the codes document.

Query

Query by the IATA code

airport_by_iata_code.n1ql

```
SELECT airports.airport_id, airports.airport_name, airports.airport_type,
    airports.iso_region, airports.municipality,
    IFNULL( airports.airport_iata, airports.airport_icao, airports.airport_ident ) AS ai
FROM `flight-data` AS codes
USE KEYS 'airport_code_ICT'
INNER JOIN `flight-data` AS airports ON KEYS 'airport_' || TOSTRING( codes.id )
LIMIT 1
```

Query by the ICAO code

airport_by_icao_code.n1ql

```
SELECT airports.airport_id, airports.airport_name, airports.airport_type,
    airports.iso_region, airports.municipality,
    IFNULL( airports.airport_iata, airports.airport_icao, airports.airport_ident ) AS ai
FROM `flight-data` AS codes
USE KEYS 'airport_code_KICT'
INNER JOIN `flight-data` AS airports ON KEYS 'airport_' || TOSTRING( codes.id )
LIMIT 1
```

Both queries will yield the same exact result.

Result


```
[
  {
    "airport_code": "ICT",
    "airport_id": 3605,
    "airport_name": "Wichita Dwight D. Eisenhower National Airport",
    "airport_type": "large_airport",
    "iso_region": "US-KS",
    "municipality": "Wichita"
  }
]
```


Airports by City

Lets say we wanted to search on cities in an ISO country to find associated airport codes for use within an auto-complete function on our site.

Index

idx_airports_cities.n1ql

```
CREATE INDEX idx_airports_cities ON `flight-data`( iso_country, municipality )
WHERE doc_type = 'airport'
    AND iso_country IS NOT NULL
    AND municipality IS NOT NULL
USING GSI
```

Query

This query will find cites and their airport_code based on a partial match of the city name.

airports_by_city.n1ql

```
SELECT a.municipality AS city,
    IFNULL( a.airport_iata, a.airport_icao, a.airport_ident ) AS airport_code
FROM `flight-data` AS a
WHERE a.iso_country = 'US'
    AND a.municipality LIKE 'San%'
    AND a.doc_type = 'airport'
ORDER BY a.municipality ASC, a.airport_name ASC
LIMIT 5
```

Results


```
[
  {
    "airport_code": "SJT",
    "city": "San Angelo"
  },
  {
    "airport_code": "SKF",
    "city": "San Antonio"
  },
  {
    "airport_code": "RND",
    "city": "San Antonio"
  },
  {
    "airport_code": "SAT",
    "city": "San Antonio"
  },
  {
    "airport_code": "SBD",
    "city": "San Bernardino"
  }
]
```


We can get the total number of matches as well:

Query

total*airports*by_city.n1ql

```
SELECT COUNT(1) AS matches
FROM `flight-data` AS a
WHERE a.iso_country = 'US'
    AND a.municipality LIKE 'San%'
    AND a.doc_type = 'airport'
```

Results

```
[
  {
    "matches": 26
  }
]
```

We can leverage the `OFFSET` clause to paginate through the results.

Query

airports*by*city_offset.n1ql

```
SELECT a.municipality AS city,
    IFNULL( a.airport_iata, a.airport_icao, a.airport_ident ) AS airport_code
FROM `flight-data` AS a
WHERE a.iso_country = 'US'
    AND a.municipality LIKE 'San%'
    AND a.doc_type = 'airport'
ORDER BY a.municipality ASC, a.airport_name ASC
LIMIT 5 OFFSET 5
```

Results

```
[
  {
    "airport_code": "SQL",
    "city": "San Carlos"
  },
  {
    "airport_code": "KNUC",
    "city": "San Clemente Island"
  },
  {
    "airport_code": "SDM",
    "city": "San Diego"
  },
  {
    "airport_code": "MYF",
    "city": "San Diego"
  },
  {
    "airport_code": "NZY",
    "city": "San Diego"
  }
]
```

Airports Near a Given Airport

For this query we want to find all airports within a given radius of a given airport code.

Since we are going to be querying on the ISO Country, Latitude and Longitude of a given airport we need to create an index.

