

Actividad 2. Laboratorio Actividad de Proyecto (Grupal): Uso de MongoDB.

BASES DE DATOS PARA EL BIG DATA

ALEJANDRO MOYANO GONZALEZ

PEDRO DURÁN PORRAS

JORGE SAN ROMAN CAPARROSO

NICOLAS LOPEZ ESTRUEL

Asignatura	Datos del alumno	Fecha
Bases de datos para el Big Data		2025-01-15

ÍNDICE

1	Actividad 2.	
	Laboratorio Actividad de Proyecto (Grupal): Uso de MongoDB.	5
1.1	Presentación.....	5
1.2	Reto de la actividad.....	5
1.2.1	Esquema de datos NoSQL.....	5
1.2.1.1	Dataset1:.....	5
1.2.1.2	Dataset2:.....	5
1.2.1.3	Dataset3:.....	6
1.2.2	Carga de datos.	6
1.2.3	Explora las colecciones.....	7
1.2.4	Consulta la colección 1.	10
1.2.5	Consulta la colección 2.	12
1.3	Conclusión.....	15

Asignatura	Datos del alumno	Fecha
Bases de datos para el Big Data		2025-01-15

Asignatura	Datos del alumno	Fecha
Bases de datos para el Big Data		2025-01-15

1.1 Presentación

Este trabajo tiene como finalidad manejar los principales comandos de MongoDB, así como trabajar con bases de datos NoSQL. Para ello, trabajaremos con el dataset de la primera actividad, para los que le hemos hecho la modificación de añadir una variable llamada `id_df` que indica el dataset del que proviene cada documento. A continuación, describiremos más en

1 Actividad 2. Laboratorio Actividad de Proyecto (Grupal): Uso de MongoDB.

profundidad los subdatasets que contiene:

1.2 Reto de la actividad

1.2.1 Esquema de datos NoSQL

1.2.1.1 Dataset1:

El primer dataset consta de 13 variables relacionadas con el crimen en una ciudad de Estados Unidos. Cada documento contiene toda la información necesaria sobre un crimen sin necesidad de referencias externas. Este primer dataset consta con un total de 10.051 documentos.

Campos organizados

- **dates:** Agrupa todos los datos relacionados con fechas y horas, facilitando consultas temporales
- **location:** Agrupa datos relacionados con la ubicación, como dirección y tipo de dirección.

Tipos de datos

- **_id:** Identificador único generado automáticamente
- **crime_id:** Identificador del crimen (tipo entero).
- **crime_type:** Tipo de crimen, puede ser texto descriptivo o código de referencia de crimen
- **offense_date:** Fecha y hora del crimen en formato ISO
- **call_time:** Hora de la llamada en formato de cadena
- **call_datetime:** Fecha y hora completas en formato ISO
- **disposition:** Resultado o disposición del caso
- **adres:** Dirección completa del lugar del crimen
- **city:** Ciudad donde ocurrió el crimen
- **state:** Estado donde ocurrió el crimen
- **adres_type:** Tipo de dirección
- **agency_id** Identificador de la agencia responsable

1.2.1.2 Dataset2:

El segundo dataset consta de 14 variables relacionadas con alojamientos turísticos en Madrid. Este dataset incluye 29.975 documentos.

Campos organizados

- **location:** Agrupa todos los datos relacionados con la dirección del alojamiento.

Asignatura	Datos del alumno	Fecha
Bases de datos para el Big Data		2025-01-25

- **details:** Agrupa información específica sobre el tipo, categoría y denominación del alojamiento

Tipos de datos detallados

- **_id:** Identificador único generado automáticamente
- **alojamiento_tipo:** Tipo de alojamiento
- **categoria:** Categoría del alojamiento
- **denominación:** Denominación o nombre del alojamiento
- **via_tipo:** Tipo de vía
- **via_nombre:** Nombre de la vía donde se encuentra el alojamiento
- **numero:** Número de la dirección del alojamiento
- **bloque:** Bloque del edificio, si aplica
- **portal:** Portal del edificio, si aplica
- **escalera:** Escalera del edificio, si aplica
- **planta:** Planta del edificio, si aplica
- **puerta:** Puerta específica, si aplica
- **cdpostal:** Código postal del alojamiento
- **localidad:** Ciudad o localidad del alojamiento
- **signatura:** Identificador oficial del alojamiento

1.2.1.3 Dataset3:

El tercer dataset de esta actividad consta de 10 variables relacionadas con el alumbrado de Navidad de una ciudad concreto. Este tercer dataset está formado por 245 documentos.

