User Queries

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

Users By ID

The following query will get a User by their ID.

Query

user*by*document_id.n1ql

```
SELECT users.user_id, users.account.username, users.account.`password`
FROM `flight-data` AS users
USE KEYS 'user_197'
```

Result

```
[
  {
    "password": "N8HERvS8btfGbmz",
    "user_id": 197,
    "username": "Eudora43"
  }
]
```

The following query will retrieve multiple Users by their ID.

Query

users*by*document_id.n1ql

```
SELECT users.details.*
FROM `flight-data` AS users
USE KEYS ['user_197', 'user_999']
```

Result

```
[
  {
    "company": null,
    "dob": "2015-09-24",
    "first_name": "Albin",
    "home_country": "GQ",
    "job_title": "Forward Markets Director",
    "last_name": "Price",
    "middle_name": null,
    "prefix": "Dr.",
    "suffix": null
  },
  {
    "company": null,
    "dob": null,
    "first_name": "Dallas",
    "home_country": "KH",
    "job_title": "Central Functionality Executive",
    "last_name": "Kunze",
    "middle_name": "Harriett",
    "prefix": null,
    "suffix": null
  }
]
```


Users By Username

The following query will get a User by their Username.

Index

idx*users*username.n1ql

```
CREATE INDEX idx_users_username ON `flight-data`( account.username )
WHERE doc_type = 'user'
USING GSI
```

Query

user*by*username.n1ql

```
SELECT users.user_id, users.details.first_name, users.details.last_name
FROM `flight-data` AS users
WHERE users.account.username = 'Eudora43'
    AND users.doc_type = 'user'
LIMIT 1
```

Result

```
[
  {
    "first_name": "Albin",
    "last_name": "Price",
    "user_id": 197
  }
]
```

The following index and query will retrieve a user by their `username` and `password`.

Index

We need to update our index from the previous example, to do that we need to drop and recreate it.

idx*users*username_drop.n1ql

```
DROP INDEX `flight-data`.idx_users_username
```

idx*users*username_password.n1ql

```
CREATE INDEX idx_users_username_password ON `flight-data`( account.username, account.`pa
WHERE doc_type = 'user'
USING GSI
```

Query

user*by*username_password.n1ql

```
SELECT users.user_id, users.details.first_name, users.details.last_name
FROM `flight-data` AS users
WHERE users.account.username = 'Eudora43'
    AND users.account.`password` = 'N8HERvS8btfGbmz'
    AND users.doc_type = 'user'
LIMIT 1
```

Result

```
[
  {
    "first_name": "Albin",
    "last_name": "Price",
    "user_id": 197
  }
]
```

Users Addresses

The following query will get a users addresses by their `user_id`

Query

[user*addresses*by*document*id.n1ql](#)

```
SELECT users.addresses
FROM `flight-data` AS users
USE KEYS 'user_197'
```

Result

```
[
  {
    "addresses": [
      {
        "address_1": "98527 Tromp Light Lodge",
        "address_2": null,
        "iso_country": "GQ",
        "iso_region": "GQ-CS",
        "locality": "South Selmerhaven",
        "postal_code": "49540-9412",
        "primary": true,
        "type": "Home"
      },
      {
        "address_1": "5783 Mathilde Vista Parkway",
        "address_2": "Apt. 899",
        "iso_country": "GQ",
        "iso_region": "GQ-CS",
        "locality": "Schinnerside",
        "postal_code": "78895",
        "primary": false,
        "type": "Home"
      }
    ]
  }
]
```


However, these results are not friendly to work with as there is actually only 1 record returned. This is because we only selected a single user document which has a single property `addresses` that is an array of multiple addresses. We need to flatten this array and return it as separate documents.

