Team-based Project 2: CouchDB, Spark, Databricks, and Lakehouses
MBA 552

Due November 22, 2023, 6:00 PM EST

TEAM MEMBERS

ASIF PATHAN

KHYATI DESAI

KUSH TIWARI

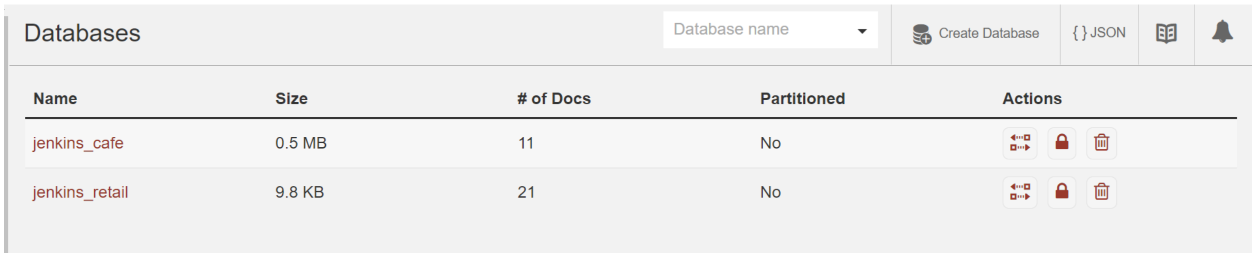
Introduction:
This project focuses mainly on 1) lakehouses and 2) showing how a NoSQL database is more flexible than a relational database. The Project 2 folder on Moodle contains four files: Project2 Description (this file), Project2Data, loan-risks.snappy.parquet, and loan.csv. After successful completion of the project, you should be able to create and query a NoSQL database using CouchDB and create and query a lakehouse using Spark and Databricks. Your team should not collaborate with other teams on your project. Instead, your submitted work should reflect the critical-thinking abilities of only those students on your team.

PART 1 (CouchDB)

Instructions: The data is modified version of data from Machine learning website here. For the rest of the section follow the instructions below. There is an Excel file labeled *Project2Data* in Project 2 folder on Moodle. Use the data in the file to create the following records in the subsequent steps.

Step 1: Launch your CouchDB via Fauxton and create a schemaless database "**jenkins_retail**". In the jenkins_retail database, create all the records in Excel file but with the following properties:

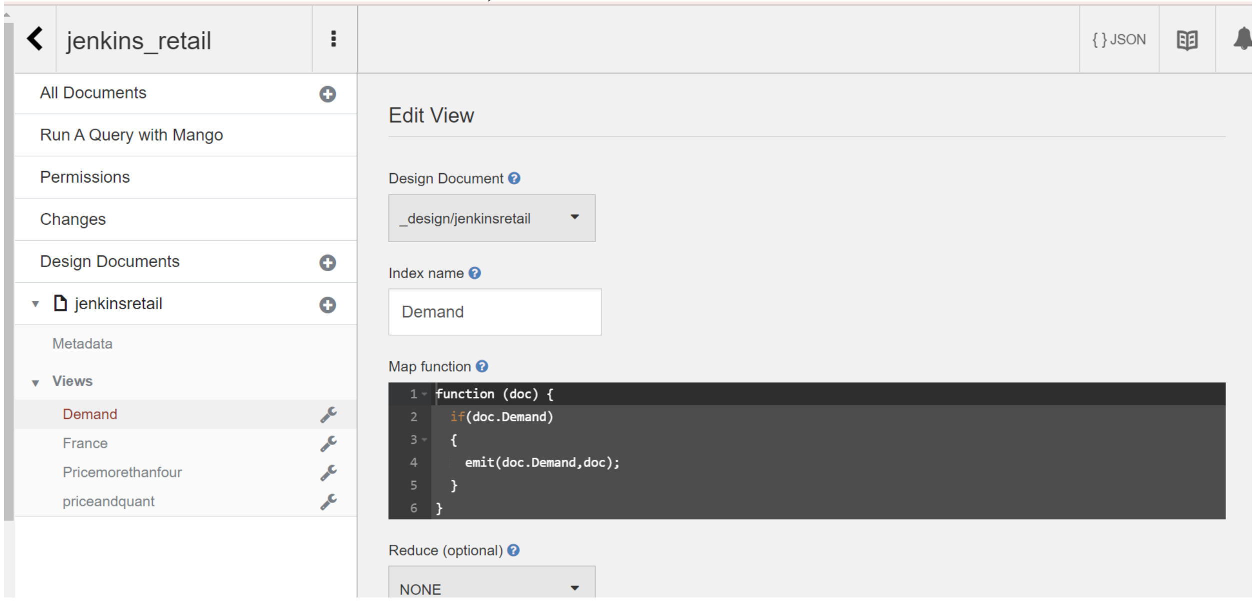
a) Create all the records with columns StockCode, Description, Quantity, Price, CustomerID and Country (that means you should **NOT** include Invoice and InvoiceDate in your columns). Notice that some of the rows do not have a value for the CustomerID column. When you enter the data for these rows, do **NOT** enter a CustomerID field or value. (Remember, in a NoSQL database different records can have different columns.)
b) Create another column "Demand" but **ONLY** for records with Quantity equal to or more than 12. In the column, put "*Moderate*" (if the quantity is between 12 to 20, inclusive of 12 and 20) or "*High*" (if the quantity is more than 20).



