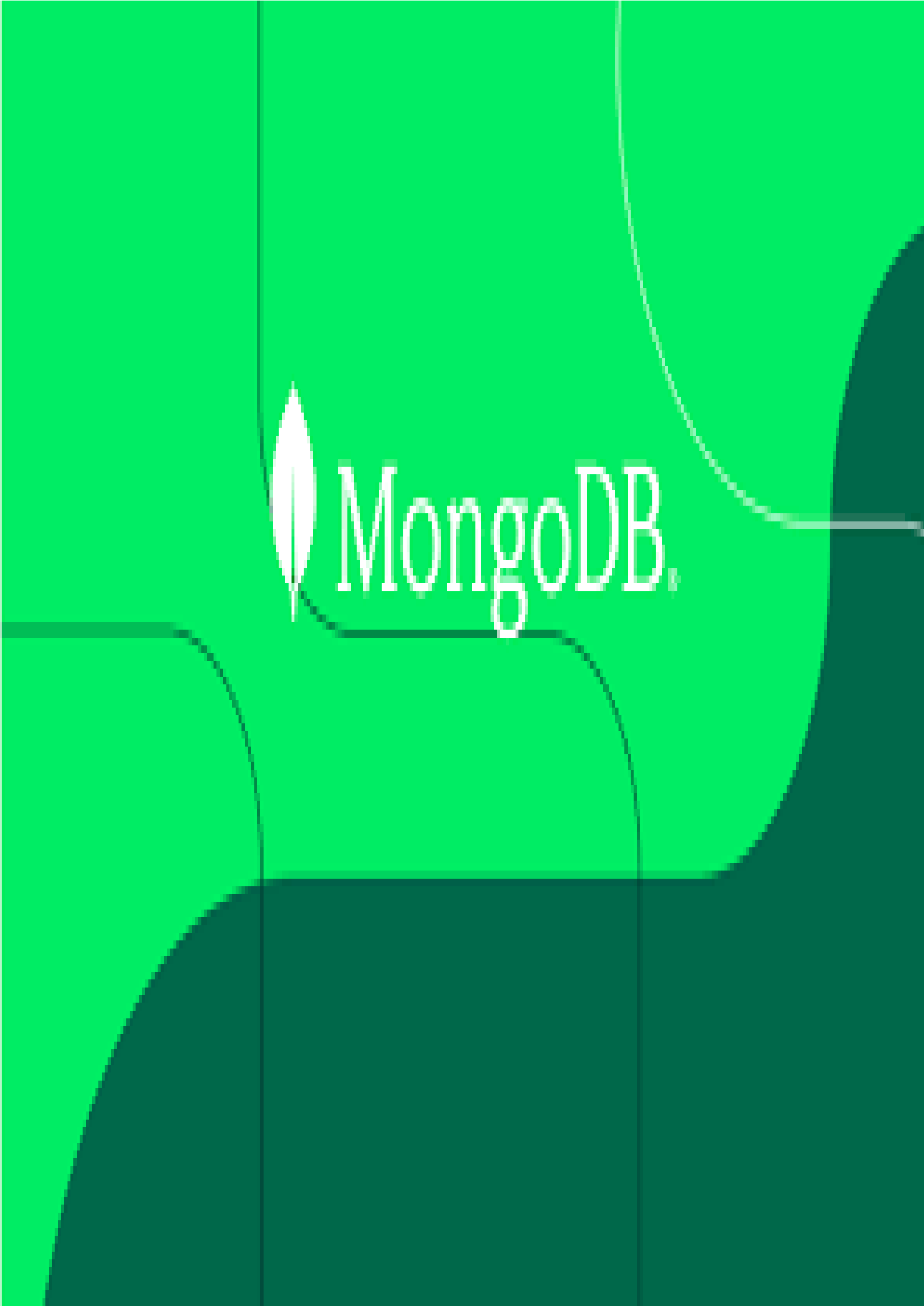

MongoDB.