Query

user*addresses*flattened.n1ql

```
SELECT flattened_addresses.*
FROM `flight-data` AS users
USE KEYS 'user_197'
UNNEST users.addresses AS flattened_addresses
```

Result

```
[
  {
    "address_1": "98527 Tromp Light Lodge",
    "address_2": null,
    "iso_country": "GQ",
    "iso_region": "GQ-CS",
    "locality": "South Selmerhaven",
    "postal_code": "49540-9412",
    "primary": true,
    "type": "Home"
  },
  {
    "address_1": "5783 Mathilde Vista Parkway",
    "address_2": "Apt. 899",
    "iso_country": "GQ",
    "iso_region": "GQ-CS",
    "locality": "Schinnerside",
    "postal_code": "78895",
    "primary": false,
    "type": "Home"
  }
]
```

We know that each of our users has a primary address, we need to be able to return just that address.

Query

user*addresses*flattened_primary.n1ql

```
SELECT flattened_addresses.*
FROM `flight-data` AS users
USE KEYS 'user_197'
UNNEST users.addresses AS flattened_addresses
WHERE flattened_addresses.`primary` = true
```

Results

```
[
  {
    "address_1": "98527 Tromp Light Lodge",
    "address_2": null,
    "iso_country": "GQ",
    "iso_region": "GQ-CS",
    "locality": "South Selmerhaven",
    "postal_code": "49540-9412",
    "primary": true,
    "type": "Home"
  }
]
```

Building on the previous examples, we want to return the full country and region names as part of each address.

Query

user*addresses*flattened*country*continent.n1ql

```
SELECT flattened_addresses.address_1, flattened_addresses.address_2, flattened_addresses
    flattened_addresses.postal_code, flattened_addresses.`primary`, flattened_addresses.
    flattened_addresses.iso_country, countries.country_name,
    flattened_addresses.iso_region, regions.region_name
FROM `flight-data` AS users
USE KEYS 'user_197'
UNNEST users.addresses AS flattened_addresses
INNER JOIN `flight-data` AS countries
    ON KEYS 'country_' || flattened_addresses.iso_country
INNER JOIN `flight-data` AS regions
    ON KEYS 'region_' || flattened_addresses.iso_region
```

Result

```
[
  {
    "address_1": "98527 Tromp Light Lodge",
    "address_2": null,
    "country_name": "Equatorial Guinea",
    "iso_country": "GQ",
    "iso_region": "GQ-CS",
    "locality": "South Selmerhaven",
    "postal_code": "49540-9412",
    "primary": true,
    "region_name": "Centro Sur",
    "type": "Home"
  },
  {
    "address_1": "5783 Mathilde Vista Parkway",
    "address_2": "Apt. 899",
    "country_name": "Equatorial Guinea",
    "iso_country": "GQ",
    "iso_region": "GQ-CS",
    "locality": "Schinnerside",
    "postal_code": "78895",
    "primary": false,
    "region_name": "Centro Sur",
    "type": "Home"
  }
]
```


And now with just the primary address information.

Query

user*addresses*flattened*primary*country_continent.n1ql

```
SELECT flattened_addresses.address_1, flattened_addresses.address_2, flattened_addresses
    flattened_addresses.postal_code, flattened_addresses.type,
    flattened_addresses.iso_country, countries.country_name,
    flattened_addresses.iso_region, regions.region_name
FROM `flight-data` AS users
USE KEYS 'user_197'
UNNEST users.addresses AS flattened_addresses
INNER JOIN `flight-data` AS countries
    ON KEYS 'country_' || flattened_addresses.iso_country
INNER JOIN `flight-data` AS regions
    ON KEYS 'region_' || flattened_addresses.iso_region
WHERE flattened_addresses.`primary` = true
```

Result

```
[
  {
    "address_1": "98527 Tromp Light Lodge",
    "address_2": null,
    "country_name": "Equatorial Guinea",
    "iso_country": "GQ",
    "iso_region": "GQ-CS",
    "locality": "South Selmerhaven",
    "postal_code": "49540-9412",
    "region_name": "Centro Sur",
    "type": "Home"
  }
]
```

Now we want to lookup our users by the region that they are in. To do this we will need to create an index on the `addresses[*].iso_region`. Prior to Couchbase 4.5 the entire array had to be indexed and data could not be efficiently queried.

