

Módulo	Tecnología y Herramientas Big Data
Nombre y apellidos	Richard Douglas Grijalba
Fecha entrega	09 agosto 2024

Hoja de respuestas

En lo personal quiero agradecer mucho a este curso y los distintos instructores que participan en la formación y estructura de los materiales y atención de dudas y consultas, porque me muestra que las diferentes tecnologías disponibles para la analítica de datos y data science son muy amplias, van más allá de las disponibles en la suite de Microsoft y Python, por lo que el haber visto Linux junto Spark y No Sql como Mongo DB y NEO4j formó un choque disruptivo en mi zona de confort y despierta la necesidad y softskill del analista de datos y de buscar respuestas y manejar y responder ante un problema de ciencia de datos, prueba y error en algunos casos, pensamiento investigativo, entendiendo que las respuestas no siempre estarán a la vista y no podemos quedarnos esperando una solución del exterior , puedo decir sin equivocarme que hasta el momento este ha sido la asignación de proyecto más difícil y retador que he tenido durante el curso y a lo largo de este programa de estudio en IMF. He trabajado con tres tecnologías diferentes que me han sacado de mi zona de confort: MongoDB, Terminal Linux, NoSQL con Neo4j, y Spark. Estas tecnologías han sido fundamentales para abordar el complejo desafío de emprender la salida en el mundo del Data Science.

De mi parte quiero solo mostrar mi agradecimiento y hacer ver que hice mi mayor esfuerzo en este módulo, aprender en modalidad virtual presenta sus retos y encontrarse con algunos issues en el camino hacen que uno deba emprender a buscar respuestas, generar pensamiento crítico y aprendizaje continuo.

Para cada caso por separado se encontrarán o se adjuntan capturas de los procesos y consultas necesarias para dar respuesta a cada una de las solicitudes, además de archivos anexos tan y como se solicita para el desarrollo de ese caso final del curso.

Tabla de Contenido

Caso Práctico Apache Spark.....	4
Tablas y/o Archivos cargados.....	6
Consideraciones Iniciales	7
Caso 1	10
Caso 2	12
Caso 3.....	13
Caso 4.....	14
Caso 5.....	15
Caso 6.....	15
Caso 7	16
Caso 8.....	18
Caso 9.....	20
Caso 10.....	21
Caso 2 Mongo DB	22
Consideraciones Iniciales	22
inicio en MongoDB en la terminal de linux.....	22
Caso 1	23
Caso 2	24
Caso 3.....	24
Caso 4.....	26
Caso 5.....	30
Caso 6.....	31
Caso 7	32
Caso 8.....	33
Caso 9.....	34
Caso 10.....	37
Caso 3 NEO4J.....	39
Consideraciones Iniciales	39
Caso 1	40
Caso 2.....	40
Caso 3.....	41
Caso 4.....	42
Caso 5.....	42
Caso 6.....	43
Caso 7	43
Caso 8.....	44
Caso 9.....	45
Caso 10.....	46
Conclusiones.....	47

Caso Práctico Apache Spark

Una compañía de telecomunicaciones quiere obtener insights de la muestra de 30 clientes obtenida al final de este mes y con información también del mes anterior. Los analistas de datos tendrán que agregar esta información, que se proporciona para responder a 10 cuestiones que ha planteado la directiva.

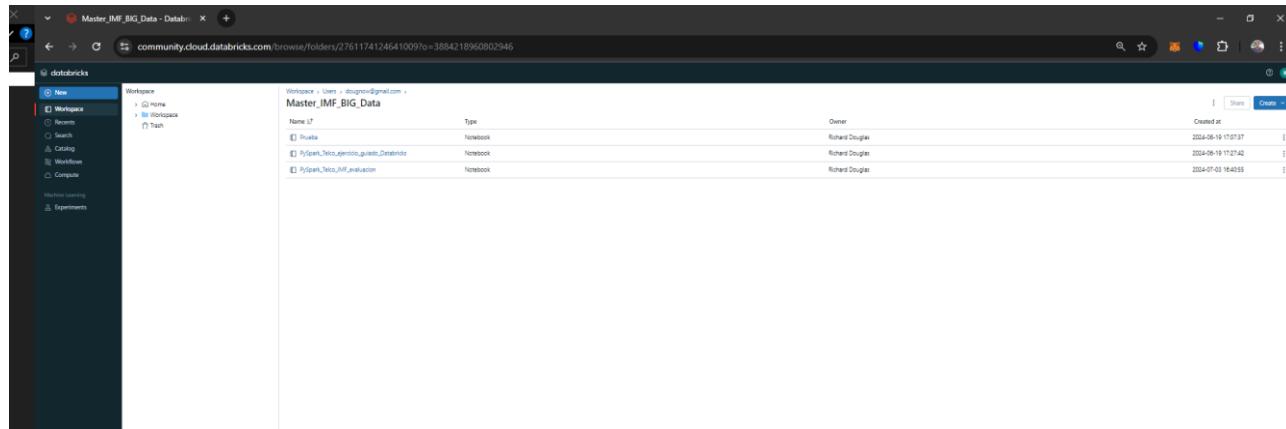
Las tablas de input son las mismas que se utilizaron en el ejercicio guiado de Spark:

- **df_clientes** : Muestra información de cada cliente (id, nombre, edad, ciudad, pais).
- **df_facturas_mes_ant**: Muestra la factura del mes anterior junto con el plan que tenían contratado para cada cliente. Un mismo cliente puede tener contratadas dos ofertas con la compañía, en cuyo caso habrá dos filas (de ahora en adelante, registros) con el mismo “id_cliente” y distinto “id_oferta”.
- **df_facturas_mes_actual**: Es una tabla equivalente a la anterior, pero muestra la información actualizada a último día del mes actual (agosto de 2020).
- **df_consumos_diarios**: Muestra para cada cliente, y diariamente, los datos móviles consumidos, el tráfico, SMS enviados y minutos de llamadas a fijos y móviles durante el mes en curso.
- **df_ofertas**: Muestra un catálogo de los distintos paquetes ofrecidos por la empresa, junto con una descripción y sus precios.

En este proyecto de Ciencia de Datos, se ha utilizado la herramienta cloud Spark Databricks para analizar y transformar datos relevantes de una empresa de telecomunicaciones. El objetivo es proporcionar insights útiles para la toma de decisiones estratégicas basados en diversas tablas de datos.

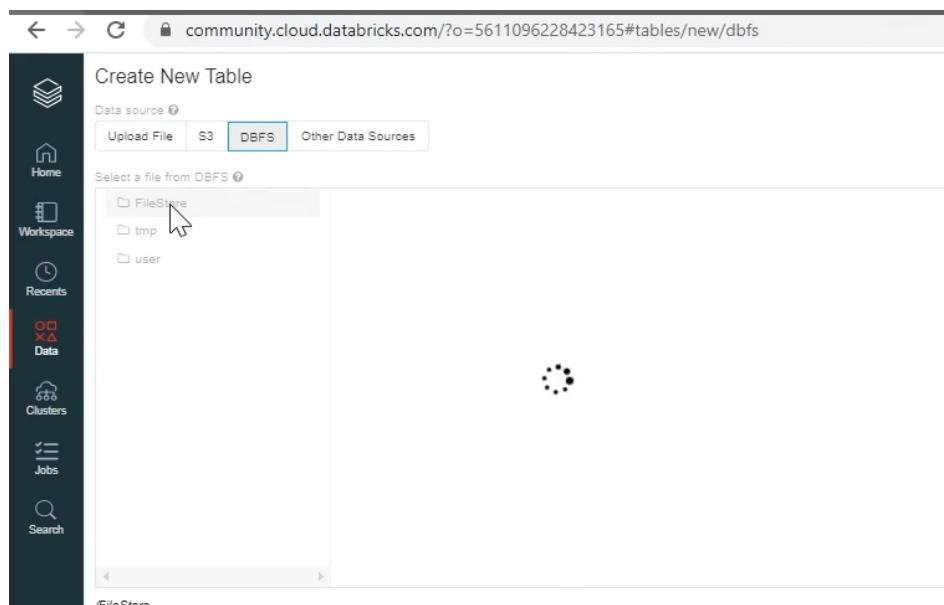
En lo que respecta al desarrollo de esta asignación corresponde a la carga y mantenimiento de datos, así como la realización de transformaciones necesarias para extraer insights valiosos, lo que busca es crear una simulación de lo que corresponde a las responsabilidades de una Analista de datos y/o Científico de Datos. El análisis y las transformaciones se realizaron utilizando Spark en Databricks, en lo que respecta como un área de Training, permitiendo el contacto con las tecnologías que se utilizan para la gestión y manejo eficiente de grandes volúmenes de datos (Big Data) y finalmente el dar respuesta a las consultas o casos específicos nos lleva a entender que se debe tomar los datos y obtener respuestas a los problemas reales de las empresas.

Data Bricks



The screenshot shows the Databricks web interface. On the left, there's a sidebar with options like 'New', 'Workspace', 'Recent', 'Search', 'Catalog', 'Workflow', and 'Compile'. The main area displays a list of notebooks under the path 'Master_IMF_BIG_Data'. The list includes:

Name	Type	Owner	Created at
Private	Notebook	Richard Douglas	2024-05-19 17:07:37
Arshan_Teico_experiments_guide_Databricks	Notebook	Richard Douglas	2024-05-19 17:27:42
Arshan_Teico_IMF_exploracion	Notebook	Richard Douglas	2024-07-03 18:40:55



The screenshot shows the 'Create New Table' dialog in Databricks. The left sidebar has 'Data' selected. The main area has 'DBFS' selected as the data source. A dropdown menu titled 'Select a file from DBFS' shows 'FileStore' as the current selection, with other options like 'tmp' and 'user' available. A loading icon is visible in the center.

Tablas y/o Archivos cargados

The screenshot shows the Databricks Data Catalog interface. On the left, there's a sidebar with options like Workspace, Recents, Search, Catalog (which is selected), Workflows, Compute, Machine Learning, and Experiments. The main area is titled "Data" and shows the "Databases" section with a "default" database selected. Below it is a "Tables" section with a "Filter Tables" search bar. A list of tables is shown, each with a preview of its data:

- df_clientes
- df_consumos_diarios
- df_facturas_mes_actual
- df_facturas_mes_ant
- df_ofertas

On the right side of the interface, there's a code editor window displaying some Python code related to data processing.

This screenshot shows a Databricks notebook interface. The sidebar on the left includes "New", "Workspace", "Recents", "Search", "Catalog" (selected), "Workflows", "Compute", "Machine Learning", and "Experiments". The main area has a "Data" tab open, showing the "Tables" section with the "default" database selected. A table preview is displayed with columns: ID, CANTIDAD, PRECIO, and FECHA. Below the table is a message: "Pantalla el número TOTAL de clientes que tenían en el mes anterior más de un contrato con la compañía".

The notebook contains the following code cells:

```

# Llamando el DF correspondiente
df_ant = spark.read.table("df_facturas_mes_ant")

```

A note below the code says: "nuevo dataframe de facturas del mes actual que asigne un 7% de descuento a todos los contratos de clientes que ya existían en el mes anterior (los contratos de los clientes nuevos seguirán importe). Mantener la columna "importe" y crear una nueva columna "importe_dto" con el nuevo importe, casteada a 2 decimales. Mostrar además el resultado ordenado por los campos idente, importe descendente."

```

# Crear el DF de facturas del mes actual
df_act = spark.read.table("df_facturas_mes_actual")
df_act = df_act.withColumn("importe_dto", df_act.importe * 0.93)
df_act = df_act.orderBy("idente", "importe", ascending=False)

```

At the bottom, there's a note about COVID-19: "Por saturación en la red debido al incremento en el uso de datos por el confinamiento debido a la COVID-19, se decide limitar el uso de datos este mes subiendo la tarifa de todas las que tienen 'datos ilimitados' en un 15%. Obtener un DF con las mismas columnas que el DF de facturas del mes actual y a mayores la columna 'importe_dto'".

Consideraciones Iniciales

Vamos a tener una serie de líneas de código en las que inciamos llamando o creando los data frame (df) las variables o sets de datos, por lo que de inicio no estamos aun dando respuesta a los enunciados indicados, sin embargo esta etapa es de suma importancia, en vista que sin datos o datos de calidad no se puede realizar un proyecto o solución de Data Science o Business Analytics.

```

columns = ['id_cliente', 'nombre', 'edad', 'sexo', 'provincia', 'pais']
data = [(1,'Pablo Perez',26,'M','Madrid','España'),(2,'Eduardo Redondo',58,'M','Bogota','Colombia'),(3,'Roberto Salazar',68,'M','Monterrey','Mexico'),
        (4,'Pedro Conde',32,'M','Madrid','España'),(5,'Ana Robles',41,'F','Valladolid','España'),(6,'David Roldan',74,'M','Guadalajara','Mexico'),
        (7,'Carmen Arauzo',19,'F','Medellin','Colombia'),(8,'Silvia Rodriguez',49,'F','Barcelona','España'),(9,'David Cardoso',24,'M','Cali','Colombia'),
        (10,'Pablo Lopez',40,'M','Sevilla','España'),(11,'Luis Rodrigues',28,'M','Bogota','Colombia'),(12,'Alvaro Monroy',75,'M','Monterrey','Mexico'),
        (13,'Victor Ruiz',53,'M','Valladolid','España'),(14,'Melisa Aguado',30,'F','Bogota','Colombia'),(15,'Cristian Cuadrado',52,'M','Guadalajara','Mexico'),
        (16,'Cristina Sanz',49,'F','Madrid','España'),(17,'Jorge Recio',18,'M','Valladolid','España'),(18,'Laura Luiz',44,'F','Cali','Colombia'),
        (19,'Juan Carlos Iglesias',38,'M','Barcelona','España'),(20,'Oscar Rico',22,'M','Valladolid','España'),(21,'Fatima Cuevas',29,'F','Cali','Colombia'),
        (22,'Clara Suarez',21,'F','Sevilla','España'),(23,'Fernanda Gomez',78,'F','Monterrey','Mexico'),(24,'Ruben Garcia',68,'M','Sevilla','España'),
        (25,'Ines Barcero',29,'F','Guadalajara','Mexico'),(26,'Celia Castro',47,'F','Barcelona','España'),(27,'Roberta Varado',64,'F','Cali','Colombia'),
        (28,'Walter Ramos',54,'M','Cali','Colombia'),(29,'Romina Verde',29,'F','Guadalajara','Mexico'),(30,'Marta Rodrigo',67,'F','Guadalajara','Mexico'))]

#(25,'Roberto Versero',66,'Sevilla','M','España'),(26,'Javier Ramos',36,'M','Madrid','España'),(27,'Ana Bolana',32,'F','Barcelona','España'),
#(28,'Lucia Alonso',23,'F','Valladolid','España'),(29,'Elena Luengo',27,'F','Valencia','España'),(30,'Rebeca Velez',69,'F','Cali','Colombia'),]

df_clientes = spark.createDataFrame(data, columns)
df_clientes.write.mode("overwrite").option("encoding", "UTF-8").saveAsTable('df_clientes')
df_clientes.show(n=30)
+-----+-----+-----+-----+
| 12 | Alvaro Monroy | 75 | M | Monterrey | Mexico |
| 13 | Victor Ruiz | 53 | M | Valladolid | España |
| 14 | Melisa Aguado | 30 | F | Bogota | Colombia |
| 15 | Cristian Cuadrado | 52 | M | Guadalajara | Mexico |
| 16 | Cristina Sanz | 49 | F | Madrid | España |
| 17 | Jorge Recio | 18 | M | Valladolid | España |
| 18 | Laura Luiz | 44 | F | Cali | Colombia |
| 19 | Juan Carlos Iglesias | 38 | M | Barcelona | España |
| 20 | Oscar Rico | 22 | M | Valladolid | España |
| 21 | Fatima Cuevas | 29 | F | Cali | Colombia |
| 22 | Clara Suarez | 21 | F | Sevilla | España |
| 23 | Fernanda Gomez | 78 | F | Monterrey | Mexico |
| 24 | Ruben Garcia | 68 | M | Sevilla | España |
| 25 | Ines Barcero | 29 | F | Guadalajara | Mexico |
| 26 | Celia Castro | 47 | F | Barcelona | España |
| 27 | Roberta Varado | 64 | F | Cali | Colombia |
| 28 | Walter Ramos | 54 | M | Cali | Colombia |
| 29 | Romina Verde | 29 | F | Guadalajara | Mexico |
| 30 | Marta Rodrigo | 67 | F | Guadalajara | Mexico |
+-----+-----+-----+-----+

```

DataFrame Ofertas estandar (truncate=False)

```
# Se presentan 10 tipos de ofertas diferentes (combinaciones entre Fibra Optica, Fijo, Movil (llamadas o mensajes ilimitados), Television)

columns = ['id_oferta','descripcion','importe']
data = [(1,'Fibra Optica 300MB + Fijo ilimitado + Television',70.00),
        (2,'Fibra Optica 600MB + Fijo ilimitado + Television', 78.50),
        (3,'Fibra Optica 300MB + Movil llamadas y datos ilimitados + Television', 85.00),
        (4,'Fibra Optica 600MB + Movil llamadas y datos ilimitados + Television', 95.00),
        (5,'Fibra Optica 300MB + Fijo ilimitado + Movil llamadas y datos ilimitados', 102.50),
        (6,'Fibra Optica 600MB + Fijo ilimitado + Movil llamadas y datos ilimitados + Television', 124.50),
        (7,'Fibra Optica 300MB + Movil llamadas ilimitadas + Television', 80.50),
        (8,'Fibra Optica 300MB + Movil datos ilimitados + Television', 84.00),
        (9,'Fijo llamadas ilimitadas + Movil llamadas y datos ilimitados', 42.50),
        (10,'Fijo llamadas ilimitadas + Movil llamadas ilimitadas', 31.99)] 

df_ofertas = spark.createDataFrame(data, columns)
df_ofertas.write.mode("overwrite").option("encoding", "UTF-8").saveAsTable('df_ofertas')
df_ofertas.show(n=10, truncate=False)
```

id_oferta	descripcion	importe
1	Fibra Optica 300MB + Fijo ilimitado + Television	70.0
2	Fibra Optica 600MB + Fijo ilimitado + Television	78.5
3	Fibra Optica 300MB + Movil llamadas y datos ilimitados + Television	85.0
4	Fibra Optica 600MB + Movil llamadas y datos ilimitados + Television	95.0
5	Fibra Optica 300MB + Fijo ilimitado + Movil llamadas y datos ilimitados	102.5
6	Fibra Optica 600MB + Fijo ilimitado + Movil llamadas y datos ilimitados + Television 124.5	
7	Fibra Optica 300MB + Movil llamadas ilimitadas + Television	80.5
8	Fibra Optica 300MB + Movil datos ilimitados + Television	84.0
9	Fijo llamadas ilimitadas + Movil llamadas y datos ilimitados	42.5
10	Fijo llamadas ilimitadas + Movil llamadas ilimitadas	31.99

DataFrame facturas_mes_ant

```
columns = ['id_cliente','id_oferta','importe','fecha']
data = [(1,6,124.5,"2020-07-31"),(1,6,118.60,"2020-07-31"),(1,9,36.5,"2020-07-31"),
        (2,4,95.0,"2020-07-31"),(3,5,102.5,"2020-07-31"),(5,7,80.5,"2020-07-31"),
        (5,9,40.5,"2020-07-31"),(6,8,84.0,"2020-07-31"),(7,4,95.0,"2020-07-31"),
        (8,5,102.5,"2020-07-31"),(10,6,124.5,"2020-07-31"),(11,3,80.0,"2020-07-31"),
        (12,1,70.0,"2020-07-31"),(13,4,95.0,"2020-07-31"),(14,9,42.5,"2020-07-31"),
        (15,2,78.50,"2020-07-31"),(15,6,115.0,"2020-07-31"),(16,1,70.0,"2020-07-31"),
        (17,8,84.0,"2020-07-31"),(18,5,102.5,"2020-07-31"),(18,9,37.5,"2020-07-31"),
        (20,4,95.0,"2020-07-31"),(22,2,78.5,"2020-07-31"),(22,7,69.5,"2020-07-31"),
        (23,3,85.0,"2020-07-31"),(24,3,85.0,"2020-07-31"),(25,9,42.5,"2020-07-31"),
        (25,10,31.99,"2020-07-31"),(26,4,95.0,"2020-07-31"),(27,8,84.0,"2020-07-31"),
        (27,9,41.5,"2020-07-31"),(29,6,124.5,"2020-07-31"),(30,1,70.0,"2020-07-31"),
        (30,2,70.0,"2020-07-31")]

df_facturas_mes_ant = spark.createDataFrame(data, columns)
df_facturas_mes_ant.write.mode("overwrite").option("encoding", "UTF-8").saveAsTable('df_facturas_mes_ant')
df_facturas_mes_ant.show(n=40, truncate=False)
```

15	2	78.5	2020-07-31
15	6	115.0	2020-07-31
16	1	70.0	2020-07-31
17	8	84.0	2020-07-31
18	5	102.5	2020-07-31
18	9	37.5	2020-07-31
20	4	95.0	2020-07-31
22	2	78.5	2020-07-31
22	7	69.5	2020-07-31
23	3	85.0	2020-07-31
24	3	85.0	2020-07-31
25	9	42.5	2020-07-31
25	10	31.99	2020-07-31
26	4	95.0	2020-07-31
27	8	84.0	2020-07-31
27	9	41.5	2020-07-31
29	6	124.5	2020-07-31
30	1	70.0	2020-07-31
30	2	70.0	2020-07-31

DataFrame factura_mes_actual

```
# Se dan de baja en el mes actual: 1y,27(solo de uno de los contratos),16,20,24
# cliente 25 rebaja por fidelidad
columns = ['id_cliente','id_oferta','importe','fecha']
data = [(1,6,124.5,"2020-08-31"),(1,6,118.60,"2020-08-31"),(2,4,95.0,"2020-08-31"),
        (3,5,102.5,"2020-08-31"),(4,1,70.0,"2020-08-31"),(5,7,80.5,"2020-08-31"),
        (5,9,40.5,"2020-08-31"),(6,8,84.0,"2020-08-31"),(7,4,95.0,"2020-08-31"),
        (8,5,100.5,"2020-08-31"),(9,7,78.50,"2020-08-31"),(9,10,29.99,"2020-08-31"),
        (10,6,124.5,"2020-08-31"),(11,3,83.0,"2020-08-31"),(12,1,70.0,"2020-08-31"),
        (13,4,95.0,"2020-08-31"),(14,9,42.5,"2020-08-31"),(15,2,78.50,"2020-08-31"),
        (15,6,115.0,"2020-08-31"),(17,8,84.0,"2020-08-31"),(18,5,102.5,"2020-08-31"),
        (18,9,39.0,"2020-08-31"),(19,3,85.0,"2020-08-31"),(21,6,124.50,"2020-08-31"),
        (22,2,78.5,"2020-08-31"),(22,7,72.5,"2020-08-31"),(23,3,85.0,"2020-08-31"),
        (25,9,42.5,"2020-08-31"),(25,10,29.70,"2020-08-31"),(26,4,95.0,"2020-08-31"),
        (27,8,84.0,"2020-08-31"),(28,5,102.50,"2020-08-31"),(29,6,121.50,"2020-08-31"),
        (30,1,70.0,"2020-08-31"),(30,2,71.0,"2020-08-31")]

df_facturas_mes_actual = spark.createDataFrame(data, columns)

# df_facturas_mes_actual.show(n=40, truncate=False)
df_facturas_mes_actual.write.mode("overwrite").option("encoding", "UTF-8").saveAsTable('df_facturas_mes_actual')
```

```
display(df_facturas_mes_actual)
```

Table

	id_cliente	id_oferta	importe	fecha
14	11	3	83	2020-08-31
15	12	1	70	2020-08-31
16	13	4	95	2020-08-31
17	14	9	42.5	2020-08-31
18	15	2	78.5	2020-08-31
19	15	6	115	2020-08-31
20	17	8	84	2020-08-31
21	18	5	102.5	2020-08-31
22	18	9	39	2020-08-31
23	19	3	85	2020-08-31
24	21	6	124.5	2020-08-31
25	22	2	78.5	2020-08-31
26	22	7	72.5	2020-08-31
27	23	3	65	2020-08-31

Caso 1

Mostrar por pantalla el número total de clientes del mes anterior con más de un contrato con la compañía.

