**3.2) Realizar las siguientes consultas en MongoDB:**

**a) Seleccionar el id y el texto de 10 "documents"**

db.tweets.find({text:{$ne:null}}, {id:1, text:1}).limit(10)

**b) Seleccionar los lenguajes distintos de los tweets**

Object.keys(db.tweets.findOne()) (para ver las claves y asi saber que se llama "lang")

db.tweets.distinct("lang")

**c) Seleccionar el id, el nombre de usuario y la cantidad de followers (key “user.follower\_count") de aquellos usuarios que tengan más de 100 followers**

db.tweets.find({"user.followers\_count":{$gt:99}}, {id:1, "user.name": 1, "user.followers\_count":1})

**d) Seleccionar la cantidad de usuarios con más 100 followers**

db.tweets.find({"user.followers\_count":{$gt:99}}).count()

**e) Seleccionar el id, el nombre de usuario y la cantidad de followers (key "user.follower\_count") de los 10 usuarios con más followers ordenado de manera descendente**

db.tweets.find({"user.followers\_count":{$ne:null}}, {id:1, "user.name": 1, user.followers\_count":1}).sort({"user.followers\_count":-1}).limit(10)

**4.Map-Reduce**

**----------**

**4.2) Realizar las siguientes consultas con map-reduce:**

**a) En base al campo "source", determinar la cantidad de usuarios que hay por cada canal.**

var mapFunction1 = function() {emit(this.source,1);};

var reduceFunction1 = function(fuente, cantidad) {return Array.sum(cantidad);};

db.tweets.mapReduce(mapFunction1,reduceFunction1,{out:"map\_reduce\_example"}) SE GENERA UNA COLECCION LLAMADA MAP\_REDUCE\_EXAMPLE CON LA COLECCIÓN

db.map\_reduce\_example.find()

**b) Determinar la cantidad de tweets por cada lenguaje**

var mapFunction1 = function() {emit(this.lang,1);};

var reduceFunction1 = function(lenguaje, cantidad) {return Array.sum(cantidad);};

db.tweets.mapReduce(mapFunction1,reduceFunction1,{out:"map\_reduce\_example"})

db.map\_reduce\_example.find()

**c) Clasificar los textos de los tweets en corto (< 10 palabras), mediano (>=10, <20 palabras) y largo (>= 20 palabras). Devolver la cantidad de tweets en cada una de las clases.**

var mapFunction1 = function() { if (this.text != null) { palabras = this.text.split(' ').length; if (palabras < 10){ emit("corto",1);} else if (palabras >= 10 && palabras < 20){emit("mediano",1);} else emit("largo",1);}};

var reduceFunction1 = function(longitudPalabra, cantidad) {return Array.sum(cantidad);};

db.tweets.mapReduce(mapFunction1,reduceFunction1,{out:"map\_reduce\_example"})

**5.Python**

**------**

**5.2) Realizar en Python los siguientes programas:**

**a) Obtener los documentos de algún usuario con más de 1 documento asociado**

var mapFunction1 = function() {if (this.user != null) { emit(this.user.name,1);}};

var reduceFunction1 = function(nombreUsuario, cantidad) {return Array.sum(cantidad);};

db.tweets.mapReduce(mapFunction1,reduceFunction1,{out:"map\_reduce"})

db.map\_reduce.find({"value":{$gt:1}}).sort({"value":-1}).limit(10)

A esto se le suma el programa realizado en Python

**b) Obtener el nombre de usuario y las coordenadas geográficas de los tweets que proveen información geográfica. Obs. la especificación de la condición "not null" en Python es: {"geo": {"$ne":None}}**

Realizado directamente en Python

**c) Determinar la cantidad de tweets por cada lenguaje**

var mapFunction1 = function() {emit(this.lang,1);};

var reduceFunction1 = function(lenguaje, cantidad) {return Array.sum(cantidad);};

db.tweets.mapReduce(mapFunction1,reduceFunction1,{out:"map\_reduce"})

db.map\_reduce.find()

A esto se le suma el programa realizado en Python