# Bonify Backend Engineer Challenge Answers – Laraib Usman

## Answer 1:

Storage of Key/Values pairs with quick access seems like an ideal problem for nosql in database Redis which is also used as a cache. As redis is a high performance data structure server that can handle lots of calls per second and is ideal to be used in production systems.

In our IP blacklist monitoring system there best approach would be to create a set of “black\_listed\_ips” i.e. the key and to add the values i.e. ips to the set whenever an ip is blacklisted.

* 1. Data model:
* Key: black\_listed\_ips
* Value: Set of ips e.g. [192.168.121.108, 192.168.120.109 ]
  1. To add
* redis> SADD black\_listed\_ips 192.168.121.108
* redis> SADD black\_listed\_ips 192.168.120.109
  1. To remove
* redis> SREM black\_listed\_ips 192.168.120.109
  1. To check if a key is blacklisted
* redis> SISMEMBER black\_listed\_ips 192.168.121.108

A set in redis can have upto 232 entries i.e. greater than 4 billion, this should be sufficient storage to store the black listed ips.

## Answer 2:

Document classification into category based on count of words present belonging to each category can be easily solved using map reduce. The document can be split into parts to be received by each mapper for parallel execution. This can be done in two stages:

##### Counting occurrences of each key word

Simple modification of the conventional word count problem where instead of counting all the words, only key words belonging all the categories are counted.

###### Map:

* Input: PartId (document split in partitions), Content (text of the partition)
* Output: Word, 1 e.g. (keyboard, 1) (pants,1)

###### Reduce:

* Input: keyword, array of 1s indicating number of times the word was seen. e.g. (keyboard, [1,1])
* Output: keyword, sum of occurrences e.g. (keyboard, 2)

##### Aggregating output from stage 1 with respect to categories

The output from stage 1 would serve as an input to this stage where the keywords and counts would be classified into categories with their individual counts aggregated.

###### Map:

* Input: keyword, sum of occurrences e.g. (keyboard, 2) (processor, 5) (shirt, 1)
* Output: category, total of keyword occurrences e.g. (computer, 7) (clothes ,1 )

###### Reduce:

* Input: output from all the mappers e.g. (computers, 7) (clothes, 1) (computers, 2) (clothes 2)
* Output: aggregated results for each category e.g. (computers, 9) (clothes,3)

This would help us identify the most suitable category i.e. the primary category by finding the category with maximum count. And if needed would also help us get the secondary categories e.g. an article about “Suitable food for cats” can have primary category as “food” but secondary category as “animals”, both of which would be correct.

##### Diagram:



## Answer 3:

Assumption: Input was to be provided via command line argument.

###### Solution:

Controller created with endpoint “*/process”*  eliminating the need for if else statements to simply output the type received.

###### Steps:

1. Compile and run the project bonify\_sb using command:

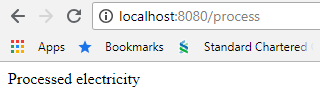
*mvn spring-boot:run -Dspring-boot.run.arguments=--type=electricity*

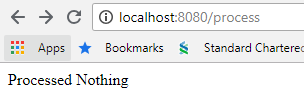
* Any type can be passed.

1. Default port for springboot tomcat is 8080. Go to the browser and hit the following url:

*http://localhost:8080/process*

1. If no type is passed and the application is run using command *“mvn spring-boot:run”*, default value i.e. *“Nothing”* is read from application.properties instead.





## Answer 4:

Assumption: No database setup was required, only two user’s Maria and Marcello were used from the input urls given.

###### Solution:

Controller with dynamic url created “*/users/{user\_name}/info/{info\_query}”* with two path variables i.e. user\_name and info\_query.

User class with predefined attributes created.

* Possible values for user\_name are Maria and Marcello
* Possible values for info\_query are birthday and location

Users are created and stored in a map with key as name and value being the user object.

Based on the info\_query received relevant user attribute is fetched and returned.