CONEXIÓN Y CONSULTAS EN MONGODB

OBJETIVO

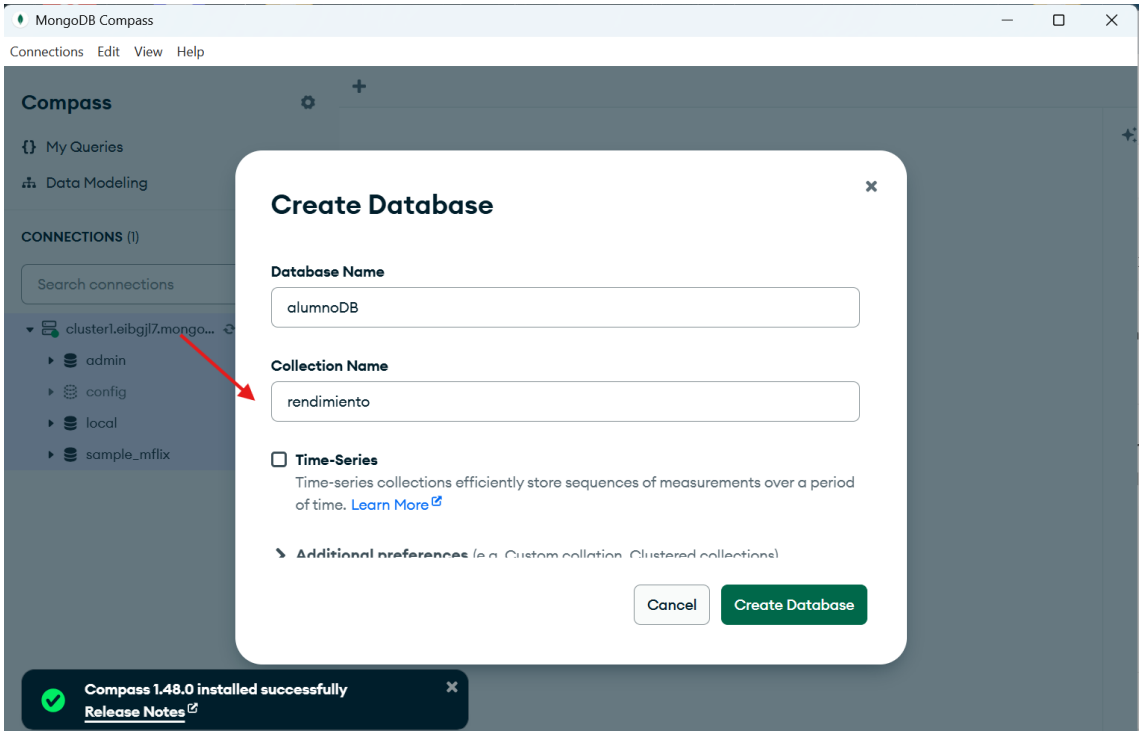
El objetivo de esta práctica es demostrar la capacidad de conectarse a un clúster de MongoDB, importar datos desde un CSV, y realizar consultas y agregaciones usando distintas herramientas: MongoDB Compass y Python.