Index

idx*users*addresses_regions.n1ql

```
CREATE INDEX idx_users_addresses_region ON `flight-data`(
    DISTINCT ARRAY address.iso_region
        FOR address IN addresses
            WHEN address.iso_region IS NOT NULL
        END
)
WHERE doc_type = 'user';
```

Query

For our query, we do not want to return the entire `addresses` property, we want to omit the `primary` and `type` fields. The results should be sorted by `iso_region DESC` and have the `primary` address listed first.

user*addresses*by_region.n1ql

```
SELECT users.details.first_name ||
    IFNULL(' ' ||  users.details.last_name, '') AS name,
    (
        ARRAY {
            "address_1": address.address_1,
            "address_2": address.address_2,
            "iso_region": address.iso_region,
            "iso_country": address.iso_country,
            "postal_code": address.postal_code,
            "locality": address.locality
        } FOR address IN users.addresses END
    ) AS addresses
FROM `flight-data` AS users
WHERE users.doc_type = 'user'
    AND (
        ANY address IN users.addresses
            SATISFIES address.iso_region IN [
                'US-AK', 'US-MN', 'US-NC'
            ]
        END
    )
ORDER BY users.addresses[*].iso_region DESC,
    users.addresses[*].`primary` DESC
```

Result


```
[
  {
    "addresses": [
      {
        "address_1": "38582 Sigrid Terrace Cape",
        "address_2": null,
        "iso_country": "US",
        "iso_region": "US-NC",
        "locality": "Sengermouth",
        "postal_code": "76275-0205",
        "primary": false,
        "type": "Work"
      },
      {
        "address_1": "1844 Krajcik Unions Garden",
        "address_2": "Apt. 925",
        "iso_country": "US",
        "iso_region": "US-NC",
        "locality": "Macieborough",
        "postal_code": "17581-8835",
```

```
        "primary": true,
        "type": "Other"
      }
    ],
    "first_name": "Marianna",
    "last_name": "Labadie"
  },
  {
    "addresses": [
      {
        "address_1": "82985 Angus Garden Mountain",
        "address_2": null,
        "iso_country": "US",
        "iso_region": "US-NC",
        "locality": "Swiftfort",
        "postal_code": "26911-4639",
        "primary": true,
        "type": "Other"
      }
    ],
    "first_name": "Yasmeen",
    "last_name": "Rippin"
  },
  {
    "addresses": [
      {
        "address_1": "8913 Rodriguez Gardens Fords",
        "address_2": "Apt. 764",
        "iso_country": "US",
        "iso_region": "US-MN",
        "locality": "Bonniestad",
        "postal_code": "07379",
        "primary": true,
        "type": "Home"
      },
      {
        "address_1": "69403 Cleora Ports Shores",
        "address_2": null,
        "iso_country": "US",
        "iso_region": "US-MN",
        "locality": "Randiside",
        "postal_code": "91175",
        "primary": false,
        "type": "Work"
      }
```

```
    ],
    "first_name": "Winnifred",
    "last_name": "Koepp"
  },
  {
    "addresses": [
      {
        "address_1": "354 Susanna Row Falls",
        "address_2": null,
        "iso_country": "US",
        "iso_region": "US-MN",
        "locality": "Champlinchester",
        "postal_code": "26170",
        "primary": true,
        "type": "Other"
      },
      {
        "address_1": "38849 Brakus Divide Keys",
        "address_2": null,
        "iso_country": "US",
        "iso_region": "US-MN",
        "locality": "Barrowston",
        "postal_code": "05343-0841",
        "primary": false,
        "type": "Home"
      }
    ],
    "first_name": "Lola",
    "last_name": "Emmerich"
  },
  {
    "addresses": [
      {
        "address_1": "90856 Stark Streets Manors",
        "address_2": null,
        "iso_country": "US",
        "iso_region": "US-MN",
        "locality": "Port Carlie",
        "postal_code": "56282-1062",
        "primary": true,
        "type": "Work"
      }
    ],
    "first_name": "Marie",
    "last_name": "Marks"
  },
```

```
  {
    "addresses": [
      {
        "address_1": "6136 Kuhlman Isle Crossroad",
        "address_2": null,
        "iso_country": "US",
        "iso_region": "US-MN",
        "locality": "Emmittshire",
        "postal_code": "41017-8748",
        "primary": true,
        "type": "Other"
      }
    ],
    "first_name": "Emmie",
    "last_name": null
  }
]
```

