**3Análisis del tráfico en la ciudad de valencia y su relación con la contaminación atmosférica. Almacenamiento de la información en MongoDB y consultas**

Paso 1. El alumno deberá pensar, en primer lugar, la estructura de los documentos que debe generar en la base de datos para guardar la información (tanto para los datos de tráfico como para los datos de las estaciones meteorológicas), esta estructura deberá optimizar la información lo máximo posible de manera que la base de datos no duplique información. Se recomienda consultar con el profesor la posible solución adoptada por cada alumno antes de empezar a materializar la base de datos. Se recomienda revisar la estructura de los datos para detectar alguna inconsistencia en los mismos (por ejemplo, puede haber identificadores duplicados, puntos sin coordenadas, ficheros vacíos, replicados…).

**Documento de trafico**

# agregación para limitar el arreglo 'valores' a solo un elemento en el primer documento

documents = db.trafico.aggregate([

{

"$project": {

"valores": { "$slice": ["$valores", 1] }, # se limita el arreglo 'valores' a un solo elemento

"id\_tramo":1, # Se agrega el campo '\_id'

"coordenadas":1,

}

},

{

"$limit": 1 # Se Limita los resultados a solo el primer documento

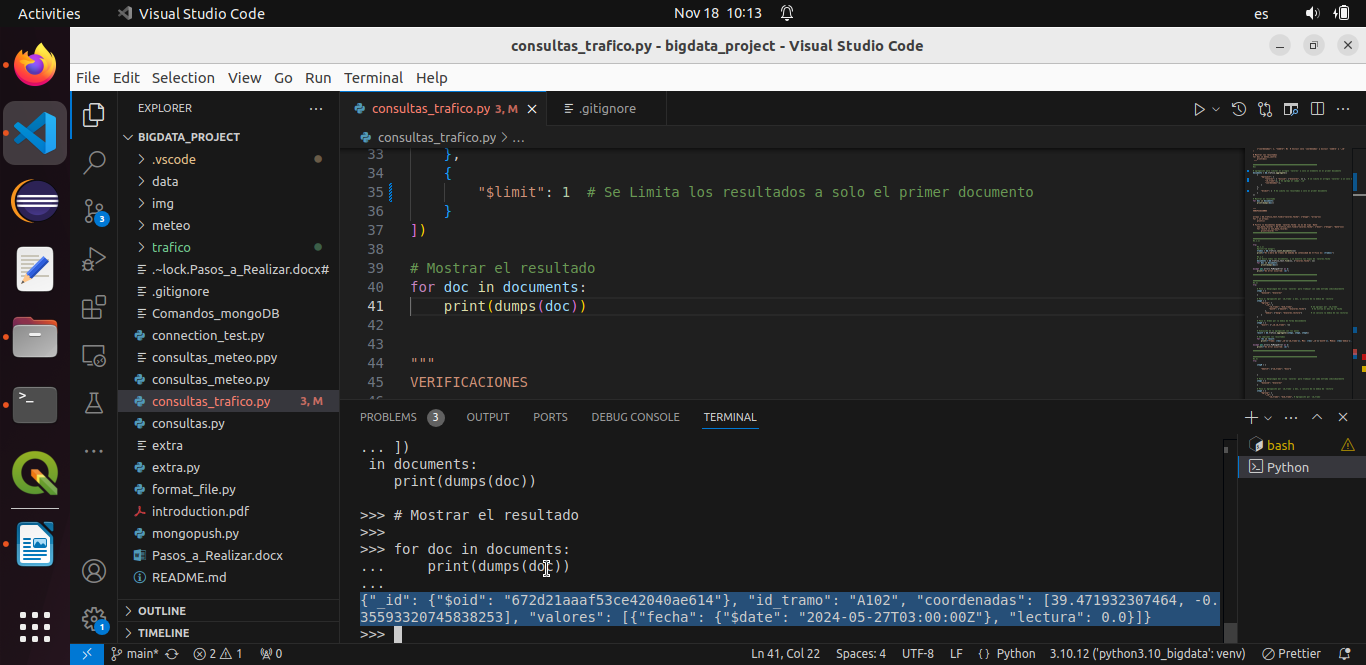
}

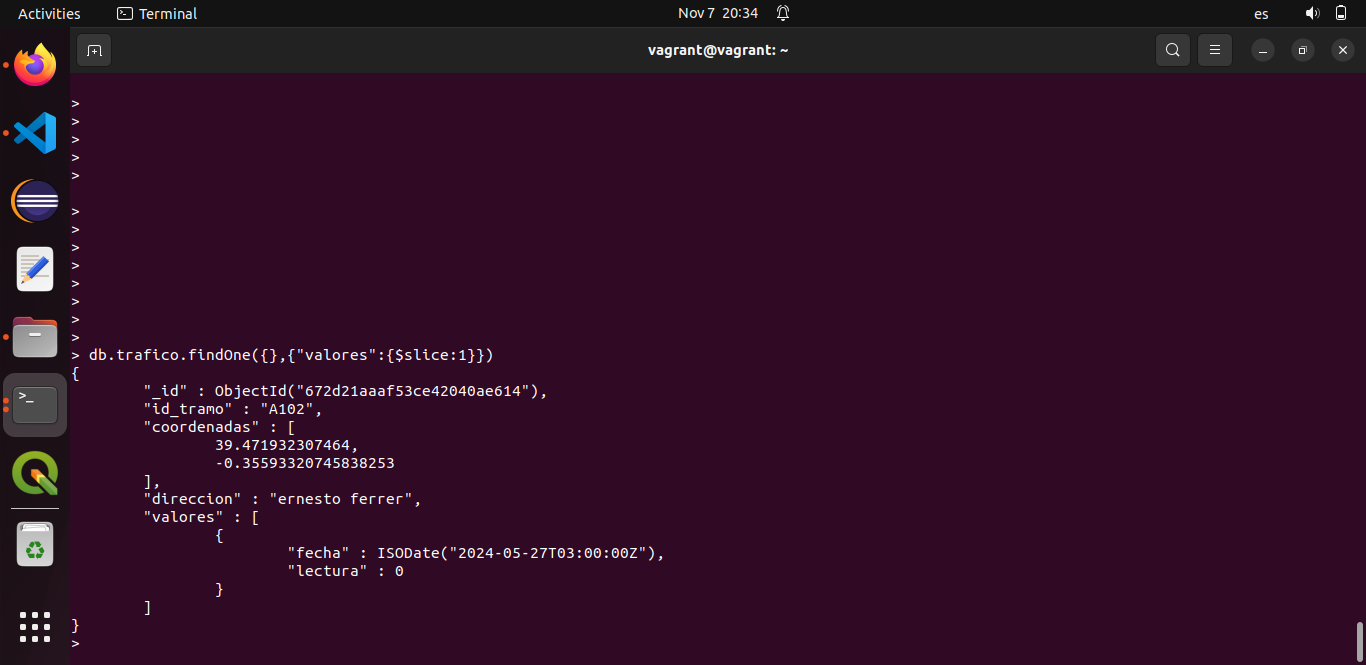
])

# Mostrar el resultado

for doc in documents:

print(dumps(doc))





**Documento de estaciones meteorológicas**

**Documentos de estaciones**

# agregación para limitar el arreglo 'valores' a solo un elemento en el primer documento

documents = db.trafico.aggregate([

{

"$project": {

"valores": { "$slice": ["$valores", 1] }, # se limita el arreglo 'valores' a un solo elemento

"id\_tramo":1, # Se agrega el campo '\_id'