DESARROLLO DE LA TAREA

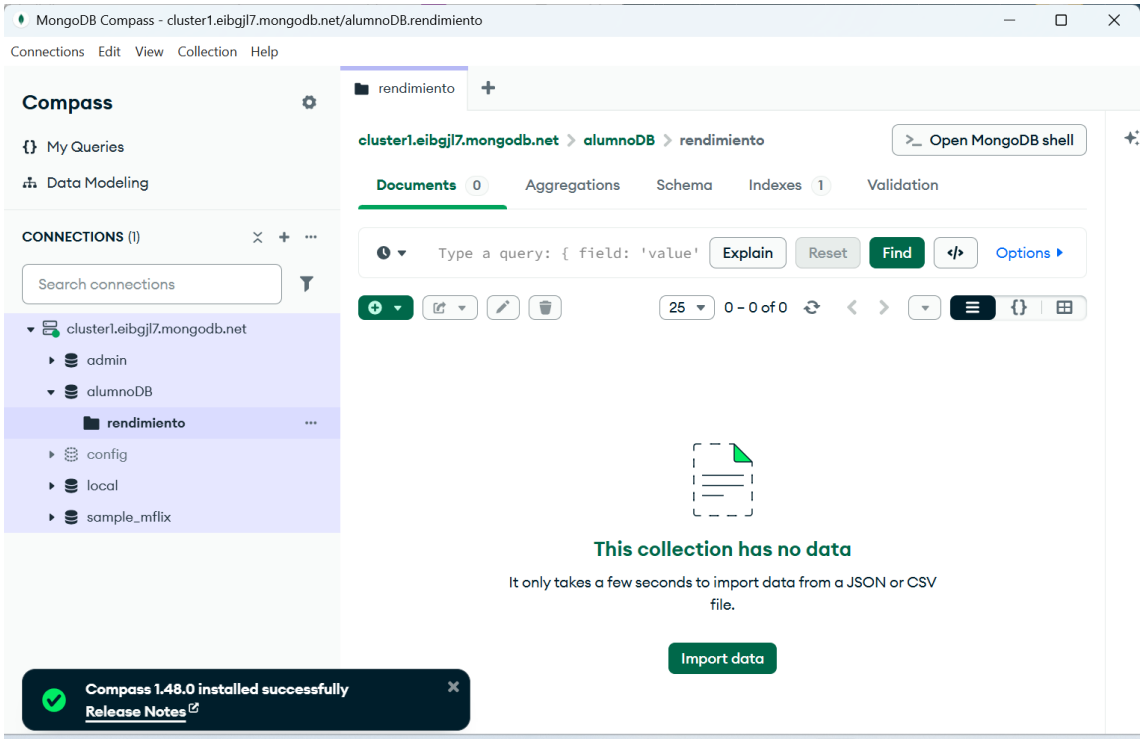
Para la tarea me descargo de Kaggle un archivo csv: Rendimiento de estudiantes, que contiene los siguientes campos:

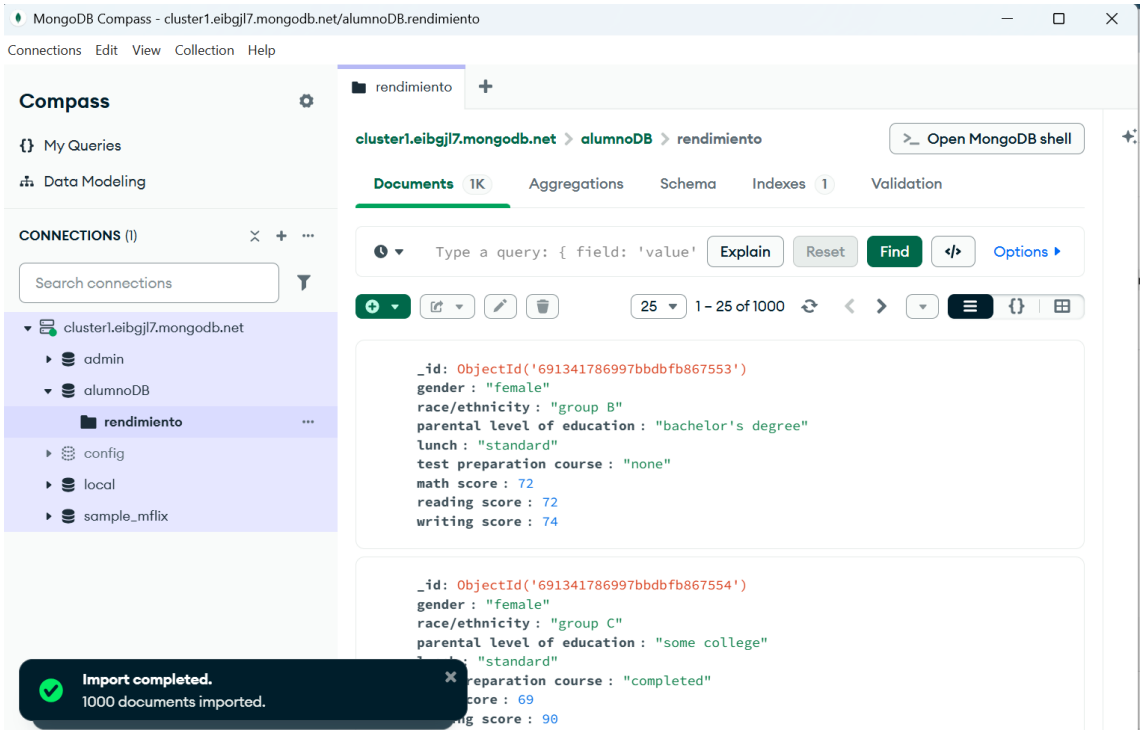
- gender
- race/ethnicity
- parental level of education
- lunch
- test preparation course
- math score
- reading score
- writing score

