Objective:

In this lab, we will utilize Eventing capabilities to locate suspicious transactions from a bucket having credit card records. The bucket, “register” consists of the following types of documents.

Merchants:

Information about various merchants in the system

```
{
  "type": "merchant",
  "merchantid": "merchant-1004906482773138605-4121",
  "name": "Rapid Cycle Solutions LLC",
  "city": {
    "name": "Verndale",
    "code": "MN",
    "state": "Minnesota",
    "county": "Wadena",
    "display": "Wing River"
  }
}
```

Cards:

Credit cards issued by various banks.

```
{
  "type": "card",
  "cardnumber": "4163-4174-9281-3991",
  "firstname": "Francoise",
  "lastname": "Navar",
  "street": "134 Yacht Rd",
  "city": {
    "name": "Skillman",
    "code": "NJ",
    "state": "New Jersey",
    "county": "Somerset",
    "display": "Montgomery"
  },
  "issued": "1/17",
  "expiry": "3/21",
  "ccv": 213,
  "issuer": "First National Bank of Las Animas",
  "threshold": 12068
}
```

Transactions:

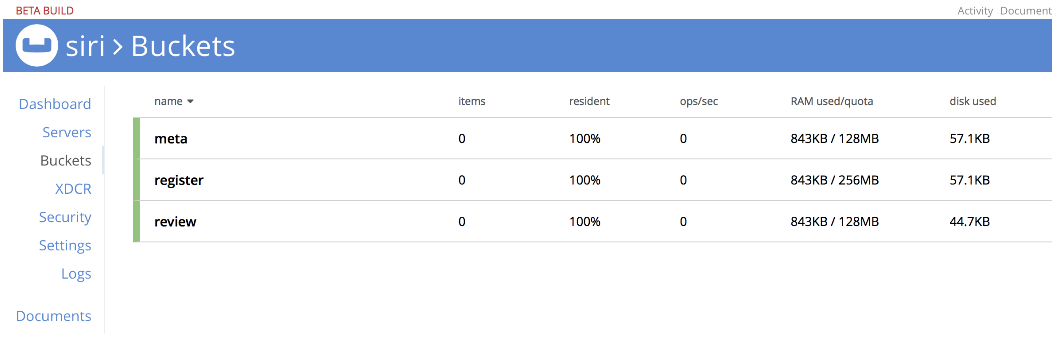
Recent transactions on the credit cards.

```
{
  "type": "transaction",
  "txnid": "tx-1504799249-129",
  "amount": 12829,
  "product": "S-Fddsdv36 Type D Flipper Unit Sold Face 36w",
  "card": "4163-4174-9281-3991",
  "merchant": "Inovalon Company",
  "city": {
    "name": "West Palm Beach",
    "code": "FL",
    "state": "Florida",
    "county": "Palm Beach",
    "display": "Royal Palm Beach"
  },
  "date": "2017-09-07T21:17:29+05:30"
}
```

Setup:

1. Create a cluster, and ensure eventing, query and indexing are enabled.

2. Create necessary buckets:
   a. A bucket named “register”. This will hold all sample data for this lab.
   b. A bucket named “meta”. This will hold eventing metadata such as timers.
   c. A bucket named “review”. This will hold suspicious transactions for further review.



name	items	resident	ops/sec	RAM used/quota	disk used
meta	0	100%	0	843KB / 128MB	57.1KB
register	0	100%	0	843KB / 256MB	57.1KB
review	0	100%	0	843KB / 128MB	44.7KB