"coordenadas":1,

}

},

{

"$limit": 1 # Se Limita los resultados a solo el primer documento

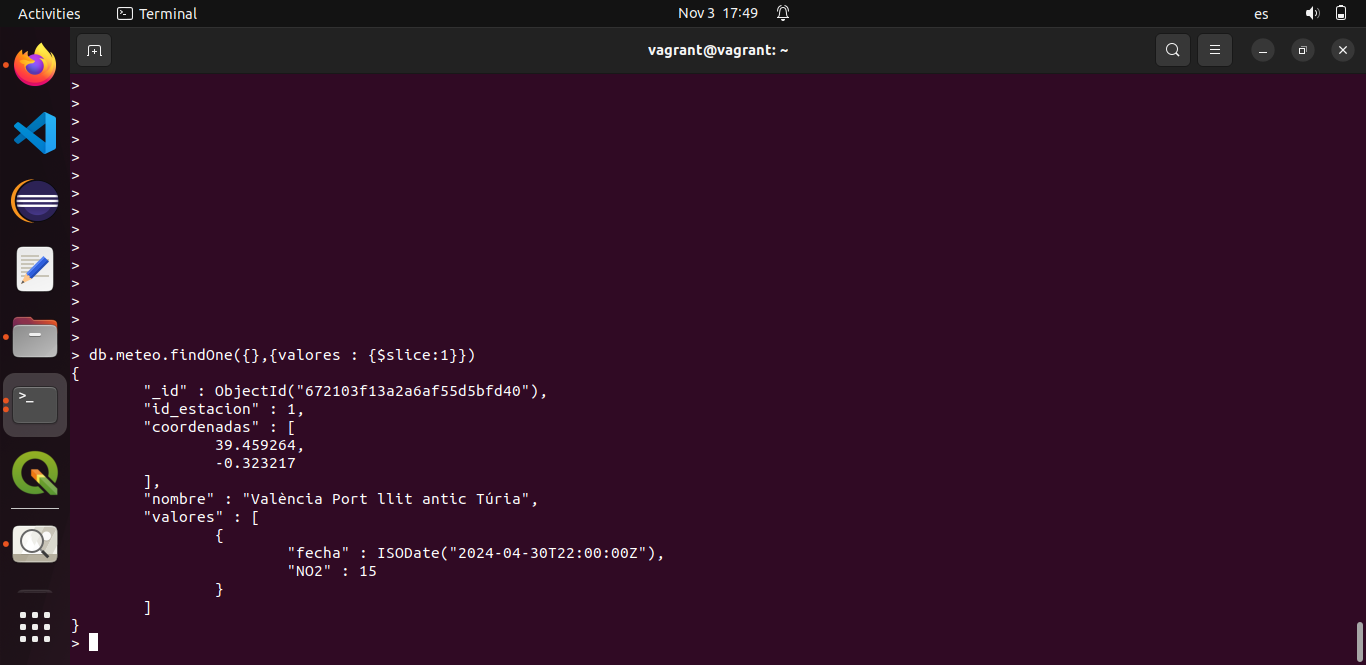
}

])

# Mostrar el resultado

for doc in documents:

print(dumps(doc))

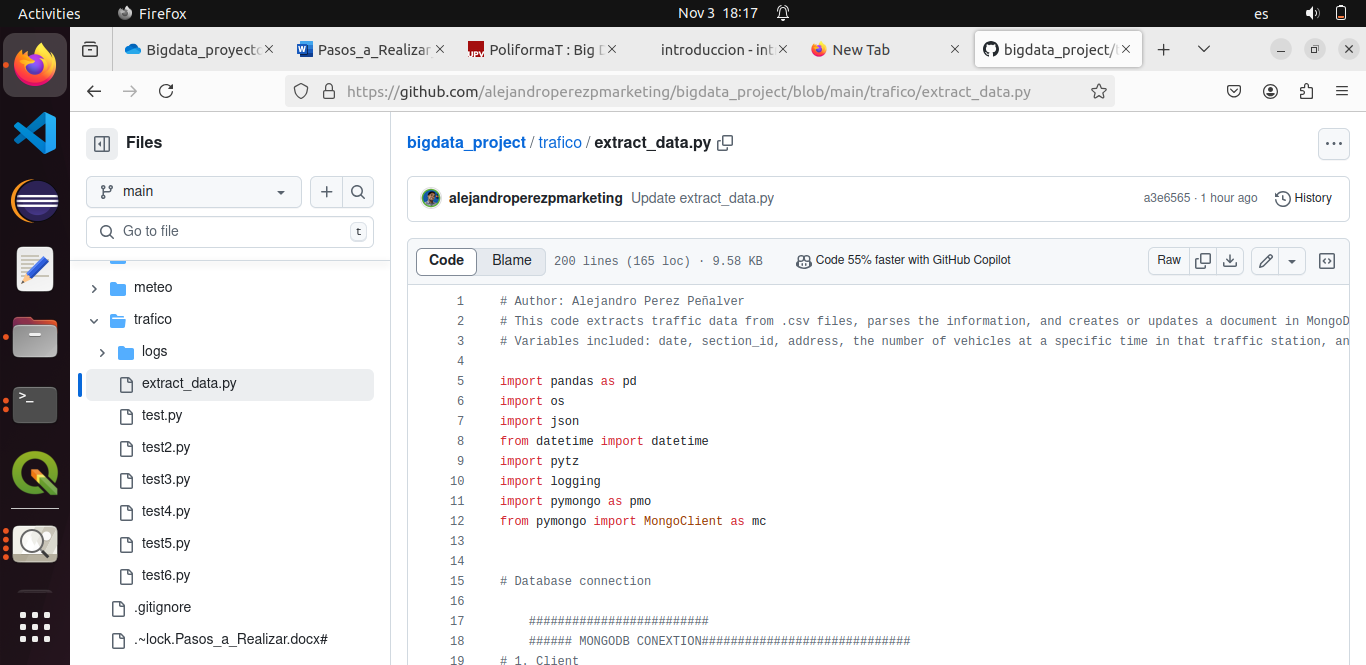


**Almacenamiento de los ficheros de tráfico (1 punto).** Se deberá realizar un programa en Python que lea los diferentes .csv originales del tráfico y los cargue en la base de datos. El documento en mongoDB debe contener el identificador el punto, la dirección, las coordenadas y los valores de lectura junto con la fecha.

Para la lectura y carga de la fecha se debe usar la librería datetime, de manera que se cargue la fecha en MongoDB en un formato reconocible por la base de datos. En caso de que la lectura sea de -1 o mayor de 5000 (datos erróneos) asignar un None a esa lectura. Tener en cuenta también la variable “estado” para cargar únicamente aquellas medidas válidas.

Los ficheros pueden presentar algún otro tipo de error o defecto en sus datos “confusión” por lo que el alumno debe tener en cuenta este tipo de errores o defectos a nivel de programación para generar una base de datos lo más consistente posible.

* **Código url:** https://github.com/alejandroperezpmarketing/bigdata\_project/blob/main/trafico/extract\_data.py

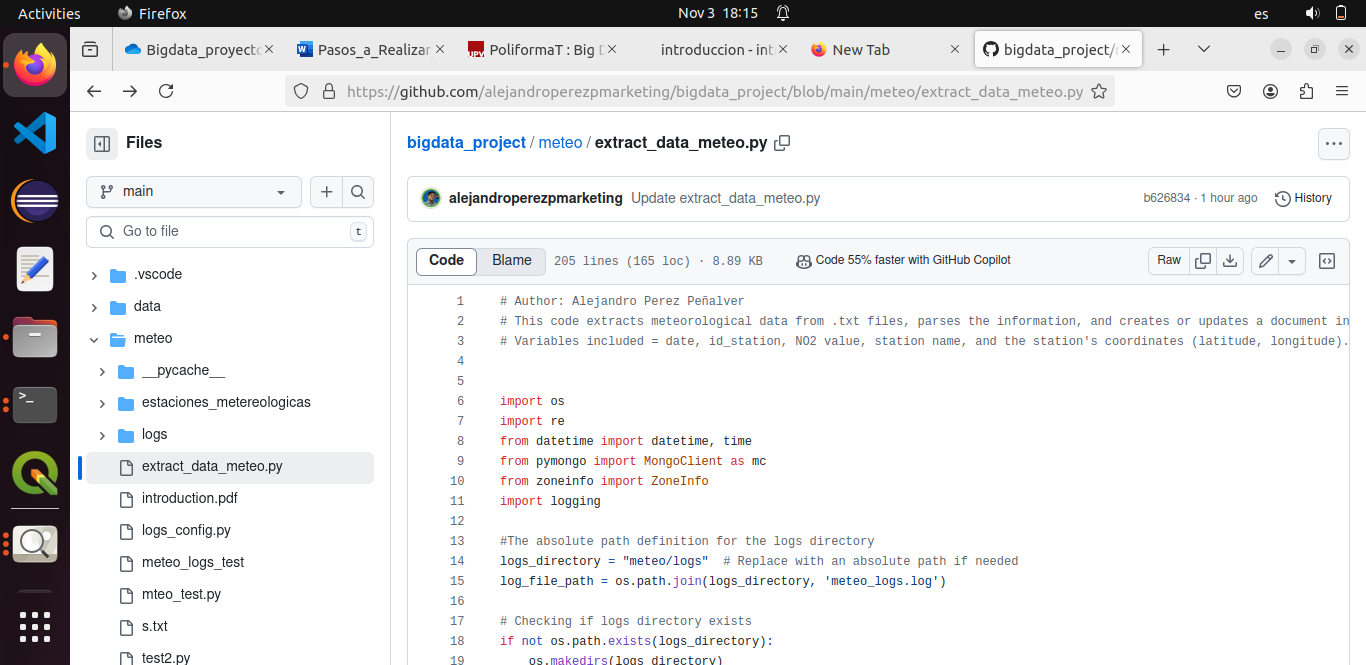


**Almacenamiento de las estaciones atmosféricas (1 punto):** Realizar un programa en Python para leer los ficheros .csv de cada estación meteorológica e importar la información a la base de datos generando una nueva colección. En este caso, el programa se deberá ejecutar una vez por cada estación meteorológica y deberá rescatar la información de NO2 únicamente para el mes de mayo y solo de aquellas estaciones que cuenten con esta información.

