**Resultados**

**¿Que resultados vamos a ofrecer?  
  
A. Estructura de datos y representación**

**<....>**

[A.1 Cassandra](https://github.com/orial/grupo_Spark/blob/master/Procesamiento/cassandra/readme.md#preprocesamiento-e-importaci%C3%B3n-de-datos)

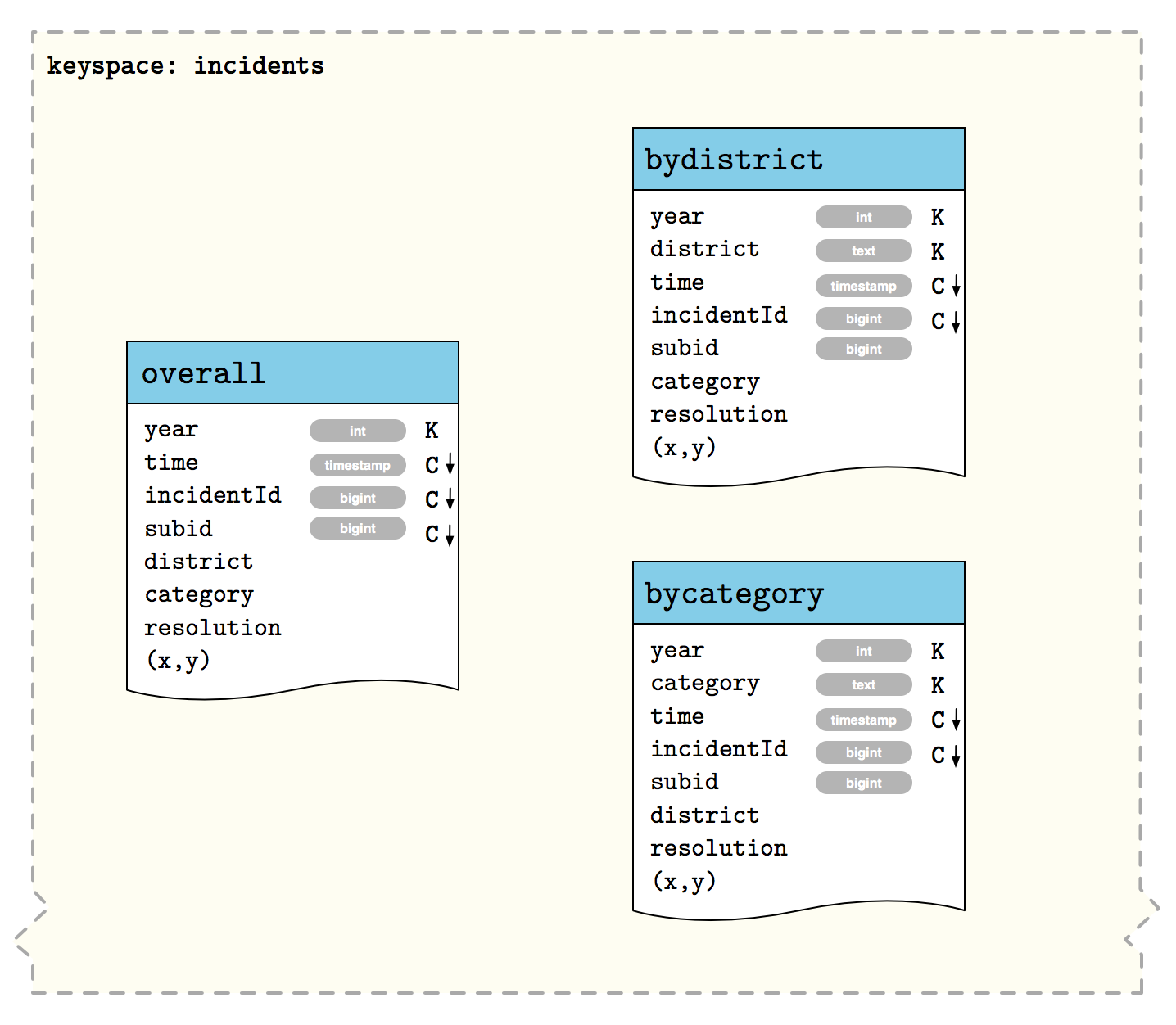
A.2. [Neo4j](https://github.com/orial/grupo_Spark/blob/master/Procesamiento/neo4j/readme.md#preprocesamiento-e-importaci%C3%B3n-de-datos)