Caso 1. Mostrar por pantalla el número TOTAL de clientes que tenían en el mes anterior más de un contrato con la compañía

```
#se debe iniciar llamando el DF correspondiente
df_facturas_mes_ant = spark.read.table("df_facturas_mes_ant")
df_facturas_mes_ant.printSchema()

root
 |-- id_cliente: long (nullable = true)
 |-- id_oferta: long (nullable = true)
 |-- importe: double (nullable = true)
 |-- fecha: string (nullable = true)
```

```
df_facturas_mes_ant = spark.read.table("df_facturas_mes_ant") #se muestra el DF de clientes del mes anterior
display(df_facturas_mes_ant)
```

Table

	id_cliente	id_oferta	importe	fecha
1	23	3	85	2020-07-31
2	24	3	85	2020-07-31
3	25	9	42.5	2020-07-31
4	25	10	31.99	2020-07-31
5	26	4	95	2020-07-31
6	27	8	84	2020-07-31
7	27	9	41.5	2020-07-31
8	29	6	124.5	2020-07-31
9	30	1	70	2020-07-31
10	30	2	70	2020-07-31
11	1	6	124.5	2020-07-31
12	1	6	118.6	2020-07-31
13	1	9	36.5	2020-07-31
14	2	4	95	2020-07-31
15	3	5	102.5	2020-07-31

34 rows

```

# se procede a dar respuesta a contar o indicar cuantos clientes tenian mas de un contrato
# Primero debemos proceder a agrupar por cliente y contar los contratos
conteo_de_clientes = df_facturas_mes_ant.groupby("id_cliente").agg(count("id_cliente").alias("conteo_contratos"))

# Se procede a crear un filtro clientes con más de un contrato
clientes_con_varios_contratos = conteo_de_clientes.filter(col("conteo_contratos") > 1)

# Con esto se puede ver el número total de estos clientes
clientes_con_varios_contratos.count()

# Finalmente se muestra el resultado
print("El número TOTAL de Clientes que tenian el mes anterior mas de un contrato corresponde a un total de: " + str(clientes_con_varios_contratos.count()))

El número TOTAL de Clientes que tenian el mes anterior mas de un contrato corresponde a un total de: 8

```

Estos son los clientes que tienen mas de un contrato Esto no es solicitado, pero lo aplico como un metodo de verificacion, que en total tenemos 8 clientes que presentan mas de un contrato

```

from pyspark.sql.window import Window
window_cliente = Window.partitionBy("id_cliente").orderBy(desc("importe"))

df_facturas_mes_ant = df_facturas_mes_ant.join(df_clientes,["id_cliente"], "left")
df_facturas_mes_ant = df_facturas_mes_ant.withColumn("count_contratos", row_number().over(window_cliente))
# Como puede haber ventanas con más de 2 elementos (por ejemplo el caso del id_cliente 1), creamos otra columna ("total_contratos") que nos informe del número maximo de la columna que acabamos de crear ("count_contratos").
df_facturas_mes_ant = df_facturas_mes_ant.withColumn("total_contratos", max("count_contratos").over(window_cliente.rowsBetween(Window.unboundedPreceding,Window.unboundedFollowing)))
df_facturas_mes_ant = df_facturas_mes_ant.filter(col("total_contratos") > 1)
df_facturas_mes_ant = df_facturas_mes_ant.dropDuplicates(["id_cliente"])
df_facturas_mes_ant = df_facturas_mes_ant.drop("id_oferta", "importe", "fecha", "count_contratos")
window_indice = Window.orderBy(sql_functions.monotonically_increasing_id())
df_facturas_mes_ant = df_facturas_mes_ant.withColumn("Indice", sql_functions.row_number().over(window_indice))
df_facturas_mes_ant.show(n=40)

```

id_cliente	nombre	total_contratos	indice
1	Pablo Perez	3	1
5	Ana Robles	2	2
15	Cristian Cuadrado	2	3
18	Laura Luis	2	4
22	Clara Suarez	2	5
25	Ines Bercero	2	6
27	Roberta Varadlo	2	7
30	Marta Rodrigo	2	8

Caso 2

Generar un nuevo dataframe de facturas del mes actual que asigne un 7 % de descuento a todos los contratos de clientes que ya existían en el mes anterior (los contratos de los clientes nuevos seguirán con el mismo importe). Mantener la columna "importe" y crear una nueva columna "importe(dto)" con el nuevo importe, casteada a dos decimales. Mostrar, además, el resultado ordenado por los campos "id_cliente" ascendente e importe descendente

```
# Procedemos a obtener los clientes que tenian contratos el mes anterior
clientes_mes_anterior = df_facturas_mes_ant.select("id_cliente").distinct()

# Con el siguiente comando se procede a unir los DataFrames del mes actual con los clientes del mes anterior
df_facturas_con_descuento = df_facturas_mes_actual.join(clientes_mes_anterior, ["id_cliente"],"left")

# Segun lo solicitado se requiere que se le aplique un 7% de descuento a los clientes que existian el mes anterior (tipo de beneficio a la fidelidad de clientes)
df_facturas_con_descuento = df_facturas_con_descuento.withColumn("importe(dto)",when(col("id_cliente").isNotNull(), round(col("importe") * 0.93, 2)).otherwise(col("importe")))

# Ordenar por id_cliente ascendente y importe descendente
df_facturas_con_descuento_ordenado = df_facturas_con_descuento.orderBy(col("id_cliente").asc(), col("importe").desc())

# Se procede a mostrar el resultado
display(df_facturas_con_descuento_ordenado)
```

Table

	id_cliente	id_oferta	importe	fecha	importe(dto)
1	1	6	124.5	2020-08-31	115.79
2	1	6	118.6	2020-08-31	110.3
3	2	4	95	2020-08-31	88.35
4	3	5	102.5	2020-08-31	95.33
5	4	1	70	2020-08-31	65.1
6	5	7	80.5	2020-08-31	74.87
7	5	9	40.5	2020-08-31	37.67
8	6	8	84	2020-08-31	78.12
9	7	4	95	2020-08-31	88.35
10	8	5	100.5	2020-08-31	93.47
11	9	7	78.5	2020-08-31	73.01
12	9	10	29.99	2020-08-31	27.89
13	10	6	124.5	2020-08-31	115.79
14	11	3	83	2020-08-31	77.19
15	12	1	70	2020-08-31	65.1

35 rows

Caso 3

Por problemas de saturación en la red debido al incremento en el uso de datos por el confinamiento debido a la COVID-19, se decide limitar el uso de datos este mes subiendo la tarifa de todas las ofertas que incluyan "datos ilimitados" en un 15 %. Obtener un DF con las mismas columnas que el DF de facturas del mes actual y, a mayores, la columna "importe(dto)".

Vamos a mostrar los mismos resultados pero con solo dos decimales

```
df_facturas_ajustadas = df_facturas_mes_actual.alias("actual").join(df_ofertas.withColumnRenamed("importe", "importe_oferta"), "id_oferta", "left").\
    withColumn("importe(dto)", round(when(lower(col("descripcion")).contains("datos ilimitados"), col("importe_oferta") * 1.15).otherwise(col("importe_oferta")),2))\
    .select("id_cliente", "id_oferta", "importe", "fecha", "importe(dto")
```

Table

	id_cliente	id_oferta	importe	fecha	importe(dto)
1	25	10	29.7	2020-08-31	31.99
2	26	4	95	2020-08-31	109.25
3	27	8	84	2020-08-31	96.6
4	28	5	102.5	2020-08-31	117.87
5	29	6	121.5	2020-08-31	143.17
6	30	1	70	2020-08-31	70
7	30	2	71	2020-08-31	78.5
8	1	6	124.5	2020-08-31	143.17
9	1	6	118.6	2020-08-31	143.17
10	2	4	95	2020-08-31	109.25
11	3	5	102.5	2020-08-31	117.87
12	4	1	70	2020-08-31	70
13	5	7	80.5	2020-08-31	80.5
14	5	9	40.5	2020-08-31	48.87
15	6	8	84	2020-08-31	96.6

35 rows

Caso 4

Crear una nueva variable "grupo_edad" que agrupe a los clientes, tanto del "mes_actual" como del "mes_anterior" en cuatro rangos según su edad asignando valores del uno al cuatro según el rango al que pertenezcan (18-25 (1), 26-40 (2), 41-65 (3), >65 (4)).

Obtener una tabla resumen que extraiga para cada para cada uno de los cuatro grupos identificados la media de consumo de datos, SMS enviados, "minutos_movil", "minutos_fijo", con todos los campos casteados a dos decimales y ordenar el DF por "grupo_edad" ascendente. Extraer conclusiones

4. Crear una nueva variable "grupo_edad" que agrupe a los clientes, tanto del mes_actual como del mes_anterior en 4 rangos según su edad asignando valores del 1 - 4 según el rango al que pertenezcan (18-25 (1), 26-40 (2), 41-65 (3), >65 (4)). Obtener una tabla resumen que extraiga para cada para cada uno de los 4 grupos identificados la MEDIA de consumo de datos, sms enviados, minutos_movil, minutos_fijo, con todos los campos casteados a 2 decimales y ordenar el DF por grupo_edad ascendente. Extraer conclusiones.

spark.catalog.listTables() # este codigo nos permite ver los DF o tales disponibles (creados hasta el momento)																																																																																																																					
[Table(name='df_clientes', database='default', description=None, tableType='MANAGED', isTemporary=False), Table(name='df_consumos_diarios', database='default', description=None, tableType='MANAGED', isTemporary=False), Table(name='df_facturas_mes_actual', database='default', description=None, tableType='MANAGED', isTemporary=False), Table(name='df_facturas_mes_ant', database='default', description=None, tableType='MANAGED', isTemporary=False), Table(name='df_ofertas', database='default', description=None, tableType='MANAGED', isTemporary=False)]																																																																																																																					
df_clientes = spark.read.table("df_clientes") #leemos o llamamos el DF_clientes de su tabla original																																																																																																																					
display(df_clientes)																																																																																																																					
<table border="1"> <thead> <tr> <th>Table</th><th>s3_id_cliente</th><th>A₂ nombre</th><th>A₃ edad</th><th>A₄ sexo</th><th>A₅ provincia</th><th>A₆ pais</th></tr> </thead> <tbody> <tr><td>10</td><td>28</td><td>Walter Ramos</td><td>54</td><td>M</td><td>Cal</td><td>Colombia</td></tr> <tr><td>11</td><td>29</td><td>Romina Verde</td><td>29</td><td>F</td><td>Guadalajara</td><td>Mexico</td></tr> <tr><td>12</td><td>30</td><td>Marta Rodriguez</td><td>67</td><td>F</td><td>Guadalajara</td><td>Mexico</td></tr> <tr><td>13</td><td>7</td><td>Carmen Araujo</td><td>19</td><td>F</td><td>Medellin</td><td>Colombia</td></tr> <tr><td>14</td><td>8</td><td>Silva Rodriguez</td><td>49</td><td>F</td><td>Barcelona</td><td>España</td></tr> <tr><td>15</td><td>9</td><td>David Cardoso</td><td>24</td><td>M</td><td>Cal</td><td>Colombia</td></tr> <tr><td>16</td><td>22</td><td>Claire Suarez</td><td>21</td><td>F</td><td>Sevilla</td><td>España</td></tr> <tr><td>17</td><td>23</td><td>Fernanda Gomez</td><td>78</td><td>F</td><td>Monterrey</td><td>Mexico</td></tr> <tr><td>18</td><td>24</td><td>Ruben Garcia</td><td>68</td><td>M</td><td>Sevilla</td><td>España</td></tr> <tr><td>19</td><td>19</td><td>Juan Carlos Iglesias</td><td>38</td><td>M</td><td>Barcelona</td><td>España</td></tr> <tr><td>20</td><td>20</td><td>Oscar Rico</td><td>22</td><td>M</td><td>Varadolid</td><td>España</td></tr> <tr><td>21</td><td>21</td><td>Fatima Cuevas</td><td>29</td><td>F</td><td>Cal</td><td>Colombia</td></tr> <tr><td>22</td><td>4</td><td>Pedro Conde</td><td>32</td><td>M</td><td>Madrid</td><td>España</td></tr> <tr><td>23</td><td>5</td><td>Ana Robles</td><td>41</td><td>F</td><td>Varadolid</td><td>España</td></tr> <tr><td>24</td><td>6</td><td>David Rodan</td><td>74</td><td>M</td><td>Guadalajara</td><td>Mexico</td></tr> </tbody> </table>						Table	s3_id_cliente	A ₂ nombre	A ₃ edad	A ₄ sexo	A ₅ provincia	A ₆ pais	10	28	Walter Ramos	54	M	Cal	Colombia	11	29	Romina Verde	29	F	Guadalajara	Mexico	12	30	Marta Rodriguez	67	F	Guadalajara	Mexico	13	7	Carmen Araujo	19	F	Medellin	Colombia	14	8	Silva Rodriguez	49	F	Barcelona	España	15	9	David Cardoso	24	M	Cal	Colombia	16	22	Claire Suarez	21	F	Sevilla	España	17	23	Fernanda Gomez	78	F	Monterrey	Mexico	18	24	Ruben Garcia	68	M	Sevilla	España	19	19	Juan Carlos Iglesias	38	M	Barcelona	España	20	20	Oscar Rico	22	M	Varadolid	España	21	21	Fatima Cuevas	29	F	Cal	Colombia	22	4	Pedro Conde	32	M	Madrid	España	23	5	Ana Robles	41	F	Varadolid	España	24	6	David Rodan	74	M	Guadalajara	Mexico
Table	s3_id_cliente	A ₂ nombre	A ₃ edad	A ₄ sexo	A ₅ provincia	A ₆ pais																																																																																																															
10	28	Walter Ramos	54	M	Cal	Colombia																																																																																																															
11	29	Romina Verde	29	F	Guadalajara	Mexico																																																																																																															
12	30	Marta Rodriguez	67	F	Guadalajara	Mexico																																																																																																															
13	7	Carmen Araujo	19	F	Medellin	Colombia																																																																																																															
14	8	Silva Rodriguez	49	F	Barcelona	España																																																																																																															
15	9	David Cardoso	24	M	Cal	Colombia																																																																																																															
16	22	Claire Suarez	21	F	Sevilla	España																																																																																																															
17	23	Fernanda Gomez	78	F	Monterrey	Mexico																																																																																																															
18	24	Ruben Garcia	68	M	Sevilla	España																																																																																																															
19	19	Juan Carlos Iglesias	38	M	Barcelona	España																																																																																																															
20	20	Oscar Rico	22	M	Varadolid	España																																																																																																															
21	21	Fatima Cuevas	29	F	Cal	Colombia																																																																																																															
22	4	Pedro Conde	32	M	Madrid	España																																																																																																															
23	5	Ana Robles	41	F	Varadolid	España																																																																																																															
24	6	David Rodan	74	M	Guadalajara	Mexico																																																																																																															
30 rows																																																																																																																					

Para cada uno de los 4 grupos identificados la MEDIA de consumo de datos, sms enviados, minutos_movil, minutos_fijo, con todos los campos casteados a 2 decimales y ordenar el DF por grupo_edad ascendente. Extraer conclusiones.

# Calcula la media de consumo de datos, sms enviados, minutos_movil y minutos_fijo por grupo_edad																																			
# vamos que ahora toca generar no solo los valores sino tambien una tabla resumen, lo cual se procede a usar un redondeo a 2 digitos																																			
df_consumos_diarios_resumen = df_consumos_diarios.join.groupby("grupo_edad").agg(
round(mean("consumo_datos", 2).alias("media_consumo_datos"),																																			
round(mean("sms_enviados", 2).alias("media_sms_enviados"),																																			
round(mean("minutos_llamadas_movil", 2).alias("media_minutos_movil"),																																			
round(mean("minutos_llamadas_fijo", 2).alias("media_minutos_fijo"),																																			
).orderBy("grupo_edad")																																			
# Mostrar el resultado																																			
display(df_consumos_diarios_resumen)																																			
<table border="1"> <thead> <tr> <th>s3_grupo_edad</th><th>1.2 media_consumo_datos</th><th>1.2 media_sms_enviados</th><th>1.2 media_minutos_movil</th><th>1.2 media_minutos_fijo</th><th></th></tr> </thead> <tbody> <tr><td>1</td><td>1338.24</td><td>0.54</td><td>18.97</td><td>10.86</td><td></td></tr> <tr><td>2</td><td>1062.69</td><td>1.13</td><td>34.88</td><td>21.76</td><td></td></tr> <tr><td>3</td><td>473.29</td><td>5.25</td><td>32.44</td><td>26.43</td><td></td></tr> <tr><td>4</td><td>171.19</td><td>6.16</td><td>18.09</td><td>35.75</td><td></td></tr> </tbody> </table>						s3_grupo_edad	1.2 media_consumo_datos	1.2 media_sms_enviados	1.2 media_minutos_movil	1.2 media_minutos_fijo		1	1338.24	0.54	18.97	10.86		2	1062.69	1.13	34.88	21.76		3	473.29	5.25	32.44	26.43		4	171.19	6.16	18.09	35.75	
s3_grupo_edad	1.2 media_consumo_datos	1.2 media_sms_enviados	1.2 media_minutos_movil	1.2 media_minutos_fijo																															
1	1338.24	0.54	18.97	10.86																															
2	1062.69	1.13	34.88	21.76																															
3	473.29	5.25	32.44	26.43																															
4	171.19	6.16	18.09	35.75																															
4 rows																																			

Conclusiones sobre el resultado del resumen anterior

rangos según su edad asignando valores del 1 - 4 según el rango al que pertenezcan (18-25 (1), 26-40 (2), 41-65 (3), >65 (4))

- **Media Consumo Datos : El grupo de edad - 1 18 a 25**, presenta la media de consumo de datos mas alta, y la misma disminuye a mayor rango de edad
- **Media Mensajes Enviados SMS : El grupo numero 4** mayor a 65. Es que presenta un mayor uso de los mensajes de texto con una media 6.
- **Media de Minutos Movil** : Los grupos de edad 2 y 3 son los que mas consumen los minutos móvil presentando una media de 34.88 los del grupo 2, y una media 32.44 los del grupo 3.
- **Media de Minutos Fijo** : Aquellos que pertenecen a los grupo de edad 4. presentan el mayor consumo de minutos fijo

Este breve ejercicio permite conocer con una breve exploracion de los datos, el comportamiento de consumo en telefonía por las personas segmentadas por la edad, a lo que se le puede dar seguimiento y la empresa podria generar promociones segun los ragos de preferencia de la segmentacion de los clientes por edad.

Caso 5

Se quiere realizar un estudio por sexo para analizar si son las mujeres o los hombres quienes consumen más datos durante el fin de semana y hacen más llamadas desde el móvil. Para ello, se deberá, sin ayuda de un calendario, extraer el día de la semana al que corresponde cada una de las fechas del mes de agosto para saber cuáles son fin de semana (se consideran días de fin de semana el viernes, sábado y domingo). El DF a obtener deberá tener dos registros con las siguientes columnas: sexo, “total_mins_movil_finde”, “total_datos_moviles_finde”. Extraer conclusiones tras presentar el DF resultante.

```
df_consumos_diarios_join_finde = df_consumos_diarios_join.filter(col('dia_semana').isin([5, 6, 7]))  
  
# Agrupar por sexo y sumar minutos de móvil y datos móviles  
df_consumos_diarios_join_finde = df_consumos_diarios_join_finde.groupBy('sexo').agg(spark_sum('minutos_llamadas_movil').alias('total_mins_movil_finde'),  
spark_sum('consumo_datos_MB').alias('total_datos_moviles_finde'))
```

El DF a obtener deberá tener 2 registros con las columnas: sexo, total_mins_movil_finde, total_datos_moviles_finde.

```
display(df_consumos_diarios_join_finde)
```

Table

	sex	total_mins_movil_finde	total_datos_moviles_finde
1	F	6732	137824
2	M	3303	127323

Conclusiones de los resultados

Segun el cuadro resumen anterior se denotan algunas diferencias claras :

- **Total de minutos móviles** Se denota un mayor consumo por parte de las mujeres en el total de llamadas o minutos móvil, Mujeres un total de 6732 minutos contra los Hombres 3303, praticamente un consumo del doble
- **Total de Datos Móviles** Existe un consumo similar entre ambos, siendo en este caso nuevamente las mujeres las que consumen mas los datos los fines de semana

Estos resultados nos permite determinar que durante los fines de semana los clientes tendrán mayor consumo de datos y llamadas , y las mujeres van a demandar un mayor cantidad de llamadas, la segmentación y promociones de los clientes se puede hacer tomando en cuenta esta informacion.