Campos organizados

- **Location:** Agrupa los datos relacionados con la posición geográfica y las zonas (distritos y barrios) donde se instala el alumbrado.
- **Details:** Agrupa información específica sobre el tipo, motivo, y diseño del alumbrado.

Tipos de datos detallados

- **_id:** Identificador único generado automáticamente por MongoDB
- **posicion:** Posición o ubicación de la instalación
- **distrito:** Distrito donde se encuentra el alumbrado público
- **barrio:** Barrio donde se encuentra el alumbrado público
- **motivo:** Motivo decorativo del alumbrado público
- **comentario:** Comentarios adicionales sobre el diseño
- **tipo:** Tipo de instalación del alumbrado público
- **diseño:** Diseño del alumbrado público, si aplica
- **texto:** Descripción textual del motivo
- **promotor:** Entidad promotora de la instalación
- **novedad:** Indica si se trata de una instalación novedosa
- **idelem:** Identificador único del elemento
- **longitud:** Longitud de la instalación en metros
- **n_elementos:** Número de elementos instalados

1.2.2 Carga de datos.

```
mongoimport --verbose --db Proyecto --collection dataset1 --jsonArray
C:\Users\nikol\Documents\02\MOYANO_GONZALEZ_ALEJANDRO_actividad_1_dataset1.json
```

Asignatura	Datos del alumno	Fecha
Bases de datos para el Big Data		2025-01-15

```
mongoimport --verbose --db Proyecto --collection dataset2 --jsonArray
C:\Users\nikol\Documents\MOYANO_GONZALEZ_ALEJANDRO_actividad_1_dataset2.json
```

```
mongoimport --verbose --db Proyecto --collection dataset3 --jsonArray
C:\Users\nikol\Documents\MOYANO_GONZALEZ_ALEJANDRO_actividad_1_dataset3.json
```

```
PS C:\MongoDB\bin> mongoimport --verbose --db Proyecto --collection dataset1 --jsonArray C:\Users\nikol\Documents\MOYANO_GONZALEZ_ALEJANDRO_actividad_1_dataset1.json
2025-01-16T12:30:30.274+0100 using write concern: &{majority <nil> 0s}
2025-01-16T12:30:30.297+0100 filesize: 3606267 bytes
2025-01-16T12:30:30.297+0100 using fields:
2025-01-16T12:30:30.298+0100 connected to: mongodb://localhost/
2025-01-16T12:30:30.298+0100 ns: Proyecto.dataset1
2025-01-16T12:30:30.299+0100 connected to node type: standalone
2025-01-16T12:30:30.615+0100 10051 document(s) imported successfully. 0 document(s) failed to import.
```

Captura 1. Carga de datos para el “dataset1”.

```
PS C:\MongoDB\bin> mongoimport --verbose --db Proyecto --collection dataset2 --jsonArray C:\Users\nikol\Documents\MOYANO_GONZALEZ_ALEJANDRO_actividad_1_dataset2.json
2025-01-16T12:31:05.580+0100 using write concern: &{majority <nil> 0s}
2025-01-16T12:31:05.600+0100 filesize: 5278118 bytes
2025-01-16T12:31:05.600+0100 using fields:
2025-01-16T12:31:05.600+0100 connected to: mongodb://localhost/
2025-01-16T12:31:05.601+0100 ns: Proyecto.dataset2
2025-01-16T12:31:05.601+0100 connected to node type: standalone
2025-01-16T12:31:05.950+0100 13865 document(s) imported successfully. 0 document(s) failed to import.
```

Captura 2. Carga de datos para el “dataset2”.

```
PS C:\MongoDB\bin> mongoimport --verbose --db Proyecto --collection dataset3 --jsonArray C:\Users\nikol\Documents\MOYANO_GONZALEZ_ALEJANDRO_actividad_1_dataset3.json
2025-01-16T12:31:28.648+0100 using write concern: &{majority <nil> 0s}
2025-01-16T12:31:28.676+0100 filesize: 115335 bytes
2025-01-16T12:31:28.676+0100 using fields:
2025-01-16T12:31:28.676+0100 connected to: mongodb://localhost/
2025-01-16T12:31:28.676+0100 ns: Proyecto.dataset3
2025-01-16T12:31:28.677+0100 connected to node type: standalone
2025-01-16T12:31:28.704+0100 245 document(s) imported successfully. 0 document(s) failed to import.
```

Captura 3. Carga de datos para el “dataset3”.