A.3. [Mongodb](https://github.com/orial/grupo_Spark/blob/master/Procesamiento/mongodb/readme.md#preprocesamiento-e-importaci%C3%B3n-de-datos)

**A.1. Cassandra**

**<,,,>**

**Estructura de la información  
  
<explicar resumidamente la estructura>**

[](https://github.com/orial/grupo_Spark/blob/master/Procesamiento/docs/cassandra/model_tablas.png)

**Representación de la información: consultas CQL**

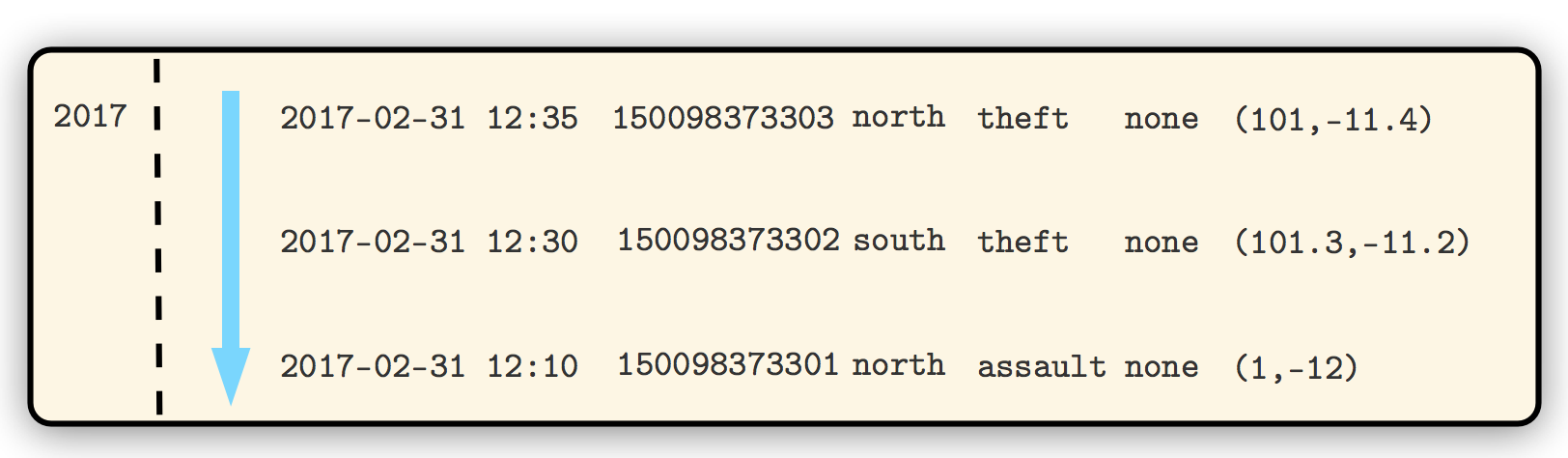
Todas las consultas se pueden encuentrar en el script CQL [cql/queries.cql](https://github.com/orial/grupo_Spark/blob/master/Procesamiento/cassandra/cql/queries.cql), pero para realizarlas es necesario crear el esquema de la estructura en la base de datos de Cassandra desde la consola a partir de los esquemas en [cql/schema.cql](https://github.com/orial/grupo_Spark/blob/master/Procesamiento/cassandra/cql/schema.cql).

1. Crear keyspace
2. Crear tablas e volcar los dataset mediante el comando COPY

### **Actividad criminal para un periodo de tiempo**

La actividad es ofrecida por la tabla: incidents.overall, con la estructura:

|  |  |
| --- | --- |
| Partition keys | year |
| Clustering keys | time,incidentid |

[](https://github.com/orial/grupo_Spark/blob/master/Procesamiento/docs/cassandra/diagrama_table1.png)

La información se encuentra particionada por una clave primaria compuesta de las claves: de partición año; pero al añadir el campo time como clave de clusterización podemos realizar una búsqueda por periodo. No se podría considerar una consulta muy eficiente ya que no se aprovecha las ventajas de particionamiento con respecto a la condición de búsqueda.