Caso 6

Obtener un DF que contenga cuatro registros, que serán el cliente de cada grupo edad que más datos móviles ha consumido durante los 15 primeros días del mes de agosto (día 15 incluido en el cálculo).

El DF deberá contener las columnas: nombre, edad, “grupo_edad”, “datos_moviles_total_15”,

“max_sms_enviados_15” (“max_sms_enviados_15” contiene el máximo de sms enviados en un día por el cliente con más “datos_moviles” consumidos de cada “grupo_edad” durante esos 15 primeros días del mes).

Extraer conclusiones en cuanto a datos consumidos y SMS enviados por cada “grupo_edad”.

```

# Filtrar primeros 15 días del mes
df_primeros_15 = df_consumos_diarios.join.filter(dayofmonth(col("fecha")) <= 15)

# Calcular el total de datos móviles y máximo de SMS por día por cliente
df_primeros_15_agg = df_primeros_15.groupBy("id_cliente", "grupo_edad").agg(
    sum("consumo_datos_MB").alias("datos_moviles_total_15"),
    max("sms_enviados").alias("max_sms_enviados_15")
)

# Obtener el cliente con más datos móviles por grupo de edad
window_edad = Window.partitionBy("grupo_edad").orderBy(col("datos_moviles_total_15").desc())
df_top_clientes = df_primeros_15_agg.withColumn("rank", row_number().over(window_edad)).filter(col("rank") == 1).drop("rank")

# Unir con información del cliente
df_top_clientes = df_top_clientes.join(df_clientes, "id_cliente", "left").select("nombre", "edad", "grupo_edad", "datos_moviles_total_15", "max_sms_enviados_15")

display(df_top_clientes)

```

Table

	nombre	edad	grupo_edad	datos_moviles_total_15	max_sms_enviados_15
1	Clara Suarez	21	1	27665	2
2	Romina Verde	29	2	23603	1
3	Walter Ramos	54	3	11892	18
4	Marta Rodrigo	67	4	4657	16

4 rows

Conclusiones

- El cliente que nos dice la tabla resumen con mayor consumo de datos móviles corresponde a **Carlos** de 21 años
- Los clientes que tienen un mayor consumo, comportamiento en general, que se pueden asignar como VIP's son importantes para cualquier empresa, ademas las empresas deben buscar fidelización de los clientes para que aquellos clientes A,B o C, se mantengan o aumenten de categoría
- Esto permite a las empresas garantizar un monto promedio de facturación de servicios, disminucion de costos en atención de quejas.

Caso 7

Interesa averiguar los minutos de llamadas de los clientes que son nuevos este mes para realizar un estudio del impacto que tendrán en caso de producirse un pico de volumen de llamadas en la red.

Obtener un DF que contenga solo a los clientes nuevos de este mes con cuatro columnas:

"nombre_cliente_nuevo", edad, "importe_total_mes_actual", "total_minutos".

Se ordenará el DF resultante por la columna "total_minutos", que contiene el total de minutos de llamadas fijas y móviles durante este mes para cada cliente, ordenándolos de manera que el primer cliente nuevo que aparezca sea el que menos minutos de llamadas ha realizado en el mes de agosto.

```
df_consumos_diarios_join = df_consumos_diarios.select("id_cliente","minutos_llamadas_movil","minutos_llamadas_fijo")
df_consumos_diarios_join.show()
```

1	46	10
1	17	0
1	31	4
1	20	0
1	21	0
1	12	0
1	0	24
1	46	0
1	0	0
1	25	0
1	41	0
1	34	102
1	0	50
1	31	0
1	78	0
1	29	32
1	0	7
1	19	0

only showing top 20 rows

```
# se procede a obtener los clientes nuevos
```

```
clientes_nuevos_join = df_facturas_mes_actual.join(df_facturas_mes_ant,["id_cliente"],"leftanti")
clientes_nuevos_join.show()
```

id_cliente	id_oferta	importe	fecha	edad	grupo_edad
4	1	70.0	2020-08-31	32	2
9	7	78.5	2020-08-31	24	1
9	10	29.99	2020-08-31	24	1
19	3	85.0	2020-08-31	38	2
21	6	124.5	2020-08-31	29	2
28	5	102.5	2020-08-31	54	3

```

# Verificar las columnas del DataFrame antes de la agrupación
clientes_nuevos_join.printSchema()

root
 |-- id_cliente: long (nullable = true)
 |-- id_oferta: long (nullable = true)
 |-- importe: double (nullable = true)
 |-- fecha: string (nullable = true)
 |-- edad: long (nullable = true)
 |-- grupo_edad: integer (nullable = true)
 |-- nombre_cliente_nuevo: string (nullable = true)
 |-- minutos_llamadas_movil: long (nullable = true)
 |-- minutos_llamadas_fijo: long (nullable = true)

# Agrupar y calcular totales
clientes_nuevos_join = clientes_nuevos_join.groupBy("id_cliente", "nombre_cliente_nuevo", "edad") \
    .agg(spark_sum("importe").alias("importe_total_mes_actual"),
         (spark_sum("minutos_llamadas_movil") + spark_sum("minutos_llamadas_fijo")).alias("total_minutos"))
    ).orderBy("total_minutos")

# Mostrar el resultado
display(clientes_nuevos_join)

```

Table

	id_cliente	$\text{nombre_cliente_nuevo}$	edad	$\text{importe_total_mes_actual}$	total_minutos
1	28	Walter Ramos	54	3177.5	528
2	9	David Cardoso	24	3363.1899999999932	736
3	19	Juan Carlos Iglesias	38	2635	923
4	4	Pedro Conde	32	2170	1939
5	21	Fatima Cuevas	29	3859.5	2366

5 rows

Caso 8

Obtener un DF que contenga, para los clientes que ya existían en el mes anterior y siguen dados de alta este mes, tres columnas: nombre, edad, “n_días_sin_sms”. La última columna se refiere a obtener el número de días en los que el cliente no ha enviado ningún SMS durante el mes de agosto. Forzar al comando show a que muestre 30 valores.

Si en todo el mes el cliente “A” no envió SMS tres días, esta columna deberá contener el valor “3”. Si hubiera algún cliente que hubiese enviado al menos un SMS todos los días del mes, esta columna tendrá valor “0”).

Una vez que tenemos el Df con los clientes que estan presentes en ambos meses, se procede a crear el proceso para el calculo de los dias sin envio de sms

```
# Crear una columna con la fecha en formato DateType
df_consumos_diarios = df_consumos_diarios.withColumn("fecha", to_date("fecha", "yyyy-MM-dd"))

# Calcular el número de días en el mes de agosto
num_days_in_august = 31

# este código nos permite agrupar por cliente y contar el número de días que no se enviaron SMS
df_n_dias_sin_sms = df_consumos_diarios.groupBy("id_cliente").agg(
    (num_days_in_august - spark_sum(when(col("sms_enviados") > 0, 1).otherwise(0))).alias("n_dias_sin_sms")
)

# Unir con el DataFrame de clientes activos
df_resultado = df_clientes_activos.join(df_n_dias_sin_sms, ["id_cliente"], "left")

# Seleccionar y renombrar las columnas necesarias
df_resultado = df_resultado.select("nombre", "edad", "n_dias_sin_sms")

# Mostrar el resultado
df_resultado.show(30)
```

Ana Robles	41	3
David Roldan	74	1
Carmen Arauzo	19	19
Silvia Rodriguez	49	2
Pablo Lopez	40	11
Luis Rodrigues	20	21
Alvaro Monroy	75	3
Victor Ruiz	53	7
Melisa Aguado	30	20
Cristian Cuadrado	52	5
Jorge Recio	18	22
Laura Luiz	44	7
Clara Suarez	21	26
Fernanda Gomez	78	21
Ines Barcerol	29	13
Celia Castro	47	0
Roberta Varado	64	1
Romina Verde	29	23
Marta Rodrigo	67	2

```
display(df_resultado) #con esto procedemos a mostar los datos del resultado de una mejor forma
```

Table

	nombre	edad	n_dias_sin_sms
1	Pablo Perez	26	14
2	Eduardo Redondo	58	7
3	Roberto Salazar	68	3
4	Ana Robles	41	3
5	David Roldan	74	1
6	Carmen Arauzo	19	19
7	Silvia Rodriguez	49	2
8	Pablo Lopez	40	11
9	Luis Rodrigues	20	21
10	Alvaro Monroy	75	3
11	Victor Ruiz	53	7
12	Melisa Aguado	30	20
13	Cristian Cuadrado	52	5
14	Jorge Recio	18	22
15	Laura Luiz	44	7

22 rows

Caso 9

Se desea obtener un coeficiente de ponderación que permita evaluar a cada cliente en función de su consumo para identificar los clientes más atractivos que forman parte de la compañía.

Este cálculo solo se realizará para los clientes que tengan una sola oferta contratada con la compañía.

Este coeficiente se obtendrá en base a los consumos diarios y, por tanto, solo se tendrán en cuenta los clientes que existen en el mes de agosto, ya que no existen datos de consumo del mes de julio. Las ponderaciones que se darán a cada uno de los consumos son las siguientes:

- 0,4 --> Llamadas desde teléfono móvil.
- 0,3 --> "datos_móviles_MB".
- 0,2 --> Llamadas desde teléfono fijo.
- 0,1 --> SMS enviados.



PySpark_Telco_IMF_evaluacion (Python)

```
df_normalizado = df_sumas_consumos.withColumn("datos_móviles_0_1", col("total_datos_móviles_MB") / max_consumos["max_datos_móviles_MB"])
    .withColumn("minutos_llamadas_movil_0_1", col("total_minutos_llamadas_movil") / max_consumos["max_minutos_llamadas_movil"])
    .withColumn("minutos_llamadas_fijo_0_1", col("total_minutos_llamadas_fijo") / max_consumos["max_minutos_llamadas_fijo"])
    .withColumn("sms_enviados_0_1", col("total_sms_enviados") / max_consumos["max_sms_enviados"])
```

```
# Aplicar ponderaciones
df_ponderado = df_normalizado.withColumn(
    "coeficiente_cliente",
    col("datos_móviles_0_1") * 0.3 +
    col("minutos_llamadas_movil_0_1") * 0.4 +
    col("minutos_llamadas_fijo_0_1") * 0.2 +
    col("sms_enviados_0_1") * 0.1
).select("id_cliente", "coeficiente_cliente")
```

```
# Unir con información de clientes para obtener nombre y edad
df_resultado = df_ponderado.join(df_clientes_una_sola_oferta, "id_cliente").select("nombre", "edad", "coeficiente_cliente")

# Ordenar por coeficiente en orden descendente
df_resultado = df_resultado.orderBy(col("coeficiente_cliente").desc())

# Mostrar el resultado
print("Esta es la tabla resumen de los cálculos de ponderación de los clientes:")
df_resultado.show()
```

Esta es la tabla resumen de los cálculos de ponderación de los clientes:

	nombre edad	coeficiente_cliente
Romina Verde	29 0.08688029518271995	
Fatima Cuevas	29 0.08072401949500864	
Silvia Rodriguez	49 0.07869446236017838	
Celia Castro	47 0.07189382964587766	
Melisa Aguado	30 0.06485328127735929	
Pedro Conde	32 0.06183052296641726	
Carmen Arauzo	19 0.05777098521283136	
Eduardo Redondo	58 0.05277221146373834	
Fernanda Gomez	78 0.047992504156016455	
Roberta Varado	64 0.0464478062305848	
Juan Carlos Iglesias	38 0.045502113581325575	
David Roidan	74 0.04537437482011635	
Jorge Recio	18 0.04172011704161756	
Walter Ramos	54 0.040952896178247614	
Roberto Salazar	68 0.037765200679635035	
Luis Rodrigues	20 0.0363224659128446	

Caso 10

Se desea averiguar la fecha en la que entre los tres clientes que más consumen de cada “grupo_edad” llegan a un consumo de 20 GB de datos móviles. En caso de que algún “grupo_edad” no llegue, entre los tres clientes, a 20 GB en todo el mes, se asignará null como valor de esta columna (1 GB = 1024 MB).

Obtener un DF que contiene cuatro registros, uno para cada “grupo_edad” y tres columnas: “grupo_edad”, “fecha_20_GB”, “datos_moviles_total_grupo_3_clientes”.

“datos_moviles_total_grupo_3_clientes” representa el total de datos consumidos en MB por los tres clientes del grupo hasta final de mes).

```
# Una vez que tenemos los DF con los consumos en GB y con los grupos de edad, se procede a sacar los 3 clientes que más consumen por grupo de edad
window_spec = Window.partitionBy("grupo_edad").orderBy(col("datos_móviles_GB").desc())
```

```
df_consumos_diarios_join = df_consumos_diarios_join.withColumn("rank", row_number().over(window_spec)).filter(col("rank") <= 3)
df_consumos_diarios_join.show()
```

id_cliente consumo_datos_MB sms_enviados minutos_llamadas_movil minutos_llamadas_fijo	fecha	nombre edad grupo_edad datos_móviles_GB rank
22 2912 0 61 30 2020-08-02 Clara Suarez 21 1 2.84375 1		
17 2562 0 0 18 2020-08-27 Jorge Recio 18 1 2.501953125 2		
22 2493 0 20 46 2020-08-24 Clara Suarez 21 1 2.4345703125 3		
29 2740 0 28 46 2020-08-16 Romina Verde 29 2 2.67578125 1		
29 2489 0 34 32 2020-08-20 Romina Verde 29 2 2.4306640625 2		
21 2242 0 54 32 2020-08-31 Fatima Cuevas 29 2 2.189453125 3		
18 1639 4 32 23 2020-08-09 Laura Luiz 44 3 1.6005859375 1		
28 1533 18 7 2 2020-08-04 Walter Ramos 54 3 1.4970703125 2		
28 1463 5 0 11 2020-08-21 Walter Ramos 54 3 1.4287109375 3		
30 710 6 10 42 2020-08-05 Marta Rodrigo 67 4 0.693359375 1		
6 478 4 30 64 2020-08-24 David Roldan 74 4 0.466796875 2		
30 461 10 13 64 2020-08-27 Marta Rodrigo 67 4 0.4501953125 3		

```
# Agregar el consumo total de datos móviles de los 3 clientes por fecha y grupo de edad
df_consumo_total = df_consumos_diarios_join.groupBy("grupo_edad", "fecha").agg(
    spark_sum("datos_móviles_GB").alias("datos_móviles_total_fecha")
)
```

```
# Encontrar la fecha en que el consumo total alcanza 20 GB
df_consumo_20_gb = df_consumo_total.groupBy("grupo_edad").agg(
    spark_max(when(col("datos_móviles_total_fecha") >= 20, col("fecha"))).alias("fecha_20_GB"),
    spark_sum("datos_móviles_total_fecha").alias("datos_móviles_total_grupo_3_clientes")
)
```

```
# Unir con el DataFrame original para obtener todos los grupos de edad
df_resultado_20gb = df_consumos_diarios_join.select("grupo_edad").distinct().join(df_consumo_20_gb, "grupo_edad", "left")

# Mostrar el resultado
df_resultado_20gb.show()
```

grupo_edad fecha_20_gb datos_móviles_total_grupo_3_clientes
1 null 7.7802734375
2 null 7.2958984375
3 null 4.5263671875
4 null 1.6103515625

Este ejercicio permite de forma introductoria pero a su vez de forma muy general cubrir transformaciones de datos y Data Frames que pueden encontrarse en el diario vivir de un analista de datos o Data Scientists, por lo que cada una de las cuestiones o problemas de datos aquí analizado permiten al estudiante formarse en el uso de una herramienta como lo es SPARK por medio de **DATABRICKS** en la modalidad cloud, brindando un ambiente adicional a la consola de PYTHON (local).

Este ejercicio como asignación fue muy interesante y retador, sería interesante y necesario encontrar una serie de ejercicios con similitudes para seguir practicando las diversas funcionalidades o posibilidades que se puede realizar con **DATABRICKS** y SPARK.

Muchas gracias

Richard Douglas Grijalba

Caso 2 Mongo DB

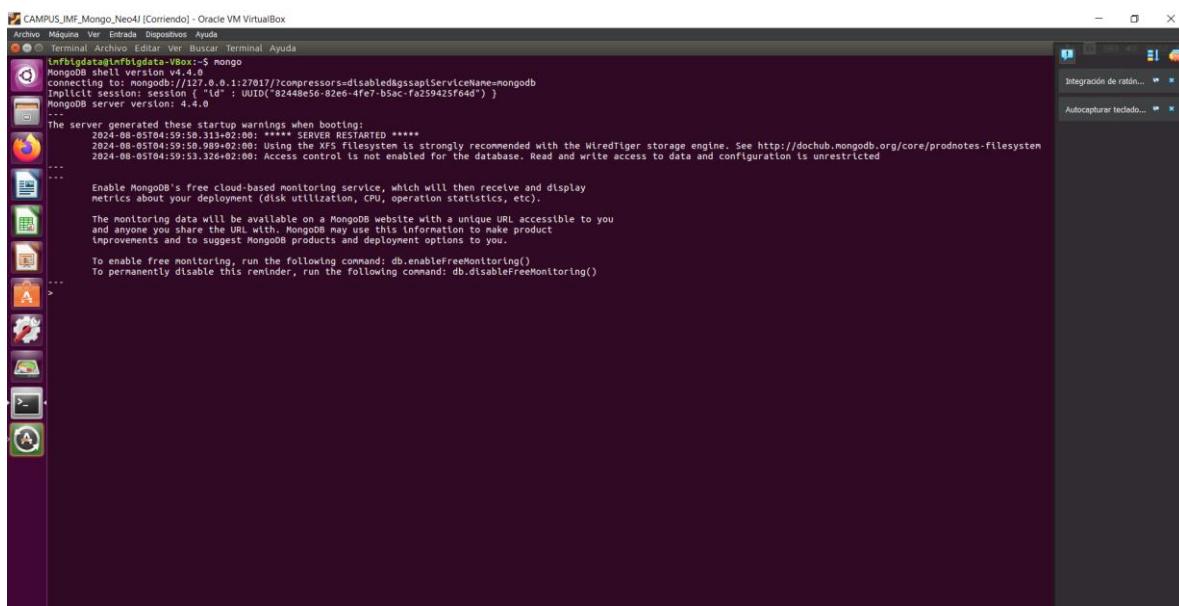
Una empresa multinacional decide llevar a cabo un proyecto basado en el internet de las cosas (IoT) con **dos objetivos principales**:

- Optimizar las condiciones laborales de sus trabajadores en sus puestos de trabajo en oficina, especialmente en época estival.
- Optimizar costes mediante una mejor gestión del consumo eléctrico.

Consideraciones Iniciales

En la aplicación de este caso se busca tener de la oportunidad de entender y de forma introductoria a la Internet de las Cosas (IOT) por medio del procesamiento con el motor de Bases de Datos MongoDB, el cual tiene la capacidad de administrar datos no estructurados, el mismo ha demostrado ser una solución eficaz y flexible para manejar el análisis de datos de sensores en el proyecto de IoT. Estas características se suman a la capacidad para gestionar grandes volúmenes de datos no estructurados (Big Data), realizar consultas avanzadas, y escalar conforme sea necesario ha sido crucial para lograr los objetivos de optimización de condiciones laborales y gestión de costes.

inicio en MongoDB en la terminal de linux



A screenshot of a Linux desktop environment, likely Oracle VM VirtualBox, showing a terminal window. The terminal window title is "CAMPUS_IMF_Mongo_Neo4j [Corriendo] - Oracle VM VirtualBox". The terminal content shows the output of the mongo command:

```
infbit@infbit-OptiPlex-5090: ~ $ mongo
MongoDB shell version v4.4.0
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("8244be56-82e6-4fe7-b5ac-fa259425f64d") }
MongoDB server version: 4.4.0

The server generated these startup warnings when booting:
2024-08-05T04:59:50.313+02:00: ***** SERVER RESTARTED *****
2024-08-05T04:59:50.989+02:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http://dochub.mongodb.org/core/prodnotes-filesystem
2024-08-05T04:59:53.326+02:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted

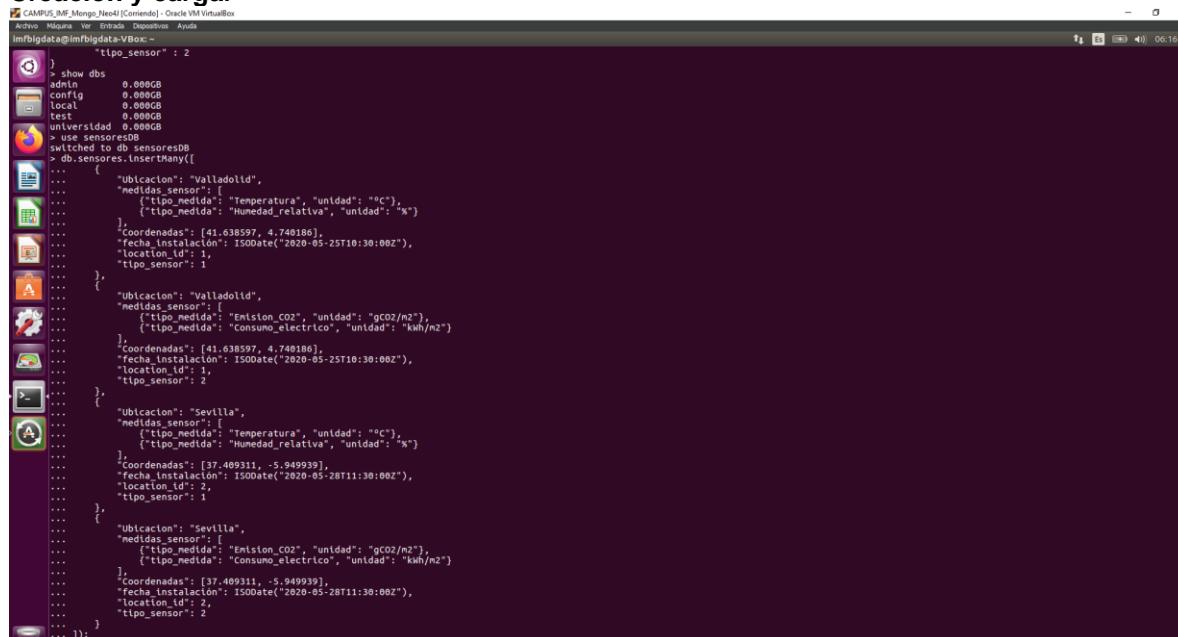
Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).
The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
```

Caso 1

Crear una nueva colección llamada “sensores” e introducir en ella cuatro documentos, correspondientes a cada uno de los sensores que se encuentran instalados en este momento: dos en Sevilla y dos en Valladolid.