El programa debe buscar de forma automática la posición del dióxido de nitrógeno (NO2) dentro de la cabecera para cargar únicamente este valor en la base de datos.

En los documentos de la colección que generan estos ficheros deberán estar incorporadas las coordenadas de cada estación.

* **Código url**: https://github.com/alejandroperezpmarketing/bigdata\_project/blob/main/meteo/extract\_data\_meteo.py



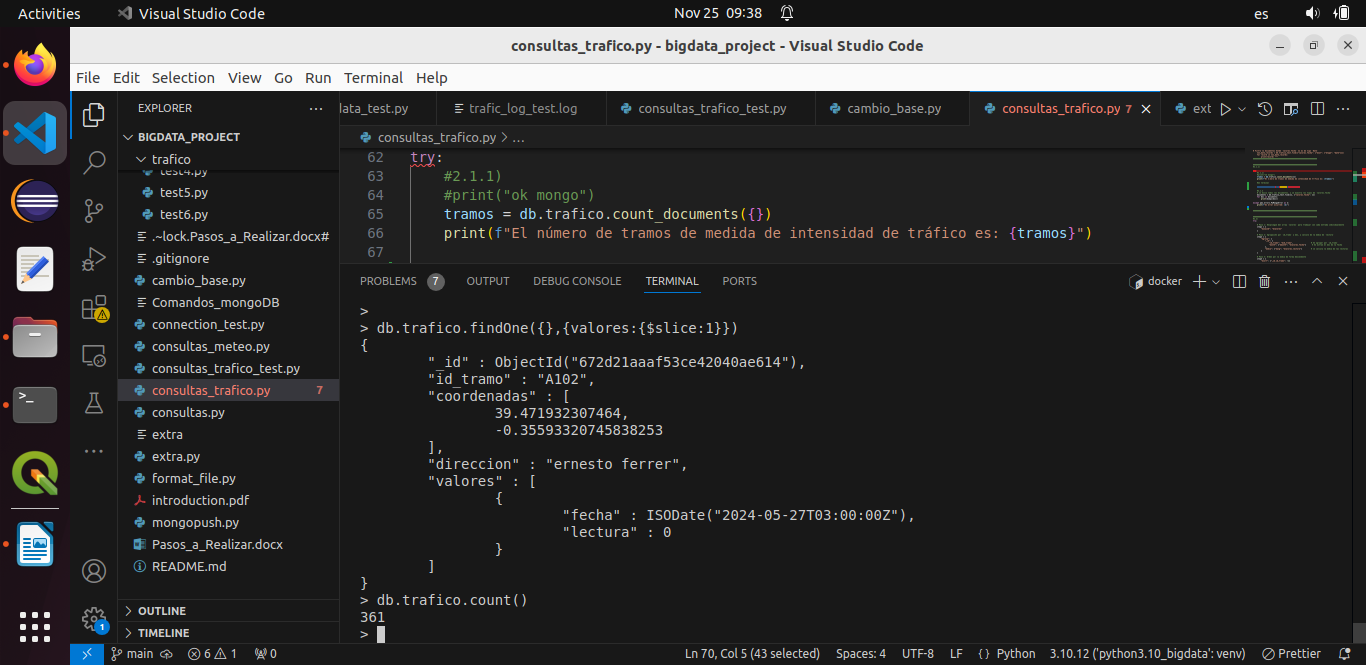
**Paso 2.** El alumno deberá contestar a las siguientes preguntas haciendo un copiar y pegar tanto de la consulta como del resultado obtenido para completar esta parte del proyecto.

**2.1 (0.5 puntos)** ¿Cuántos tramos de medida de intensidad de tráfico hay? ¿Cuál es la estructura del documento (usa la orden findOne() añadiendo el comando $slice en caso de que algún campo contenga mucha información)?

Cantidad de tramos de medida de intensidad de tráfico

#en terminal

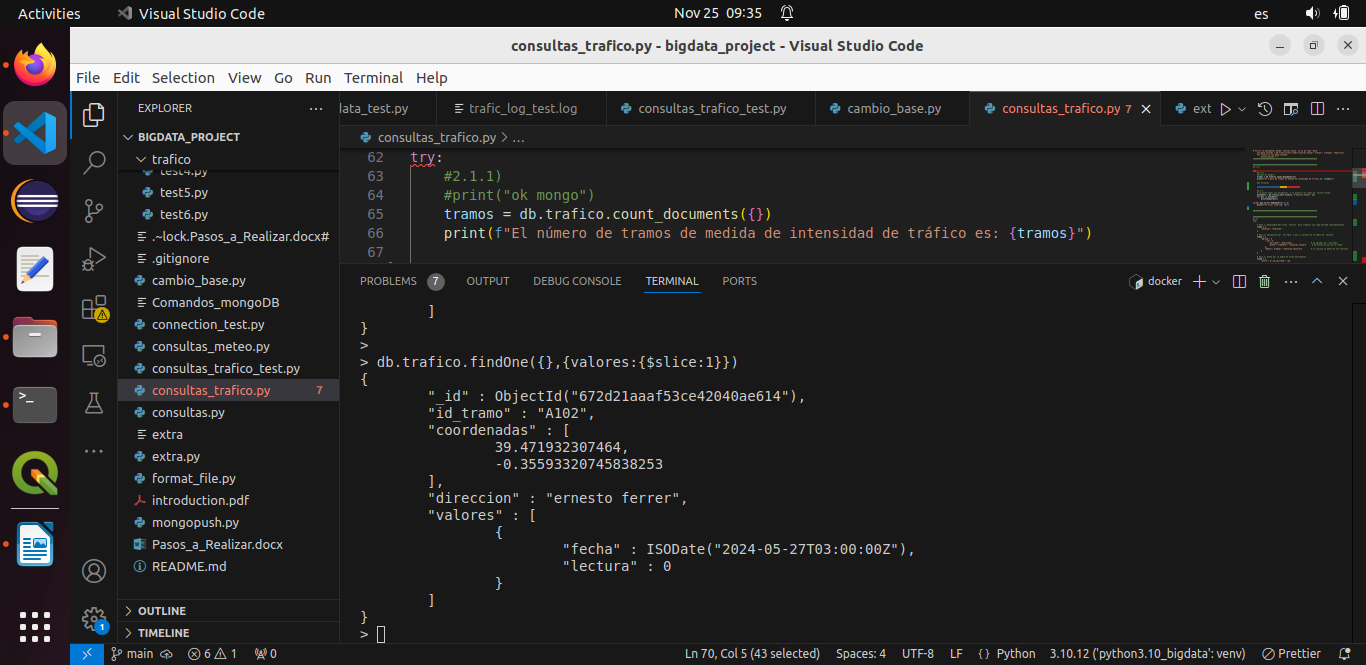
db.trafico.count()



Estructura del documento

# en terminal

db.trafico.findOne({},{valores:{$slice:1}})



import pymongo as pmo

from pymongo import MongoClient as mc

# Database connection

#########################

###### MONGODB CONEXTION#############################

# 1. Client

#client = mc('mongodb://vagrant:vagrant@localhost:27017/bigdata')