Se creó un clúster en MongoDB Atlas y se importó el CSV a la base de datos ***alumnoDB*** y la colección ***rendimiento.***



Importamos los datos





Explicación del proceso:

1. Abrir MongoDB Compass.
2. Crear la base de datos alumnoDB.
3. Crear la colección rendimiento.
4. Usar la opción **Import Data** y seleccionar el CSV.

La importación también se podría hacer de forma manual desde el terminal con el comando mongoimport:

mongoimport --uri "mongodb+srv://<user>:<pwd>@<cluster-url>/alumnoDB" \

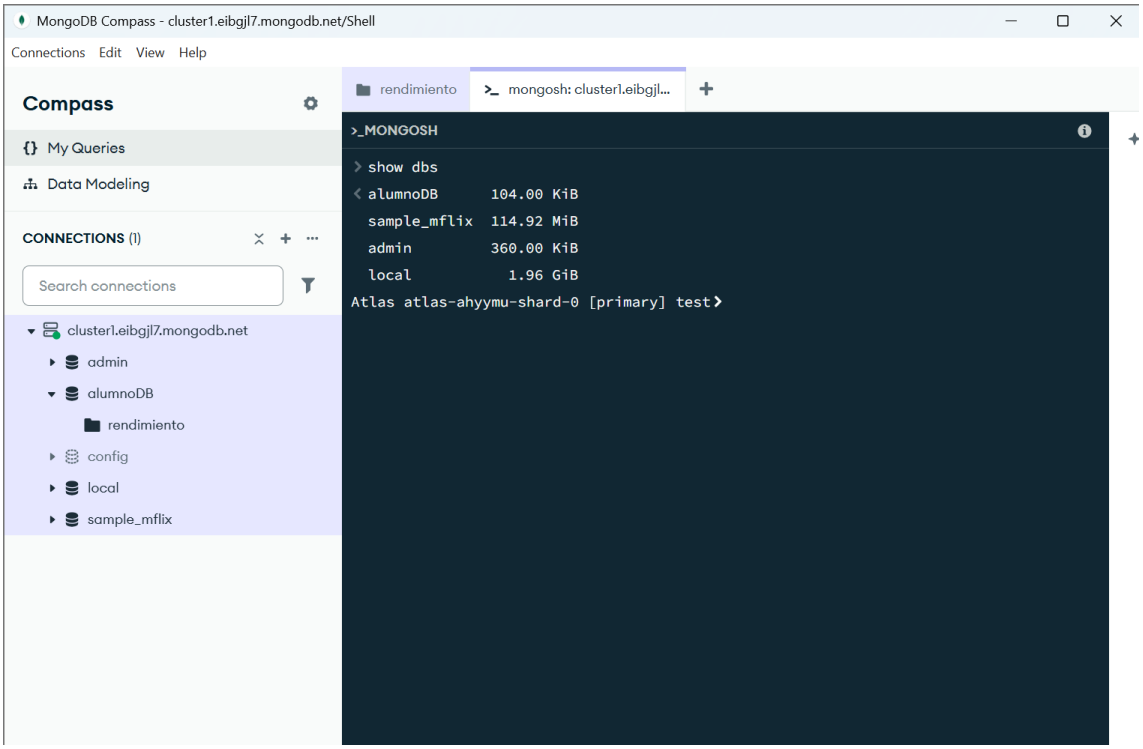
--collection rendimiento \

--type csv --headerline \

--file /ruta/a/rendimiento.csv

LISTAR BASES DE DATOS DEL SERVIDOR

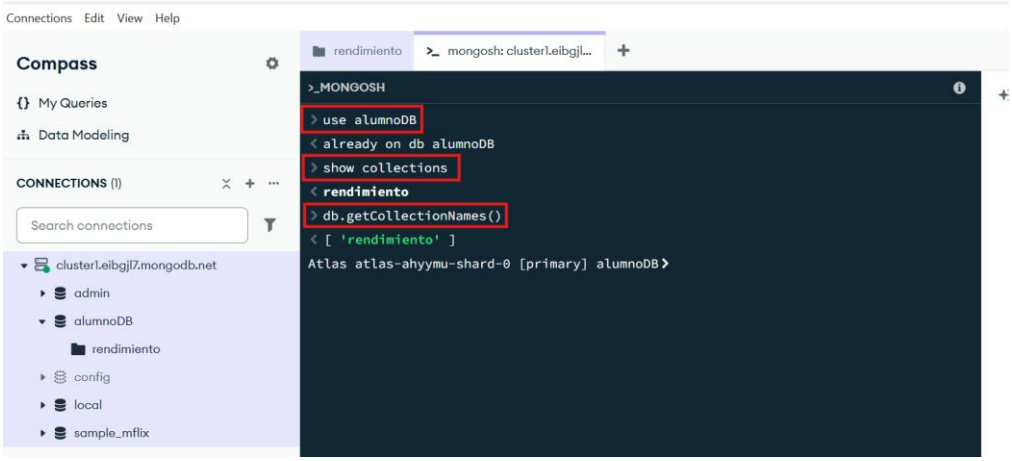
Show dbs



Para realizar consultas de una Base de Datos, lo primero que tendremos que hacer, será ubicarnos en ella →***use <nombre base de datos>***