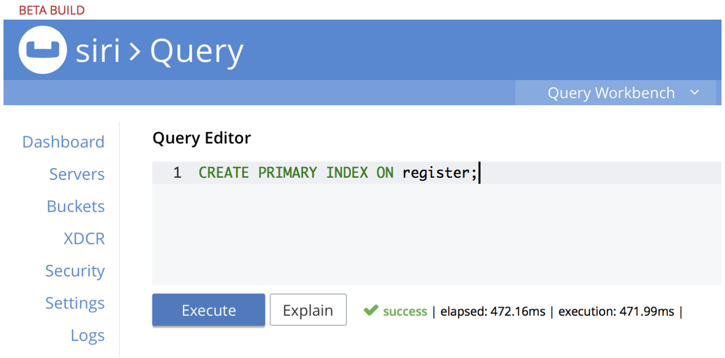
3. Import the data as follows:

```
cbimport json -c http://localhost:8091 -u Administrator -p password \
-b register -f list -g '%type%:%txnid%' -d file://txns.json

cbimport json -c http://localhost:8091 -u Administrator -p password \
-b register -f list -g '%type%:%cardnumber%' -d file://cards.json

cbimport json -c http://localhost:8091 -u Administrator -p password \
-b register -f list -g '%type%:%merchantid%' -d file://merchants.json
```

https://s3.amazonaws.com/eventing-workshop/cards.json
https://s3.amazonaws.com/eventing-workshop/txns.json
https://s3.amazonaws.com/eventing-workshop/merchants.json

(Note: Above are samples. These can be generated to any desired volume using https://github.com/couchbase/eventing/tree/unstable/tools/cc_data and running `go run *.go`)

4. Go to Query Workbench and create a primary index on “register” bucket as follows:
   ```
   CREATE PRIMARY INDEX ON register;
   ```

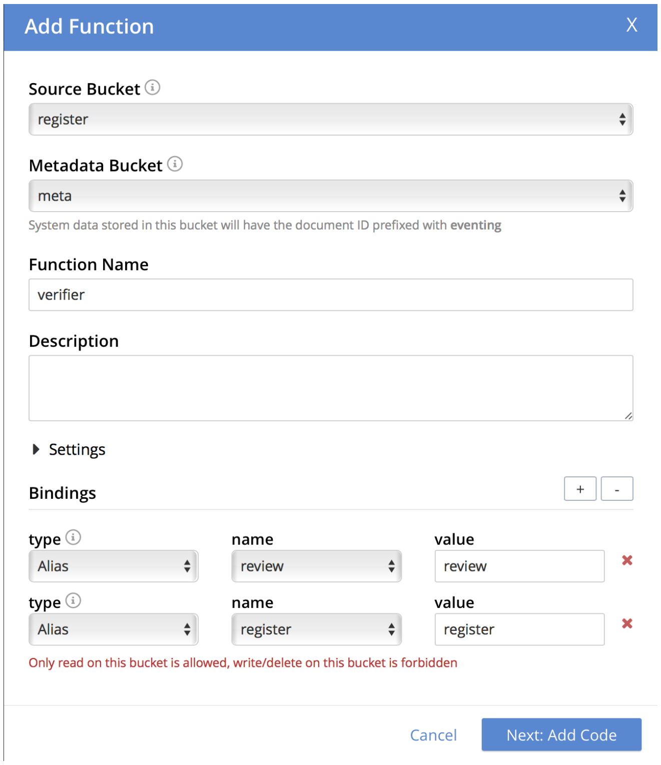


Query Editor

```
CREATE PRIMARY INDEX ON register;
```

✓ success | elapsed: 472.16ms | execution: 471.99ms |

5. Create a new Eventing Handler called “verifier”;
   a. Set source to “register”
   b. Set metadata bucket to “meta”
   c. Name the function “verifier”
   d. Add binding to bucket “review” with alias “review”
   e. Add binding to bucket “register” with alias “register”



Source Bucket: register

Metadata Bucket: meta

System data stored in this bucket will have the document ID prefixed with **eventing**

Function Name: verifier

Description:

Bindings

type	name	value
Alias	review	review
Alias	register	register

Only read on this bucket is allowed, write/delete on this bucket is forbidden

6. Create a new Eventing Handler called “verifier”;
   a. Set source to “register”
   b. Set metadata bucket to “meta”

c. Name the function “verifier”

d. Add binding to bucket “review” with alias “review”