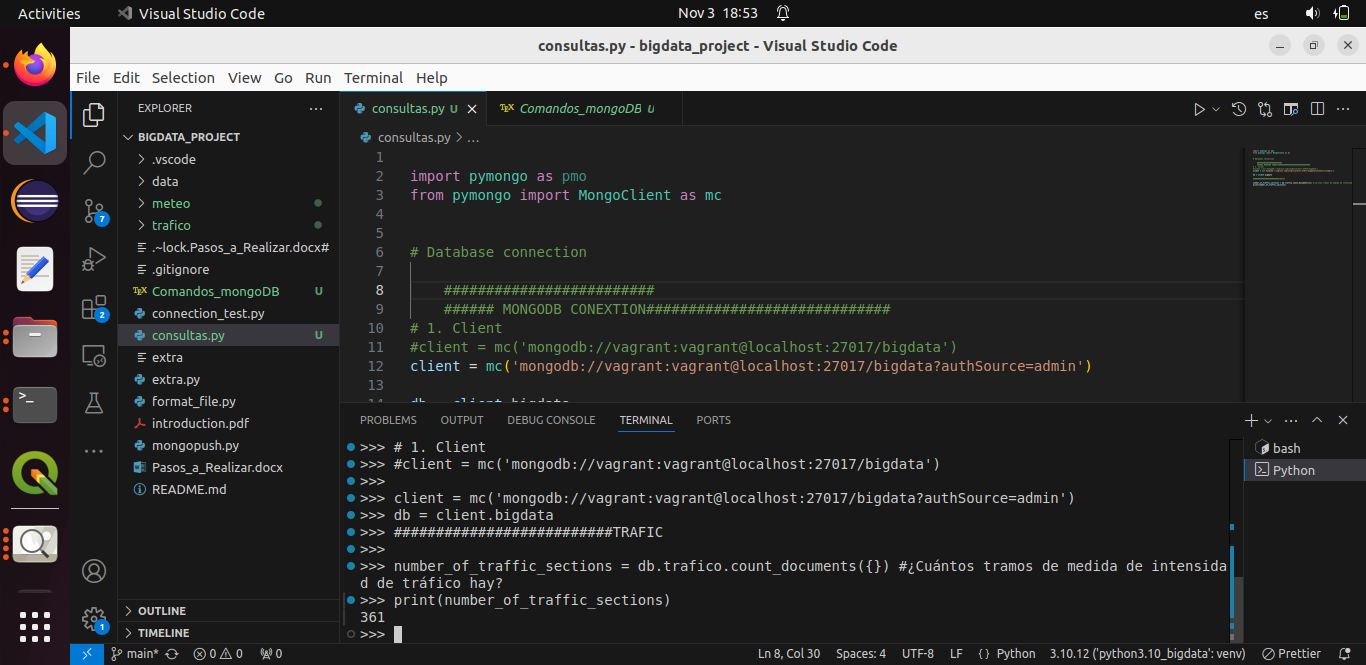
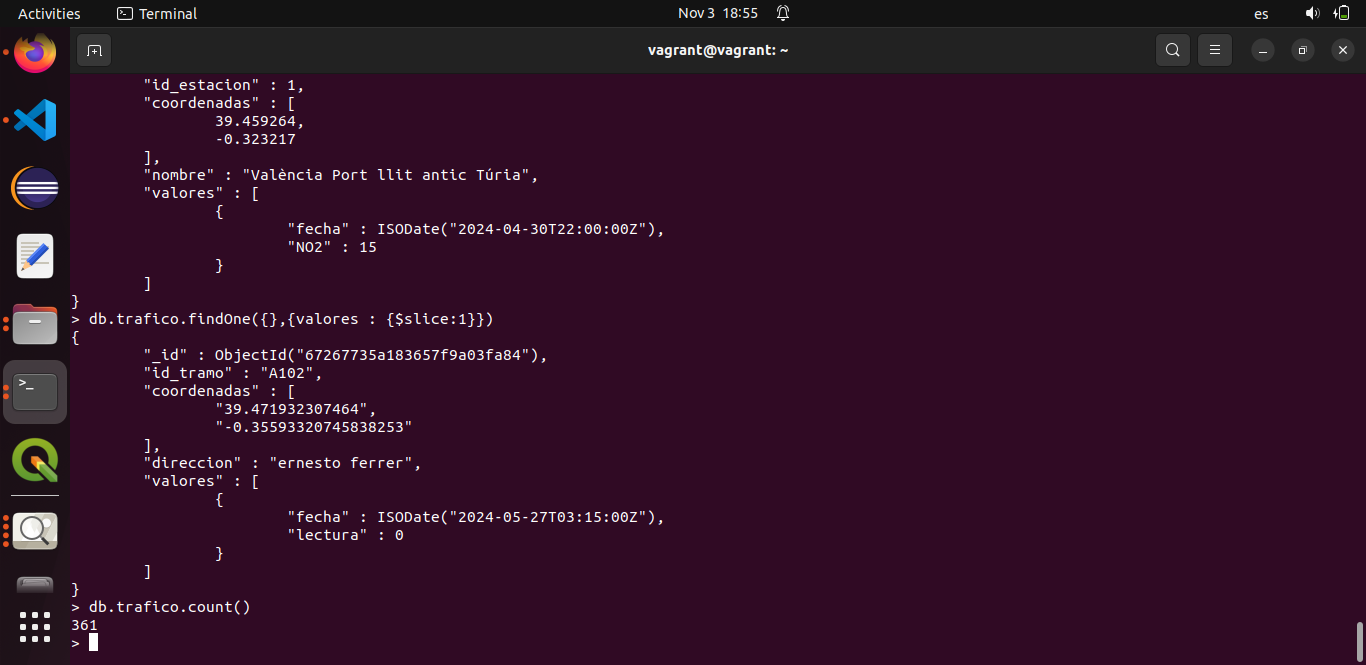
client = mc('mongodb://vagrant:vagrant@localhost:27017/bigdata?authSource=admin')

db = client.bigdata

##########################TRAFIC

number\_of\_traffic\_sections = db.trafico.count\_documents({}) #¿Cuántos tramos de medida de intensidad de tráfico hay?

print(number\_of\_traffic\_sections)



**Resultado: 361 documentos.**

**2.2 (1 punto).** Obtener el valor mensual medio de intensidad de tráfico para cada tramo, ordenando la salida de mayor a menor valor de intensidad de tráfico (operador $sort de la función aggregate). Como ejemplo, para comprobar la solución, para el tramo “A122” el resultado es 2420.4.

#### en terminal

db.trafico.aggregate([

{$unwind: "$valores"},

{$group: {

\_id: {

month: {$month:"$valores.fecha"},

id\_tramo: "$id\_tramo"