Name	Size	# of Docs	Partitioned	Actions
jenkins_cafe	0.5 MB	11	No	
jenkins_retail	9.8 KB	21	No	

Step 2: Now that you have created your database, query it through the map function to answer the following questions. Even though the data is small, the code should work with many records. Therefore, the target is to write successful code. Include a screenshot in this document showing all the data. Include screenshots of the code and output for each of the following map functions.

a) What is the count of records that have the column "Demand"? Write the code to which will display the answer. (Note: One way to answer this is to use code that displays all the records except those with the column Demand and then subtract the number from total number of records.)



Edit View

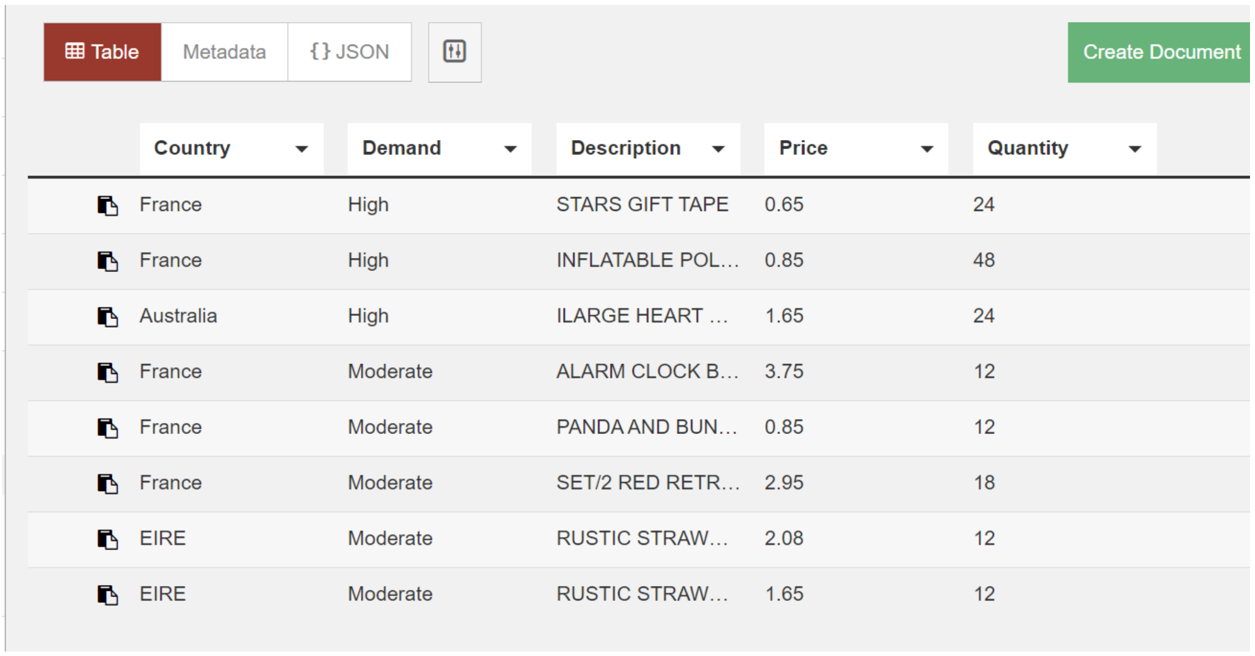
Design Document: _design/jenkinsretail

Index name: Demand

Map function

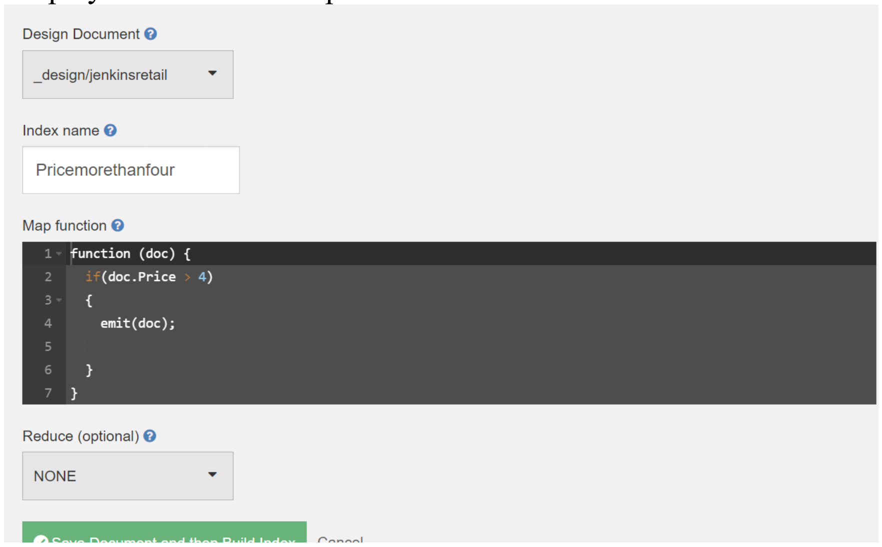
```
function (doc) {
  if(doc.Demand)
  {
    emit(doc.Demand,doc);
  }
}
```

Reduce (optional): NONE



Country	Demand	Description	Price	Quantity
France	High	STARS GIFT TAPE	0.65	24
France	High	INFLATABLE POL…	0.85	48
Australia	High	ILARGE HEART …	1.65	24
France	Moderate	ALARM CLOCK B…	3.75	12
France	Moderate	PANDA AND BUN…	0.85	12
France	Moderate	SET/2 RED RETR…	2.95	18
EIRE	Moderate	RUSTIC STRAW…	2.08	12
EIRE	Moderate	RUSTIC STRAW…	1.65	12

b) Display the records with price more than 4.



Design Document: _design/jenkinsretail

Index name: Pricemorethanfour

Map function

```
function (doc) {
  if(doc.Price > 4)
  {
    emit(doc);

  }
}
```