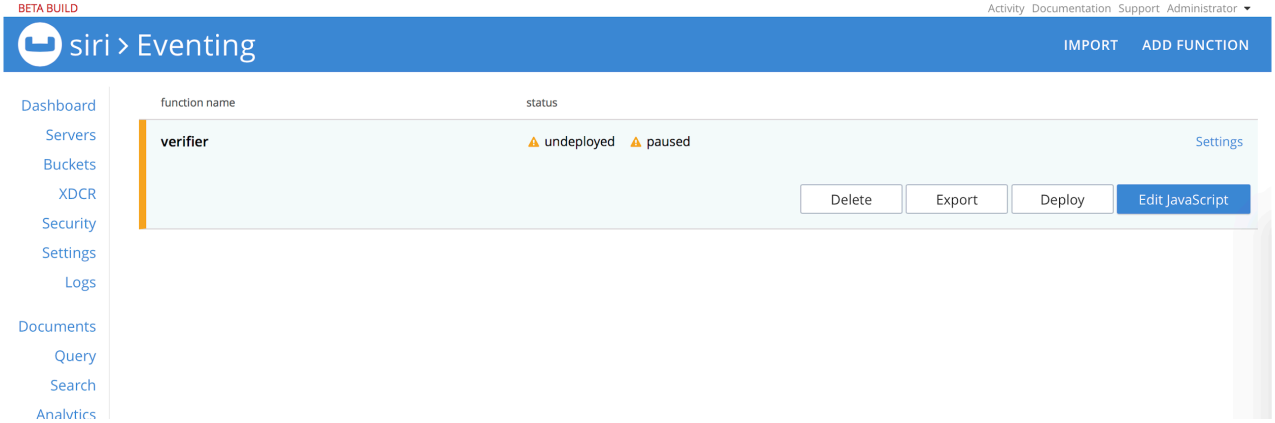
e. Add binding to bucket “register” with alias “register”

7. Next, in the eventing javascript editor, add below code:
   https://s3.amazonaws.com/eventing-workshop/verifier.json

```
function OnUpdate(doc, meta) {
    /* Check if further analysis is needed */
    if (doc.type !== "transaction") return;
    var card = register["card:" + doc.card];
    if (card.threshold > doc.amount) return;

    /* Schedule analysis at night (but for workshop: 60s later) */
    review[doc.txnid] = {"txnid": doc.txnid, "status": "open", "reason": "threshold"};
    var tm = new Date();
    tm.setSeconds(tm.getSeconds() + 15);
    createTimer(ExtendedVerify, tm, doc.txnid, doc);
}

function ExtendedVerify(doc) {
    /* Check if user has transacted with this merchant before */
    var tcard = doc.card;
    var score = 0;
    var reasons = {"product": false, "merchant": false, "proximity": false};

    var history =
        SELECT txnid, product, merchant, city
        FROM register
        WHERE
            type = "transaction" AND
            card = $tcard;

    for (var txn of history) {
        if (txn.txnid === doc.txnid) {
            continue;
        }
        /* Check past purchases of this product */
        if (doc.product === txn.product) {
            score += 8;
            reasons["product"] = true;
        }
        /* Check interaction with this merchant */
        if (doc.merchant === txn.merchant) {
            score += 5;
            reasons["merchant"] = true;
        }
        /* Check locality to city and county */
        if (doc.city.name === txn.city.name) {
            score += 2;
            reasons["proximity"] = true;
        }
        if (doc.city.county === txn.city.county) {
            score += 1;
            reasons["proximity"] = true;
        }
    }

    var status = {"txnid": doc.txnid, "score": score};
    for (var key in reasons) {
        if (!reasons[key]) {
            status[key] = "suspected";
        } else {
            status[key] = "expected";
```

```
                }
            }

            if (score > 300) {
                status["disposition"] = ["closed"];
            }
            else if (score > 100) {
                status["disposition"] = ["human-review"];
            }
            else {
                status["disposition"] = ["sms-alert", "human-review"];
                log("Suspicious transaction: " + doc.txnid)
            }

            review[doc.txnid] = status;
        }
```