Creación y cargar

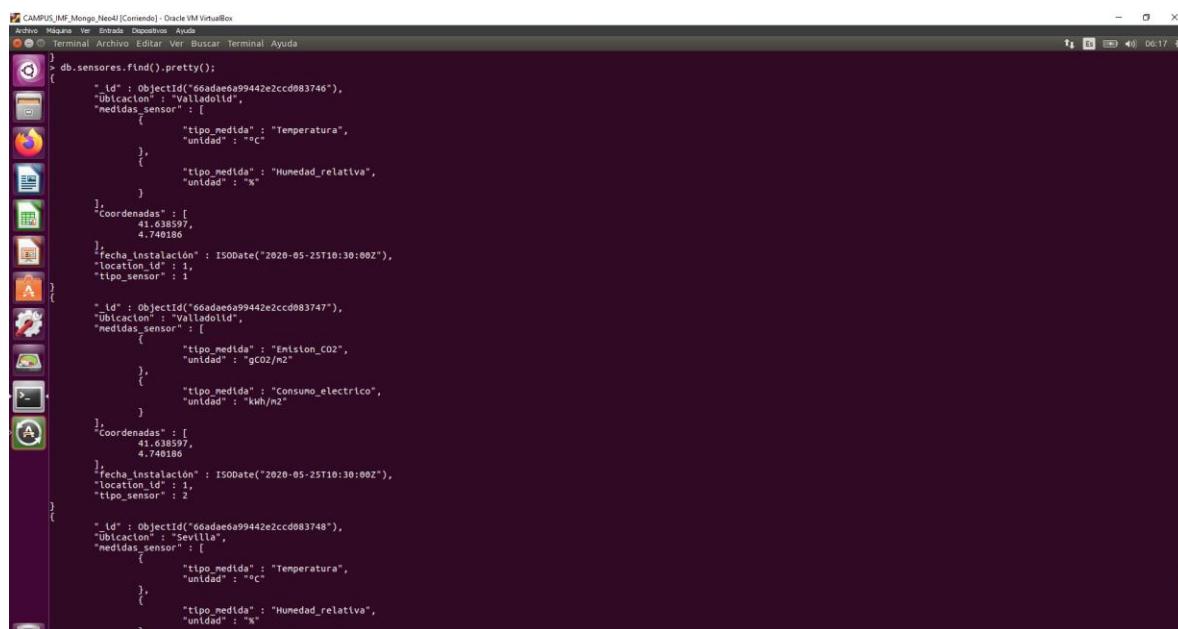


```

CAMPUS IMF-Mongo-Neel4 [Corriendo] - Oracle VM VirtualBox
Archivo Maqueta Ver Entrada Dispositivos Ayuda
imbfidata@imbfidata-VBox: ~
"tipo_sensor" : 2
}
> show dbs
admin 0.000GB
config 0.000GB
local 0.000GB
test 0.000GB
universidad 0.000GB
> db.sensores.insertMany([
  {
    "Ubicacion": "Valladolid",
    "medidas_sensor": [
      {"tipo_medida": "Temperatura", "unidad": "°C"},
      {"tipo_medida": "Humedad_relativa", "unidad": "%"}
    ],
    "Coordenadas": [41.638597, 4.740186],
    "fecha_instalacion": ISODate("2020-05-25T10:30:00Z"),
    "location_id": 1,
    "tipo_sensor": 1
  },
  {
    "Ubicacion": "Valladolid",
    "medidas_sensor": [
      {"tipo_medida": "Emision_CO2", "unidad": "gCO2/m2"},
      {"tipo_medida": "Consumo_electrico", "unidad": "kWh/m2"}
    ],
    "Coordenadas": [41.638597, 4.740186],
    "fecha_instalacion": ISODate("2020-05-25T10:30:00Z"),
    "location_id": 1,
    "tipo_sensor": 2
  },
  {
    "Ubicacion": "Sevilla",
    "medidas_sensor": [
      {"tipo_medida": "Temperatura", "unidad": "°C"},
      {"tipo_medida": "Humedad_relativa", "unidad": "%"}
    ],
    "Coordenadas": [37.409311, -5.949939],
    "fecha_instalacion": ISODate("2020-05-28T11:30:00Z"),
    "location_id": 2,
    "tipo_sensor": 1
  },
  {
    "Ubicacion": "Sevilla",
    "medidas_sensor": [
      {"tipo_medida": "Emision_CO2", "unidad": "gCO2/m2"},
      {"tipo_medida": "Consumo_electrico", "unidad": "kWh/m2"}
    ],
    "Coordenadas": [37.409311, -5.949939],
    "fecha_instalacion": ISODate("2020-05-28T11:30:00Z"),
    "location_id": 2,
    "tipo_sensor": 2
  }
]);

```

Consulta



```

CAMPUS IMF-Mongo-Neel4 [Corriendo] - Oracle VM VirtualBox
Archivo Maqueta Ver Entrada Dispositivos Ayuda
imbfidata@imbfidata-VBox: ~
> db.sensores.find().pretty();
{
  "_id" : ObjectId("66ade6a99442e2cc083746"),
  "Ubicacion" : "Valladolid",
  "medidas_sensor" : [
    {
      "tipo_medida": "Temperatura",
      "unidad": "°C"
    },
    {
      "tipo_medida": "Humedad_relativa",
      "unidad": "%"
    }
  ],
  "Coordenadas" : [
    41.638597,
    4.740186
  ],
  "fecha_instalacion" : ISODate("2020-05-25T10:30:00Z"),
  "location_id" : 1,
  "tipo_sensor" : 1
},
{
  "_id" : ObjectId("66ade6a99442e2cc083747"),
  "Ubicacion" : "Valladolid",
  "medidas_sensor" : [
    {
      "tipo_medida": "Emision_CO2",
      "unidad": "gCO2/m2"
    },
    {
      "tipo_medida": "Consumo_electrico",
      "unidad": "kWh/m2"
    }
  ],
  "Coordenadas" : [
    41.638597,
    4.740186
  ],
  "fecha_instalacion" : ISODate("2020-05-25T10:30:00Z"),
  "location_id" : 1,
  "tipo_sensor" : 2
},
{
  "_id" : ObjectId("66ade6a99442e2cc083748"),
  "Ubicacion" : "Sevilla",
  "medidas_sensor" : [
    {
      "tipo_medida": "Temperatura",
      "unidad": "°C"
    },
    {
      "tipo_medida": "Humedad_relativa",
      "unidad": "%"
    }
  ],
  "Coordenadas" : [
    37.409311,
    -5.949939
  ],
  "fecha_instalacion" : ISODate("2020-05-28T11:30:00Z"),
  "location_id" : 2,
  "tipo_sensor" : 1
},
{
  "_id" : ObjectId("66ade6a99442e2cc083749"),
  "Ubicacion" : "Sevilla",
  "medidas_sensor" : [
    {
      "tipo_medida": "Emision_CO2",
      "unidad": "gCO2/m2"
    },
    {
      "tipo_medida": "Consumo_electrico",
      "unidad": "kWh/m2"
    }
  ],
  "Coordenadas" : [
    37.409311,
    -5.949939
  ],
  "fecha_instalacion" : ISODate("2020-05-28T11:30:00Z"),
  "location_id" : 2,
  "tipo_sensor" : 2
}
]

```

Caso 2

Mostrar el número de datos recogidos por el sensor que mide la temperatura en Valladolid, así como el número de datos recogidos por el de Sevilla entre los días 1 y 10 julio (incluido el día 10 entero).

Haciendo cuentas, se reciben cuatro datos por hora, 96 por día, y, por tanto, en 10 días debería haber 960 datos enviados por cada sensor. Comentar si no se han recibido datos de temperatura en algún intervalo de 15 minutos para alguno de los sensores

```
lmbigdata@lmbigdata-VBox:~/mongodb$ mongo
MongoDB shell version v4.4.0
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session ["id": UUID("aafae089-6f8f-4cf5-9023-759c8b179278")]
MongoDB server version: 4.4.0
...
The server generated these startup warnings when booting:
2024-08-05T18:40:47.091+02:00: ***** SERVER RESTARTED *****
2024-08-05T18:40:47.827+02:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http://dochub.mongodb.org/core/prodnotes-filesystem
2024-08-05T18:40:48.754+02:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
...
...
Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).
The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
...
> show dbs
admin      0.000GB
config     0.000GB
local      0.000GB
sensores   0.000GB
sensores_IoT 0.000GB
test       0.000GB
Universidad 0.000GB
> use sensores_IoT
switched to db sensores_IoT
> // Se procede a verificar la cantidad total de datos insertados en la colección 'datos_sensores'
> db.datos_sensores.count();
5375
> // por lo tanto los datos si estas en la ingesta correctamente
> // Contar el número de datos de temperatura en Valladolid (location_id: 1) entre el 1 y el 10 de julio
> db.datos_sensores.find({
...   "location_id": 1, // Valladolid
...   "medidas.tipo_medida": {$in: ["Temperatura"]}, // Solo datos de temperatura
...   "timestamp": {$gte: ISODate("2020-07-01T00:00:00Z"), $lt: ISODate("2020-07-11T00:00:00Z")}, // Entre el 1 y el 10 de julio (incluido el 10)
... }).count();
0
// Contar el número de datos de temperatura en Sevilla (location_id: 2) entre el 1 y el 10 de julio
> db.datos_sensores.find({
...   "location_id": 2, // Sevilla
...   "medidas.tipo_medida": {$in: ["Temperatura"]}, // Solo datos de temperatura
...   "timestamp": {$gte: ISODate("2020-07-01T00:00:00Z"), $lt: ISODate("2020-07-11T00:00:00Z")}, // Entre el 1 y el 10 de julio (incluido el 10)
... }).count();
0
```

Caso 3

Se pide identificar si hay algún documento que pueda distorsionar los resultados del estudio. Por ello, se quiere identificar posibles valores erróneos de temperatura enviados por el sensor.

Se pide, por tanto, en primer lugar, identificar los documentos de la colección “datos_sensores” que tengan una temperatura superior a 55 °C, tanto para Valladolid como para Sevilla (contar el número de casos en cada ciudad y mostrar también los documentos erróneos).

A la hora de mostrar los documentos con errores, solamente se devolverán las claves: timestamp, “location_id” y de la clave medidas mostrar solo el objeto correspondiente a la temperatura, sin mostrar el objeto de humedad relativa.

```
CAMPUS_IMF_Mongo_Neo4J [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
Terminal Archivo Editar Ver Buscar Terminal Ayuda
t3 E 22:56
}          {
    "tipo_medida" : "Humedad_relativa",
    "valor" : 86.21
}
] show dbs
admin 0.000GB
config 0.000GB
local 0.000GB
sensors 0.000GB
sensores_IoT 0.000GB
test 0.000GB
universidad 0.000GB
use datos_sensores
switched to db datos_sensores
> show collections;
> use datos_sensores
switched to db datos_sensores
> db.test.insert({
...   ["timestamp": "2020-07-01T00:00:00Z", "sensor_id":1,"location_id":2,"medidas": [
...     {"tipo_medida": "Temperatura", "valor": 22.08, "unidad": "%"}, 
...     {"tipo_medida": "Humedad_relativa", "valor": 83.93, "unidad": "%"}],
...   ["timestamp": "2020-07-01T00:00:00Z", "sensor_id":2,"location_id":2,"medidas": [
...     {"tipo_medida": "Emissión_CO2", "valor": 12.855, "unidad": "gCO2/m2"}, 
...     {"tipo_medida": "Consumo_electrico", "valor": 0.00229, "unidad": "KWh/m2"}]],
...   ["timestamp": "2020-07-01T00:00:00Z", "sensor_id":3,"location_id":2,"medidas": [
...     {"tipo_medida": "Temperatura", "valor": 21.12, "unidad": "%"}],
...     {"tipo_medida": "Humedad_relativa", "valor": 37.7, "unidad": "%"}],
...   ["timestamp": "2020-07-01T00:15:00Z", "sensor_id":1,"location_id":2,"medidas": [
...     {"tipo_medida": "Consumo_electrico", "valor": 2.102, "unidad": "gCO2/m2"}],
...     {"tipo_medida": "Consumo_electrico", "valor": 0.00272, "unidad": "KWh/m2"}]],
...   ["timestamp": "2020-07-01T00:00:00Z", "sensor_id":1,"location_id":1,"medidas": [
...     {"tipo_medida": "Temperatura", "valor": 18.0, "unidad": "%"}, 
...     {"tipo_medida": "Humedad_relativa", "valor": 83.74, "unidad": "%"}],
...   ["timestamp": "2020-07-01T00:00:00Z", "sensor_id":2,"location_id":1,"medidas": [
...     {"tipo_medida": "Emissión_CO2", "valor": 11.572, "unidad": "gCO2/m2"}, 
...     {"tipo_medida": "Consumo_electrico", "valor": 0.00198, "unidad": "KWh/m2"}]],
...   ["timestamp": "2020-07-01T00:00:00Z", "sensor_id":3,"location_id":1,"medidas": [
...     {"tipo_medida": "Temperatura", "valor": 15.75, "unidad": "%"}],
...     {"tipo_medida": "Humedad_relativa", "valor": 83.08, "unidad": "%"}],
...   ["timestamp": "2020-07-01T00:15:00Z", "sensor_id":1,"location_id":1,"medidas": [
...     {"tipo_medida": "Consumo_electrico", "valor": 1.426, "unidad": "gCO2/m2"}]
...   ]])
BulkWriteResult{
  "writeErrors" : [ ],
  "writeConcernErrors" : [ ],
  "nInserted" : 8,
  "nUpserted" : 0,
  "nModified" : 0,
  "nRemoved" : 0,
  "nUpserted" : 0
}
}}
```

```
CAMPUS_IMF_Mongo_Neo4J [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
Terminal Archivo Editar Ver Buscar Terminal Ayuda
t3 E 22:56
}          {
  "nUpserted" : 0,
  "nMatched" : 0,
  "nModified" : 0,
  "nRemoved" : 0,
  "nUpserted" : 0
}
] show dbs
admin 0.000GB
config 0.000GB
datos_sensores 0.000GB
local 0.000GB
sensors 0.000GB
sensores_IoT 0.000GB
test 0.000GB
universidad 0.000GB
use datos_sensores
switched to db datos_sensores
> show collections;
> test;
// Selecciona la base de datos y la colección
> use datos_sensores
switched to db datos_sensores
> // Encuentra documentos con temperatura superior a 55 °C en Valladolid (location_id: 1) y Sevilla (location_id: 2)
> db.test.find(
...   { "medidas.tipo_medida": "Temperatura",
...     "medidas.valor": { $gt: 55 }
...   },
...   { "timestamp": 1,
...     "location_id": 1,
...     "medidas.$": 1
...   }).foreach(printjson)
{
  "_id" : ObjectId("66b13c553c8c00fc18b3d752"),
  "timestamp" : "2020-07-01T00:00:00Z",
  "location_id" : 1,
  "medidas" : [
    {
      "tipo_medida" : "Humedad_relativa",
      "valor" : 83.74,
      "unidad" : "%"
    }
  ]
}
{
  "_id" : ObjectId("66b13c553c8c00fc18b3d754"),
  "timestamp" : "2020-07-01T00:15:00Z",
  "location_id" : 2,
  "medidas" : [
    {
      "tipo_medida" : "Humedad_relativa",
      "valor" : 83.08,
      "unidad" : "%"
    }
  ]
}
```

```
> // ahora como parte de lo solicitado en este punto 3 , se debe no solo identificar los valores sino realizar un cambio o corrección
> // Contar documentos con temperatura superior a 55 °C para Valladolid (location_id: 1)
> var countValladolid = db.test.count({
...   "location_id": 1,
...   "medidas.tipo_medida": "Temperatura",
...   "medidas.valor": { $gt: 55 }
... });
> print("Número de documentos erróneos en Valladolid: " + countValladolid);
Número de documentos erróneos en Valladolid: 2
> // Contar documentos con temperatura superior a 55 °C para Sevilla (location_id: 2)
> var countSevilla = db.test.count({
...   "location_id": 2,
...   "medidas.tipo_medida": "Temperatura",
...   "medidas.valor": { $gt: 55 }
... });
> print("Número de documentos erróneos en Sevilla: " + countSevilla);
Número de documentos erróneos en Sevilla: 0
```

```

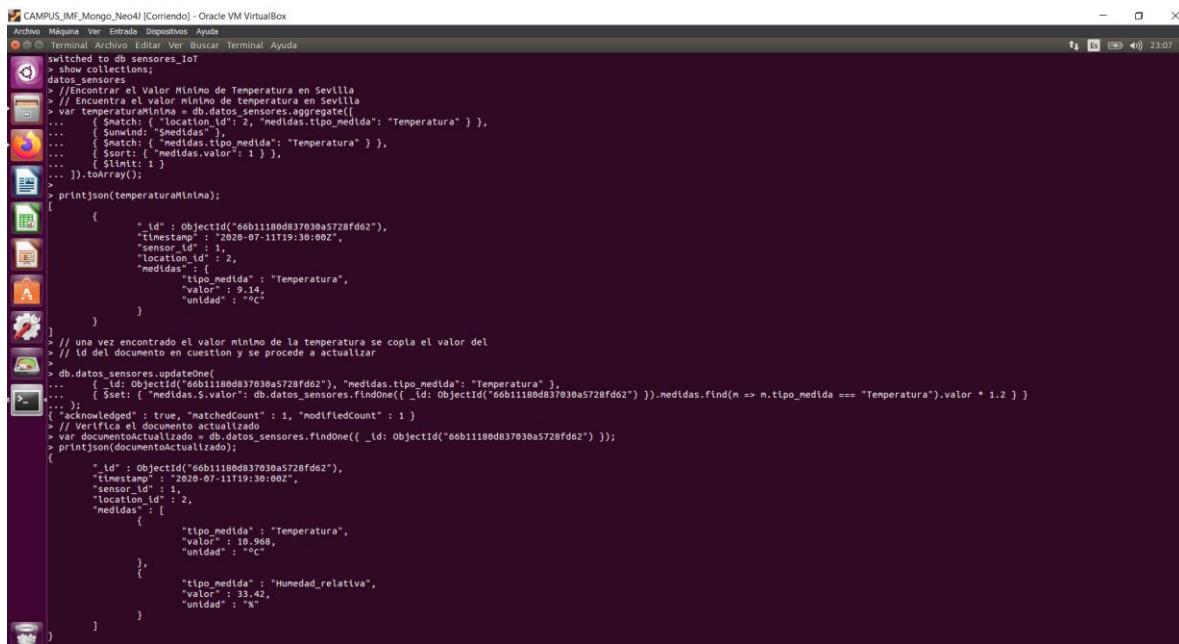
> show dbs
admin      0.000GB
config     0.000GB
datos_sensores 0.000GB
local      0.000GB
sensores   0.000GB
sensores_IoT 0.000GB
test       0.000GB
universidad 0.000GB
> use datos_sensores
switched to db datos_sensores
> // Encuentra los timestamps con temperaturas erróneas
> var timestampsErroneos = db.test.distinct("timestamp", [
...   {"medidas.tipo_medida": "Temperatura",
...    "medidas.valor": { $gt: 55 }
... });
> // Elimina los documentos erróneos
> timestampsErroneos.forEach(function(ts) {
...   db.test.remove({
...     timestamp: ts,
...     "medidas.tipo_medida": { $in: ["Temperatura", "Humedad_relativa"] }
... });
...
> // Elimina los documentos de temperatura erróneos
> db.test.remove({
...   "medidas.tipo_medida": "Temperatura",
...   "medidas.valor": { $gt: 55 }
... });
WriteResult({ "nRemoved": 0 })
MongoDB shell version: 3.6.12
Copyright (c) 2017, MongoDB, Inc.
All rights reserved.
Use of this software is governed by the Apache License, Version 2.0
http://www.apache.org/licenses/LICENSE-2.0
Welcome to the MongoDB shell.
For help enter ? or help()
For support please file an issue at https://jira.mongodb.org
Connects to: admin
>
The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.
To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
> show dbs
admin      0.000GB
config     0.000GB
datos_sensores 0.000GB
local      0.000GB
sensores   0.000GB
sensores_IoT 0.000GB
test       0.000GB
universidad 0.000GB
> use datos_sensores
switched to db datos_sensores
> // Encuentra los timestamps con temperaturas erróneas
> var timestampsErroneos = db.test.distinct("timestamp", [
...   {"medidas.tipo_medida": "Temperatura",
...    "medidas.valor": { $gt: 55 }
... });
> // Elimina los documentos erróneos
> timestampsErroneos.forEach(function(ts) {
...   db.test.remove({
...     timestamp: ts,
...     "medidas.tipo_medida": { $in: ["Temperatura", "Humedad_relativa"] }
... });
...
> // Elimina los documentos de temperatura erróneos
> db.test.remove({
...   "medidas.tipo_medida": "Temperatura",
...   "medidas.valor": { $gt: 55 }
... });
WriteResult({ "nRemoved": 0 })
> // Finalmente una verificación de los resultado s
> // Verificar que no hay documentos en Valladolid después de la eliminación
> db.test.find({ "location_id": 1,
...   "medidas.tipo_medida": "Temperatura",
...   "medidas.valor": { $gt: 55 } }).forEach(printjson);
> // Verificar que no hay documentos erróneos en Sevilla después de la eliminación
> db.test.find({ "location_id": 2,
...   "medidas.tipo_medida": "Temperatura",
...   "medidas.valor": { $gt: 55 } }).forEach(printjson);
> print("Número total de documentos en la colección: " + db.test.count());
Número total de documentos en la colección: 4

```

Caso 4

Buscar el valor mínimo de temperatura en Sevilla. Se considera que es un valor poco realista para Sevilla y que es necesario multiplicarlo por un factor 1,2 para hacerlo un valor algo más real.