Reduce (optional): NONE

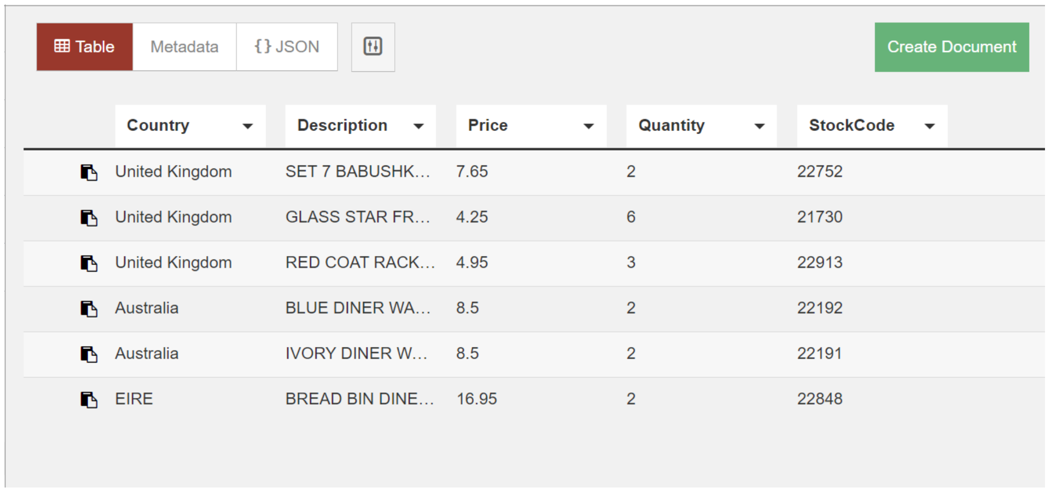
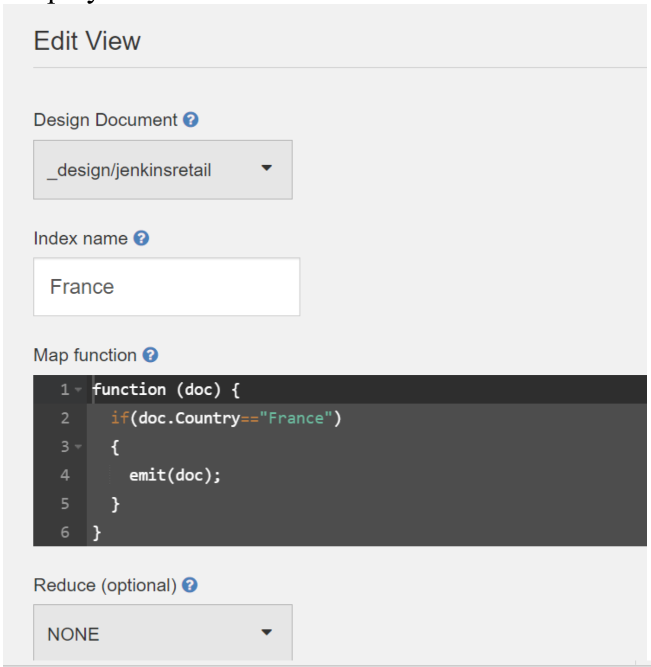


Table | Metadata | {} JSON | Create Document

Country	Description	Price	Quantity	StockCode
United Kingdom	SET 7 BABUSHK…	7.65	2	22752
United Kingdom	GLASS STAR FR…	4.25	6	21730
United Kingdom	RED COAT RACK…	4.95	3	22913
Australia	BLUE DINER WA…	8.5	2	22192
Australia	IVORY DINER W…	8.5	2	22191
EIRE	BREAD BIN DINE…	16.95	2	22848

c) Display the records from France.



Edit View

Design Document: _design/jenkinsretail

Index name: France

Map function

```
function (doc) {
  if(doc.Country=="France")
  {
    emit(doc);
  }
}
```