},

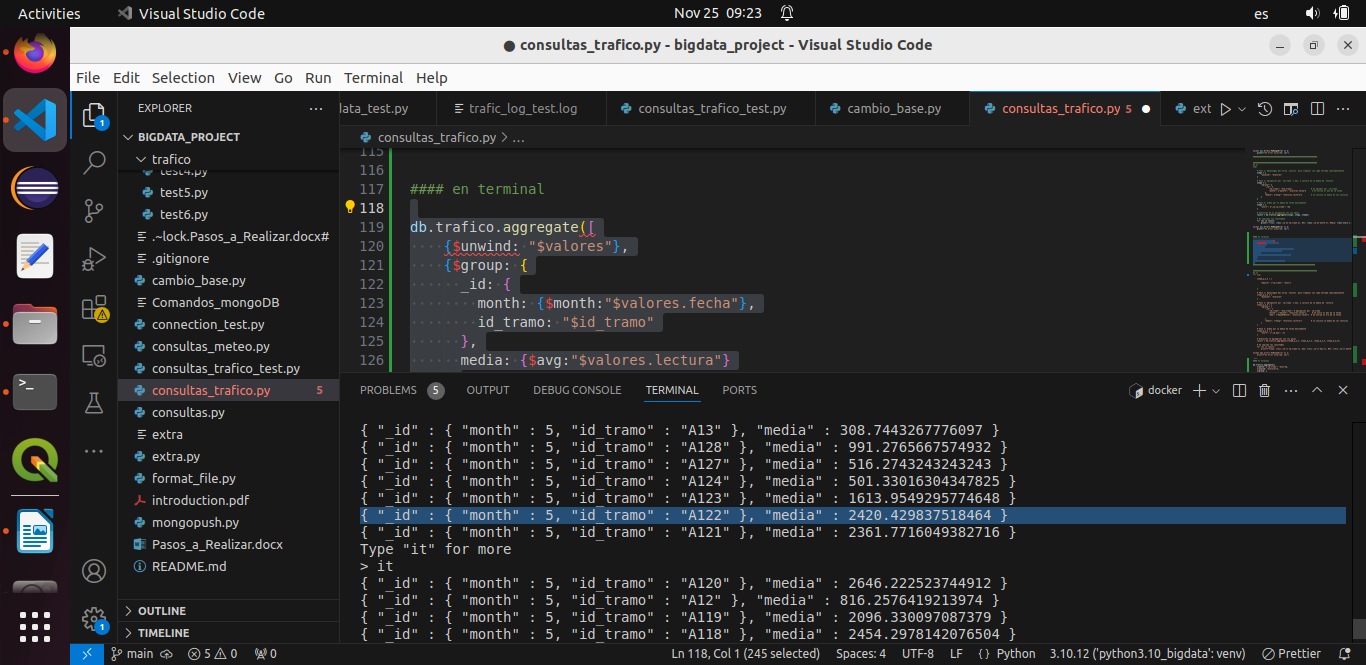
media: {$avg:"$valores.lectura"}

}

},

{$sort: {"\_id.id\_tramo":-1}}

])



**2.3 (0.5 puntos).** Obtener el valor medio de tráfico por día del mes para la espira con identificador igual a “A111”, ordenando la salida por día del mes. Como ejemplo, para comprobar la solución, para el primer día del mes el resultado es 975.7.

#2.3)

try:

step0 = {

"$match": {"id\_tramo": "A111"}

}

# Paso 1: Despliegue del array `valores` para trabajar con cada entrada individualmente

step1 = {

"$unwind": "$valores"

}

# Paso 2: Agrupación por `id\_tramo` y mes, y calculo de la media de `lectura`

step2 = {

"$group": {

"\_id": {

"id\_tramo": "$id\_tramo", # Agrupación por `id\_tramo`

"month": {"$month": "$valores.fecha"}, # Se extra el mes de la fecha

"day": {"$dayOfMonth": "$valores.fecha"} # Se extrae el día de la fecha

},

"media": {"$avg": "$valores.lectura"} # se calcula la media de las lecturas

}

}

# Paso 3: Orden por la media de forma descendente

step3 = {

"$sort": {"\_id.day": -1}

}

# Ejecución la agregación con los pasos

result = db.trafico.aggregate([step0, step1, step2, step3])

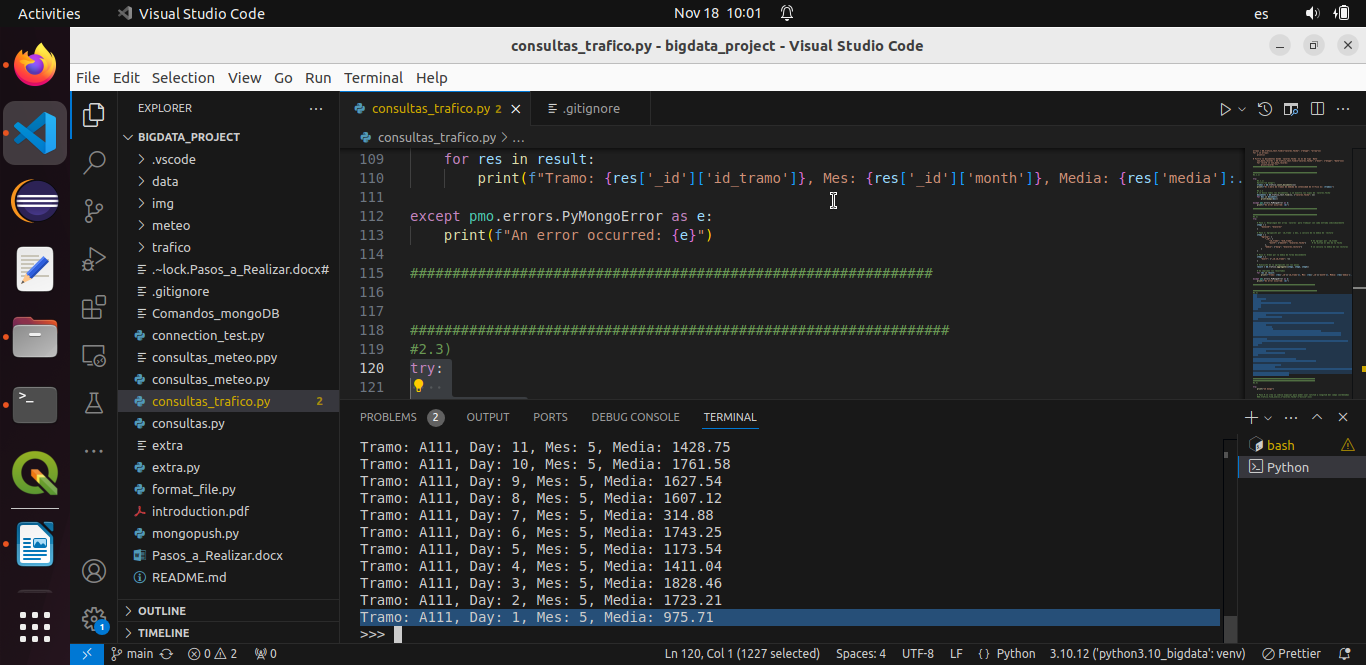
# Se imprime los resultados

for res in result:

print(f"Tramo: {res['\_id']['id\_tramo']}, Day: {res['\_id']['day']}, Mes: {res['\_id']['month']}, Media: {res['media']:.2f}")

except pmo.errors.PyMongoError as e:

print(f"An error occurred: {e}")



#### en terminal

db.trafico.aggregate([

{$match: {"id\_tramo": "A111"}},

{$unwind: "$valores"},

{$group: {

\_id: {

month: {$month:"$valores.fecha"},

day: {$dayOfMonth: "$valores.fecha"}

},

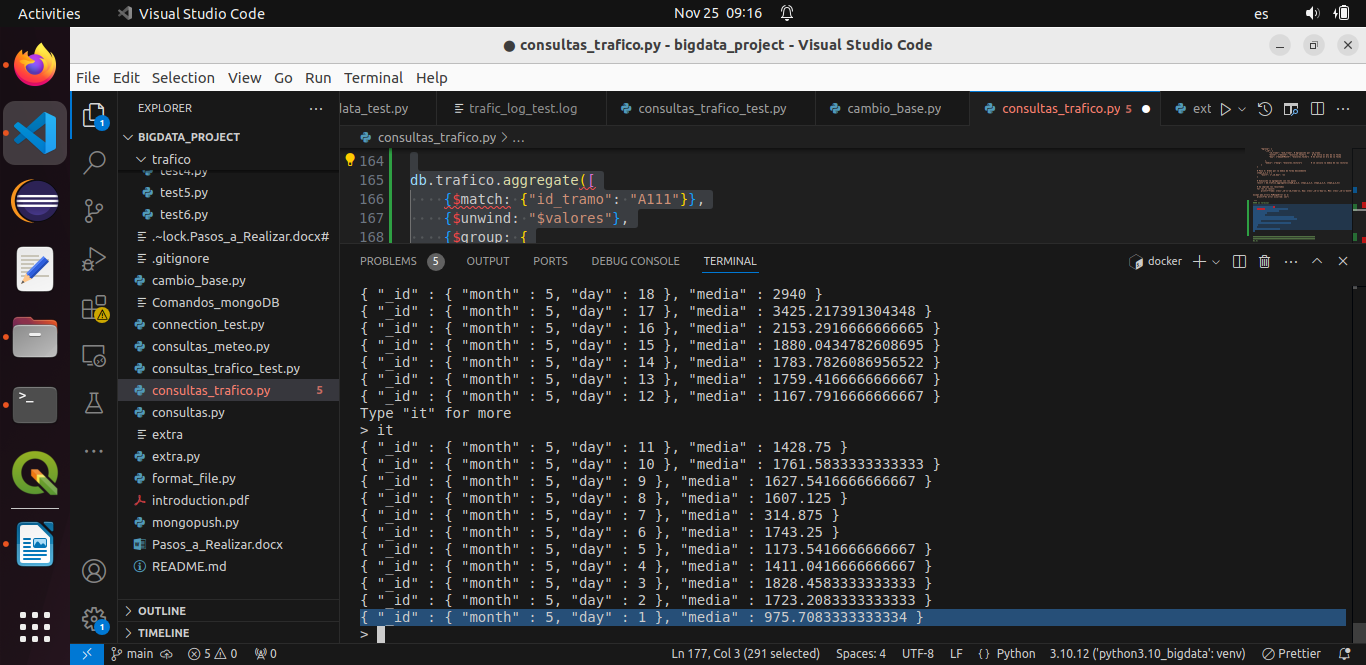
media: {$avg:"$valores.lectura"}

}

},

{$sort: {"\_id.day":-1}}

])