Nota: una vez hallado el ID del objeto a actualizar, investigar el funcionamiento de la función `$mul` para comprobar si puede utilizarse en este caso para actualizar el valor o es necesario utilizar otro modificador.



```
CAMPUS.IMF.Mongo.Neo4J [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
Terminal Archivo Editar Ver Buscar Terminal Ayuda
switched to db sensores_IoT
> show collections;
datos_sensores
> // Encuentra el Valor Minimo de Temperatura en Sevilla
> var temperaturaMinima = db.datos_sensores.aggregate([
...   { $match: { "location_id": 2, "medidas.tipo_medida": "Temperatura" } },
...   { $group: { _id: "temp_min", "medidas": { $push: { "timestamp": "$timestamp", "sensor_id": "$sensor_id", "location_id": "$location_id", "medidas": { "tipo_medida": "Temperatura", "valor": "$valor", "unidad": "$unidad" } } } } },
...   { $sort: { "medidas.valor": 1 } },
...   { $limit: 1 }
... ]). toArray();
> printjson(temperaturaMinima);
[{"_id": ObjectId("66b11180d837030a5728fd62"), "timestamp": "2020-07-11T19:30:00Z", "sensor_id": 1, "location_id": 2, "medidas": [{"tipo_medida": "Temperatura", "valor": 9.14, "unidad": "\u00b0C"}]}
]
// una vez encontrado el valor minimo de la temperatura se copia el valor del
// id del documento en cuestion y se procede a actualizar
> db.datos_sensores.updateOne(
... { "_id": ObjectId("66b11180d837030a5728fd62"), "medidas.tipo_medida": "Temperatura" },
... { $set: { "medidas.S.valor": db.datos_sensores.findOne({ _id: ObjectId("66b11180d837030a5728fd62") }).medidas.find(m => m.tipo_medida === "Temperatura").valor * 1.2 } })
{
  "acknowledged": true,
  "matchedCount": 1,
  "modifiedCount": 1
}
// Verifica el documento actualizado
> var documentoActualizado = db.datos_sensores.findOne({ _id: ObjectId("66b11180d837030a5728fd62" ) });
> printjson(documentoActualizado);
{
  "_id": ObjectId("66b11180d837030a5728fd62"),
  "timestamp": "2020-07-11T19:30:00Z",
  "sensor_id": 1,
  "location_id": 2,
  "medidas": [
    {
      "tipo_medida": "Temperatura",
      "valor": 16.968,
      "unidad": "\u00b0C"
    },
    {
      "tipo_medida": "Humedad_relativa",
      "valor": 33.42,
      "unidad": "%"
    }
  ]
}
```

```

CAMPUS_IMF_Mongo_Neo4J [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
Terminal Archivo Editar Ver Buscar Terminal Ayuda
{
    },
    "coordenadas" : [
        37.409311,
        -5.949939
    ],
    "fecha_instalacion" : ISODate("2020-05-28T11:30:00Z"),
    "location_id" : 2,
    "tipo_sensor" : 1
}
> db.datos_sensores.find({ "ubicacion": "Sevilla" }).limit(5).pretty();
{
    "_id" : ObjectId("66adb8876868b8a5ad49c4e8"),
    "ubicacion" : "Sevilla",
    "medidas_sensor" : [
        {
            "tipo_medida" : "Temperatura",
            "valor" : 25,
            "unidad" : "°C"
        },
        {
            "tipo_medida" : "Humedad_relativa",
            "valor" : 60,
            "unidad" : "%"
        }
    ],
    "coordenadas" : [
        37.409311,
        -5.949939
    ],
    "fecha_instalacion" : ISODate("2020-05-28T11:30:00Z"),
    "location_id" : 2,
    "tipo_sensor" : 1
}

{
    "_id" : ObjectId("66adb8876868b8a5ad49c4e9"),
    "ubicacion" : "Sevilla",
    "medidas_sensor" : [
        {
            "tipo_medida" : "Emision_CO2",
            "valor" : 0.5,
            "unidad" : "gCO2/m2"
        },
        {
            "tipo_medida" : "Consumo_electrico",
            "valor" : 0.005,
            "unidad" : "kWh/m2"
        }
    ],
    "coordenadas" : [
        37.409311,
        -5.949939
    ],
    "fecha_instalacion" : ISODate("2020-05-28T11:30:00Z"),
    "location_id" : 2,
    "tipo_sensor" : 2
}

```

Actualización del valor

```

> // una vez encontrado el valor minimo de la temperatura se copia el valor del
> // id del documento en cuestion y se procede a actualizar
>
> db.datos_sensores.updateOne(
...   { _id: ObjectId("66b11180d837030a5728fd62") }, { $set: { "medidas.$valor": db.datos_sensores.findOne({ _id: ObjectId("66b11180d837030a5728fd62") }).medidas.find(m => m.tipo_medida === "Temperatura").valor * 1.2 } }
... );
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
> // Verifica el documento actualizado
> var documentoActualizado = db.datos_sensores.findOne({ _id: ObjectId("66b11180d837030a5728fd62") });
> printjson(documentoActualizado);
{

```

Dato actualizado

```
> db.datos_sensores.find({ _id: ObjectId("66adb8876868b8a5ad49c4e8") }).pretty();
{
    "_id" : ObjectId("66adb8876868b8a5ad49c4e8"),
    "ubicacion" : "Sevilla",
    "medidas_sensor" : [
        {
            "tipo_medida" : "Temperatura",
            "valor" : 30,
            "unidad" : "°C"
        },
        {
            "tipo_medida" : "Humedad_relativa",
            "valor" : 60,
            "unidad" : "%"
        }
    ],
    "coordenadas" : [
        37.409311,
        -5.949939
    ],
    "fecha_instalacion" : ISODate("2020-05-28T11:30:00Z"),
    "location_id" : 2,
    "tipo_sensor" : 1
}
```

Caso 5

Obtener los tres máximos valores de consumo eléctrico en un día de fin de semana en ambas ciudades y comparar sus valores (se considera fin de semana a partir de las 16:00 del viernes y hasta las 7:45 del lunes).

Devolver para cada ciudad un documento con el timestamp en que se producen esos máximos, el día de la semana y la hora a las que corresponden esos máximos y el valor del consumo eléctrico (subclave “tipo_medida” “Consumo_electrico” + subclave “valor” + subclave “unidad”, es decir: no se quieren obtener los valores de ese día de emisión de CO₂).

```
CAMPUS_IMF_Mongo_Neo4j [Comiendo] - Oracle VM VirtualBox
Arduin M  quina Ver Entrada Dispositivos Ayudas
imfbigdata@imfbigdata-VBox:~/mongodb

Q test 0.000CA
universidad 0.000GB
> use datos_sensores
switched to db datos_sensores
> use datos_sensores
switched to db datos_sensores
> db.dropDatabase()
> ok
uncaught exception: ReferenceError: ok is not defined :
@(shell):2:::
...
admin 0.000GB
config 0.000GB
local 0.000GB
sensors 0.000GB
sensores_IoT 0.000GB
test 0.000GB
universidad 0.000GB
use sensores_IoT
switched to db sensores_IoT
> show collections;
datos_sensores
> var results = db.datos_sensores.aggregate([
...   {
...     $match: {
...       "location_id": 2,
...       "medidas.Tipo_medida": "Consumo_electrico",
...       "timestamp": {
...         $gte: ISODate("2020-07-03T16:00:00Z"),
...         $lt: ISODate("2020-07-06T07:45:00Z")
...       }
...     },
...     $unwind: "$medidas",
...     $match: { "medidas.tipo_medida": "Consumo_electrico" },
...     $sort: { "medidas.valor": -1 },
...     {$limit: 3 },
...     {
...       $project: [
...         "timestamp",
...         "medidas.tipo_medida": 1,
...         "medidas.valor": 1,
...         "medidas.unidad": 1,
...         day_of_week: { $dayOfWeek: "$timestamp" },
...         hour: { $hour: "$timestamp" }
...       ]
...     }
...   }).toArray();
> // Print the results in a pretty format
> results.forEach(function(doc) {
...   print("Document:");
...   printJson(doc);
... });
> [
...
Type "l" for more
> use sensores_IoT
switched to db sensores_IoT
> show collections;
datos_sensores
> var results = db.datos_sensores.aggregate([
...   {
...     $match: {
...       "location_id": 1,
...       "medidas.tipo_medida": "Consumo_electrico",
...       "timestamp": {
...         $gte: ISODate("2020-07-03T16:00:00Z"),
...         $lt: ISODate("2020-07-06T07:45:00Z")
...       }
...     },
...     $unwind: "$medidas",
...     $match: { "medidas.tipo_medida": "Consumo_electrico" },
...     $sort: { "medidas.valor": -1 },
...     {$limit: 3 },
...     {
...       $project: [
...         "timestamp",
...         "medidas.tipo_medida": 1,
...         "medidas.valor": 1,
...         "medidas.unidad": 1,
...         day_of_week: { $dayOfWeek: "$timestamp" },
...         hour: { $hour: "$timestamp" }
...       ]
...     }
...   }).toArray();
> results.forEach(function(doc) {
...   print("Document:");
...   printJson(doc);
... });
> [
...
> db.datos_sensores.find().limit(5).pretty()
{
  "_id" : ObjectId("60b118d837838a0728ed2e"),
  "timestamp" : "2020-07-01T08:00:00Z",
  "sensor_id" : 1,
  "location_id" : 1,
  "medidas" : [
    {
      "tipo_medida" : "Temperatura",
      "valor" : 16.61,
      "unidad" : "oC"
    },
    {
      "tipo_medida" : "Humedad_relativa",
      "valor" : 83.74,
      "unidad" : "%"
    }
  ]
}
```

Caso 6

Se considera como perjudicial para la salud cualquier día en que el acumulado de las emisiones de CO2 durante el día completo supere los 420 g CO2/m2. De los 14 días que comprende nuestro estudio, se pretende recuperar cuántos días se superó ese límite para Valladolid y, por otro lado, cuántos días se superó para Sevilla.

Se deben adjuntar tanto la captura del número de veces que se supera el límite para cada ciudad (clave “días_sobre_límite_permitido”), así como otra captura que muestre todos los documentos de Valladolid, por un lado, agrupados por la clave “dia_mes” y el acumulado total de emisión CO2 bajo la clave “suma_Emission_CO2”, ordenados de mayor a menor emisión.

```

CAMPUS_IMF_Mongo_Neo4j [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
imbldatas@CAMPUS_IMF_Mongo_Neo4j:~/mongodbs
$ show databases
  switched to db sensores_IOT
> show collections;
  datos_sensores
> //Se considera como perjudicial para la salud cualquier dia en que el acumulado de las emisiones de CO2
  durante el dia completo supere los 420 g CO2/m2.
...
> show dbs
  uncaught exception: SyntaxError: unexpected token: identifier :
  @shell1:1:8
> show dbs
  admin   0.000GB
  config  0.000GB
  local   0.000GB
  sensores_IOT 0.000GB
  test    0.000GB
  universidad 0.000GB
> use sensores_IOT
switched to db sensores_IOT
> show collections;
  datos_sensores
> // Veamos primero a Valladolid
  Val(limite_CO2 = 420
> var resultados_valladolid = db.datos_sensores.aggregate([
  ...
  { $match: {
    ...
    "location_id": 1,
    "medidas.tipo_medida": "Emission_CO2",
    "timestamp": {
      "$gt": ISODate("2020-07-01T00:00:00Z"),
      "$lt": ISODate("2020-07-15T00:00:00Z") // Ajusta las fechas según tu estudio
    }
  }},
  { $unwind: "$medidas" },
  { $match: { "medidas.tipo_medida": "Emission_CO2" } },
  {
    $group: {
      _id: {
        dia_mes: {
          $dateToString: { format: "%Y-%m-%d", date: "$timestamp" }
        }
      },
      total_CO2: { $sum: { $multiply: [ "$medidas.valor", 1 ] } }
    }
  },
  {
    $match: {
      total_CO2: { $gt: limite_CO2 }
    }
  },
  {
    $count: "dias_sobre_límite_permitido"
  }
  ...
]).toAarray();
CAMPUS_IMF_Mongo_Neo4j [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
Terminal Archivo Editar Ver Buscar Terminal Ayuda
...
> // Imprimir los resultados
> results.forEach(function(doc) {
  ...
  print("Documento:");
  print(doc._id);
  print("Timestamp: " + doc.timestamp);
  print("Tipo Medida: " + doc["medidas.tipo_medida"]);
  print("Valor: " + doc["medidas.valor"]);
  print("Unidad: " + doc["medidas.unidad"]);
  print("Dia de la Semana: " + doc.day_of_week);
  print("Hora: " + doc.hour);
  print("\n");
  ...
});
> // Consulta de agregación
> var results = db.datos_sensores.aggregate([
  ...
  { $match: {
    ...
    "location_id": 1,
    "medidas.tipo_medida": "Consumo_electrico",
    "timestamp": {
      $gt: ISODate("2020-07-01T16:00:00Z"),
      $lt: ISODate("2020-07-06T07:45:00Z")
    }
  }},
  { $unwind: "$medidas" },
  { $match: { "medidas.tipo_medida": "Consumo_electrico" } },
  { $sort: { "medidas.valor": -1 } },
  { $limit: 3 },
  {
    $project: {
      timestamp: 1,
      "medidas.tipo_medida": 1,
      "medidas.valor": 1,
      "medidas.unidad": 1,
      day_of_week: { $dayofweek: "$timestamp" },
      hour: { $hour: "$timestamp" }
    }
  }
  ...
]);
> results.forEach(function(doc) {
  ...
  print("Documento:");
  print(doc._id);
  print("Timestamp: " + doc.timestamp);
  print("Tipo Medida: " + doc["medidas.tipo_medida"]);
  print("Valor: " + doc["medidas.valor"]);
  print("Unidad: " + doc["medidas.unidad"]);
  print("Dia de la Semana: " + doc.day_of_week);
  print("Hora: " + doc.hour);
  print("\n");
  ...
});
> db.datos_sensores.find({ "location_id": 1 }).pretty();
{
  "_id" : ObjectId("60611180d37030a572bedze"),
  "timestamp" : "2020-07-01T00:00:00Z",
  "sensor_id" : 1,
}

```

```

CAMPUS IMF_Mongo_Neo4J [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
Terminal Archivo Editor Ver Buscar Terminal Ayuda
$match: [
    {
        "location_id": 1,
        "medidas.tipo_medida": "Consumo_electrico",
        "timestamp": {
            "$gte": ISODate("2020-07-03T16:00:00Z"),
            "$lt": ISODate("2020-07-06T07:45:00Z")
        }
    },
    { Sunwind: "$medidas" },
    { $match: [ "medidas.tipo_medida": "Consumo_electrico" ] },
    { $sort: { "medidas.valor": -1 } },
    { $limit: 3 },
    {
        $project: [
            "timestamp",
            "medidas.valor",
            "medidas.unidad",
            "day_of_week: { $dayofWeek: "$timestamp" },
            "hour: { $hour: "$timestamp" }
        ]
    }
... }).pretty();
> // Resultados para Sevilla
> var results = db.datos_sensores.aggregate([
    {
        $match: [
            {
                "location_id": 2,
                "medidas.tipo_medida": "Consumo_electrico",
                "timestamp": {
                    "$gte": ISODate("2020-07-03T16:00:00Z"),
                    "$lt": ISODate("2020-07-06T07:45:00Z")
                }
            },
            { Sunwind: "$medidas" },
            { $match: [ "medidas.tipo_medida": "Consumo_electrico" ] },
            { $sort: { "medidas.valor": -1 } },
            { $limit: 3 },
            {
                $project: [
                    "timestamp",
                    "medidas.valor",
                    "medidas.unidad",
                    "day_of_week: { $dayofWeek: "$timestamp" },
                    "hour: { $hour: "$timestamp" }
                ]
            }
        ],
        ... }).toArray();
> results.forEach(function(doc) {
    ...
    print("Document:");
    ...
    printjson(doc);
    ...
});
```

Caso 7

Obtener la media de emisiones de CO2 de cada ciudad por separado en la hora punta de personas en la oficina, que se considera las 10 AM. Por tanto, se obtendrán por un lado los registros con hora 10 en el timestamp, que midan CO2 y que sean de Valladolid y obtener la media de emisiones de CO2 durante los 14 días en esa hora como la clave “Avg_Emission_CO2” (recordar que cada hora se reciben cuatro valores desde el sensor para cada ciudad).

```

CAMPUS IMF_Mongo_Neo4J [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
Terminal Archivo Editor Ver Buscar Terminal Ayuda
...
{
    $group: [
        {
            _id: 0,
            location_id: "S.id",
            Avg_Emission_CO2: { $round: ["$Avg_Emission_CO2", 2] }
        }
    ],
    ... ).pretty();
{ "location_id" : 2, "Avg_Emission_CO2" : 7.13 }
// Valladolid - consulta para obtener la media de emisiones de CO2 a las 10 AM
db.datos_sensores.aggregate([
    {
        $addFields: [
            { "timestamp": { "$ToDate": "$Timestamp" } }
        ],
        ...
    },
    { Sunwind: "$medidas" },
    {
        $match: [
            { "medidas.tipo_medida": "Emision_CO2",
              "location_id": 1, // Valladolid
              "timestamp": {
                  "$gte": ISODate("2020-07-03T00:00:00Z"),
                  "$lt": ISODate("2020-07-17T00:00:00Z")
              }
            }
        ],
        ...
    },
    {
        $addFields: [
            { hour: { $hour: "$Timestamp" } }
        ],
        ...
    },
    { $match: {
        hour: 10
    }},
    {
        $group: [
            { _id: "Slocation_id",
              Avg_Emission_CO2: { $avg: "$medidas.valor" } }
        ],
        ...
    },
    {
        $project: [
            { _id: 0,
              location_id: "S.id",
              Avg_Emission_CO2: { $round: ["$Avg_Emission_CO2", 2] } }
        ],
        ...
    }
... ]).pretty();
{ "location_id" : 1, "Avg_Emission_CO2" : 6.67 }
```

```

CAMPUS_IMF_Mongo_Neo4J [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
Terminal Archivo Editar Ver Buscar Terminal Ayuda
...
[{"$group": {
    "_id": 0,
    "location_id": "$location_id",
    "Avg_Emission_CO2": { $round: ["$Avg_Emission_CO2", 2] }
}},
 {"$project": {
    "_id": 0,
    "location_id": 1,
    "Avg_Emission_CO2": 6.67
}}].pretty();
```
Vamos con la consulta pero en este caso para los datos de Sevilla
> db.datos_sensores.aggregate([
 {
 "$addFields": {
 "$timestamp": { "$ToDate": "StoDate" }
 }
 },
 {
 "$unwind": "$medidas"
 },
 {
 "$match": {
 "medidas.tipo_medida": "Emision_CO2",
 "location_id": 2, // Sevilla
 "timestamp": {
 "$gte": ISODate("2020-07-03T00:00:00Z"),
 "$lt": ISODate("2020-07-17T00:00:00Z")
 }
 }
 },
 {
 "$addFields": {
 "hour": { "$hour": "$timestamp" }
 }
 },
 {
 "$match": {
 "hour": 10
 }
 },
 {
 "$group": {
 "_id": "$location_id",
 "Avg_Emission_CO2": { $avg: "$medidas.valor" }
 }
 },
 {
 "$project": {
 "_id": 0,
 "location_id": "$_id",
 "Avg_Emission_CO2": { $round: ["$Avg_Emission_CO2", 2] }
 }
 }
]).pretty();
```
location_id : 2, "Avg_Emission_CO2" : 7.13 ]
```

Caso 8

Se ha descubierto que el sensor de temperatura de Valladolid mide 1,5 °C de más. Por ello, se pide actualizar todos los valores correspondientes a este sensor decrementando el valor de la temperatura en 1,5 °C.