Reduce (optional): NONE

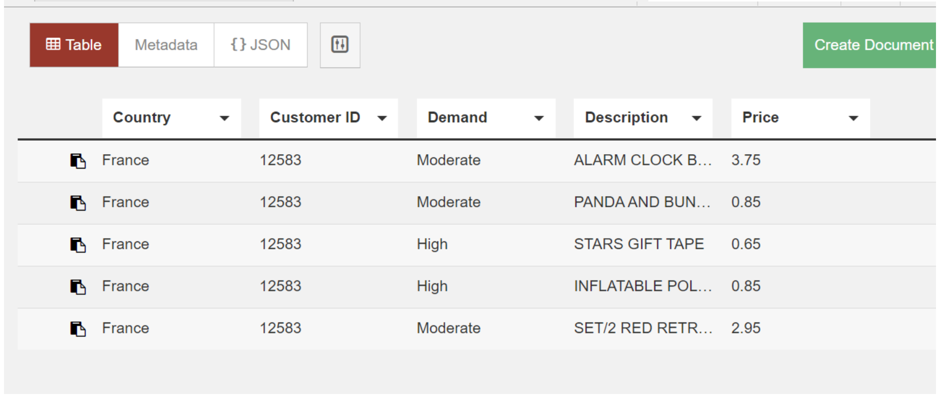
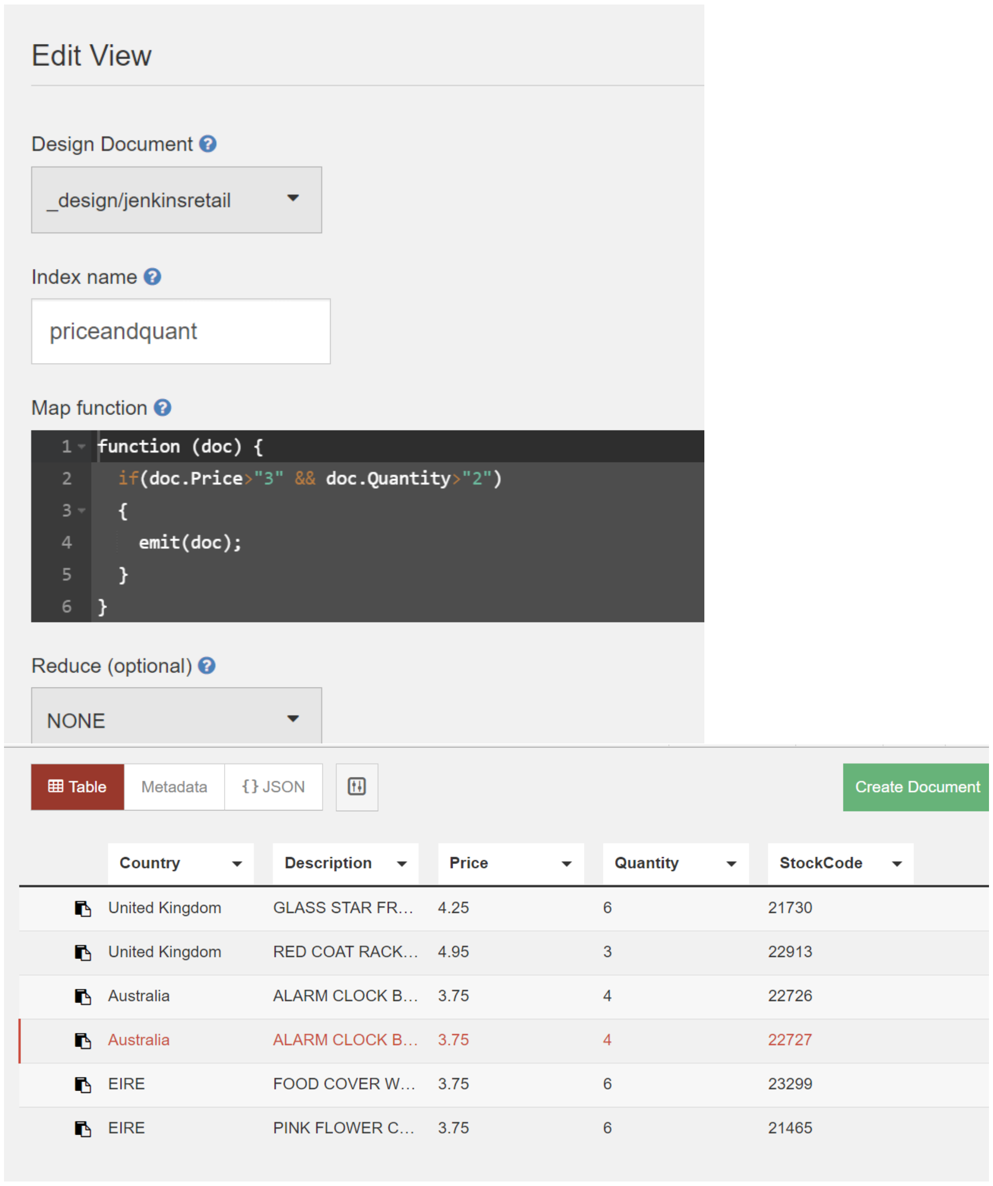