1.2.3 Explora las colecciones.

Para la exploración, hemos dividido el dataset en tres colecciones, formadas cada una por los documentos que pertenecen a **id_df = '1'**, **id_df = '2'** e **id_df = '3'**. Además, hemos utilizado el comando `db.<collection>.findOne()`. Con ello, mostramos un documento de cada colección:

```
db.dataset1.findOne() db.dataset2.findOne() db.dataset3.findOne()
```

```
> db.dataset1.findOne()
< {
  _id: ObjectId('678691e2e3f7b041e9fdd01'),
  CrimeId: 160903280,
  OriginalCrimeTypeName: 'Assault / Battery',
  OffenseDate: '2016-03-30T00:00:00',
  CallTime: '18:42',
  CallDateTime: '2016-03-30T18:42:00',
  Disposition: 'REP',
  Address: '100 Block Of Chilton Av',
  City: 'San Francisco',
  State: 'CA',
  AgencyId: 1,
  AddressType: 'Premise Address',
  id_df: '1'
}
```

```
> db.dataset2.findOne()
< {
  _id: ObjectId('67869227e3f7b041e9fe063d'),
  alojamiento_tipo: 'HOTEL',
  categoria: '3-HOTEL',
  denominacion: 'GRAN LEGAZPI',
  via_tipo: 'PASEO',
  via_nombre: 'de la Chopera',
  numero: 71,
  bloque: '',
  portal: '',
  escalera: '',
  planta: '',
  puerta: '',
  cdpostal: 28045,
  localidad: 'Madrid',
  signature: 'HM-127',
  id_df: '2'
}
```

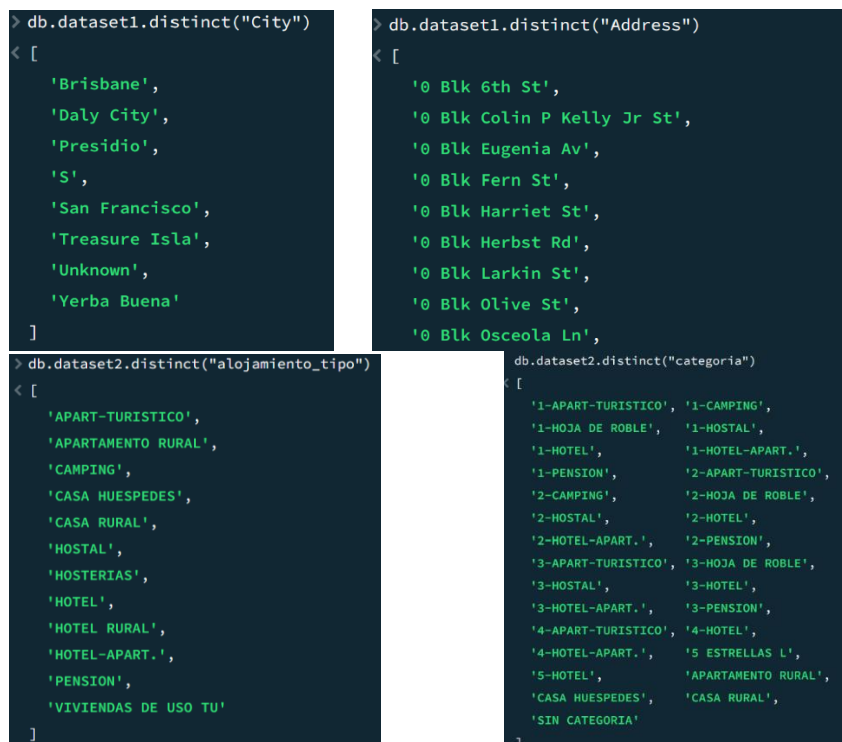
Captura 4. Exploración de los “dataset1” y “dataset2”.

- Identifica 2 campos categóricos en cada fichero y muestra todos los valores distintos de dichos campos.