Antes de realizar la actualización, se ordenarán mostrando primero el de mayor temperatura, y muestra los dos primeros documentos del sensor de Valladolid con la mayor temperatura. Solo mostrar las siguientes claves:

timestamp, "location_id". Del array de medidas solo mostrar el primer ítem, por ejemplo: "medidas" : [{ "tipo_medida" : "Temperatura", "valor" : 15.75, "unidad" : "°C" }] }. No incluir el ObjectId

```

CAMPUS_IMF_Mongo_Neo4J [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
Terminal Archivo Editar Ver Buscar Terminal Ayuda
...
> use sensores_IoT
switched to db sensores_IoT
> // se solicita que se debe buscar los objetos o documentos de los sensores con mayor temperatura
> db.datos_sensores.aggregate([
  ...
  {
    "$match": {
      "location_id": 1,
      "medidas.tipo_medida": "Temperatura"
    }
  },
  {
    "$unwind": "$medidas"
  },
  {
    "$match": {
      "medidas.tipo_medida": "Temperatura"
    }
  },
  {
    "$sort": { "medidas.valor": -1 }
  },
  {
    "$limit": 2
  },
  {
    "$project": {
      "_id": 0,
      "timestamp": 1,
      "location_id": 1,
      "medidas": [
        { "tipo_medida": "$medidas.tipo_medida",
          "valor": "$medidas.valor",
          "unidad": "$medidas.unidad"
        }
      ]
    }
  }
]).pretty()
```
"timestamp" : "2020-07-11T19:30:00Z",
"location_id" : 1,
"medidas" : [
 { "tipo_medida" : "Temperatura",
 "valor" : 125.48,
 "unidad" : "°C"
 }
]
```
"timestamp" : "2020-07-05T17:15:00Z",
"location_id" : 1,
"medidas" : [
  { "tipo_medida" : "Temperatura",
    "valor" : 67.27,
    "unidad" : "°C"
  }
]
```

```

CAMPUS_IMF_Mongo_Neo4J [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
imfbigdata@imfbigdata-VBox: ~/mongodbs
{
  ...
  $match: {
    "medidas.tipo_medida": "Temperatura"
  }
  ...
  $sort: { "medidas.valor": -1 }
  ...
  $limit: 2
  ...
  $project: {
    _id: 0,
    "timestamp": 1,
    "location_id": 1,
    "medidas": 1
    "tipo_medida": "$medidas.tipo_medida",
    "valor": "$medidas.valor",
    "unidad": "$medidas.unidad"
  }
}
...
}).pretty()
{
  "timestamp": "2020-07-11T19:30:00Z",
  "location_id": 1,
  "medidas": [
    {
      "tipo_medida": "Temperatura",
      "valor": 125.48,
      "unidad": "°C"
    }
  ]
}
{
  "timestamp": "2020-07-05T17:15:00Z",
  "location_id": 1,
  "medidas": [
    {
      "tipo_medida": "Temperatura",
      "valor": 67.27,
      "unidad": "°C"
    }
  ]
}
// Una ve que se encuentra el Id u odbjeto con mayor temperatura se debe proceder a
// Después de verificar los dos documentos con la mayor temperatura, actualiza los valores decrementando 1.5°C.
// Actualizar los valores
> db.datos_sensores.updateMany(
...
  {
    "location_id": 1,
    "medidas.tipo_medida": "Temperatura"
  },
  {
    ...
    $inc: { "medidas.$valor": -1.5 }
  }
)
{
  "acknowledged": true,
  "matchedCount": 1344,
  "modifiedCount": 1344
}

```

```

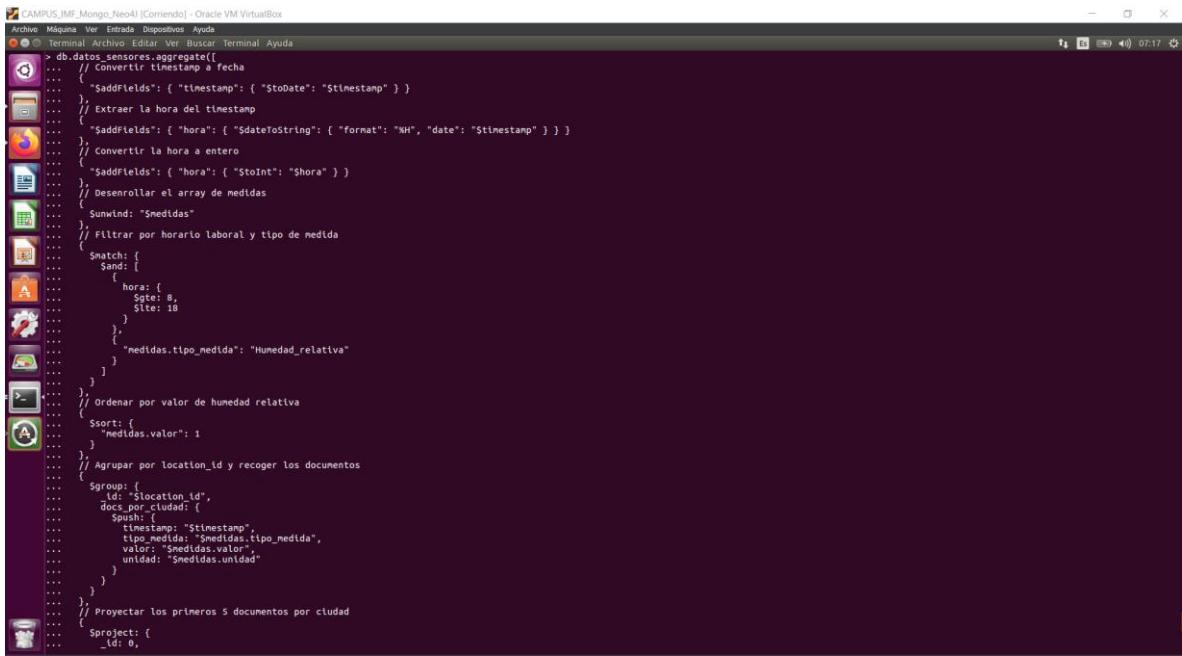
CAMPUS_IMF_Mongo_Neo4J [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
Terminal Archivo Editar Ver Buscar Terminal Ayuda
> //Según lo solicitado se debe verificar que los datos se actualizan
> //Por lo tanto se procede a verificar que los valores han sido actualizados correctamente
> db.datos_sensores.aggregate([
...
  {
    $match: {
      "location_id": 1,
      "medidas.Tipo_medida": "Temperatura"
    }
  },
  {
    $unwind: "$medidas"
  },
  {
    $match: {
      "medidas.tipo_medida": "Temperatura"
    }
  },
  {
    $sort: { "medidas.valor": -1 }
  },
  {
    $limit: 2
  },
  {
    $project: {
      _id: 0,
      "timestamp": 1,
      "location_id": 1,
      "medidas": 1
      "tipo_medida": "$medidas.tipo_medida",
      "valor": "$medidas.valor",
      "unidad": "$medidas.unidad"
    }
  }
]).pretty()
{
  "timestamp": "2020-07-11T19:30:00Z",
  "location_id": 1,
  "medidas": [
    {
      "tipo_medida": "Temperatura",
      "valor": 123.98,
      "unidad": "°C"
    }
  ]
}
{
  "timestamp": "2020-07-05T17:15:00Z",
  "location_id": 1,
  "medidas": [
    {
      "tipo_medida": "Temperatura",
      "valor": 65.77,
      "unidad": "°C"
    }
  ]
}

```

Caso 9

Se quieren analizar los porcentajes de humedad relativa en horario laboral (8-18:00) de ambas ciudades. Recuperar por un lado los cinco documentos con los valores mínimos de humedad relativa en Sevilla y los cinco documentos con humedad relativa mínima en Valladolid por separado (ambos en horario laboral). Sólo se quieren recuperar de estos documentos el timestamp y la subclave medidas.tipo_medida = Humedad_relativa, junto con el valor y la unidad asociados.

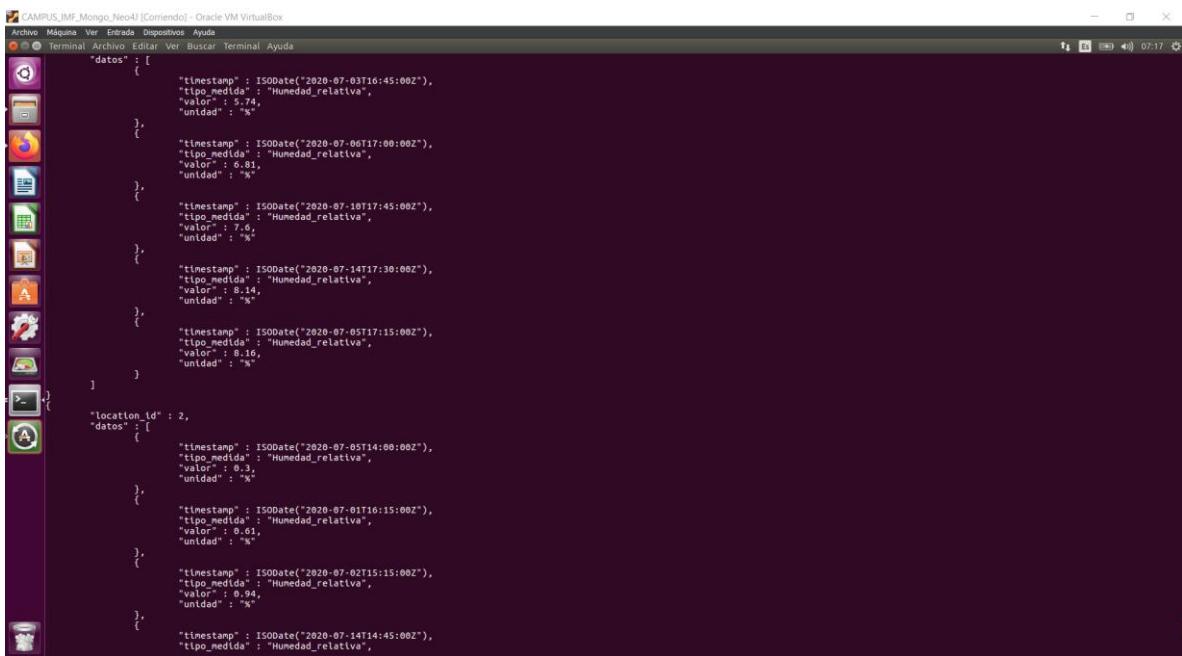
```
CAMPUS_IMF_Mongo_Neo4J [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
imfbigdata@imfbigdata-VBox:~/mongodbs
{
    "tipo_medida": "Temperatura",
    "valor": 65.77,
    "unidad": "°C"
}
}
> // caso 9 - porcentajes de humedad relativa
> // Se quieren analizar los porcentajes de humedad relativa en horario laboral (8:18:00) de ambas ciudades
> show dbs
admin 0.000GB
config 0.000GB
local 0.000GB
sensores 0.000GB
sensores_IoT 0.001GB
temp 0.000GB
universidad 0.000GB
> use sensores_IoT
switched to db sensores_IoT
> db.sensores
datos sensores
> // Consulta de Valladolid
> db.datos_sensores.aggregate([
  {
    $match: {
      "location.id": 1,
      "medidas.tipo_medida": "Humedad_relativa",
      "timestamp": {
        "$gte: ISODate("2020-07-01T08:00:00Z"),
        "$lt: ISODate("2020-07-01T18:00:00Z")
      }
    },
    $unwind: "$medidas"
  },
  {
    $match: {
      "medidas.tipo_medida": "Humedad_relativa",
      "timestamp": {
        "$gte: ISODate("2020-07-01T08:00:00Z"),
        "$lt: ISODate("2020-07-01T18:00:00Z")
      }
    }
  },
  {
    $project: {
      timestamp: 1,
      medidas.tipo_medida: 1,
      medidas.valor: 1,
      medidas.unidad: 1
    }
  },
  {
    $sort: {
      "medidas.valor": 1
    }
  }
])
CAMPUS_IMF_Mongo_Neo4J [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
○ ○ Terminal Archivo Editar Ver Buscar Terminal Ayuda
> db.datos_sensores.aggregate([
  {
    $match: {
      "location.id": 2,
      "medidas.tipo_medida": "Humedad_relativa",
      "timestamp": {
        "$gte: ISODate("2020-07-01T08:00:00Z"),
        "$lt: ISODate("2020-07-01T18:00:00Z")
      }
    },
    $unwind: "$medidas"
  },
  {
    $match: {
      "medidas.tipo_medida": "Humedad_relativa",
      "timestamp": {
        "$gte: ISODate("2020-07-01T08:00:00Z"),
        "$lt: ISODate("2020-07-01T18:00:00Z")
      }
    }
  },
  {
    $project: {
      timestamp: 1,
      medidas.tipo_medida: 1,
      medidas.valor: 1,
      medidas.unidad: 1
    }
  },
  {
    $sort: {
      "medidas.valor": 1
    }
  },
  {
    $limit: 5
  }
]).pretty();
>
> // Sevilla
> db.datos_sensores.aggregate([
  {
    $match: {
      "location.id": 3,
      "medidas.tipo_medida": "Humedad_relativa",
      "timestamp": {
        "$gte: ISODate("2020-07-01T08:00:00Z"),
        "$lt: ISODate("2020-07-01T18:00:00Z")
      }
    },
    $unwind: "$medidas"
  },
  {
    $match: {
      "medidas.tipo_medida": "Humedad_relativa",
      "timestamp": {
        "$gte: ISODate("2020-07-01T08:00:00Z"),
        "$lt: ISODate("2020-07-01T18:00:00Z")
      }
    }
  },
  {
    $project: {
      timestamp: 1,
      medidas.tipo_medida: 1,
      medidas.valor: 1,
      medidas.unidad: 1
    }
  },
  {
    $sort: {
      "medidas.valor": 1
    }
  },
  {
    $limit: 5
  }
]).pretty();
>
> db.datos_sensores.aggregate([
  {
    // Convertir timestamp a fecha
  }
])
```



```

CAMPUS-INF-Mongo-Neo4j [Comiendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
Terminal Archivo Editar Ver Buscar Terminal Ayuda
> db.datos.sensores.aggregate([
  // Convertir timestamp a fecha
  { "$addFields": { "ToDate": { "$ToDate": "$timestamp" } } },
  // Extraer la hora del timestamp
  { "$addFields": { "Hora": { "$dateToString": { "format": "%H", "date": "$timestamp" } } } },
  // Convertir la hora a entero
  { "$addFields": { "Hora": { "$toInt": "$Hora" } } },
  // Desenrollar el array de medidas
  { "$unwind": "$Medidas" },
  // Filtrar por horario laboral y tipo de medida
  { "$match": {
    $and: [
      { "Hora": { "$gte": 8, "$lt": 18 } },
      { "Medidas.tipo_medida": "Humedad_relativa" }
    ]
  }},
  // Ordenar por valor de humedad relativa
  { "$sort": { "Medidas.valor": 1 } },
  // Agrupar por location_id y recoger los documentos
  { "$group": {
    "_id": "$location_id",
    "docs_por_ciudad": {
      "$push": {
        "timestamp": "$timestamp",
        "tipo_medida": "$Medidas.tipo_medida",
        "valor": "$Medidas.valor",
        "unidad": "$Medidas.unidad"
      }
    }
  }},
  // Proyectar los primeros 5 documentos por ciudad
  { "$project": {
    "_id": 0,
  }}
])

```



```

CAMPUS-INF-Mongo-Neo4j [Comiendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
Terminal Archivo Editar Ver Buscar Terminal Ayuda
{
  "location_id": 2,
  "datos": [
    {
      "timestamp": ISODate("2020-07-03T16:45:00Z"),
      "tipo_medida": "Humedad_relativa",
      "valor": 5.74,
      "unidad": "%"
    },
    {
      "timestamp": ISODate("2020-07-06T17:00:00Z"),
      "tipo_medida": "Humedad_relativa",
      "valor": 6.81,
      "unidad": "%"
    },
    {
      "timestamp": ISODate("2020-07-10T17:45:00Z"),
      "tipo_medida": "Humedad_relativa",
      "valor": 7.04,
      "unidad": "%"
    },
    {
      "timestamp": ISODate("2020-07-14T17:30:00Z"),
      "tipo_medida": "Humedad_relativa",
      "valor": 8.14,
      "unidad": "%"
    },
    {
      "timestamp": ISODate("2020-07-05T17:15:00Z"),
      "tipo_medida": "Humedad_relativa",
      "valor": 8.16,
      "unidad": "%"
    }
  ]
}

```

Caso 10

Se quieren actualizar todos los documentos para introducir en el array de medidas dos nuevos elementos que van a ser constantes y que van a tener el valor del precio del kWh y de la superficie total de la sede. Solamente se añadirán a los documentos que tengan como medida el consumo eléctrico (los que tienen temperatura y humedad no).

```
CAMPUS_IMF_Mongo_Neo4j [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivo Ayuda
Terminal Archivo Editar Ver Buscar Terminal Ayuda
[{"acknowledged": true, "matchedCount": 1344, "modifiedCount": 1344}
> // Actualizar documentos para Sevilla
> db.datos_sensores.updateMany(
... {
...     "location_id": 2,
...     "medidas.tipo_medida": "Consumo_electrico"
... },
... {
...     $push: {
...         medidas: {
...             $each:
...                 [
...                     { "precio_kwh": 0.107, "unidad": "€/kWh" },
...                     { "superficie": 550, "unidad": "m2" }
...                 ]
...         }
...     }
... }
);
{"acknowledged": true, "matchedCount": 1344, "modifiedCount": 1344}
> // Verificar los cambios
> // Verificar documentos para Valladolid
> db.datos_sensores.find(
... {
...     "location_id": 1,
...     "timestamp": {
...         "$in": [
...             ISODate("2020-07-01T08:00:00Z"),
...             ISODate("2020-07-01T23:15:00Z")
...         ]
...     }
... },
... {
...     "timestamp": 1,
...     "location_id": 1,
...     "medidas": 1
... }
... ).limit(4).pretty();
> // Verificar documentos para Sevilla
> db.datos_sensores.find(
... {
...     "location_id": 2,
...     "timestamp": {
...         "$in": [
...             ISODate("2020-07-01T08:00:00Z"),
...             ISODate("2020-07-01T23:15:00Z")
...         ]
...     }
... },
... {
...     "timestamp": 1,
...     "location_id": 2,
...     "medidas": 1
... }
... ).limit(4).pretty();
```

```
CAMPUS_IMF_Mongo_Neo4j [Corriendo] - Oracle VM VirtualBox
Archivo Maquina Ver Entrada Dispositivos Ayuda
Archivo Terminal Archivo Editar Ver Buscar Terminal Ayuda
...
}
    ...
    > db.datas_sensors.find({
    ...   $or: [
    ...     { "timestamp": "2020-07-01T08:00:00Z", "location_id": 1 },
    ...     { "timestamp": "2020-07-01T23:15:00Z", "location_id": 1 }
    ...   ]
    ... }).limit(4).pretty()
{
  "_id" : ObjectId("66b11180d837030a5728edae"),
  "timestamp" : "2020-07-01T08:00:00Z",
  "sensor_id" : 1,
  "location_id" : 1,
  "medidas" : [
    {
      "tipo_medida" : "Temperatura",
      "valor" : 18.7,
      "unidad" : "°C"
    },
    {
      "tipo_medida" : "Humedad_relativa",
      "valor" : 77.36,
      "unidad" : "%"
    }
  ]
},
{
  "_id" : ObjectId("66b11180d837030a5728edb0"),
  "timestamp" : "2020-07-01T08:00:00Z",
  "sensor_id" : 2,
  "location_id" : 1,
  "medidas" : [
    {
      "tipo_medida" : "Emision_CO2",
      "valor" : 6.7,
      "unidad" : "gCO2/m2"
    },
    {
      "tipo_medida" : "Consumo_electrico",
      "valor" : 0.01788,
      "unidad" : "kWh/m2"
    },
    {
      "precio_kwh" : 0.102,
      "unidad" : "€/kWh"
    },
    {
      "superficie" : 450,
      "unidad" : "m2"
    }
  ]
},
{
  "_id" : ObjectId("66b11180d837030a5728ea2"),
  "timestamp" : "2020-07-01T23:15:00Z",
  "sensor_id" : 3,
  "location_id" : 2,
  "medidas" : [
    {
      "tipo_medida" : "Temperatura",
      "valor" : 20.5,
      "unidad" : "°C"
    },
    {
      "tipo_medida" : "Humedad_relativa",
      "valor" : 72.5,
      "unidad" : "%"
    }
  ]
}
```


Caso 3 NEO4J

Consideraciones Iniciales

Una vez se ha construido y preparado completamente la base de datos “Rastreo_COVID” durante el ejercicio de repaso, se procede a resolver las cuestiones que se plantean en el caso de evaluación. En primer lugar, se recuerdan los nodos y relaciones que forman la BBDD:

Se parte de la complicada situación que se está atravesando en plena pandemia de la COVID-19.

Hoy en día, se habla mucho sobre los planes de acción cuando la situación haya concluido y sobre cómo países y localidades comenzarán de nuevo a reactivar sus economías.

Una acción que la mayor parte de los países están tomando es el “rastreo de contactos”. Utilizando la tecnología que nos proporcionan nuestros teléfonos móviles, a través de aplicaciones como puede ser Radar Covid, se busca una forma de lograr el distanciamiento o aislamiento de personas vulnerables o en riesgo de contraer la enfermedad.

Esto ha despertado el interés por la utilidad que tienen las bases de datos orientadas a grafos para resolver este tipo de problemas, puesto que tan importante es ser capaces de identificar a contactos directos como indirectos.

Nodo persona. Propiedades:

- Id de la persona (id_persona).
- Nombre de la persona (nombre_presona).
- Estado de la persona (estado_salud: contagiado/sano).
- Timestamp que representa el momento en que se notifica a una persona que se había realizado el test PCR, si está o no contagiada (hota_test_resultado).
- Latitud: coordenada de latitud del domicilio donde reside la persona (latitud_domicilio).
- Longitud: coordenada de longitud del domicilio donde reside la persona (longitud_domicilio).
- Corresponden a coordenadas de Valladolid y Salamanca.

Nodo Ubicación. Propiedades:

- Id del lugar (id_ubicacion).
- Nombre del establecimiento (nombre_establecimiento).
- Tipo de establecimiento (tipo_establecimiento).