Building on the previous index and query, we want to retrieve all of the users, a unique array of each region the user is in, as well as the total # of addresses.

Query

user*addresses*by*region*distinct.n1ql

```
SELECT users.details.first_name ||
    IFNULL(' ' ||  users.details.last_name, '') AS name,
    ARRAY_LENGTH(users.addresses) AS addresses,
    ARRAY_DISTINCT(
        ARRAY address.iso_region FOR address IN users.addresses END
    ) AS regions
FROM `flight-data` AS users
WHERE users.doc_type = 'user'
    AND (
        ANY address IN users.addresses
            SATISFIES address.iso_region IN ['US-SC']
        END
    )
ORDER BY users.addresses[*].iso_region DESC
```

Result

```
[
  {
    "addresses": 2,
    "name": "Lenny Borer",
    "regions": [
      "US-KS",
      "US-SC"
    ]
  },
  {
    "addresses": 2,
    "name": "Jasper Donnelly",
    "regions": [
      "US-SC"
    ]
  },
  {
    "addresses": 1,
    "name": "Cindy Thiel",
    "regions": [
      "US-SC"
    ]
  },
  {
    "addresses": 2,
    "name": "Zoila Koepp",
    "regions": [
      "US-NM",
      "US-SC"
    ]
  }
]
```


Users Phones

The following query will get a users phones by their `user_id`

Query

user*phones*by*document*id.n1ql

```
SELECT users.phones
FROM `flight-data` AS users
USE KEYS 'user_197'
```

Result


```
[
  {
    "phones": [
      {
        "extension": null,
        "phone_number": "(570) 615-4605",
        "primary": false,
        "type": "Home"
      },
      {
        "extension": "1735",
        "phone_number": "(923) 578-2435",
        "primary": true,
        "type": "Mobile"
      }
    ]
  }
]
```


Just like the addresses, we need to flatten these results to make them more useful.

Query

user*phones*flattened.n1ql

```
SELECT flattened_phones.*
FROM `flight-data` AS users
USE KEYS 'user_197'
UNNEST users.phones AS flattened_phones
```

Result

```
[
  {
    "extension": null,
    "phone_number": "(570) 615-4605",
    "primary": false,
    "type": "Home"
  },
  {
    "extension": "1735",
    "phone_number": "(923) 578-2435",
    "primary": true,
    "type": "Mobile"
  }
]
```

We know that each of our users has a primary phone, we need to be able to return just that phone.

Query

user*phones*flattened_primary.n1ql

```
SELECT flattened_phones.*
FROM `flight-data` AS users
USE KEYS 'user_197'
UNNEST users.phones AS flattened_phones
WHERE flattened_phones.`primary` = true
```

Results

```
[
  {
    "extension": "1735",
    "phone_number": "(923) 578-2435",
    "primary": true,
    "type": "Mobile"
  }
]
```

Users Emails

The following query will get a users emails by their `user_id`

Query

useremailsbydocumentid.n1ql

```
SELECT users.emails
FROM `flight-data` AS users
USE KEYS 'user_1997'
```

Result

```
[
  {
    "emails": [
      {
        "email_address": "Chase.Kohler63@gmail.com",
        "primary": false,
        "type": "Home"
      },
      {
        "email_address": "Judah66@gmail.com",
        "primary": true,
        "type": "Home"
      },
      {
        "email_address": "Creola_Little34@gmail.com",
        "primary": false,
        "type": "Work"
      }
    ]
  }
]
```

Just like the addresses and phones, we need to flatten these results to make them more useful.