Asignatura	Datos del alumno	Fecha
Bases de datos para el Big Data		2025-01-25

Para extraer 2 campos categóricos en cada fichero, utilizaremos el comando `db.<collection>.distinct(Campo)`, con el que podemos identificar en la salida si el campo es de tipo categórico:

```
db.dataset1.distinct("City")
db.dataset1.distinct("OriginalCrimeTypeName")
db.dataset2.distinct("alojamiento_tipo")
db.dataset2.distinct("categoria")
db.dataset3.distinct("MOTIVO")
db.dataset3.distinct("DISEÑO")
```



```
> db.dataset1.distinct("City")
< [
  'Brisbane',
  'Daly City',
  'Presidio',
  'S',
  'San Francisco',
  'Treasure Isla',
  'Unknown',
  'Verba Buena'
]

> db.dataset1.distinct("Address")
< [
  '0 Blk 6th St',
  '0 Blk Colin P Kelly Jr St',
  '0 Blk Eugenia Av',
  '0 Blk Fern St',
  '0 Blk Harriet St',
  '0 Blk Herbst Rd',
  '0 Blk Larkin St',
  '0 Blk Olive St',
  '0 Blk Osceola Ln',
]

> db.dataset2.distinct("alojamiento_tipo")
< [
  'APART-TURISTICO',
  'APARTAMENTO RURAL',
  'CAMPING',
  'CASA HUESPEDES',
  'CASA RURAL',
  'HOSTAL',
  'HOSTERIAS',
  'HOTEL',
  'HOTEL RURAL',
  'HOTEL-APART.',
  'PENSION',
  'VIVIENDAS DE USO TU'
]

db.dataset2.distinct("categoria")
< [
  '1-APART-TURISTICO', '1-CAMPING',
  '1-HOJA DE ROBLE', '1-HOSTAL',
  '1-HOTEL', '1-HOTEL-APART.',
  '1-PENSION', '2-APART-TURISTICO',
  '2-CAMPING', '2-HOJA DE ROBLE',
  '2-HOSTAL', '2-HOTEL',
  '2-HOTEL-APART.', '2-PENSION',
  '3-APART-TURISTICO', '3-HOJA DE ROBLE',
  '3-HOSTAL', '3-HOTEL',
  '3-HOTEL-APART.', '3-PENSION',
  '4-APART-TURISTICO', '4-HOTEL',
  '4-HOTEL-APART.', '5 ESTRELLAS L',
  '5-HOTEL', 'APARTAMENTO RURAL',
  'CASA HUESPEDES', 'CASA RURAL',
  'SIN CATEGORIA'
]
```

Captura 5. Salida por MongoDB Shell de la colección 1 y 2.

- b) Sobre la colección más poblada lanza la siguiente consulta reemplazando los campos que consideres más adecuados: `db.getCollection('COLECCION').find({CAMPO1: {$gte: "A", $lt: "B"}}, {CAMPO2: 1, CAMPO3: 1 })`

Sobre la colección 2 utilizamos la consulta propuesta, ya que es la que tiene mayor número de documentos (posteriormente, en el apartado 4 comprobaremos el número de documentos que la colección tiene a través de una consulta en MongoDB). Hemos utilizado la siguiente instrucción que aparece en la captura de pantalla, filtrando la variable **código postal** y mostrando en pantalla **denominación** y **localidad**

```
db.getCollection('dataset2').find({ cdpostal: { $gte: 28000, $lt: 29000 } }, { denominacion: 1, localidad: 1, _id: 0 })
```


Asignatura	Datos del alumno	Fecha
Bases de datos para el Big Data		2025-01-15

```
> db.getCollection('dataset2').find(
  { cdpostal: { $gte: 28000, $lt: 29000 } }, // Rango de códigos postales
  { denominacion: 1, localidad: 1, _id: 0 } // Proyección: Muestra "denominacion" y "localidad"
)
< {
  denominacion: 'GRAN MELIA FENIX',
  localidad: 'Madrid'
}
{
  denominacion: 'BLESS HOTEL MADRID',
  localidad: 'Madrid'
}
```

Captura 6. Salida por MongoDB Shell de la colección 2.

- c) Utiliza la consulta anterior para recuperar aquellos documentos que, adicionalmente, cumplan dos condiciones más donde utilices el operador \$type y \$exists. Devuelve cuatro campos que elijas a excepción de “_id”.

Hemos utilizado para ello la siguiente instrucción que aparece en pantalla, tal y como se explica en la captura. Las variables que se muestran son **categoría**, **denominación**, **cdpostal** y **localidad**

```
db.getCollection('dataset2').find({cdpostal:{ $gte: 28000, $lt: 29000},
denominacion: { $exists: true },categoria: { $type: "string" } }, {
denominacion: 1, localidad: 1, categoria: 1, cdpostal: 1, _id: 0 })
```

```
> db.getCollection('dataset2').find(
  {
    cdpostal: { $gte: 28000, $lt: 29000 }, // Condición 1: Rango de códigos postales
    denominacion: { $exists: true }, // Condición 2: Campo denominacion debe existir
    categoria: { $type: "string" } // Condición 3: Campo categoria debe ser de tipo string
  },
  {
    denominacion: 1, // Campos a devolver
    localidad: 1,
    categoria: 1,
    cdpostal: 1,
    _id: 0 // Excluir el campo _id
  }
)
< {
  categoria: '5-HOTEL',
  denominacion: 'GRAN MELIA FENIX',
  cdpostal: 28001,
  localidad: 'Madrid'
}
{
  categoria: '5-HOTEL',
  denominacion: 'BLESS HOTEL MADRID',
  cdpostal: 28001,
  localidad: 'Madrid'
}
```

Captura 7. Salida por MongoDB Shell de la colección 2.