Table | Metadata | {} JSON | Create Document

Country	Customer ID	Demand	Description	Price
France	12583	Moderate	ALARM CLOCK B…	3.75
France	12583	Moderate	PANDA AND BUN…	0.85
France	12583	High	STARS GIFT TAPE	0.65
France	12583	High	INFLATABLE POL…	0.85
France	12583	Moderate	SET/2 RED RETR…	2.95

d) Create a new view to answer another question of your choosing.


Edit View

Design Document

_design/jenkinsretail

Index name

priceandquant

Map function

```
function (doc) {
  if(doc.Price>"3" && doc.Quantity>"2")
  {
    emit(doc);
  }
}
```

Reduce (optional)

NONE

Country	Description	Price	Quantity	StockCode
United Kingdom	GLASS STAR FR…	4.25	6	21730
United Kingdom	RED COAT RACK…	4.95	3	22913
Australia	ALARM CLOCK B…	3.75	4	22726
Australia	ALARM CLOCK B…	3.75	4	22727
EIRE	FOOD COVER W…	3.75	6	23299
EIRE	PINK FLOWER C…	3.75	6	21465

PART 2 (Lakehouses, Spark, Delta Lake, Databricks)

Instructions: The data is similar to the modified version of data from Lending Club used in a recent lab. Lending Club is a peer-to-peer Lending company based in the US. They match people looking to invest money with people looking to borrow money. When investors invest their money through Lending Club, this money is passed onto borrowers, and when borrowers pay their loans back, the capital plus the interest passes on back to the investors. It is a win for everybody as they can get typically lower loan rates and higher investor returns. Read more about the data here. In this project, you will create a lakehouse with two delta lake tables. The two data files are in two different formats, csv and parquet. For the rest of the section follow the instructions below.

Step 1: Set up a Databricks notebook. Load in the *loan-risks.snappy.parquet* file and the loan.csv file.

Step 2: Create a view for the parquet file. Now, you can read and explore the data as easily as any other table. Therefore, use the view to answer the following questions:

a) How many people paid off their debt?
b) How many people get the maximum loan and what is it?
c) How many lenders are from Indiana (IN)?
d) Write the code and answer ***three*** additional questions that you deem interesting.

Step 3: Create a view for the .csv file, which will allow you to read and explore the data as easily as any other table. Then, use the view to answer the following questions:

a) How many records exist in the created table?
b) How many loan applicants own a house?
c) Using 3a) and 3b) above, what percentage of applicants are homeowners?
d) What is the ratio between the newly-employed (emp_length less than 1 year) applicants versus the experienced employee (emp_length = 10+ years) applicants? (Here, you should divide number of experienced employees by the number of applicants that are newly-employed and round your answer to nearest whole number. For example, suppose you get a 3.7 after the division, then your ratio will be 1:4 (1 to 4), which means for every newly-employed applicant there are four applicants that are experienced employees).

Step 4: Publish your team's notebook and include the link here. Be sure all team members' names are included as a comment in the notebook.

Part 2.1 – ".parquet"
https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/7601106624631126/653598871334307/5698629569878374/latest.html

Part 2.2 – ".csv"
https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/7601106624631126/1697094075595148/5698629569878374/latest.html