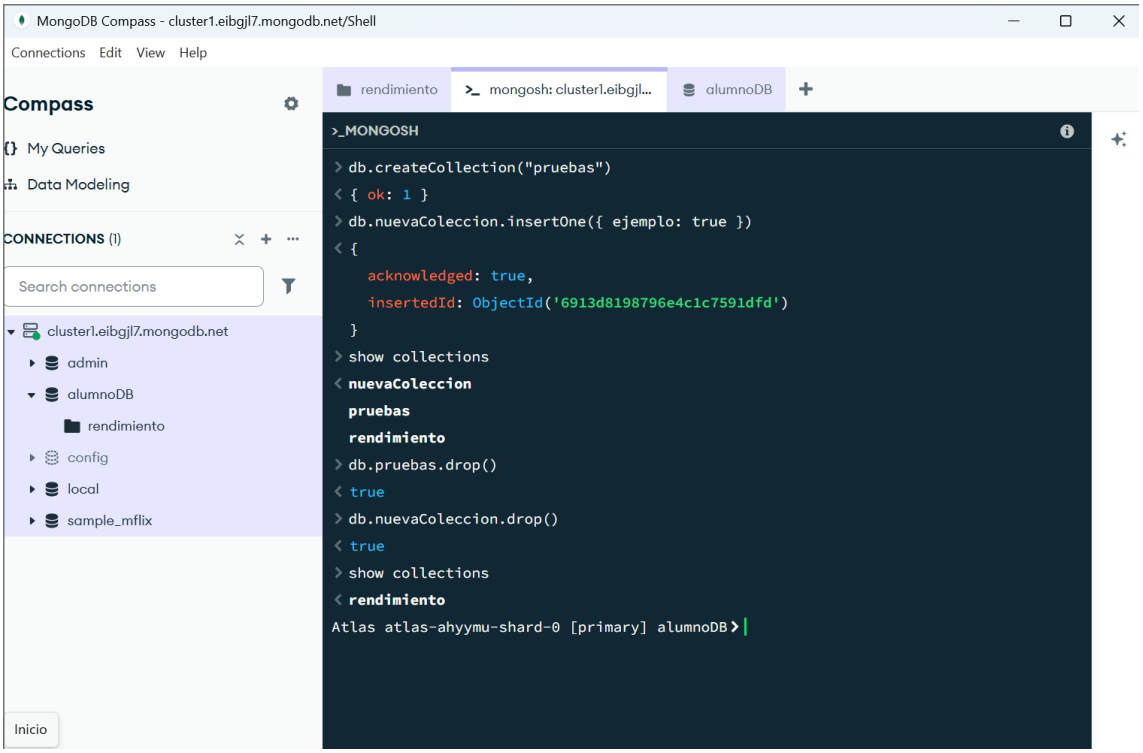
LISTAR COLECCIONES DE UNA BASE DE DATOS

Podremos utilizar ***show collections*** o con JS ***db.getCollectionNames()***



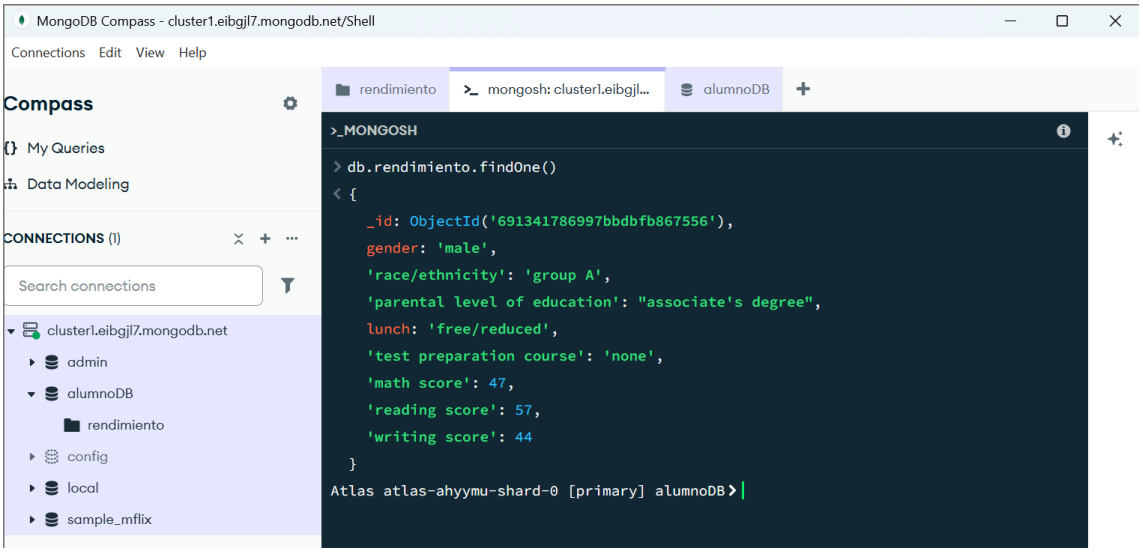
CREAR Y ELIMINAR COLECCIONES

Para crear usamos el comando ***db.createCollection*** o con ***insertOne***. Para eliminar, lo haremos con drop