- d) Describe brevemente qué ocurre si a la consulta del punto anterior, le añades al final la siguiente instrucción: **.toArray()**

Lo que ocurre al añadirlo es que el resultado de la consulta se convierte en un array de documentos que es devuelto completamente en memoria.

```
db.getCollection('dataset2').find( {cdpostal:{ $gte:28000,$lt:
29000},denominacion:{ $exists: true },categoria: { $type: "string" } },
denominacion:1,localidad:1,categoria:1,cdpostal: 1, _id: 0 }).toArray()
```

Asignatura	Datos del alumno	Fecha
Bases de datos para el Big Data		2025-01-25

```

> db.getCollection('dataset2').find(
  {
    cdpostal: { $gte: 28000, $lt: 29000 },
    denominacion: { $exists: true },
    categoria: { $type: "string" }
  },
  {
    denominacion: 1,
    localidad: 1,
    categoria: 1,
    cdpostal: 1,
    _id: 0
  }
).toArray()
< [
  {
    categoria: '5-HOTEL',
    denominacion: 'GRAN MELIA FENIX',
    cdpostal: 28001,
    localidad: 'Madrid'
  },
  {
    categoria: '5-HOTEL',
    denominacion: 'BLESS HOTEL MADRID',
    cdpostal: 28001,
    localidad: 'Madrid'
  },
]

```

Captura 8. Salida por MongoDB Shell de la colección 2.

- e) ¿Qué ocurre si además le añades la siguiente instrucción: `.forEach(function(valor, indice, array){print("CAMPO2: " + valor.CAMPO2 + "CAMPO4: " + valor.CAMPO4[o] + ": " + valor.CAMPO4[1] + " Registro No. " + indice);})`. Si reemplazas los campos por aquellos que se ajusten a la consulta, ¿Para qué crees que sería útil esto último?

Si a la consulta le añades la instrucción `.forEach(function(valor, indice, array) { ... })`, lo que sucede es que procesas cada documento del resultado de forma individual y puedes aplicar una función personalizada a cada uno. Este enfoque es útil porque permite trabajar directamente con los documentos recuperados, procesándolos de forma personalizada. Es ideal para generar informes con datos formateados específicamente según las necesidades del análisis, facilitando la presentación de resultados.

Además, ofrece la posibilidad de realizar cálculos adicionales, generar estadísticas o validar la información mientras se recorren los documentos. También resulta práctico para la depuración, ya que permite inspeccionar cada registro y asegurarse de que los valores sean los esperados, ayudando a identificar posibles errores o inconsistencias en los datos.

```

db.getCollection('dataset2').find({cdpostal: { $gte: 28000, $lt: 29000 },
denominacion: { $exists: true },categoria: { $type: "string" }},{
denominacion: 1, localidad: 1, categoria: 1, cdpostal: 1, _id: 0
}).forEach(function(valor, indice, array) {print("Denominación:" +
valor.denominacion + ", Categoría: " + valor.categoria + ", Código Postal: "
+ valor.cdpostal + " (Registro No. " + indice + ")");});

```

```

Denominación: MARVAEZ, 58, Categoría: SIN CATEGORIA, Código Postal: 28009 (Registro No. undefined)
Denominación: LOPE DE VEGA 4 PAX, Categoría: SIN CATEGORIA, Código Postal: 28014 (Registro No. undefined)
Denominación: HERRADORES, 8, Categoría: SIN CATEGORIA, Código Postal: 28013 (Registro No. undefined)
Denominación: AYALA, 81, Categoría: SIN CATEGORIA, Código Postal: 28086 (Registro No. undefined)
Denominación: MOGUER 5, Categoría: SIN CATEGORIA, Código Postal: 28040 (Registro No. undefined)
Denominación: YOGI SONGS, Categoría: SIN CATEGORIA, Código Postal: 28013 (Registro No. undefined)
Denominación: DOCTOR GARCIA TAPIA, 62, Categoría: SIN CATEGORIA, Código Postal: 28030 (Registro No. undefined)
Denominación: POVEDILLA, 6, Categoría: SIN CATEGORIA, Código Postal: 28009 (Registro No. undefined)
Denominación: GYL MADRID, Categoría: SIN CATEGORIA, Código Postal: 28013 (Registro No. undefined)
Denominación: CARRETAS, 25. 2 H, Categoría: SIN CATEGORIA, Código Postal: 28012 (Registro No. undefined)
Denominación: THE NICE GREEN HOUSE, Categoría: SIN CATEGORIA, Código Postal: 28007 (Registro No. undefined)
Denominación: COMADRE 4, Categoría: SIN CATEGORIA, Código Postal: 28012 (Registro No. undefined)
Denominación: PURO MADRID, Categoría: SIN CATEGORIA, Código Postal: 28005 (Registro No. undefined)
Denominación: ESCALINATA 1, Categoría: SIN CATEGORIA, Código Postal: 28012 (Registro No. undefined)
Denominación: RUDA, 8, Categoría: SIN CATEGORIA, Código Postal: 28005 (Registro No. undefined)