Nodo visita (de personas a distintos lugares (hospital, colegio, bar, restaurante, etc.) de dos ciudades:

Valladolid y Salamanca). Propiedades:

- Identificador de la visita (id_visita).
- Identificador de la persona (id_ubicacion).
- Hora de comienzo de la visita (inicio_visita).
- Hora de fin de la visita (fin_visita).
- Hay tres tipos de relaciones entre los nodos:
- (Persona) [VISITA_EMPLAZAMIENTO] (Ubicacion)
- (Persona) [REALIZA_VISITA] (Visita) [A_ESTABLECIMIENTO] (Ubicacion)

Caso 1

Identificar el número de personas contagiadas y de personas sanas en la muestra de 40 personas. Devolver el resultado en formato table o text. Los campos a devolver se muestran a continuación:

The screenshot shows the Neo4j Browser interface. On the left, there's a sidebar with 'Database Information' containing sections for 'Node Labels' (with 'Person' selected), 'Relationship Types' (with 'VISITA_EMPLAZAMIENTO' selected), and 'Property Keys'. The main area displays a query in the command line and its results in a table.

```
//Identificar las personas contagiadas y sanas , que se pueda ver en una tabla
MATCH (p:Persona) WITH p LIMIT 40 RETURN COUNT(CASE WHEN p.estado = 'Contagiado' THEN 1 END) AS Total_contagiados,
COUNT(CASE WHEN p.estado = 'Sano' THEN 1 END) AS Total_sanos
```

Total_contagiados	Total_sanos
10	30

Started streaming 1 records after 2 ms and completed after 2 ms.

Caso 2

Encontrar las personas sanas que han estado en contacto con una persona que ha dado positivo.

Debido a que nos encontramos en las primeras fases de estudio de la enfermedad, se da por supuesto que la COVID puede infectar a personas sanas que hayan estado en un mismo lugar en el que ha estado una persona contagiada, aunque sea en diferentes días.

Identificar todas las personas sanas que han estado en el mismo lugar (sin importar fecha ni hora) en el que también ha estado una persona contagiada (lógicamente el comienzo de la visita de la persona sana tiene que ser posterior al de la persona enferma, ya que es imposible contagiarse si la persona sana ha ido antes que la contagiada).

Devolver el resultado en formato table o text

The screenshot shows the Neo4j Browser interface. On the left, there's a sidebar with 'Database Information' containing sections for 'Node Labels' (with 'Person' selected), 'Relationship Types' (with 'VISITA_EMPLAZAMIENTO' selected), and 'Property Keys'. The main area displays a query in the command line and its results in a table.

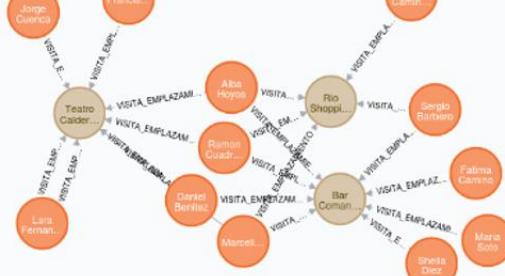
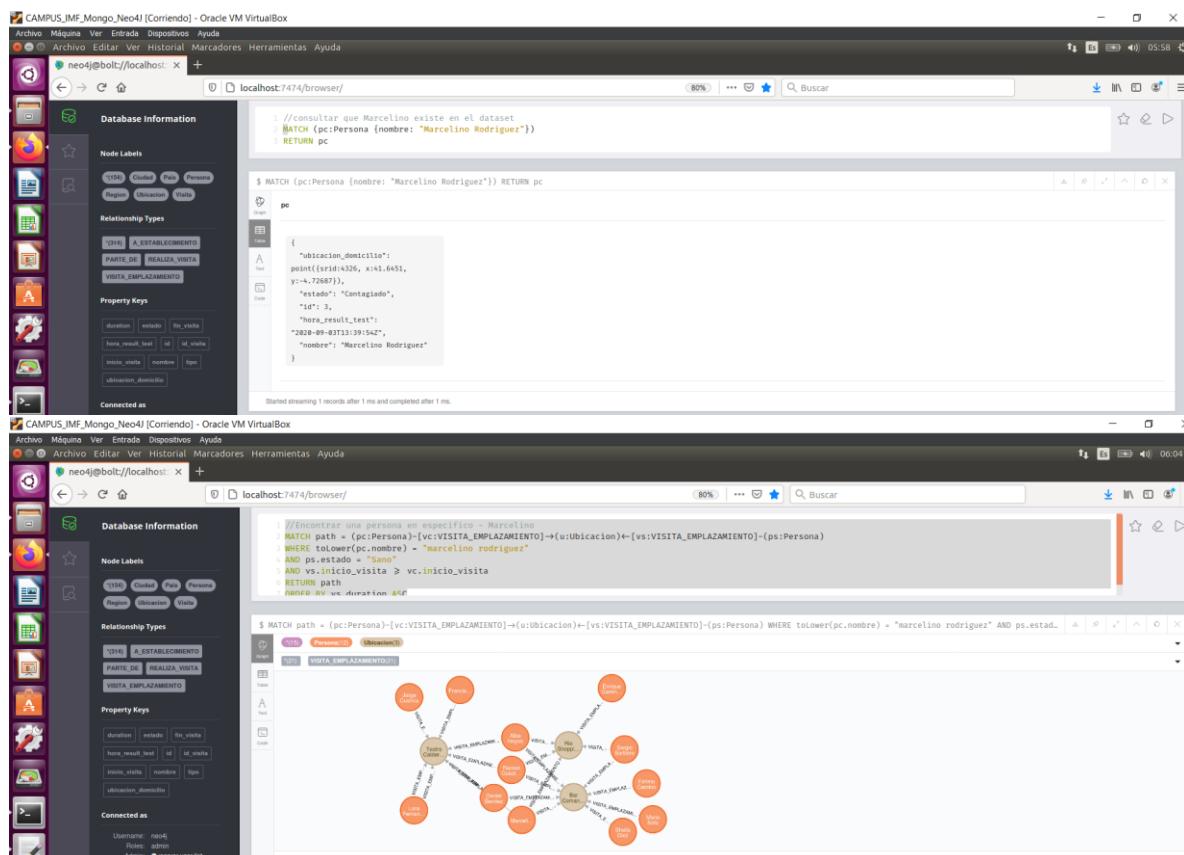
```
//Encontrar las personas sanas que han estado en contacto con una persona que ha dado positivo.
MATCH (pc:Persona{estado:"Contagiado"})-[vc:VISITA_EMPLAZAMIENTO]->(u:Ubicacion)-[vs:VISITA_EMPLAZAMIENTO]-(ps:Persona{estado:"Sano"})
WHERE vc.inicio_visita < vs.fin_visita AND vs.fin_visita < vc.fin_visita
RETURN DISTINCT ps.nombre AS Persona_en_riesgo
```

Persona_en_riesgo
"Celia Cuenca"
"Alvaro Fernandez"
"Maria Soto"
"Alba Hoyos"
"Esther Liebana"
"Javier Montes"
"Daniel Beriles"
"Ramon Cuadrado"
"Shelia Diaz"

Started streaming 16 records after 8 ms and completed after 20 ms.

Caso 3

Mostrar grafo con las personas sanas que han coincidido con una persona contagiada, concretamente con "Marcelino Rodriguez". Hay que mostrar en un grafo con este nodo persona, junto con todos los lugares que ha visitado, y con los nodos de etiqueta persona sana que han visitado también ese lugar posteriormente. Observando los resultados del grafo, comentar seis personas de las que han estado en alguna ubicación en la que ha estado Marcelino después de haber estado él, tienen menos riesgo que el resto de haberse contagiado que el resto:



Caso 4

Construir la misma consulta anterior, pero mostrando el resultado como una tabla y no como un grafo (formato table o text) mostrando los campos:

- Esparcidor_virus.
- Comienzo_esparcimiento_virus.
- Establecimiento.
- Persona_en_riesgo.
- Inicio_visita_persona_en_riesgo.

The screenshot shows the Neo4j Browser interface with a query results table. The table has five columns: Esparcidor_virus, Comienzo_esparcimiento_virus, Establecimiento, Persona_en_riesgo, and Inicio_visita_persona_en_riesgo. The data is as follows:

Esparcidor_virus	Comienzo_esparcimiento_virus	Establecimiento	Persona_en_riesgo	Inicio_visita_persona_en_riesgo
"Marcelino Rodriguez"	"2020-08-31T11:39:54Z"	"Bar Comandante"	"Daniel Benites"	"2020-08-31T16:52:36Z"
"Marcelino Rodriguez"	"2020-08-31T11:39:54Z"	"Bar Comandante"	"Alba Hoyos"	"2020-09-01T16:12:30Z"
"Marcelino Rodriguez"	"2020-08-31T11:39:54Z"	"Bar Comandante"	"Sergio Barbero"	"2020-09-01T17:21:13Z"
"Marcelino Rodriguez"	"2020-08-31T11:39:54Z"	"Bar Comandante"	"Maria Soto"	"2020-09-01T17:21:23Z"
"Marcelino Rodriguez"	"2020-08-31T11:39:54Z"	"Bar Comandante"	"Ramon Cuadra"	"2020-09-01T17:21:27Z"

Caso 5

5.1 Construir una tabla (formato text y table) que identifique para cada persona contagiada (columna uno), las personas sanas con las que ha coincidido en un establecimiento en el mismo tiempo. Construir como segunda columna un array de elementos JSON llamado "Contactos" con claves:

Persona_en_contacto. Establecimiento. Fecha_comienzo_solapamiento.

Fecha_fin_solapamiento.

Se muestra a continuación un ejemplo del formato de cada documento JSON y de los dos campos a obtener:

The screenshot shows the Neo4j Browser interface with a query results table and a JSON document. The table has two columns: Persona_contagiada and Contactos. The data is as follows:

Persona_contagiada	Contactos
"Bartolino Castillo"	[{"Fecha_fin_solapamiento": "2020-09-02T08:59:23Z", "Establecimiento": "Bar La Chica de Ayer", "Fecha_comienzo_solapamiento": "2020-09-02T09:17:29Z", "Persona_en_contacto": "Celia Cuenca"}]

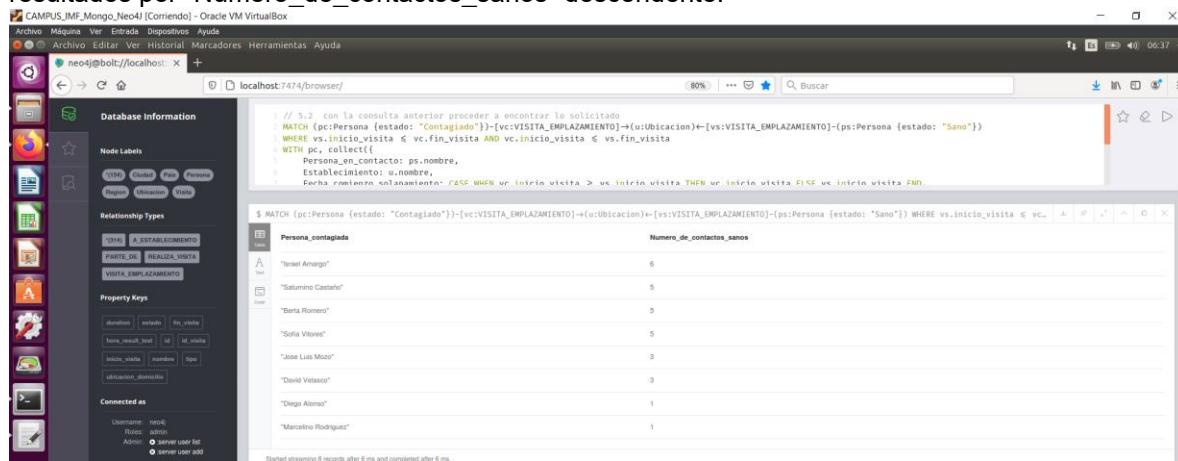
Below the table, a JSON document is shown:

```

{
    "Fecha_fin_solapamiento": "2020-09-02T08:59:23Z",
    "Establecimiento": "Bar La Chica de Ayer",
    "Fecha_comienzo_solapamiento": "2020-09-02T09:17:29Z",
    "Persona_en_contacto": "Celia Cuenca"
}

```

5.2 Una vez obtenida la consulta anterior, conseguir, añadiendo tres sentencias a esta, una tabla que tenga el nombre de la persona contagiada y otra columna con el número de personas sanas con las que ha tenido contacto, obteniendo ese número a partir de los elementos del array de elementos JSON. Ordenar los resultados por "Numero_de_contactos_sanos" descendente:



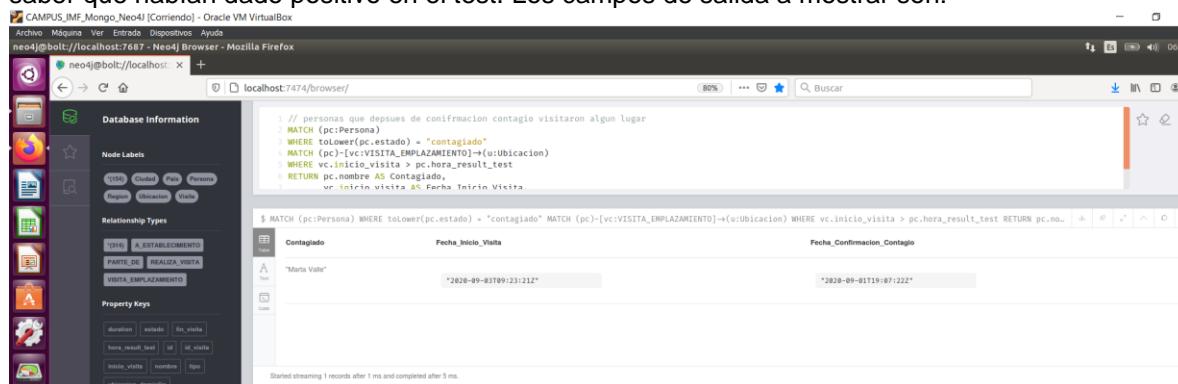
```

// 5.2 con la consulta anterior proceder a encontrar lo solicitado
MATCH (pc:Persona {estado: "Contagiado"})-(vc:VISITA_EMPLAZAMIENTO)->(u:Ubicacion)->(vs:VISITA_EMPLAZAMIENTO)-(ps:Persona {estado: "Sano"})
WHERE vc.inicio_visita <= vc.fin_visita AND vc.fin_visita <= vs.fin_visita
WITH pc, vc, ps
SET pc.sanear = true
RETURN pc.nombre,
       Establecimiento: u.nombre,
       Fecha_comienzo_sanamiento: CASE WHEN vc.inicio_visita > vc.fin_visita THEN vc.inicio_visita ELSE vc.fin_visita END
    
```

Nombre	Numero_de_contactos_sanos
"Isabel Amieiro"	6
"Saturnino Castaño"	5
"Berta Romeo"	5
"Sofía Vilares"	5
"José Luis Mozo"	3
"David Velasco"	3
"Diego Almeida"	1
"Manuelino Rodríguez"	1

Caso 6

Encontrar a aquellas personas (si es que hay alguna) que visitaron un establecimiento incluso después de saber que habían dado positivo en el test. Los campos de salida a mostrar son:



```

// Personas que despues de confirmacion contagio visitaron algun lugar
MATCH (pc:Persona {hora_result_test > pc.hora_result_test})
WHERE tolower(pc.estado) = "contagiado"
MATCH (pc)-[vc:VISITA_EMPLAZAMIENTO]->(u:Ubicacion)
WHERE vc.inicio_visita > pc.hora_result_test
RETURN pc.nombre AS Contagiado,
       vc.inicio_visita AS Fecha_Inicio_Visita
    
```

Contagiado	Fecha_Inicio_Visita	Fecha_Confirmacion_Contagio
"Marta Valls"	"2020-09-03T09:23:21Z"	"2020-09-01T19:07:22Z"

Caso 7

Ahora que se han obtenido todas las personas sanas que coincidieron en algún establecimiento con alguna contagiada, se quiere averiguar el tiempo exacto (duración) que coincidió cada persona sana con la persona contagiada.

