Exercise 6 - Custom Logic in Map Functions

Exercise 6.a - Create a Sale Products View in Couchbase

Open up the Couchbase Admin Console by going to http://127.0.0.1:8091/ in a web browser

Choose Data Buckets

Click the "Views" button next to the **default** bucket

Click the "Development Views" tab, since we already have a Design Document, click the "Add View" button next to `design/dev_products`

For the **View Name** use `on_sale`

Click "Save"

Use the following JavaScript as the **Map** function for the view. This will create a view whose value is the the Document ID. The value in the index is *null*. This will only emit documents to the index that match the following.

1. Have a `doc_type` property with a value of "product"
2. Have a `product_id` property
3. Have an `availability` property with a value of "In-Stock"
4. Has a `price` and `sale_price` properties where `sale_price` is less than the `price`

```
function (doc, meta) {
  if(
    doc.doc_type &&
    doc.doc_type === "product" &&
    doc.product_id &&
    doc.availability &&
    doc.availability === "In-Stock" &&
    doc.price &&
    doc.sale_price &&
    doc.sale_price < doc.price
  ){
    emit(doc.product_id, null);
  }
}
```

For the **Reduce** function enter the value of `_count` and click the "Save" button

You can click on the "Show Results" button to see partial data from index.

We now need to publish our Development View to Production. To do this scroll up to the top of the page, and click on Views.

Make sure you are on the **Development Views** tab and next to `_design/dev_products` click on the "Publish" button. You will get a notification that the Design Document already exists and it will ask you to confirm that you want to overwrite it, click "Confirm". This will move your `product` Design Document to production and create the index against all of the documents in the entire bucket for all of the views in the view.On Sale

---

Exercise 6.b - Querying the On Sale View

On the homepage we need to output products whose data indicates that they are actually on sale, instead of a hard-coded list, we will want to limit out results to 8 products.

Open `exercise6/com/example/ProductService.cfc` in your IDE

Modify the `getSaleProducts` method to query Couchbase View. This will call the CFCouchbase `query()` method with the following arguments:

```
query = cb.query(
    designDocumentName = "products",
    viewName = "on_sale",
    options = {
        reduce = false,
        limit = arguments.limit,
        offset = arguments.offset,
        includeDocs = true
    }
);
```

Notice how the reduce option is set to false, this tells Couchbase to not run the reduce function and just return the results from the map function only.

Open the homepage (/exercise6/index.cfm) and see if your On Sale Products is displaying.

For your reference the data from the `getSaleProducts` method is used in the following views:

- exercise6/view/includes/home.sale.cfm
- exercise6/view/includes/template.product.cfm

Exercise 6.c - Browsing All On Sale Products

From our listing of products on the homepage, there is a "View More" button that will allow our users to browse through all of the On Sale products. This view will need to know the total number of On Sale products to calculate the paging correctly.

Open `exercise6/com/example/ProductService.cfc` in your IDE

Modify the `getSaleProductsTotal` method to query Couchbase View. This will call the CFCouchbase `query()` method with the following arguments:

```
query = cb.query(
    designDocumentName = "products",
    viewName = "on_sale"
);
```

Open the homepage (/exercise6/index.cfm) and click on the "View More" button next to the On Sale Products Listing

For your reference the data from the `getSaleProducts` and `getSaleProductsTotal` methods are used in the following views:

- exercise6/view/includes/home.sale.cfm
- exercise6/view/includes/template.product.cfm
- exercise6/view/sale.cfm