```

Captura 9. Salida por MongoDB Shell de la colección 2.

1.2.4 Consulta la colección 1.

- a) ¿Cuál es el tamaño de la colección (en bytes)?

El tamaño de la colección es de 3243486 bytes. `db.dataset1.dataSize()`

```

> db.dataset1.dataSize()
< 3243486

```

Captura 10. Salida por MongoDB Shell de la colección 1.

Asignatura	Datos del alumno	Fecha
Bases de datos para el Big Data		2025-01-15

- b) ¿Cuántos documentos tiene la colección?

La colección tiene 10051 documentos. `db.dataset1.countDocuments()`

```
> db.dataset1.countDocuments()
< 10051
```

Captura 11. Salida por MongoDB Shell de la colección 1.

- c) Diseña y ejecuta una consulta donde utilices el operados `$lt` y `$gt`

La consulta que realizamos filtra en la variable **CrimeId** y muestra los documentos que pasan el filtro, mostrando el campo junto el campo **OriginalCrimeTypeName**.

```
Db.dataset1.find ({ CrimeId: { $gt: 160900000, $lt: 160910000 } },{CrimeId: 1, OriginalCrimeTypeName: 1, _id: 0 })
```

```
db.dataset1.find(
  { CrimeId: { $gt: 160900000, $lt: 160910000 } },
  { CrimeId: 1, OriginalCrimeTypeName: 1, _id: 0 }
)
{
  CrimeId: 160903280,
  OriginalCrimeTypeName: 'Assault / Battery'
}
```

Captura 12. Salida por MongoDB Shell de la colección 1.

- d) Diseña y ejecuta una consulta donde utilices el operados `$AND`, `$lte` y `$gte`

La consulta es análoga al apartado anterior, mostrando además **City**.

```
db.dataset1.find({$and:[{ CrimeId: { $gte: 160900000 }},{ CrimeId: { $lte: 160910000}}]}, { CrimeId: 1, OriginalCrimeTypeName: 1, City:1, _id: 0 })
```

```
db.dataset1.find(
  {
    $and: [
      { CrimeId: { $gte: 160900000 } },
      { CrimeId: { $lte: 160910000 } }
    ]
  },
  { CrimeId: 1, OriginalCrimeTypeName: 1, City: 1, _id: 0 }
)
{
  CrimeId: 160903280,
  OriginalCrimeTypeName: 'Assault / Battery',
  City: 'San Francisco'
}
```

Captura 13. Salida por MongoDB Shell de la colección 1.

- e) Diseña y ejecuta una consulta con algún tipo de agregación donde calcules un valor máximo, un valor mínimo y un valor medio.

La consulta que emplearemos devolverá un documento con el valor medio, el valor máximo y el valor mínimo de los **CrimeID**.

```
db.dataset1.aggregate([{$group: {_id: null, maxCrimeId:{$max: "$CrimeId" }, minCrimeId: { $min: "$CrimeId" },avgCrimeId: { $avg: "$CrimeId" } }]])
```

Asignatura	Datos del alumno	Fecha
Bases de datos para el Big Data		2025-01-25

```

> db.dataset1.aggregate([
  {
    $group: {
      _id: null,
      maxCrimeId: { $max: "$CrimeId" },
      minCrimeId: { $min: "$CrimeId" },
      avgCrimeId: { $avg: "$CrimeId" }
    }
  }
])
< {
  _id: null,
  maxCrimeId: 160964249,
  minCrimeId: 160903280,
  avgCrimeId: 160939382.37956423
}

```

Captura 14. Salida por MongoDB Shell de la colección 1.

- f) Elige una variable categórica y calcula el total de documentos agrupando por dicha variable.