**2.4 (1.5 puntos).** Obtener un .csv donde se guarde el valor medio de tráfico para cada día del mes teniendo en cuenta los datos de tráfico de cada espira electromagnética cuyas coordenadas se encuentren dentro de un radio de 600 metros a partir de las coordenadas de la estación meteorológica Centre. Para ello se deben generar un índice “2dsphere” y utilizar la función $geoNear del método $aggregate para realizar la consulta geoespacial. A continuación, se deberá realizar la consulta guardando el resultado en una nueva colección, esta nueva colección se deberá exportar al correspondiente fichero csv y el resultado se deberá dibujar en una gráfica (usando LibreOffice, por ejemplo, dentro de la máquina virtual). Como ejemplo, para comprobar la solución, para el primer día del mes el resultado es 162.1.

### en terminal

db.trafico.createIndex({"coordenadas":"2dsphere"}) # creacion del indice

db.trafico.aggregate([

{$geoNear: {near: {"coordinates": [39.470718,-0.376384]},

distanceField: "DistanciaCalculada",

maxDistance: 600}},

{$unwind: "$valores"},

{$group: {

\_id: {

month: {$month:"$valores.fecha"},

day: {$dayOfMonth: "$valores.fecha"}

},

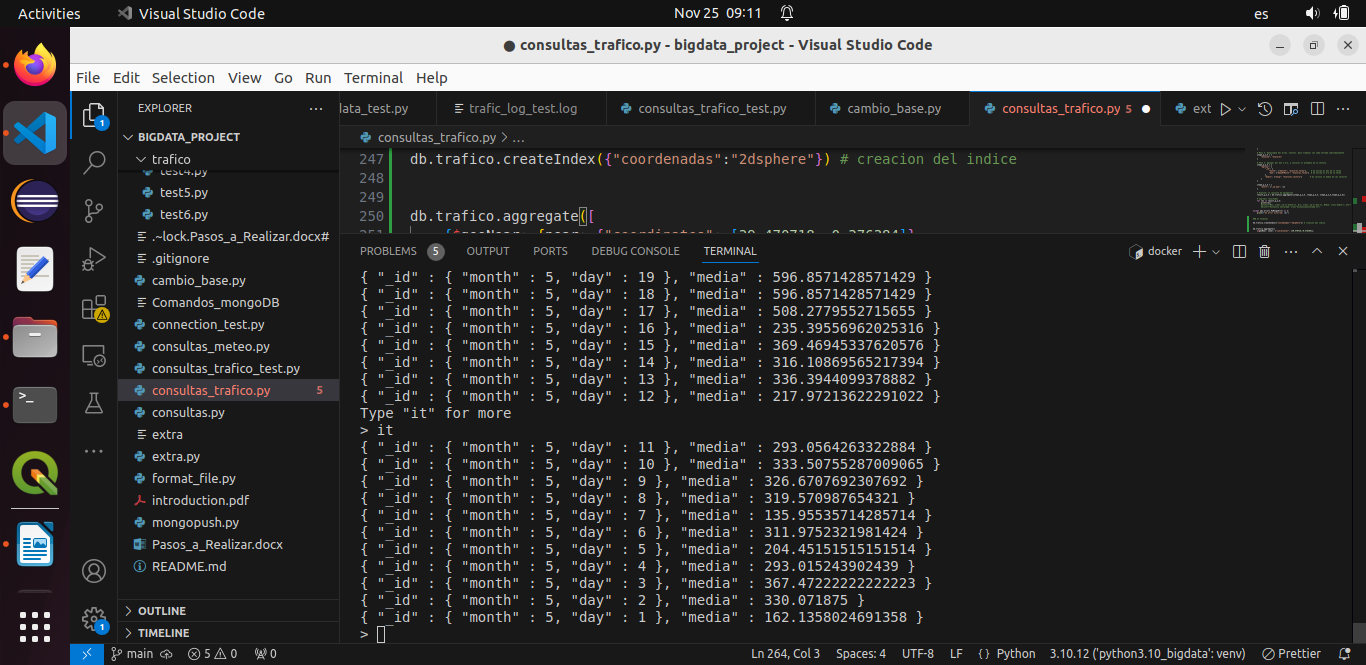
media: {$avg:"$valores.lectura"}

}

},

{$sort: {"\_id.day":-1}}

])