CREATE TABLE incidents.overall (

year int,

time timestamp,

...

PRIMARY KEY ((year), time, incidentId, subid)

) WITH CLUSTERING ORDER BY (time DESC, incident ASC, subId ASC);

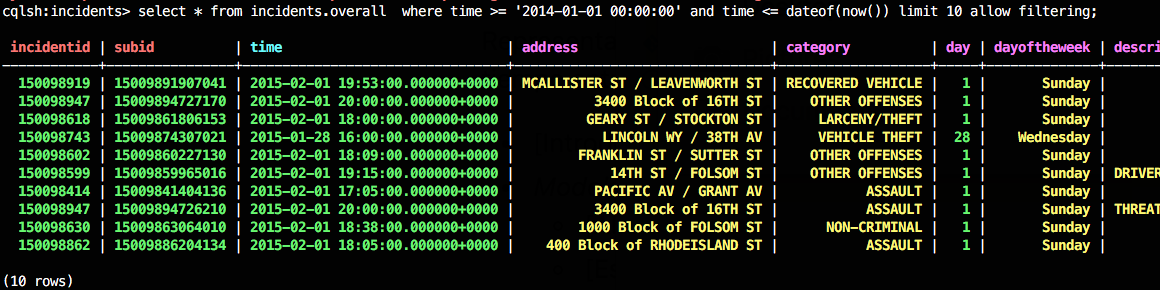
* Obtener toda las incidencias para un periodo de tiempo.

select \* from incidents.overall

where time >= '2014-01-01 00:00:00' and time <= dateof(now())

and yearh = 2017

allow filtering;

[](https://github.com/orial/grupo_Spark/blob/master/Procesamiento/docs/cassandra/queries/query_overall_periodtime.png)

* Actividad criminal por zona

Para esta sentencia si se realiza una partición de datos adecuada, con respecto a la zona y año: district:year.

La actividad es ofrecida por la tabla: incidents.overall, con la estructura:

|  |  |
| --- | --- |
| Partition keys | year, district |
| Clustering keys | time,... |

* Obtener información de incidencias por zonas (para un determinado año)

select district, year, incidentid, category, time, location

from incidents.bydistrict

where year = ?

[](https://github.com/orial/grupo_Spark/blob/master/Procesamiento/docs/cassandra/queries/query_bydistrict_incidents_year_2015.png)

Si queremos añadir condicion de periodo de tiempo, necesitamos añadir filtering:

select district, year, incidentid, category, time, location

from incidents.bydistrict

where year = 2015 and time >= '2015-02-01 00:00:00' and time <= dateof(now())

allow filtering;

* Actividad criminal por tipo de delito

Para esta sentencia si se realiza una partición de datos adecuada, con respecto al tipo de incidencia y año: category:year.

|  |  |
| --- | --- |
| Partition keys | year, category |
| Clustering keys | time,... |

* Obtener información de incidencias por categorias (para un determinado año)

select category, year, incidentid, category, time, location

from incidents.bycategory

where year = ?

[](https://github.com/orial/grupo_Spark/blob/master/Procesamiento/docs/cassandra/queries/query_bydistrict_incidents_year_2015.png)