En la consulta, nos devuelve un documento por cada valor que toma la variable **City**, junto con el total de veces que aparece. San Francisco aparece 19332 veces, Daly City 10, Yerba Buena y Presidio 6, Brisbane 2 y hay 642+2=644 ciudades desconocidas.

```

db.dataset1.aggregate([ { $group: { _id: "$City", total: { $sum: 1 } } },
{ $sort: { total: -1 } } ])

```

```

> db.dataset1.aggregate([
  {
    $group: {
      _id: "$City",
      total: { $sum: 1 }
    }
  },
  { $sort: { total: -1 } }
])
< {
  _id: 'San Francisco',
  total: 19332
}
{
  _id: 'Unknown',
  total: 642
}

```

```

{
  _id: 'Daly City',
  total: 10
}
{
  _id: 'Yerba Buena',
  total: 6
}
{
  _id: 'Presidio',
  total: 6
}
{
  _id: 'S',
  total: 2
}
{
  _id: 'Brisbane',
  total: 2
}

```

Captura 15. Salida por MongoDB Shell de la colección 1.

1.2.5 Consulta la colección 2.

- a) ¿Cuál es el tamaño de la colección (en bytes)? :

La colección ocupa 4620014 bytes. `db.dataset2.dataSize()`

```

> db.dataset2.dataSize()
< 4620014

```

Captura 16. Salida por MongoDB Shell de la colección 2.

- b) ¿Cuántos documentos tiene la colección?

La colección consta de 13865 documentos. `db.dataset2.countDocuments()`

```

> db.dataset2.countDocuments()
< 13865

```

Captura 17. Salida por MongoDB Shell de la colección 2.

Asignatura	Datos del alumno	Fecha
Bases de datos para el Big Data		2025-01-15

- c) Diseña y ejecuta una consulta donde puedas utilizar el operador \$in y un array de valores [val1, val2, val3, val4...].

La consulta devuelve los campos **categoría** y **denominación** de aquellos documentos que cumplen con el filtro impuesto en el array de valores. `db.dataset2.find({ categoria: { $in: ["3-HOTEL", "2-HOSTAL", "4-HOTEL"] } }, { categoria: 1, denominacion: 1, _id: 0 })`

```
> db.dataset2.find(
  { categoria: { $in: ["3-HOTEL", "2-HOSTAL", "4-HOTEL"] } },
  { categoria: 1, denominacion: 1, _id: 0 }
)
< {
  categoria: '3-HOTEL',
  denominacion: 'GRAN LEGAZPI'
}
{
  categoria: '2-HOSTAL',
  denominacion: 'BESAYA'
}
```

Captura 18. Salida por MongoDB Shell de la colección 2.

- d) Diseña y ejecuta una consulta con la que puedas generar un informe recuperando solo campos de tipo String. Utiliza Aggregate para generar dicho informe como una vista materializada.

La consulta garantiza que solo se incluyan valores válidos (del tipo "string") para los campos **alojamiento_tipo**, **categoría** y **denominación**. Si un campo tiene otro tipo de dato o no existe, se reemplaza con null.

```
db.dataset2.aggregate([{$project: {alojamiento_tipo: { $cond: { if: { $eq:
[{$type: "$alojamiento_tipo" }, "string"] }, then: "$alojamiento_tipo",
else: null } },categoria: { $cond: { if: { $eq: [{ $type: "$categoria" },
"string"] }, then: "$categoria", else: null } }, denominacion: { $cond: {
if: { $eq: [{ $type: "$denominacion" }, "string"] },then:"$denominacion",
else: null } } } }])
```

```
> db.dataset2.aggregate([
  {
    $project: {
      alojamiento_tipo: { $cond: { if: { $eq: [{ $type: "$alojamiento_tipo" }, "string"] }, then: "$alojamiento_tipo", else: null } },
      categoria: { $cond: { if: { $eq: [{ $type: "$categoria" }, "string"] }, then: "$categoria", else: null } },
      denominacion: { $cond: { if: { $eq: [{ $type: "$denominacion" }, "string"] }, then: "$denominacion", else: null } }
    }
  }
])
< {
  _id: ObjectId('67869227e3f7b041e9fe063d'),
  alojamiento_tipo: 'HOTEL',
  categoria: '3-HOTEL',
  denominacion: 'GRAN LEGAZPI'
}
{
  _id: ObjectId('67869227e3f7b041e9fe063e'),
  alojamiento_tipo: 'PENSION',
  categoria: '2-PENSION',
  denominacion: 'ISABEL'
}
```

Captura 19. Salida por MongoDB Shell de la colección 2.