Index

idx*airports*distance.sql

```
CREATE INDEX idx_airports_distance ON `flight-data`( iso_country, geo.latitude, geo.long
WHERE doc_type = 'airport'
    AND iso_country IS NOT NULL
    AND geo.latitude IS NOT NULL
    AND geo.longitude IS NOT NULL
USING GSI
```

This query is based on a MySQL example provided by Ollie Jones.

To perform this query we need to provide 5 pieces of information to the query, these are represented in the query below as `{{tokens}}`

Input

- The Source Airports
  - `iso_country` i.e. US
  - `latitude` i.e. `36.09780121`
  - `longitude` i.e. `-79.93730164`
- A `distance_unit`
  - Kilometers: 111.045
  - Miles: 69
- A `radius` in which to contain results in, i.e. `100`

Radius Query

```
SELECT results.airport_name, results.airport_code, ROUND( results.distance, 2 ) AS dista
FROM (
    SELECT airports.airport_name,
        /* assign an airport_code */
        IFNULL( airports.airport_iata, airports.airport_icao, airports.airport_ident ) A
        /* calculate the distance */
        {{distance_unit}} * DEGREES(ACOS(COS(RADIANS( {{source_latitude}} ))
        * COS(RADIANS( airports.geo.latitude ))
        * COS(RADIANS( {{source_longitude}} ) - RADIANS( airports.geo.longitude ))
        + SIN(RADIANS( {{source_latitude}} ))
        * SIN(RADIANS( airports.geo.latitude )))) AS distance
    FROM `flight-data` AS airports
    WHERE airports.iso_country = '{{iso_country}}'
        /* limit results to latitudes within {{distance}} north or south of the source l
        AND airports.geo.latitude BETWEEN
            {{source_latitude}} - ( {{radius}} / {{distance_unit}} )
            AND
            {{source_latitude}} + ( {{radius}} / {{distance_unit}} )
        /* limit results to longitudes within {{distance}} east or west of the source lo
        AND airports.geo.longitude BETWEEN
            {{source_longitude}} - ( {{radius}} / ( {{distance_unit}} * COS(RADIANS( {{s
            AND
            {{source_longitude}} + ( {{radius}} / ( {{distance_unit}} * COS(RADIANS( {{s
        AND airports.doc_type = 'airport'
    ) AS results
WHERE results.distance > 0 /* remove the source from the results as its distance 0 */
    AND results.distance <= {{radius}} /* remove any of the results that are not within
ORDER BY results.distance ASC /* sort the results by closest distance */
```

To provide the source airports `iso_country`, `latitude`, and `longitude` we can use the previous Airport Codes query.

Source Airport Query

source*airport*ICT.n1ql

```
SELECT airports.iso_country, airports.geo.latitude AS latitude, airports.geo.longitude A
FROM `flight-data` AS codes
USE KEYS 'airport_code_ICT'
INNER JOIN `flight-data` AS airports ON KEYS 'airport_' || TOSTRING( codes.id )
LIMIT 1
```

Result

```
[
  {
    "iso_country": "US",
    "latitude": 37.64989853,
    "longitude": -97.43309784
  }
]
```

Next we replace the tokens from our base radius query with the returned values.

Airports Near a Given Airport in Miles Query

For our example we want to find any airports within 100 miles of "ICT". Our `{{distance_unit}}` is miles, this value needs to be `69` and our `{{radius}}` is `100`. Replace the `{{source_latitude}}`, `{{source_longitude}}` and `{{iso_country}}` with the values from the previous query.

airports*near*airport*by*miles.n1ql

```
SELECT results.airport_name, results.airport_code, ROUND( results.distance, 2 ) AS dista
FROM (
    SELECT airports.airport_name,
        IFNULL( airports.airport_iata, airports.airport_icao, airports.airport_ident ) A
        69 * DEGREES(ACOS(COS(RADIANS( 37.64989853 ))
        * COS(RADIANS( airports.geo.latitude ))
        * COS(RADIANS( -97.43309784 ) - RADIANS( airports.geo.longitude ))
        + SIN(RADIANS( 37.64989853 ))
        * SIN(RADIANS( airports.geo.latitude )))) AS distance
    FROM `flight-data` AS airports
    WHERE airports.iso_country = 'US'
        AND airports.geo.latitude BETWEEN
            37.64989853 - (100 / 69)
            AND
            37.64989853 + (100 / 69)
        AND airports.geo.longitude BETWEEN
            -97.43309784 - (100 / (69 * COS(RADIANS( 37.64989853 ))))
            AND
            -97.43309784 + ( 100 / ( 69 * COS(RADIANS( 37.64989853 ))))
        AND airports.doc_type = 'airport'
    ) AS results
WHERE results.distance > 0
    AND results.distance <= 100
ORDER BY results.distance ASC
```