* Nº incidencias agrupadas por año

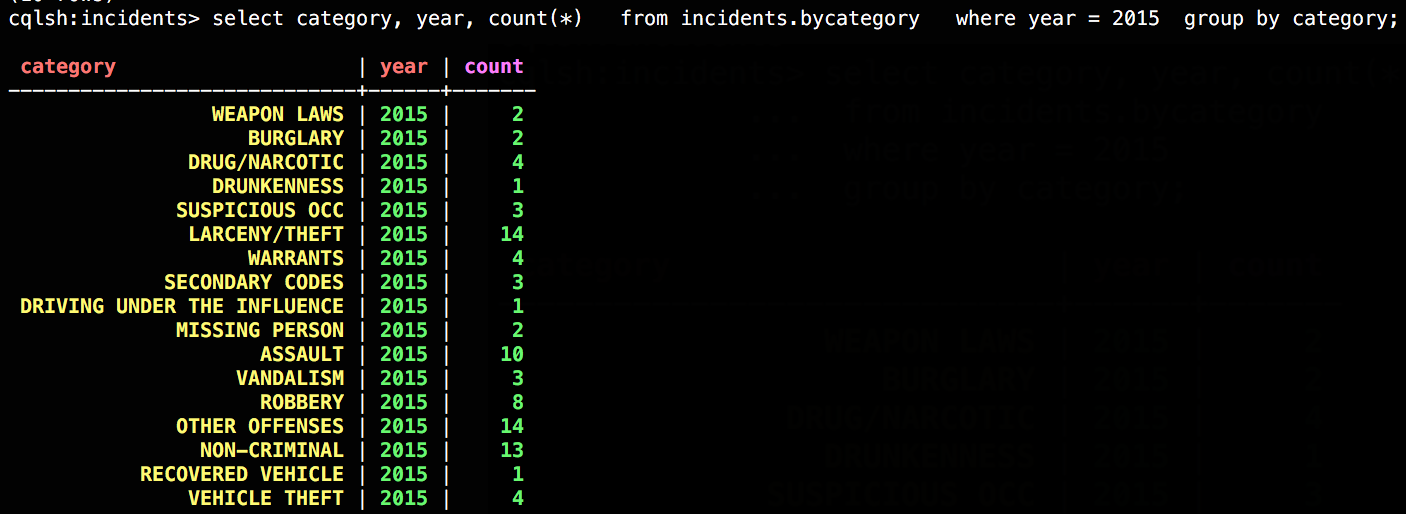
Para obtener número de incidencias producidas de cierta categoria (por un determinado año). Sin añadir filtro, más eficiente.

select category, year, count(\*)

from incidents.bycategory

where year = ?

group by category;

[](https://github.com/orial/grupo_Spark/blob/master/Procesamiento/docs/cassandra/queries/query_bycategory_groupby_withoutfilter2.png)

* Nº incidencias agrupadas por año

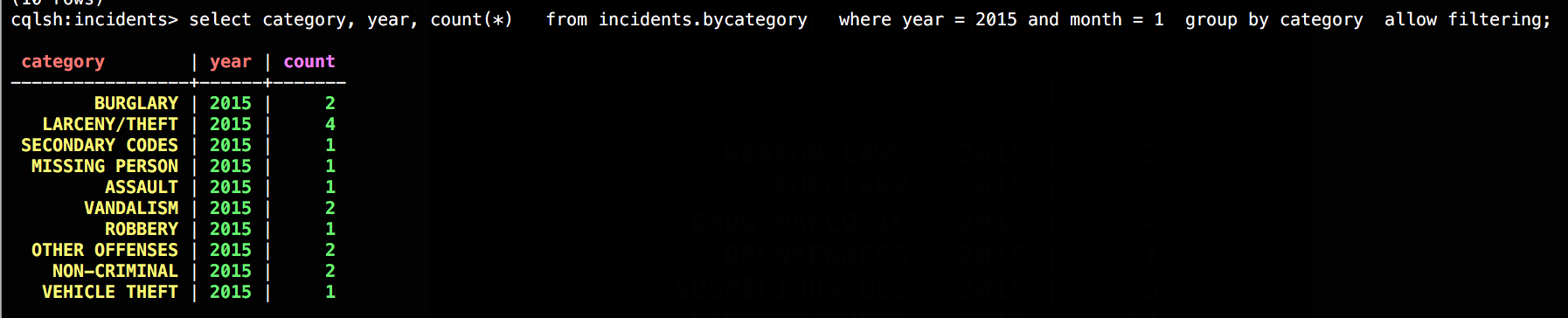
select category, year, count(\*)

from incidents.bycategory

where year = 2015 and month = 1

group by category

allow filtering;

[](https://github.com/orial/grupo_Spark/blob/master/Procesamiento/docs/cassandra/queries/query_bycategory_groupby_with_filter2.png)