Expresar la duración en horas y redondeada a cuatro decimales. Devolver el resultado en formato table o text.

```

// se procede a buscar lo solicitado en lo que indica
// se quiere averiguar el tiempo exacto (duración) que coincidió cada persona sana con la persona contagiada.
MATCH (pc:Persona {estado: "Contagiado"})-[vc:VISITA_EMPLAZAMIENTO]->(u:Ubicacion)-[vs:VISITA_EMPLAZAMIENTO]->(ps:Persona {estado: "Sano"})
WHERE vc.fin_visita > vs.inicio_visita AND vs.fin_visita > vc.inicio_visita
WITH pc, ps, u
// Calcula el inicio y fin de la intersección de visitas
MATCH (v1:Visita)->|<->|(v2:Visita)
    WHERE v1.fin_visita > v2.inicio_visita AND v2.fin_visita > v1.inicio_visita
    WITH v1, v2
    ORDER BY v1.fin_visita ASC, v2.inicio_visita ASC
    GROUP BY v1.fin_visita, v2.inicio_visita
    AS interseccion_inicio
    WITH pc, ps, u, interseccion_inicio
    MATCH (v1:Visita)->|<->|(v2:Visita)
        WHERE v1.fin_visita > v2.inicio_visita AND v2.fin_visita > v1.inicio_visita
        WITH v1, v2
        ORDER BY v1.fin_visita DESC, v2.inicio_visita DESC
        GROUP BY v1.fin_visita, v2.inicio_visita
        AS interseccion_fin
        WITH pc, ps, u, interseccion_inicio, interseccion_fin
        MATCH (v1:Visita)->|<->|(v2:Visita)
            WHERE v1.fin_visita > v2.inicio_visita AND v2.fin_visita > v1.inicio_visita
            WITH v1, v2
            ORDER BY v1.fin_visita ASC, v2.inicio_visita ASC
            GROUP BY v1.fin_visita, v2.inicio_visita
            AS interseccion_final
            WITH pc, ps, u, interseccion_inicio, interseccion_fin, interseccion_final
            MATCH (v1:Visita)->|<->|(v2:Visita)
                WHERE v1.fin_visita > v2.inicio_visita AND v2.fin_visita > v1.inicio_visita
                WITH v1, v2
                ORDER BY v1.fin_visita DESC, v2.inicio_visita DESC
                GROUP BY v1.fin_visita, v2.inicio_visita
                AS interseccion_final
                WITH pc, ps, u, interseccion_inicio, interseccion_fin, interseccion_final
                MATCH (v1:Visita)->|<->|(v2:Visita)
                    WHERE v1.fin_visita > v2.inicio_visita AND v2.fin_visita > v1.inicio_visita
                    WITH v1, v2
                    ORDER BY v1.fin_visita ASC, v2.inicio_visita ASC
                    GROUP BY v1.fin_visita, v2.inicio_visita
                    AS interseccion_final
                    WITH pc, ps, u, interseccion_inicio, interseccion_fin, interseccion_final
                    MATCH (v1:Visita)->|<->|(v2:Visita)
                        WHERE v1.fin_visita > v2.inicio_visita AND v2.fin_visita > v1.inicio_visita
                        WITH v1, v2
                        ORDER BY v1.fin_visita DESC, v2.inicio_visita DESC
                        GROUP BY v1.fin_visita, v2.inicio_visita
                        AS interseccion_final
                        WITH pc, ps, u, interseccion_inicio, interseccion_fin, interseccion_final
                        MATCH (v1:Visita)->|<->|(v2:Visita)
                            WHERE v1.fin_visita > v2.inicio_visita AND v2.fin_visita > v1.inicio_visita
                            WITH v1, v2
                            ORDER BY v1.fin_visita ASC, v2.inicio_visita ASC
                            GROUP BY v1.fin_visita, v2.inicio_visita
                            AS interseccion_final
                            WITH pc, ps, u, interseccion_inicio, interseccion_fin, interseccion_final
                            MATCH (v1:Visita)->|<->|(v2:Visita)
                                WHERE v1.fin_visita > v2.inicio_visita AND v2.fin_visita > v1.inicio_visita
                                WITH v1, v2
                                ORDER BY v1.fin_visita DESC, v2.inicio_visita DESC
                                GROUP BY v1.fin_visita, v2.inicio_visita
                                AS interseccion_final
                                WITH pc, ps, u, interseccion_inicio, interseccion_fin, interseccion_final
                                MATCH (v1:Visita)->|<->|(v2:Visita)
                                    WHERE v1.fin_visita > v2.inicio_visita AND v2.fin_visita > v1.inicio_visita
                                    WITH v1, v2
                                    ORDER BY v1.fin_visita ASC, v2.inicio_visita ASC
                                    GROUP BY v1.fin_visita, v2.inicio_visita
                                    AS interseccion_final
                                    WITH pc, ps, u, interseccion_inicio, interseccion_fin, interseccion_final
                                    MATCH (v1:Visita)->|<->|(v2:Visita)
                                        WHERE v1.fin_visita > v2.inicio_visita AND v2.fin_visita > v1.inicio_visita
                                        WITH v1, v2
                                        ORDER BY v1.fin_visita DESC, v2.inicio_visita DESC
                                        GROUP BY v1.fin_visita, v2.inicio_visita
                                        AS interseccion_final
                                        WITH pc, ps, u, interseccion_inicio, interseccion_fin, interseccion_final
                                        MATCH (v1:Visita)->|<->|(v2:Visita)
                                            WHERE v1.fin_visita > v2.inicio_visita AND v2.fin_visita > v1.inicio_visita
                                            WITH v1, v2
                                            ORDER BY v1.fin_visita ASC, v2.inicio_visita ASC
                                            GROUP BY v1.fin_visita, v2.inicio_visita
                                            AS interseccion_final
                                            WITH pc, ps, u, interseccion_inicio, interseccion_fin, interseccion_final
                                            MATCH (v1:Visita)->|<->|(v2:Visita)
                                                WHERE v1.fin_visita > v2.inicio_visita AND v2.fin_visita > v1.inicio_visita
                                                WITH v1, v2
                                                ORDER BY v1.fin_visita DESC, v2.inicio_visita DESC
                                                GROUP BY v1.fin_visita, v2.inicio_visita
                                                AS interseccion_final
                                                WITH pc, ps, u, interseccion_inicio, interseccion_fin, interseccion_final
                                                MATCH (v1:Visita)->|<->|(v2:Visita)
                                                    WHERE v1.fin_visita > v2.inicio_visita AND v2.fin_visita > v1.inicio_visita
                                                    WITH v1, v2
                                                    ORDER BY v1.fin_visita ASC, v2.inicio_visita ASC
                                                    GROUP BY v1.fin_visita, v2.inicio_visita
                                                    AS interseccion_final
                                                    WITH pc, ps, u, interseccion_inicio, interseccion_fin, interseccion_final
                                                    MATCH (v1:Visita)->|<->|(v2:Visita)
                                                        WHERE v1.fin_visita > v2.inicio_visita AND v2.fin_visita > v1.inicio_visita
                                                        WITH v1, v2
                                                        ORDER BY v1.fin_visita DESC, v2.inicio_visita DESC
                                                        GROUP BY v1.fin_visita, v2.inicio_visita
                                                        AS interseccion_final
                                                        WITH pc, ps, u, interseccion_inicio, interseccion_fin, interseccion_final
                                                        MATCH (v1:Visita)->|<->|(v2:Visita)
                                                            WHERE v1.fin_visita > v2.inicio_visita AND v2.fin_visita > v1.inicio_visita
                                                            WITH v1, v2
                                                            ORDER BY v1.fin_visita ASC, v2.inicio_visita ASC
                                                            GROUP BY v1.fin_visita, v2.inicio_visita
                                                            AS interseccion_final
                                                            WITH pc, ps, u, interseccion_inicio, interseccion_fin, interseccion_final
                                                            MATCH (v1:Visita)->|<->|(v2:Visita)
                                                                WHERE v1.fin_visita > v2.inicio_visita AND v2.fin_visita > v1.inicio_visita
                                                                WITH v1, v2
                                                                ORDER BY v1.fin_visita DESC, v2.inicio_visita DESC
                                                                GROUP BY v1.fin_visita, v2.inicio_visita
                                                                AS interseccion_final
                                                                WITH pc, ps, u, interseccion_inicio, interseccion_fin, interseccion_final
                                                                MATCH (v1:Visita)->|<->|(v2:Visita)
                                                                    WHERE v1.fin_visita > v2.inicio_visita AND v2.fin_visita > v1.inicio_visita
                                                                    WITH v1, v2
                                                                    ORDER BY v1.fin_visita ASC, v2.inicio_visita ASC
                                                                    GROUP BY v1.fin_visita, v2.inicio_visita
                                                                    AS interseccion_final
                                                                    WITH pc, ps, u, interseccion_inicio, interseccion_fin, interseccion_final
                                                                    MATCH (v1:Visita)->|<->|(v2:Visita)
                                                                        WHERE v1.fin_visita > v2.inicio_visita AND v2.fin_visita > v1.inicio_visita
                                                                        WITH v1, v2
                                                                        ORDER BY v1.fin_visita DESC, v2.inicio_visita DESC
                                                                        GROUP BY v1.fin_visita, v2.inicio_visita
                                                                        AS interseccion_final
                                                                        WITH pc, ps, u, interseccion_inicio, interseccion_fin, interseccion_final
                                                                        MATCH (v1:Visita)->|<->|(v2:Visita)
                                                                            WHERE v1.fin_visita > v2.inicio_visita AND v2.fin_visita > v1.inicio_visita
                                                                            WITH v1, v2
                                                                            ORDER BY v1.fin_visita ASC, v2.inicio_visita ASC
                                                                            GROUP BY v1.fin_visita, v2.inicio_visita
                                                                            AS interseccion_final
                                                                            WITH pc, ps, u, interseccion_inicio, interseccion_fin, interseccion_final
                                                                            MATCH (v1:Visita)->|<->|(v2:Visita)
                                                                                WHERE v1.fin_visita > v2.inicio_visita AND v2.fin_visita > v1.inicio_visita
                                                                                WITH v1, v2
                                                                                ORDER BY v1.fin_visita DESC, v2.inicio_visita DESC
                                                                                GROUP BY v1.fin_visita, v2.inicio_visita
                                                                                AS interseccion_final
                                                                                WITH pc, ps, u, interseccion_inicio, interseccion_fin, interseccion_final
                                                                                MATCH (v1:Visita)->|<->|(v2:Visita)
                                                                                    WHERE v1.fin_visita > v2.inicio_visita AND v2.fin_visita > v1.inicio_visita
                                                                                    WITH v1, v2
                                                                                    ORDER BY v1.fin_visita ASC, v2.inicio_visita ASC
                                                                                    GROUP BY v1.fin_visita, v2.inicio_visita
                                                                                    AS interseccion_final
                                                                                    WITH pc, ps, u, interseccion_inicio, interseccion_fin, interseccion_final
                                                                                    MATCH (v1:Visita)->|<->|(v2:Visita)
                                                                                        WHERE v1.fin_visita > v2.inicio_visita AND v2.fin_visita > v1.inicio_visita
                                                                                        WITH v1, v2
                                                                                        ORDER BY v1.fin_visita DESC, v2.inicio_visita DESC
                                                                                        GROUP BY v1.fin_visita, v2.inicio_visita
                                                                                        AS interseccion_final
                                                                                        WITH pc, ps, u, interseccion_inicio, interseccion_fin, interseccion_final
                                                                                        MATCH (v1:Visita)->|<->|(v2:Visita)
                                                                                            WHERE v1.fin_visita > v2.inicio_visita AND v2.fin_visita > v1.inicio_visita
                                                                                            WITH v1, v2
                                                                                            ORDER BY v1.fin_visita ASC, v2.inicio_visita ASC
                                                                                            GROUP BY v1.fin_visita, v2.inicio_visita
                                                                                            AS interseccion_final
                                                                                            WITH pc, ps, u, interseccion_inicio, interseccion_fin, interseccion_final
                                                                                            MATCH (v1:Visita)->|<->|(v2:Visita)
                                                                                                WHERE v1.fin_visita > v2.inicio_visita AND v2.fin_visita > v1.inicio_visita
                                                                                                WITH v1, v2
                                                                                                ORDER BY v1.fin_visita DESC, v2.inicio_visita DESC
                                                                                                GROUP BY v1.fin_visita, v2.inicio_visita
                                                                                                AS interseccion_final
                                                                                                WITH pc, ps, u, interseccion_inicio, interseccion_fin, interseccion_final
                                                                                                MATCH (v1:Visita)->|<->|(v2:Visita)
                                                                                                    WHERE v1.fin_visita > v2.inicio_visita AND v2.fin_visita > v1.inicio_visita
                                                                                                    WITH v1, v2
                                                                                                    ORDER BY v1.fin_visita ASC, v2.inicio_visita ASC
                                                                                                    GROUP BY v1.fin_visita, v2.inicio_visita
                                                                                                    AS interseccion_final
                                                                                                    WITH pc, ps, u, interseccion_inicio, interseccion_fin, interseccion_final
                                                                                                    MATCH (v1:Visita)->|<->|(v2:Visita)
                                                                                                        WHERE v1.fin_visita > v2.inicio_visita AND v2.fin_visita > v1.inicio_visita
                                                                                                        WITH v1, v2
                                                                                                        ORDER BY v1.fin_visita DESC, v2.inicio_visita DESC
                                                                                                        GROUP BY v1.fin_visita, v2.inicio_visita
                                                                                                        AS interseccion_final
................................................................

```

Personas_Contagiadas	Personas_Sanas	Establecimiento	Duración_en_Horas
"Santiago Castillo"	"Celia Cuenca"	"Bar La Chica de Ayer"	2.5839
"Berta Romeo"	"Ariana Fernández"	"Río Shopping"	3.05
"Sofía Vitoras"	"Ariana Fernández"	"Colegio San José"	5.2131
"Berta Romeo"	"María Soto"	"Colegio San José"	4.95
"Sofía Vitoras"	"María Soto"	"Colegio San José"	5.2131
"José Luis Muñoz"	"Alba Hoyos"	"Teatro Calderón"	1.4657
"Israel Amargo"	"Alba Hoyos"	"Teatro Calderón"	1.9942
"Berta Romeo"	"Esther Liebana"	"Colegio San José"	4.95

Started streaming 29 records after 34 ms and completed after 47 ms.

Caso 8

Una persona ha estado en dos sitios diferentes con personas contagiadas, en uno estuvo una hora y media en contacto y en el otro dos. El total de exposición de esa persona habrá sido de tres horas y media.

La duración de cada contacto entre persona sana y contagiada será el resultado que se obtenga en la cuestión anterior. Por tanto, se puede utilizar la consulta anterior como base y será necesario añadirle algo más para conseguir el resultado esperado.

Si una persona sana coincidió con dos contagiadas el mismo día en el mismo establecimiento, también se sumará el tiempo que estuvo en contacto con cada contagiado, entendiendo que el haber estado rodeado de más contagiados supone que esa persona tenga un mayor riesgo de contraer la enfermedad.

Solamente se mostrarán en la tabla (formato table o text) las cinco personas sanas con más tiempo de exposición. A esas cinco personas se les realizará inmediatamente una llamada para que comiencen a guardar cuarentena. El tiempo total se mostrará en horas con redondeo a cuatro decimales (por ejemplo: 9,4972 horas, que serán nueve horas y 30 minutos).

```

1 // Caso 8 Mostrar por persona sana el tiempo de exposición total a personas contagiadas
2 // Paso 1: Calcular la duración de coincidencia entre personas sanas y contagiadas
3 MATCH (pc:Persona {estado: "Contagiado"})-[vc:VISITA_EMPLAZAMIENTO]->(u:Ubicacion)-[vs:VISITA_EMPLAZAMIENTO]->(ps:Persona {estado: "Sano"})
4 WHERE vc.fin_visita > vs.inicio_visita AND vs.fin_visita > vc.inicio_visita
5 WITH pc, ps, u
6 MAX(vc.inicio_visita, vs.inicio_visita) AS interseccion_inicio,
7 MIN(vc.fin_visita, vs.fin_visita) AS interseccion_fin
8
9 MATCH (pc:Persona {estado: "Contagiado"})-[vc:VISITA_EMPLAZAMIENTO]->(u:Ubicacion)-[vs:VISITA_EMPLAZAMIENTO]->(ps:Persona {estado: "Sano"})
10 WHERE vc.fin_visita > vs.inicio_visita AND vs.fin_visita > vc.inicio_visita
11 WITH pc, ps, u, interseccion_inicio, interseccion_fin
12 MATCH (v1:Visita)->|<->|(v2:Visita)
13 WHERE v1.fin_visita > v2.inicio_visita AND v2.fin_visita > v1.inicio_visita
14 WITH ps.nombre AS Persona_Sana, SUM(duracion_en_horas) AS Tiempo_Exposicion_Total
15 // Paso 2: Agrupar por persona sana y sumar el tiempo de exposición total
16 // Paso 3: Ordenar por tiempo de exposición total y limitar a las 5 principales
17 ORDER BY Tiempo_Exposicion_Total DESC
18 LIMIT 5
19 RETURN Persona_Sana, Tiempo_Exposicion_Total

```

Persona_Sana	Tiempo_Exposición_Total
"Irene Montes "	5.3706
"María Soto"	4.8731
"Esther Liebana"	4.8731
"Daniel Benítez"	4.7556
"Javier Montes"	3.225

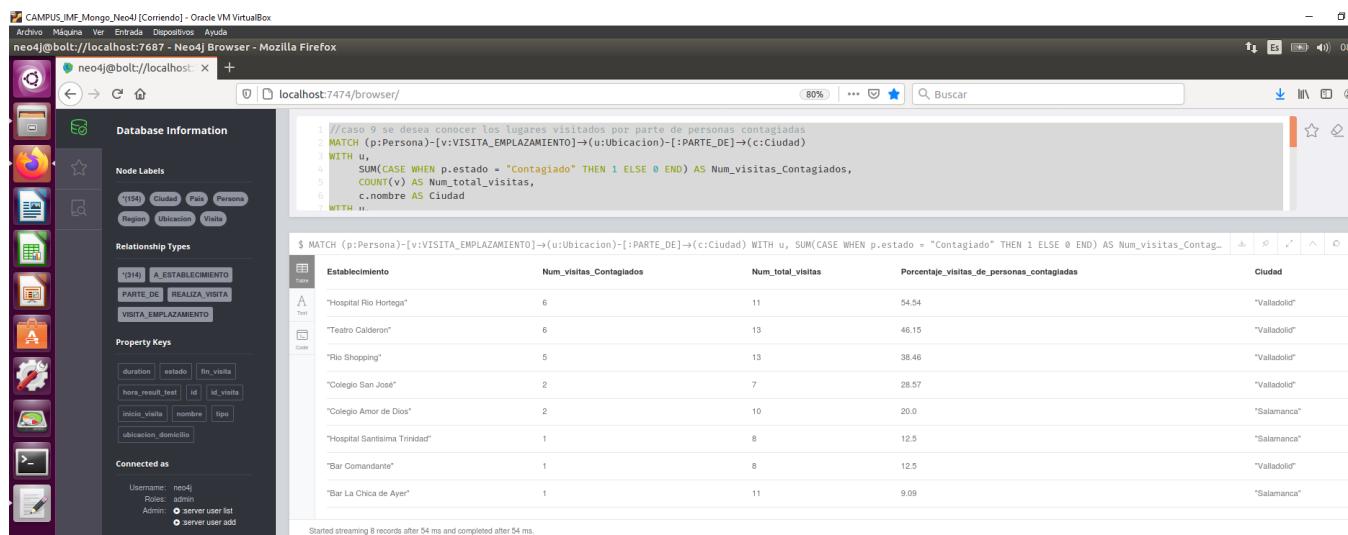
Started streaming 5 records after 56 ms and completed after 57 ms.

Persona_Sana	Tiempo_Exposición_Total
"Irene Montes "	5.3706
"María Soto"	4.8731
"Esther Liebana"	4.8731
"Daniel Benítez"	4.7556
"Javier Montes"	3.225

Caso 9

Se pretende tratar de reducir la afluencia e implementar aún más medidas de precaución en aquellos establecimientos en los que hayan estado más tiempo personas contagiadas.

Se pide devolver una tabla que contenga cada establecimiento, que ha sido visitado por al menos una persona contagiada, el total de visitas de contagiados en cada establecimiento, el total de visitas en cada establecimiento, el porcentaje de visitas de contagiados respecto al total de visitas de cada establecimiento, y la ciudad a la que pertenece el establecimiento. Expresar el porcentaje redondeado a dos decimales. Comentar cuáles son los dos establecimientos con mayor y los dos con menor porcentaje de visitas de contagiados respecto del total de cada establecimiento.

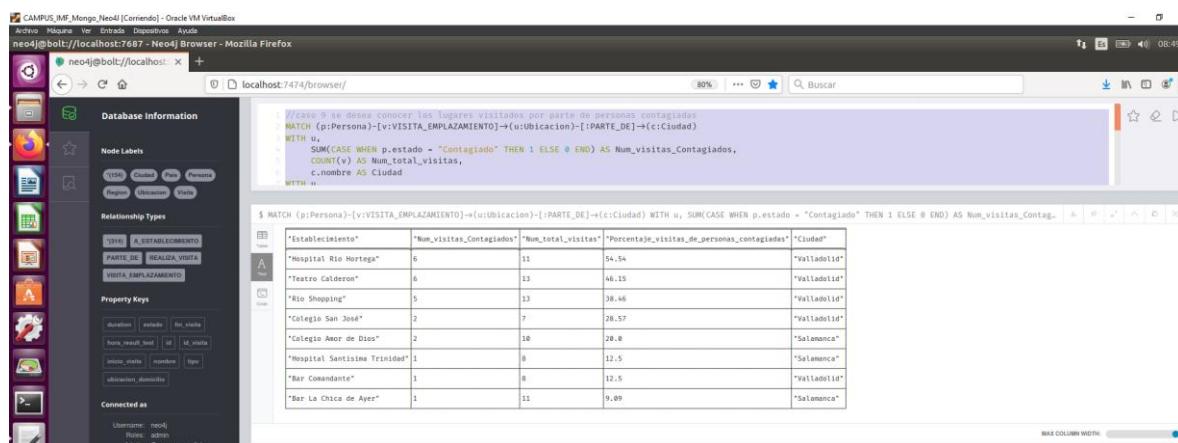


```

//caso 9 se desea conocer los lugares visitados por parte de personas contagiadas
MATCH (p:Persona)-[v:VISITA_EMPLAZAMIENTO]-(u:Ubicacion)-[:PARTE_DE]-(c:Ciudad)
WITH u,
    SUM(CASE WHEN p.estado = "Contagiado" THEN 1 ELSE 0 END) AS Num_visitantes_Contagiados,
    COUNT(v) AS Num_total_visitantes,
    c.nombre AS Ciudad
WITHIN u
$ MATCH (p:Persona)-[v:VISITA_EMPLAZAMIENTO]-(u:Ubicacion)-[:PARTE_DE]-(c:Ciudad) WITH u,
    SUM(CASE WHEN p.estado = "Contagiado" THEN 1 ELSE 0 END) AS Num_visitantes_Contagiados,
    COUNT(v) AS Num_total_visitantes,
    c.nombre AS Ciudad
    
```

Establecimiento	Num_visitantes_Contagiados	Num_total_visitantes	Porcentaje_visitantes_de_personas_contagiadas	Ciudad
"Hospital Rio Hontega"	6	11	54.54	"Valadolid"
"Teatro Calderon"	6	13	46.15	"Valadolid"
"Rio Shopping"	5	13	38.46	"Valadolid"
"Colegio San Jose"	2	7	28.57	"Valadolid"
"Colegio Amor de Dios"	2	10	20.0	"Salamanca"
"Hospital Santisima Trinidad"	1	8	12.5	"Salamanca"
"Bar Comandante"	1	8	12.5	"Valadolid"
"Bar La Chica de Ayer"	1	11	9.09	"Salamanca"

Started streaming 8 records after 54 ms and completed after 54 ms.



```

//caso 9 se desea conocer los lugares visitados por parte de personas contagiadas
MATCH (p:Persona)-[v:VISITA_EMPLAZAMIENTO]-(u:Ubicacion)-[:PARTE_DE]-(c:Ciudad)
WITH u,
    SUM(CASE WHEN p.estado = "Contagiado" THEN 1 ELSE 0 END) AS Num_visitantes_Contagiados,
    COUNT(v) AS Num_total_visitantes,
    c.nombre AS Ciudad
WITHIN u
$ MATCH (p:Persona)-[v:VISITA_EMPLAZAMIENTO]-(u:Ubicacion)-[:PARTE_DE]-(c:Ciudad) WITH u,
    SUM(CASE WHEN p.estado = "Contagiado" THEN 1 ELSE 0 END) AS Num_visitantes_Contagiados,
    COUNT(v) AS Num_total_visitantes,
    c.nombre AS Ciudad
    
```

Establecimiento	Num_visitantes_Contagiados	Num_total_visitantes	Porcentaje_visitantes_de_personas_contagiadas	Ciudad
"Hospital Rio Hontega"	6	11	54.54	"Valadolid"
"Teatro Calderon"	6	13	46.15	"Valadolid"
"Rio Shopping"	5	13	38.46	"Valadolid"
"Colegio San Jose"	2	7	28.57	"Valadolid"
"Colegio Amor de Dios"	2	10	20.0	"Salamanca"
"Hospital Santisima Trinidad"	1	8	12.5	"Salamanca"
"Bar Comandante"	1	8	12.5	"Valadolid"
"Bar La Chica de Ayer"	1	11	9.09	"Salamanca"

MAX COLUMN WIDTH: 1000

Caso 10

Mostrar las distancias entre los domicilios de los contagiados de Valladolid que hayan ido a un mismo establecimiento aunque haya sido en diferentes fechas.

Hay que obtener el resultado mediante una sola consulta en el editor de Neo4J. Se quiere devolver solo los tres registros Persona1 – Persona2 CONTAGIADAS de Valladolid que vivan a más distancia.

Importante: evitar obtener dos registros con la misma distancia y nodos intercambiados como este:

Devolver el nombre de ambas personas, la ciudad a la que pertenecen (que tendrá que ser Valladolid) y la distancia entre los domicilios de ambas personas contagiadas como se muestra en la imagen anterior.

Tener en cuenta que aproximadamente Valladolid tiene una distancia de unos 7,5 km de punta a punta, por lo que si alguna de las distancias es mayor de este valor, será indicativo de que hay algún tipo de error.

Person1	Person2	Ciudad	Distancia_domicilios_km
"Berta Romero"	"Sofía Vitoro"	"Valladolid"	2.223942025913843
"Fidel Figueroa"	"Berta Romero"	"Valladolid"	2.016001427283632
"Marcelino Rodriguez"	"Berta Romero"	"Valladolid"	1.6746342880783673

Person1	Person2	Ciudad	Distancia_domicilios_km
"Persona_Contagiada_1"	"Persona_Contagiada_2"	"Ciudad"	"Distancia_domicilios_km"
"Berta Romero"	"Sofía Vitoro"	"Valladolid"	2.223942025913843
"Fidel Figueroa"	"Berta Romero"	"Valladolid"	2.016001427283632
"Marcelino Rodriguez"	"Berta Romero"	"Valladolid"	1.6746342880783673

Conclusiones

Una vez realizado el análisis del dataset de contagios y rastreo de COVID-19, podemos decir que el avance tecnológico permite combatir de una forma distinta una propagación de un contagio o enfermedad en condiciones de pandemia, sin embargo también presenta retos como lo es el manejo de la información y datos sensibles de las personas, tomando en consideración que es un caso educativo o de práctica, pero en un ambiente con datos sobre información sensible sobre la salud e una persona, podría acontecer en multas o demandas por mal manejo y divulgación de información.

El uso de tecnologías Big Data en el control y combate de enfermedades infecciosas como el COVID-19 ofrece una capacidad sin precedentes para analizar y gestionar grandes volúmenes de datos en tiempo real, lo que resulta crucial para la toma de decisiones y la implementación de estrategias efectivas.