Airports Near a Given Airport in Miles Results

```
[
```

```
[
  {
    "airport_code": "IAB",
    "airport_name": "Mc Connell Afb",
    "distance": 9.21
  },
  {
    "airport_code": "BEC",
    "airport_name": "Beech Factory Airport",
    "distance": 12.3
  },
  {
    "airport_code": "EGT",
    "airport_name": "Wellington Municipal",
    "distance": 22.65
  },
  {
    "airport_code": "EWK",
    "airport_name": "Newton City-County Airport",
    "distance": 29.47
  },
  {
    "airport_code": "HUT",
    "airport_name": "Hutchinson Municipal Airport",
    "distance": 36.94
  },
  {
    "airport_code": "PNC",
    "airport_name": "Ponca City Rgnl",
    "distance": 65.93
  },
  {
    "airport_code": "SLN",
    "airport_name": "Salina Municipal Airport",
    "distance": 79.63
  },
  {
    "airport_code": "EMP",
    "airport_name": "Emporia Municipal Airport",
    "distance": 82.32
  },
  {
    "airport_code": "GBD",
    "airport_name": "Great Bend Municipal",
    "distance": 91.15
  },
```

```
  {
    "airport_code": "END",
    "airport_name": "Vance Afb",
    "distance": 94.28
  }
]
```

Airports Near a Given Airport in Kilometers Query

For our example we want to find any airports within 75 kilometers of "Berlin" (TXL). Our `{{distance_unit}}` is kilometers, this value needs to be `111.045` and our `{{radius}}` is `75`. Replace the `{{source_latitude}}`, `{{source_longitude}}` and `{{iso_country}}` with the values from the previous query.

Source Airport Query

sourceairportTXL.n1ql

```
SELECT airports.iso_country, airports.geo.latitude AS latitude, airports.geo.longitude A
FROM `flight-data` AS codes
USE KEYS 'airport_code_TXL'
INNER JOIN `flight-data` AS airports ON KEYS 'airport_' || TOSTRING( codes.id )
LIMIT 1
```

Result

```
[
  {
    "iso_country": "DE",
    "latitude": 52.55970001,
    "longitude": 13.2876997
  }
]
```

Airports Near a Given Airport in Kilometers Query

airportsnearairportbykilometers.n1ql

```sql
SELECT results.airport_name, results.airport_code, ROUND( results.distance, 2 ) AS dista
FROM (
    SELECT airports.airport_name,
        IFNULL( airports.airport_iata, airports.airport_icao, airports.airport_ident ) A
        111.045 * DEGREES(ACOS(COS(RADIANS( 52.55970001 ))
        * COS(RADIANS( airports.geo.latitude ))
        * COS(RADIANS( 13.2876997 ) - RADIANS( airports.geo.longitude ))
        + SIN(RADIANS( 52.55970001 ))
        * SIN(RADIANS( airports.geo.latitude )))) AS distance
    FROM `flight-data` AS airports
    WHERE airports.iso_country = 'DE'
        AND airports.geo.latitude BETWEEN
            52.55970001 - ( 75 / 111.045 )
            AND
            52.55970001 + ( 75 / 111.045 )
        AND airports.geo.longitude BETWEEN
            13.2876997 - ( 75 / ( 111.045 * COS(RADIANS( 52.55970001 ))))
            AND
            13.2876997 + ( 75 / ( 111.045 * COS(RADIANS( 52.55970001 ))))
        AND airports.doc_type = 'airport'
    ) AS results
WHERE results.distance > 0
    AND results.distance <= 75
ORDER BY results.distance ASC
```