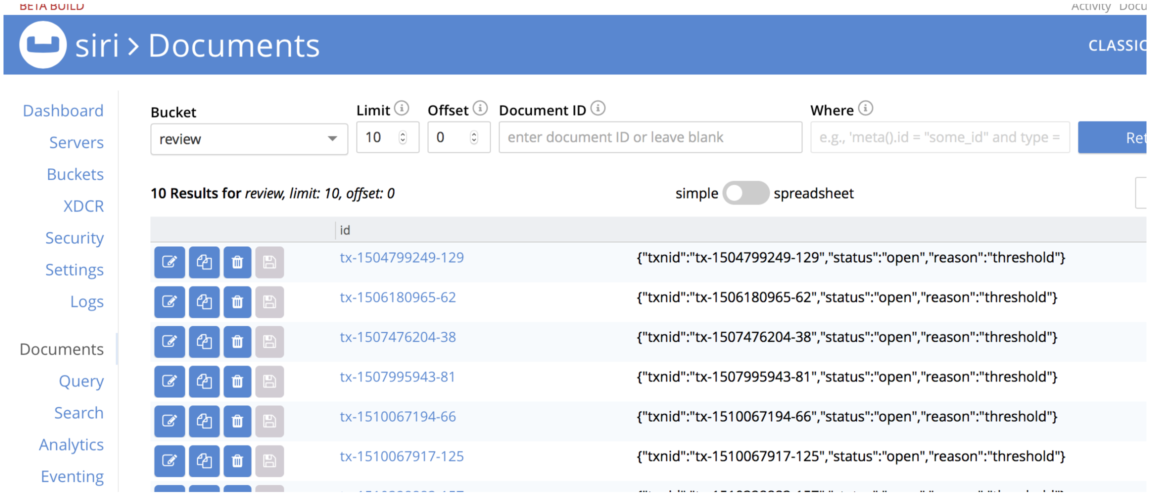
8. Save and go back to Eventing. Then click deploy



function name	status
verifier	undeployed paused

9. Now, the eventing handler will go over every transaction and apply the logic in the handler, to identify transactions that exceed threshold of the credit card. Transactions below the threshold require no further action.

    a. This initial analysis happens at “near real time” as the transactions come into the system.
    b. Identified transactions are uploaded to “review” bucket with status: “open”



Bucket: review
Limit: 10
Offset: 0
Document ID: enter document ID or leave blank
Where: e.g., 'meta().id = "some_id" and type =