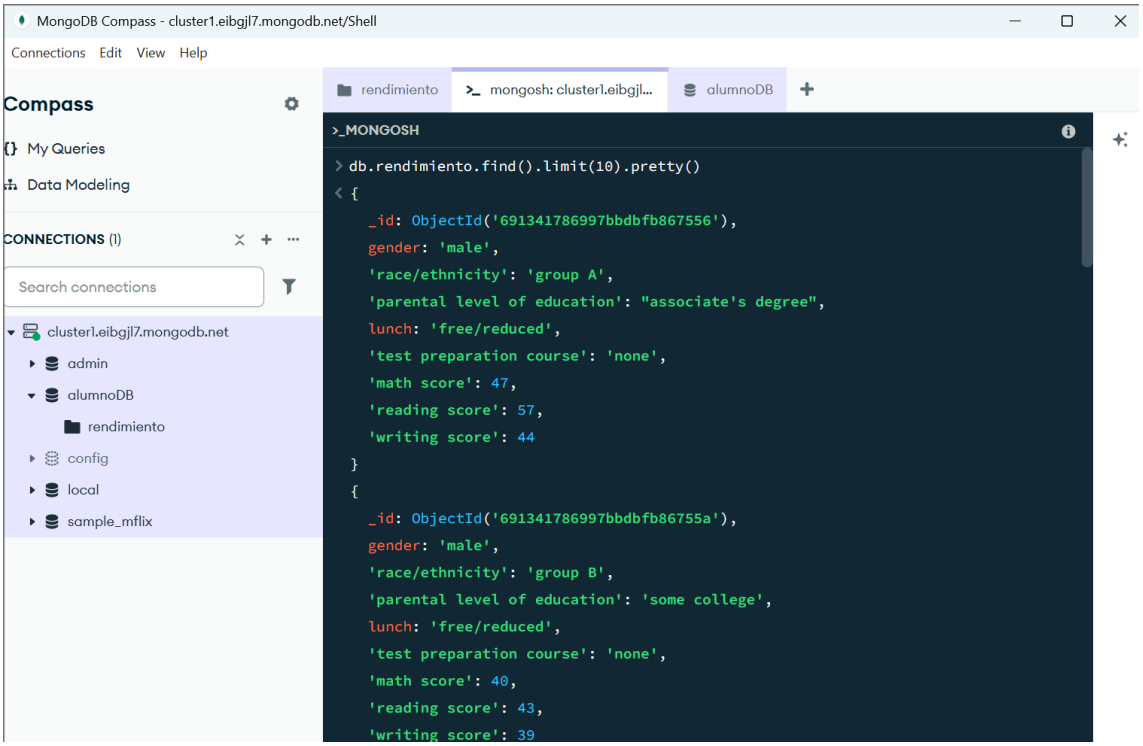


CONSULTA SIMPLE SOBRE UNA COLECCIÓN

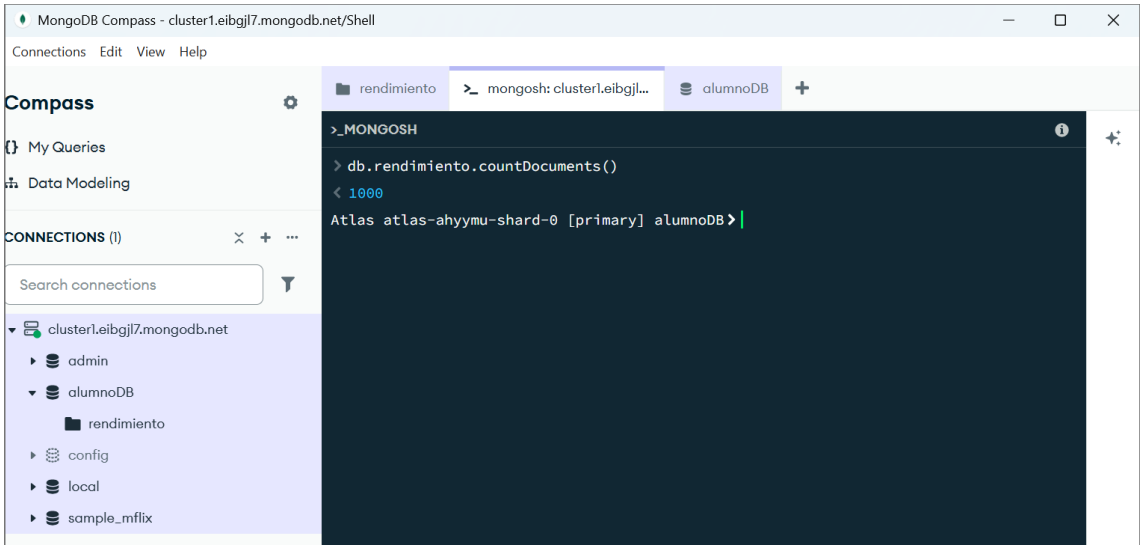