Airport Radius in Kilometers Results


```json
[
  {
    "airport_code": "SXF",
    "airport_name": "Berlin Brandenburg Willy Brandt",
    "distance": 25.5
  },
  {
    "airport_code": "EDCS",
    "airport_name": "Saarmund Airport",
    "distance": 30.65
  },
  {
    "airport_code": "EDOI",
    "airport_name": "Bienenfarm Airport",
    "distance": 38.25
  },
  {
    "airport_code": "EDAV",
    "airport_name": "Flugplatz Finow",
```

```
    "distance": 40.36
  },
  {
    "airport_code": "QXH",
    "airport_name": "Schonhagen",
    "distance": 40.53
  },
  {
    "airport_code": "EDAY",
    "airport_name": "Strausberg",
    "distance": 42.51
  },
  {
    "airport_code": "EDAI",
    "airport_name": "Segeletz Airport",
    "distance": 58.29
  },
  {
    "airport_code": "EDOJ",
    "airport_name": "Luesse Airport",
    "distance": 62.82
  },
  {
    "airport_code": "EDBK",
    "airport_name": "Kyritz",
    "distance": 70.38
  }
]
```


Airports Near a Given Airport in Miles with Source Airport Query

Our previous examples have required us to perform a separate query to determine the source airport information. We add the source airports information to the query by utilizing the `NEST` statement. For our example we want to find any airports within 100 miles of "ICT". Our `{{distance_unit}}` is miles, this value needs to be `69` and our `{{radius}}` is `100`.

airports*near*airport*by*miles*with*lookup.n1ql

```
SELECT results.airport_name, results.airport_code, ROUND( results.distance, 2 ) AS dista
FROM (
    SELECT airports.airport_name,
        IFNULL( airports.airport_iata, airports.airport_icao, airports.airport_ident ) A
        69 * DEGREES(ACOS(COS(RADIANS( source_airport[0].geo.latitude ))
        * COS(RADIANS( airports.geo.latitude ))
        * COS(RADIANS( source_airport[0].geo.longitude ) - RADIANS( airports.geo.longitu
        + SIN(RADIANS( source_airport[0].geo.latitude ))
        * SIN(RADIANS( airports.geo.latitude )))) AS distance
    FROM `flight-data` AS airports
    INNER NEST `flight-data` AS source_airport ON KEYS (
        ARRAY 'airport_' || TOSTRING(a.id) FOR a IN (
            SELECT lookup_code.id
            FROM `flight-data` AS lookup_code
            USE KEYS 'airport_code_ICT'
            LIMIT 1
        ) END
    )
    WHERE airports.iso_country = source_airport[0].iso_country
        AND airports.geo.latitude BETWEEN
            source_airport[0].geo.latitude - ( 100 / 69 )
            AND
            source_airport[0].geo.latitude + ( 100 / 69 )
        AND airports.geo.longitude BETWEEN
            source_airport[0].geo.longitude - ( 100 / ( 69 * COS(RADIANS( source_airport
            AND
            source_airport[0].geo.longitude + ( 100 / ( 69 * COS(RADIANS( source_airport
        AND airports.doc_type = 'airport'
    ) AS results
WHERE results.distance > 0
    AND results.distance <= 100
ORDER BY results.distance ASC
```

Airports Near a Given Airport in Miles Results


```
[
  {
    "airport_code": "IAB",
    "airport_name": "Mc Connell Afb",
    "distance": 9.21
  },
  {
    "airport_code": "BEC",
    "airport_name": "Beech Factory Airport",
    "distance": 12.3
  },
```

```
  {
    "airport_code": "EGT",
    "airport_name": "Wellington Municipal",
    "distance": 22.65
  },
  {
    "airport_code": "EWK",
    "airport_name": "Newton City-County Airport",
    "distance": 29.47
  },
  {
    "airport_code": "HUT",
    "airport_name": "Hutchinson Municipal Airport",
    "distance": 36.94
  },
  {
    "airport_code": "PNC",
    "airport_name": "Ponca City Rgnl",
    "distance": 65.93
  },
  {
    "airport_code": "SLN",
    "airport_name": "Salina Municipal Airport",
    "distance": 79.63
  },
  {
    "airport_code": "EMP",
    "airport_name": "Emporia Municipal Airport",
    "distance": 82.32
  },
  {
    "airport_code": "GBD",
    "airport_name": "Great Bend Municipal",
    "distance": 91.15
  },
  {
    "airport_code": "END",
    "airport_name": "Vance Afb",
    "distance": 94.28
  }
]
```


While this executes and we get the same results as before, it is exponentially slower because the `NEST` is happening on every record.