If a user\_name is not present then message *“User not found”* is returned

If the info\_query is anything other than the possible values then the message is returned *“Exception occurred: Incorrect information query”.*

###### Steps:

1. Compile and run the project bonify\_sb using command:

*mvn spring-boot:run*

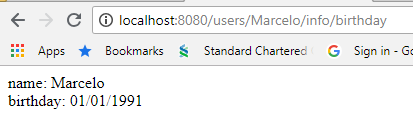
1. Default port for springboot tomcat is 8080. Go to the browser and hit the following urls:

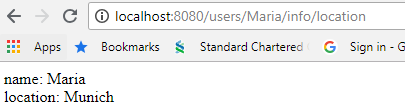
*http://localhost:8080/users/Marcelo/info/birthday*

*http://localhost:8080/users/Marcelo/info/location*

*http://localhost:8080/users/Maria/info/birthday*

*http://localhost:8080/users/Maria/info/location*





## Answer 5:

Assumption: No object relational mapping used for postgres as simple csv load and fetch by id was required.

###### Solution:

bankData.csv created with sample data present in the document and placed in application resources. Class PGCSVLoader.java created with the main method to call the csv load and find by ID method.

* Postgres DB used, csv loaded using COPY function
* Fetch by id performed by using given id

Unit Tests also added.

###### Steps:

1. Install Postgres SQL on local machine with following configurations.
   * Machine: localhost or 127.0.0.1
   * Port: 5432
   * Server: postgres
   * User: postrgres
   * Password: admin

These are read from application.properties so relevant database configurations can be changed there.

1. Create the table bank using the following sql

*CREATE TABLE public.bank*

*(*

*name character varying(25) NOT NULL,*

*bank\_identifier integer NOT NULL,*

*PRIMARY KEY (bank\_identifier)*

*)*

*WITH (*

*OIDS = FALSE*

*);*

*ALTER TABLE public.bank*

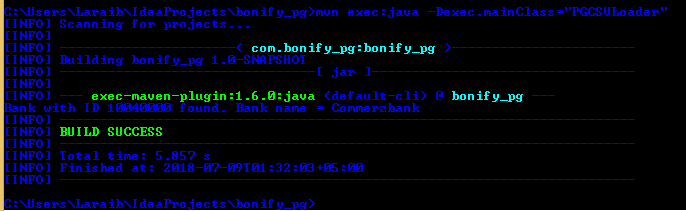
*OWNER to postgres;*

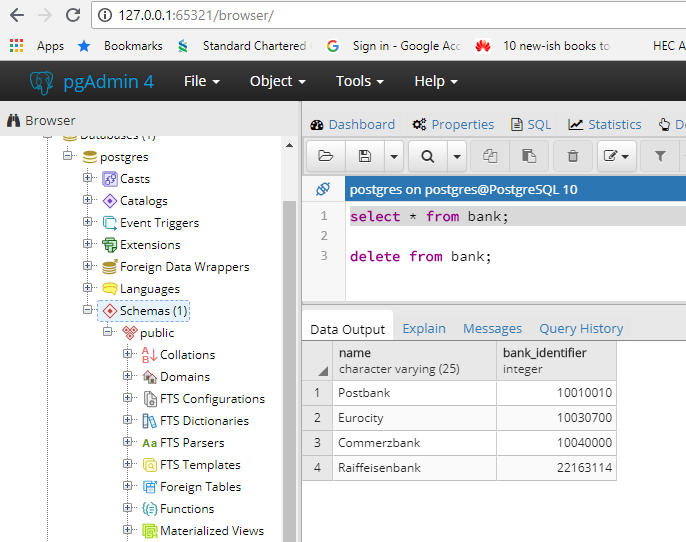
1. Compile project bonify\_pg using command:

*mvn clean install*

1. Run the main method of the csv loader class using command:

*mvn exec:java -Dexec.mainClass="PGCSVLoader"*





## Question 6 not attempted