Query

useremailsflattened.n1ql

```
SELECT flattened_emails.*
FROM `flight-data` AS users
USE KEYS 'user_1997'
UNNEST users.emails AS flattened_emails
```

Result

```
[
  {
    "email_address": "Chase.Kohler63@gmail.com",
    "primary": false,
    "type": "Home"
  },
  {
    "email_address": "Judah66@gmail.com",
    "primary": true,
    "type": "Home"
  },
  {
    "email_address": "Creola_Little34@gmail.com",
    "primary": false,
    "type": "Work"
  }
]
```

We know that each of our users has a primary email address, we need to be able to return just that email.

Query

user*emails*flattened_primary.n1ql

```
SELECT flattened_emails.*
FROM `flight-data` AS users
USE KEYS 'user_1997'
UNNEST users.emails AS flattened_emails
WHERE flattened_emails.`primary` = true
```

Results

```
[
  {
    "email_address": "Judah66@gmail.com",
    "primary": true,
    "type": "Home"
  }
]
```

The following query will retrieve only the email address from the array of objects, omitting the `type` and `primary` attributes.

Query

user*emails*flattened*email*only.n1ql

```
SELECT email
FROM `flight-data` AS users
USE KEYS 'user_1997'
UNNEST users.emails[*].email_address AS email
```

Results

```
[
  {
    "email": "Chase.Kohler63@gmail.com"
  },
  {
    "email": "Judah66@gmail.com"
  },
  {
    "email": "Creola_Little34@gmail.com"
  }
]
```

Building on the previous query, what if we wanted to ensure that the first email listed was the primary email address. We can perform the following query.

user*emails*flattened*primary*email_only.n1ql

```
SELECT emails.email_address AS email
FROM `flight-data` AS users
USE KEYS 'user_1997'
UNNEST users.emails AS emails
WHERE emails.`primary` = true
```

```
[
  {
    "email": "Judah66@gmail.com"
  },
  {
    "email": "Chase.Kohler63@gmail.com"
  },
  {
    "email": "Creola_Little34@gmail.com"
  }
]
```

User By ID as Flat Object

Our user model uses nested attributes, we need to retrieve a users record as a single level object with just the primary address, phone and email.

Query

user*by*document*id*flat.n1ql

```
SELECT users.account.*, users.details.*,
    primary_email.email_address,
    primary_address.address_1, primary_address.address_2, primary_address.iso_country,
    primary_address.iso_region, primary_address.locality, primary_address.postal_code,
    primary_phone.phone_number, primary_phone.extension AS phone_extension
FROM `flight-data` AS users
USE KEYS 'user_1997'
UNNEST users.emails AS primary_email
UNNEST users.addresses AS primary_address
UNNEST users.phones AS primary_phone
WHERE primary_email.`primary` = true
    AND primary_address.`primary` = true
    AND primary_phone.`primary` = true
```

Result

```
[
  {
    "address_1": "07363 Trantow Garden Crossroad",
    "address_2": "Suite 108",
    "company": null,
    "created_on": 1447034465570,
    "dob": "2016-02-17",
    "email_address": "Judah66@gmail.com",
    "first_name": "Donnell",
    "home_country": "VC",
    "iso_country": "VC",
    "iso_region": "VC-01",
    "job_title": null,
    "last_login": 1463565402328,
    "last_name": "Ortiz",
    "locality": "South Leland",
    "middle_name": "Winifred",
    "modified_on": 1463561681928,
    "password": "cbEsvC1RxKRv0gi",
    "phone_extension": null,
    "phone_number": "(569) 409-9444",
    "postal_code": "26561",
    "prefix": null,
    "suffix": null,
    "username": "Garett31"
  }
]
```