**2.5 (0.5 puntos)** Ordenar las estaciones meteorológicas por latitud de mayor a menor.

#en terminal

db.meteo.find(

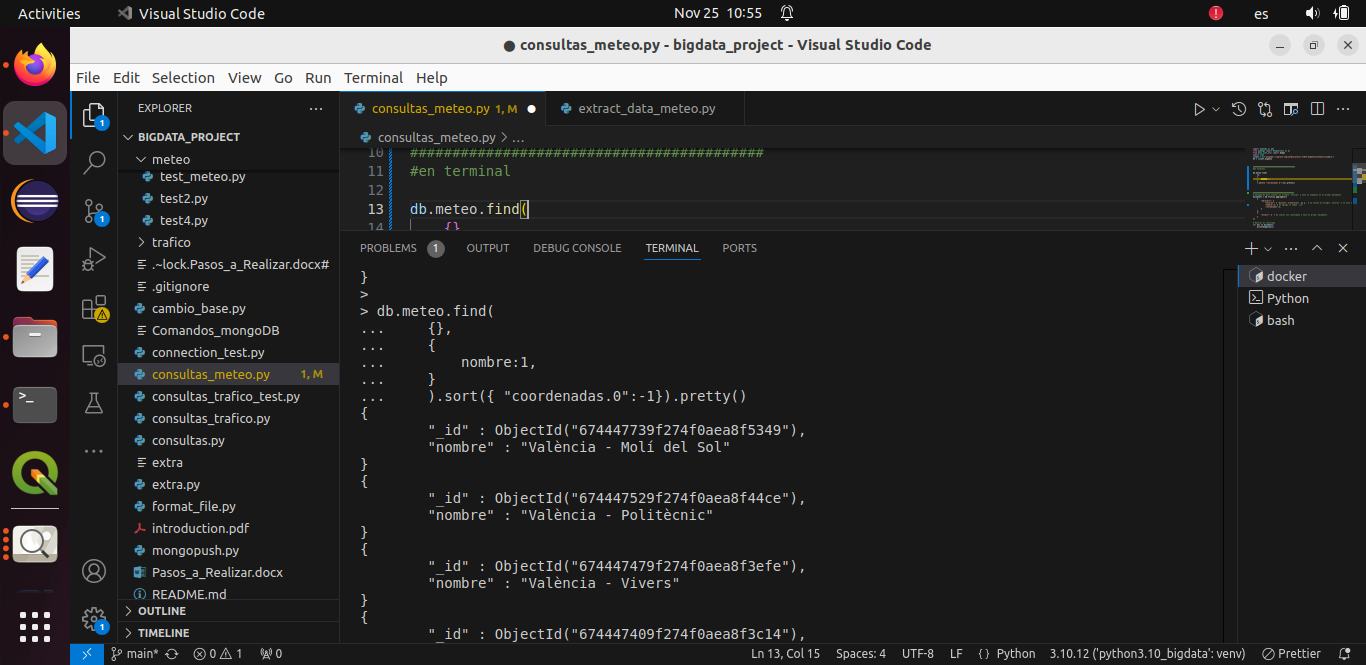
{},

{

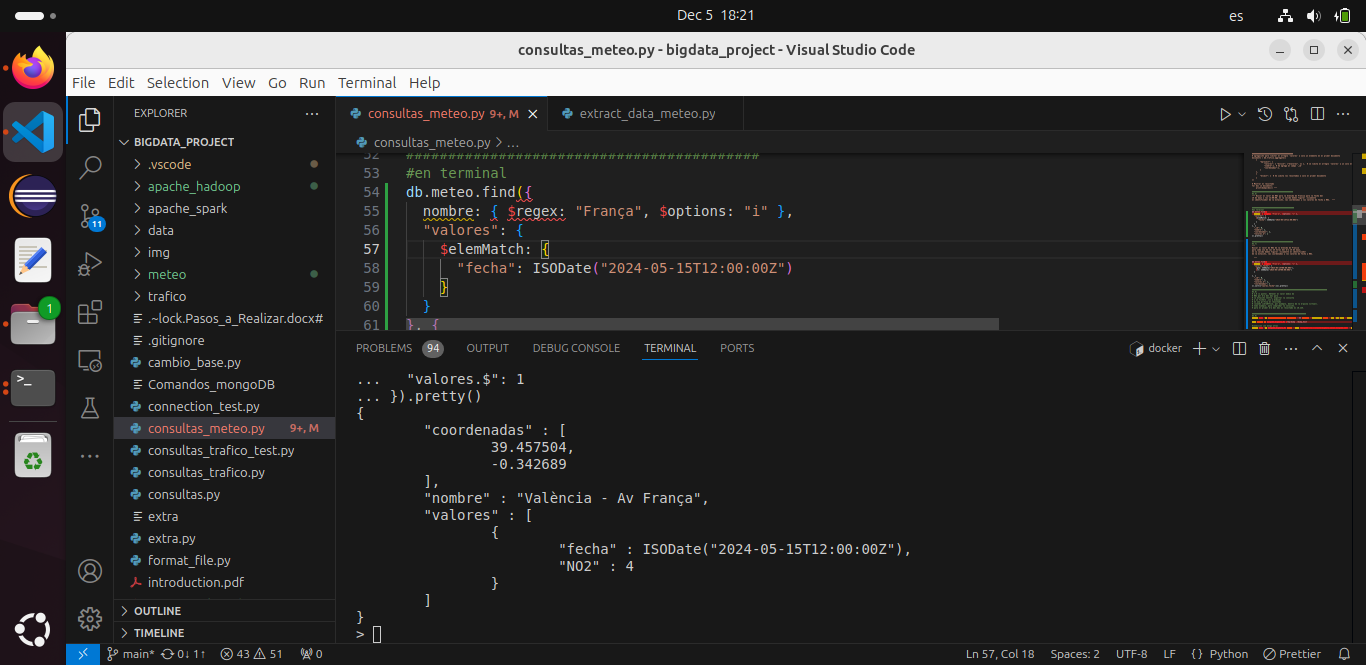
nombre:1,

}

).sort({ "coordenadas.0":-1}).pretty()



**2.6 (1 punto)** Buscar el valor de NO2 para la Avenida de Francia para la fecha del 15 de mayo a las 12:00:00, en la salida deben figurar únicamente el identificador de la estación, las coordenadas y los valores de fecha y NO2.



**2.7 (0.5 puntos)** Buscar el valor de NO2 de la Avenida de Francia para cada una de las 24 horas del día 15 de mayo, en la salida deben figurar únicamente el identificador de la estación, las coordenadas y los valores de fecha y NO2.

db.meteo.aggregate([

{

$match: {

nombre: { $regex: "França", $options: "i" }

}

},

{

$unwind: "$valores"

},

{

$match: {

"valores.fecha": {

$gte: ISODate("2024-05-15T00:00:00Z"),

$lt: ISODate("2024-05-16T00:00:00Z")

}

}

},

{

$project: {

\_id: 0,

nombre: 1,

valores: 1,

coordenadas: 1

}

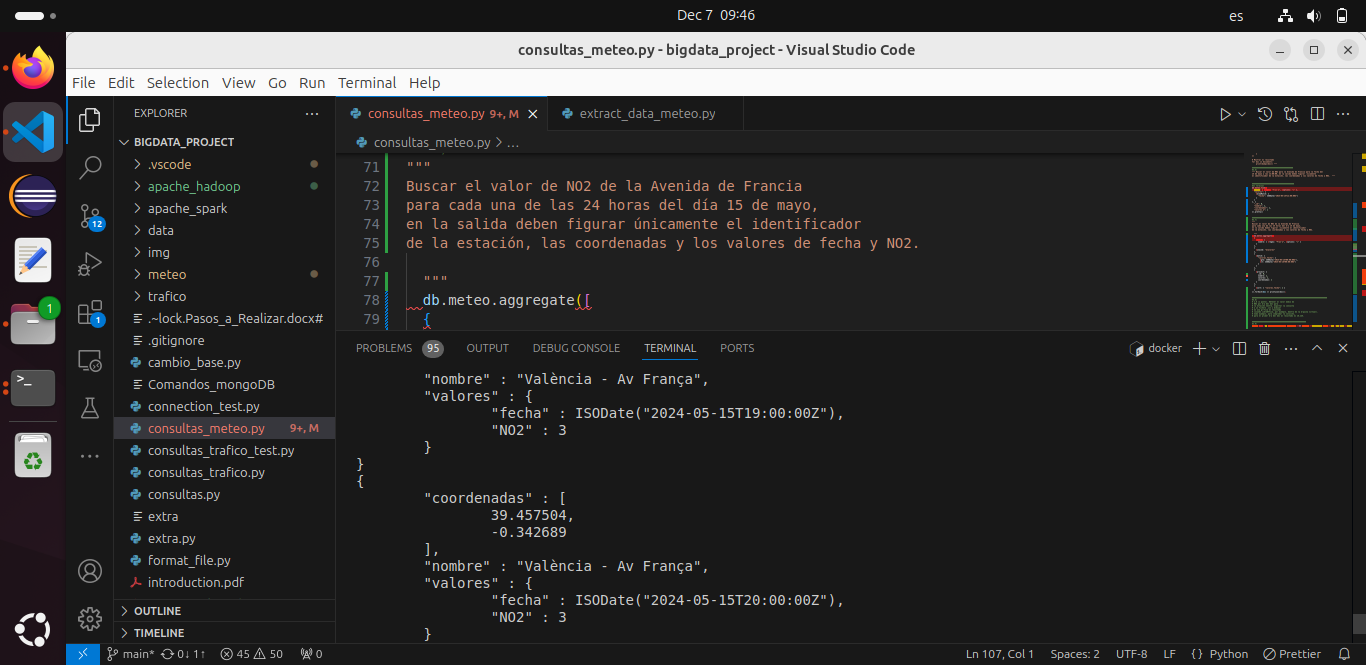
},

{

$sort: { "valores.fecha": 1 }

}

]).forEach(doc => printjson(doc));



**2.8 (1 punto).** Obtener el valor medio de NO2 por día del mes para la estación Centre, exportar la consulta a un fichero .csv y dibujar en una gráfica el resultado (usando LibreOffice, por ejemplo, dentro de la máquina virtual). Como ejemplo, para comprobar la solución, para el primer día del mes el resultado es 8.25.