**A.2. Neo4j**

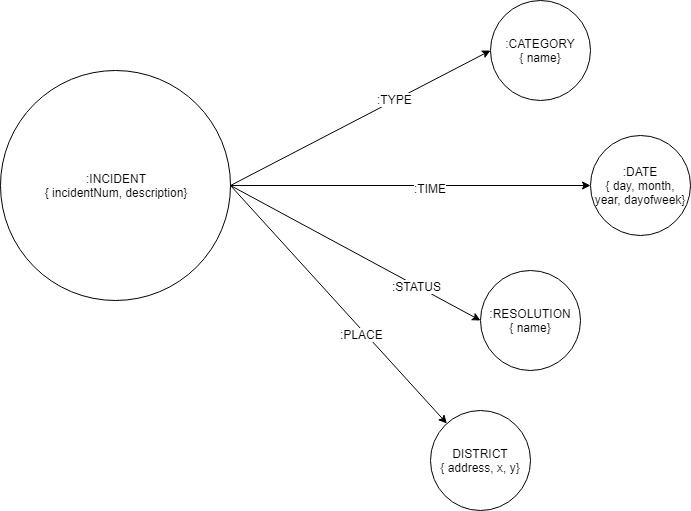
**Estructura de la información**

**<breve descripción de la estructura de la información, VENTAJAS Y DESVENTAJAS>**

Dada la estructura del archivo .CSV proporcionado, titulado con los siguientes **headers**:

IncidntNum, Category, Descript, DayOfWeek, Date, Time, PdDistrict, Resolution, Address, X, Y, Location.

Se ha decidido realizar este diseño de nodos y relaciones:

[](https://camo.githubusercontent.com/c0aa6d56d9091360794caf24a23066d3b616a9ce/68747470733a2f2f696d6167652e6962622e636f2f6e63724263782f4e6f646f732e6a7067)

Dado que, a excepción de los incidentes, un gran número de elementos estaban duplicados, se ha procedido a crear sus propios nodos y a evitar su inserción múltiple. Los datos que podían causar la aparición doble de un mismo valor, como un distrito con mismo nombre pero coordenadas X e Y diferentes a las de otro mismo distrito, se han añadido a las relaciones entre los nodos de incidentes y el resto. Esto aumenta la eficiencia y rendimiento final tanto de la inserción como de las búsquedas futuras.

Formato de fechas. Mientras que en el archivo sólo se guarda la fecha como cadena de caracteres en la columna Date, en el esquema aparecen tres campos: Day,Month y Year, los cuales serán obtenidos durante el apartado de preprocesamiento [aquí](https://github.com/orial/grupo_Spark/blob/master/Procesamiento/neo4j/readme.md#limpieza-y-preprocesamiento-de-dato).

**Consultas**

* Actividad criminal para un periodo de tiempo En este caso se obtendría el número de incidencias por día. **Count**es una función agregativa que reúne todas las entradas de un mismo tipo.

MATCH (n:INCIDENT)-[r:TIME]->(d:DATE) return d,count(n)

* Nº incidencias agrupadas por día

MATCH (n:INCIDENT)-->(s:DATE) return s,count(n); // Incidentes por día

* Nº incidencias agrupadas por año

MATCH (n:INCIDENT)-->(s:DATE) return s.year,count(n) // Incidentes por año

* Actividad criminal por zona

MATCH (n:INCIDENT)-[r:PLACE]->(d:DISTRICT) return distinct d,count(n)

**Actividad criminal por tipo de delito**

* Nº incidencias agrupadas por día

MATCH (c:CATEGORY)<--(n:INCIDENT)-->(s:DATE) return s.year,c.name,count(n) order by c.name // Incidentes y su tipo, por año

* Nº incidencias agrupadas por año

MATCH (c:CATEGORY)<--(n:INCIDENT)-->(s:DATE) return s,c.name,count(n) order by c.name // Incidentes y su tipo, por día

* Actividad criminal por dia de la semana

MATCH (n:INCIDENT)-->(s:DATE) return s.dayofweek,s.year,count(n) order by s.dayofweek,s.year // Totales de incidentes los días de la semana de cada año

**A.3. Mongodb**

**Estructura de la información**

**....no hay nada....**

**Consultas**

Cuando accedemos a los datos de los campos de un documento en MongoDB podemos encontrarnos la necesidad de realizar consultas MongoDB like. Es decir, realizar consultas por cadenas similares de texto. Por ejemplo, nombres que empiecen por una letra o letras, palabras que contengan una cierta cadena de caracteres,...