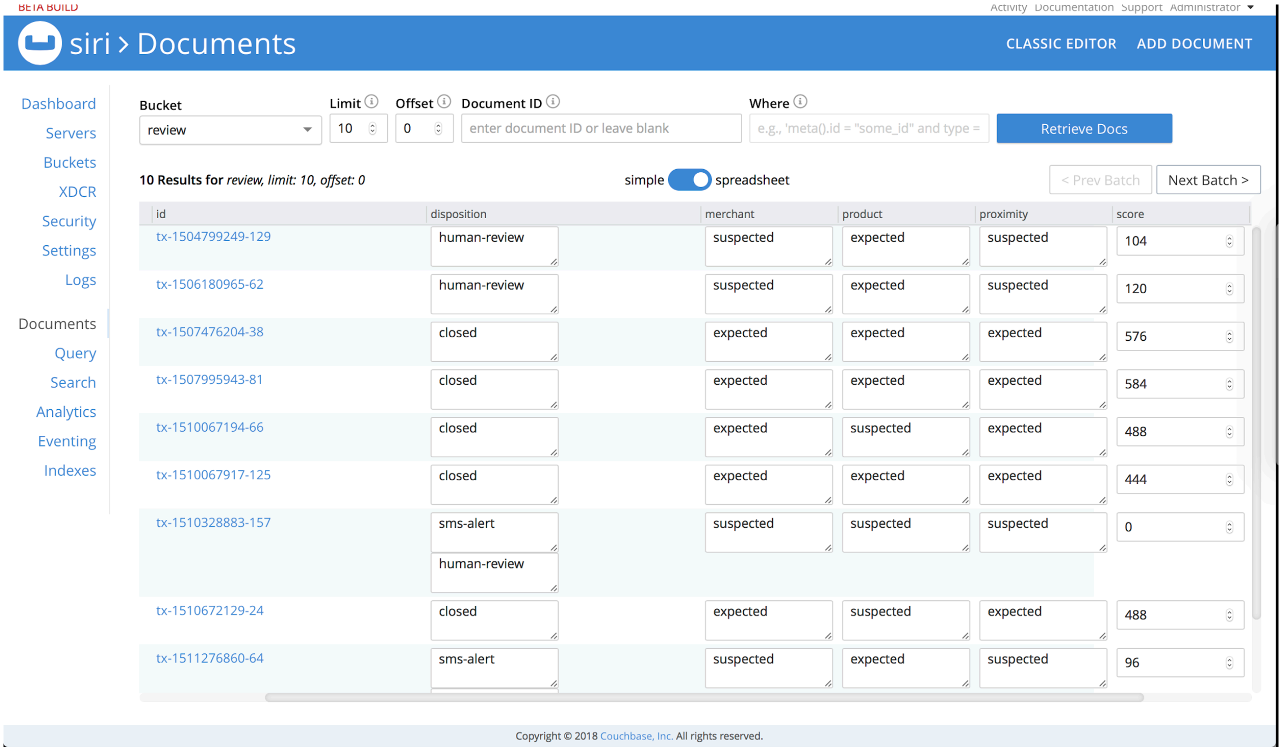
10 Results for *review, limit: 10, offset: 0*

id	
tx-1504799249-129	{"txnid":"tx-1504799249-129","status":"open","reason":"threshold"}
tx-1506180965-62	{"txnid":"tx-1506180965-62","status":"open","reason":"threshold"}
tx-1507476204-38	{"txnid":"tx-1507476204-38","status":"open","reason":"threshold"}
tx-1507995943-81	{"txnid":"tx-1507995943-81","status":"open","reason":"threshold"}
tx-1510067194-66	{"txnid":"tx-1510067194-66","status":"open","reason":"threshold"}
tx-1510067917-125	{"txnid":"tx-1510067917-125","status":"open","reason":"threshold"}

10. Transactions that are identified as needing further review are scheduled to run further analysis. Such analysis is expensive, so we run them at 12am when the cluster is generally not busy. For the workshop, we run 15 seconds later instead of waiting till 12am.

11. Now, the advanced analysis considers
    a. All merchants this card has transacted with in the past
    b. All products this card has purchased in the past
    c. Cities and counties where this card has been used in the past

12. The risk score is computed, the review record is updated with recommendation to either close, for human to review, or to alert the cardholder along with a reason. This can be seen in the “review” bucket after about 15-30 seconds.



Bucket: review
Limit: 10
Offset: 0
Document ID: enter document ID or leave blank
Where: e.g., 'meta().id = "some_id" and type =

10 Results for *review, limit: 10, offset: 0*

simple ● spreadsheet

id	disposition	merchant	product	proximity	score
tx-1504799249-129	human-review	suspected	expected	suspected	104
tx-1506180965-62	human-review	suspected	expected	suspected	120
tx-1507476204-38	closed	expected	expected	expected	576
tx-1507995943-81	closed	expected	expected	expected	584
tx-1510067194-66	closed	expected	suspected	expected	488
tx-1510067917-125	closed	expected	expected	expected	444
tx-1510328883-157	sms-alert human-review	suspected	suspected	suspected	0
tx-1510672129-24	closed	expected	suspected	expected	488
tx-1511276860-64	sms-alert	suspected	expected	suspected	96