**2.9 (1 punto):** Hacer copia de seguridad (ejecutable *Mongodump*) de la colección de estaciones meteorológicas e importarla a Mongo Atlas utilizando un clúster gratuito (M0). Buscar, en este clúster, para la estación Avda. Francia y el mes de mayo la fecha exacta de mayor y menor valor de NO2, la salida debe contener únicamente el nombre de la estación y los valores de fecha y NO2, el resultado se puede obtener a partir de dos consultas, una para el valor máximo y otra para el valor mínimo.

docker exec -it mongodb\_bigdata\_01 mongodump --db bigdata --collection meteo --out /tmp/dump --username vagrant --password vagrant --authenticationDatabase admin

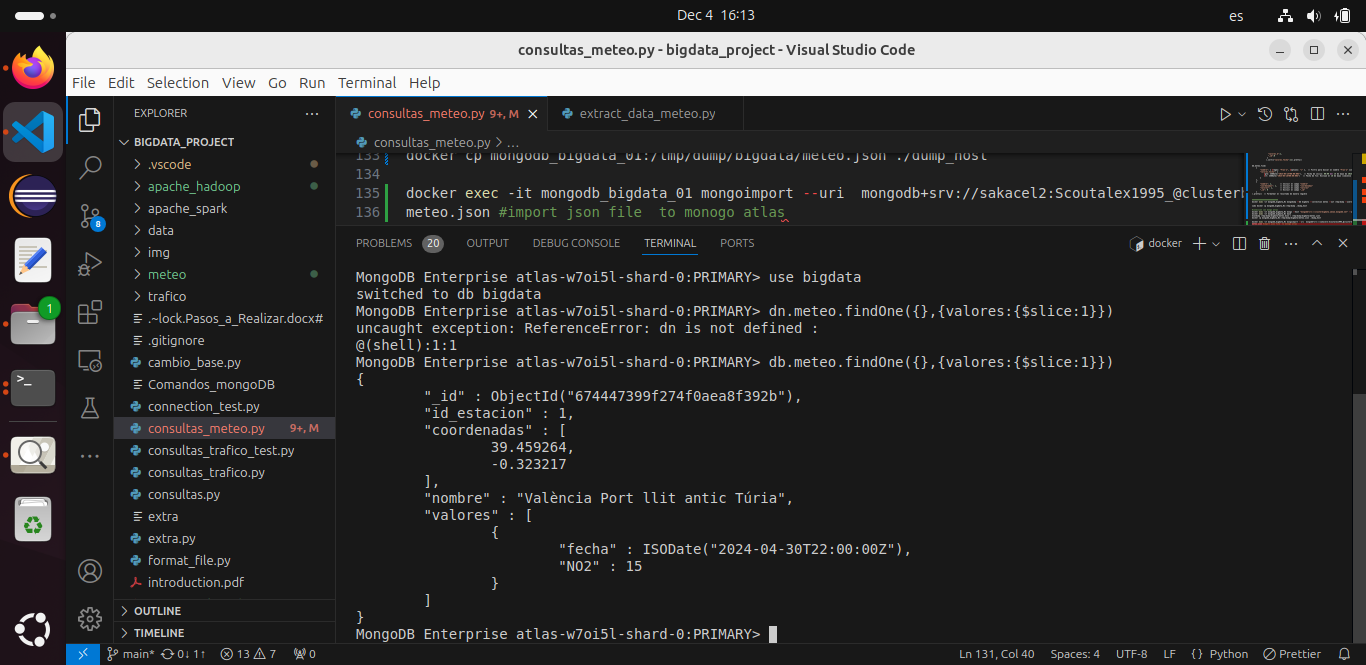
docker exec -it mongodb\_bigdata\_01 mongo --host "mongodb+srv://clusterbigdata.umnae.mongodb.net" --username sakacel2 --password Scoutalex1995\_ --authenticationDatabase admin

docker exec -it mongodb\_bigdata\_01 bash

bsondump /tmp/dump/bigdata/meteo.bson > /tmp/dump/bigdata/meteo.json

docker cp mongodb\_bigdata\_01:/tmp/dump/bigdata/meteo.json ./dump\_host

docker exec -it mongodb\_bigdata\_01 mongoimport --uri mongodb+srv://sakacel2:#######@clusterbigdata.umnae.mongodb.net/bigdata --collection meteo --type JSON --file tmp/dump/bigdata/



db.meteo.aggregate([

{

$match: {

nombre: { $regex: "França", $options: "i" } // Filtra documentos por nombre

}

},

{

$unwind: "$valores" // Descompón el array de "valores" para cada tupla de NO2 y fecha

},

{

$sort: { "valores.NO2": 1 } // Orden de los documentos por NO2 (de menor a mayor)

},

{

$group: {

\_id: null,

nombre: { $first: "$nombre" }, // Toma el primer valor de "nombre"

min: { $first: { NO2: "$valores.NO2", fecha: "$valores.fecha" } }, // Primer valor (mínimo)

max: { $last: { NO2: "$valores.NO2", fecha: "$valores.fecha" } } // Último valor (máximo)

}

},

{

$project: {

\_id: 0,

nombre: 1,

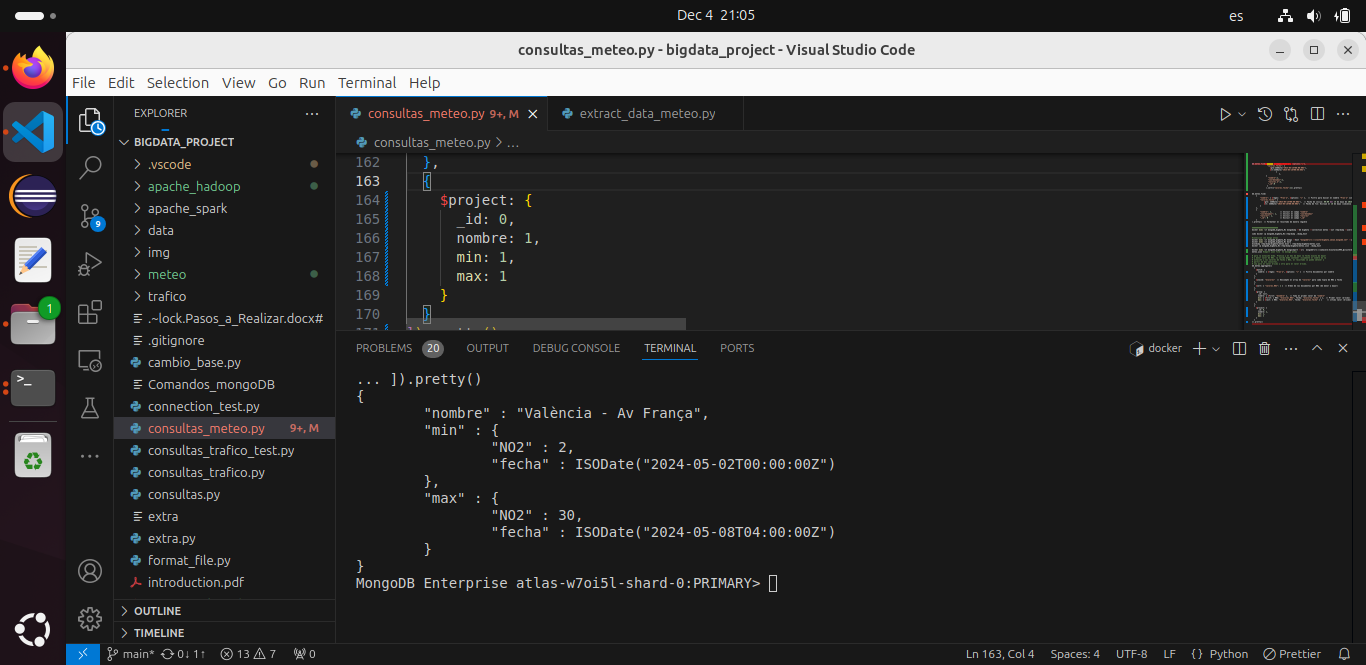
min: 1,

max: 1

}

}

]).pretty()



**Conclusiones (Evaluación de la competencia de responsabilidad y toma de decisiones)**

Se deberán presentar las conclusiones relativas a los datos con los que se ha trabajado (precisión, granularidad -resolución espacial y temporal-, confusión, valor, validez para los objetivos perseguidos, etc.), la plataforma o framework usado para su almacenamiento y/o análisis tanto de forma local como en la nube (facilidad de uso, generación de resultados, etc.), de los resultados obtenidos (correlaciones observadas, consecuencias, predicciones, agrupaciones, etc.) y sobre la bibliografía consultada por el alumno.

**Bibliografía (Evaluación de la competencia de responsabilidad y toma de decisiones)**

Se deberá presentar la bibliografía correspondiente que tendrá que contener todas las búsquedas realizadas por el alumno, tanto de carácter teórico en libros y artículos como de carácter técnico (búsqueda en internet de soluciones para la correcta instalación de las herramientas, para la generación de código -p.e. en blogs, plataforma Stack Overflow, información libre de diferentes cursos, etc.), todas estas referencias deberán estar adecuadamente referenciadas en el texto del proyecto de forma que no haya bibliografía sin su correspondiente referencia en el texto o en el código elaborado. Por último, en el apartado de conclusiones debe haber una parte sobre la bibliografía usada donde el alumno deberá considerar la importancia, fiabilidad y sencillez de uso o implementación de las diferentes fuentes usadas.

**GeoJson documentation**

https://datatracker.ietf.org/doc/html/rfc7946