- e) Diseña y ejecuta una consulta donde puedas utilizar los siguientes operadores de agregación: \$match, \$out, \$project, \$skip, \$sort, \$sortByCount o \$unwind Utiliza Aggregations de MongoDB Compass, con la opción "Add Stages" por cada operador utilizado. En la memoria muestra el resultado por etapa.

La consulta filtra los documentos donde categoría sea "3-HOTEL", ordena los resultados alfabéticamente, proyecta en pantalla los campos denominación y categoría, excluyendo _id.

Asignatura	Datos del alumno	Fecha
Bases de datos para el Big Data		2025-01-25

Posteriormente, omite los primeros 5 resultados de la consulta y devuelve un máximo de 10 documentos después de los pasos anteriores.

```
db.dataset2.aggregate([{$match: { categoria: "3-HOTEL" } },{$sort: { denominacion: 1 } },{$project: { denominacion: 1, categoria: 1, _id: 0 } }, { $skip: 5 },{$limit: 10 }])
```

```
> db.dataset2.aggregate([
  { $match: { categoria: "3-HOTEL" } }, // Filtra por una condición
  { $sort: { denominacion: 1 } }, // Ordena alfabéticamente por denominacion
  { $project: { denominacion: 1, categoria: 1, _id: 0 } }, // Muestra solo algunos campos
  { $skip: 5 }, // Salta los primeros 5 documentos
  { $limit: 10 } // Limita el resultado a 10 documentos
])
< {
  categoria: '3-HOTEL',
  denominacion: 'AP HOTEL MADRID AEROPUERTO'
}
{
  categoria: '3-HOTEL',
  denominacion: 'ASADOR DE ENRIQUE'
}
```

Captura 20. Salida por MongoDB Shell de la colección 2.

f) Crea un índice compuesto, tres índices simples y un índice de texto en esta colección.

```
db.dataset2.createIndex({ categoria: 1, denominacion: 1 })
db.dataset2.createIndex({ categoria: 1 })
db.dataset2.createIndex({ cdpostal: 1 })
db.dataset2.createIndex({ localidad: 1 })
db.dataset2.createIndex({ denominacion: "text" })
```

```
> db.dataset2.createIndex({ categoria: 1, denominacion: 1 })
< categoria_1_denominacion_1
> db.dataset2.createIndex({ categoria: 1 }) // Índice en categoría
db.dataset2.createIndex({ cdpostal: 1 }) // Índice en código postal
db.dataset2.createIndex({ localidad: 1 }) // Índice en localidad
< localidad_1
> db.dataset2.createIndex({ denominacion: "text" })
< denominacion_text
```

Captura 21. Salida por MongoDB Shell de la colección 2.

g) Diseña y ejecuta una consulta donde utilices el índice de texto creado.

La consulta devolverá todos los documentos en los que la palabra "LEGAZPI" aparezca en los campos indexados. `db.dataset2.find({ $text: { $search: "LEGAZPI" } })`

```
> db.dataset2.find({ $text: { $search: "LEGAZPI" } })
< {
  _id: ObjectId('67869227e3f7b041e9fe063d'),
  alojamiento_tipo: 'HOTEL',
  categoria: '3-HOTEL',
  denominacion: 'GRAN LEGAZPI',
  via_tipo: 'PASEO',
  via_nombre: 'de la Chopera',
  numero: 71,
  bloque: '',
  portal: '',
  escalera: '',
  planta: '',
  puerta: '',
  cdpostal: 28045,
  localidad: 'Madrid',
  signatura: 'HW-127',
  id_df: '2'
}
```

Captura 22. Salida por MongoDB Shell de la colección 2.

Asignatura	Datos del alumno	Fecha
Bases de datos para el Big Data		2025-01-15

1.3 Conclusión

Este trabajo nos ha servido para entender la utilidad de las consultas y de los elementos de agregación de documentos que posee MongoDB. Así mismo, ha sido una tarea enriquecedora al poder hacerla en equipo, ya que como suele ocurrir en muchos contextos académicos, las actividades grupales permiten el desarrollo de habilidades de trabajo en equipo, así como entender en profundidad la asignatura. Así mismo, tenemos que resaltar que no ha habido ningún problema con la realización de la actividad.

	Sí	No	A veces
Todos los miembros se han integrado en el trabajo del grupo.	x		
Todos los miembros participan activamente.	x		
Todos los miembros respetan otras ideas aportadas.	x		
Todos los miembros participan en la elaboración del informe.	x		
Me he preocupado por realizar un trabajo cooperativo con mis compañeros.	x		
Señala si consideras que algún aspecto del trabajo en grupo no ha sido adecuado.		x	