Las consultas MongoDB like se resuelven mediante expresiones regulares. Lo que realizaremos mediante la siguinte sintaxis:

db.coleccion.find({campo:expresión\_regular});

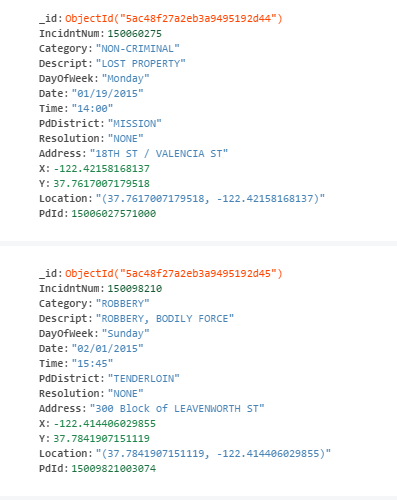
Para los patrones de las expresiones regulares MongoDB utiliza “Perl Compatible Regular Expressions” (PCRE).

**Consultas comunes**

**Actividad criminal para un periodo de tiempo.** En este caso se obtendría el *número de incidencias* por día. **Count** es una función agregativa que reúne todas las entradas de un mismo tipo.

* Incidencias en dia Sunday

db.find({DayOfWeek:"Sunday"}).pretty()

[](https://github.com/orial/grupo_Spark/blob/master/Procesamiento/docs/sunday.PNG)

* Número de incidencias en Sunday

db.incidents.count({DayOfWeek:"Sunday"})

290936

* Incidencias en el dia 02/01/2015

db.find({Date:"02/01/2015"}).pretty()

[](https://github.com/orial/grupo_Spark/blob/master/Procesamiento/docs/02012015.PNG)

* Número de incidencias el 02/01/2015

db.incidents.count({Date:"02/01/2015"})

466

* Incidencias en la Zona de LEAVENWORTH

db.incidents.find({Address:/LEAVENWORTH/}).pretty()

* Número de incidencias en LEAVENWORTH

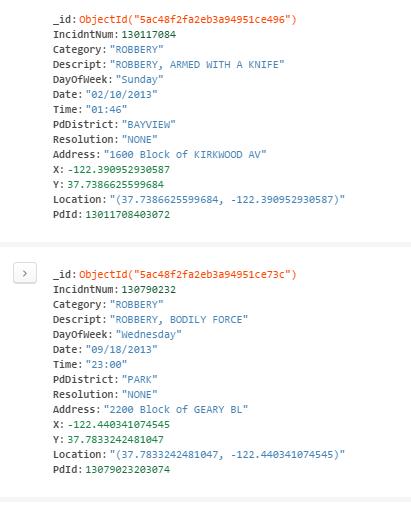
db.incidents.count({Address:/LEAVENWORTH/})

28163

**Actividad criminal por tipo de delito, por año/dia (por tipo de delito)**

Incidencias en el año 2013 para los delitos ROBBERY

db.incidents.find({Date:/2013/,Category:"ROBBERY"}).pretty()

[](https://github.com/orial/grupo_Spark/blob/master/Procesamiento/docs/2013ROBBERY.PNG)

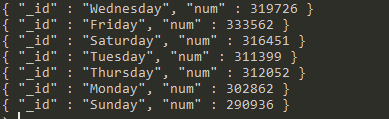
* + Número de incidencias de la busqueda anterior

db.incidents.count({Date:/2013/,Category:"ROBBERY"})

4196

* Actividad criminal por dia de la semana

db.incidents.aggregate([{$group:{day:"$DayOfWeek",num:{$sum:1}}}])

[](https://github.com/orial/grupo_Spark/blob/master/Procesamiento/docs/dias.png)