- Mostrar un documento



- Ver los primeros 10 documentos

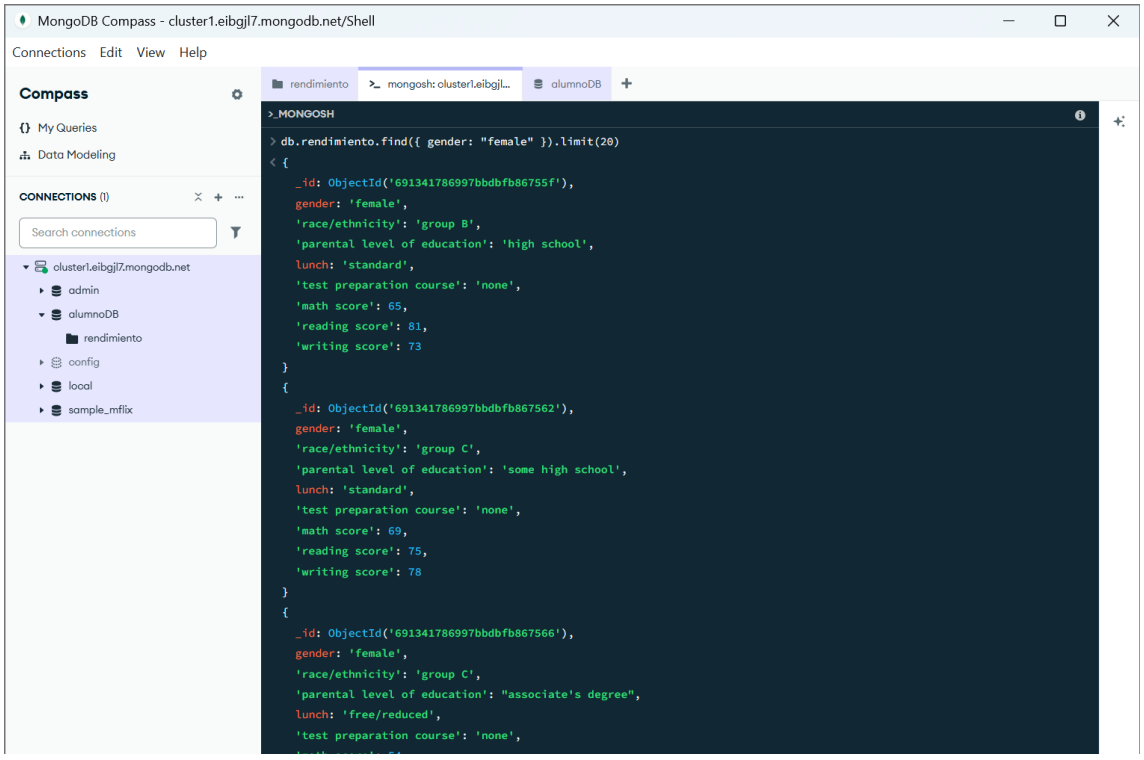


- Contar documentos

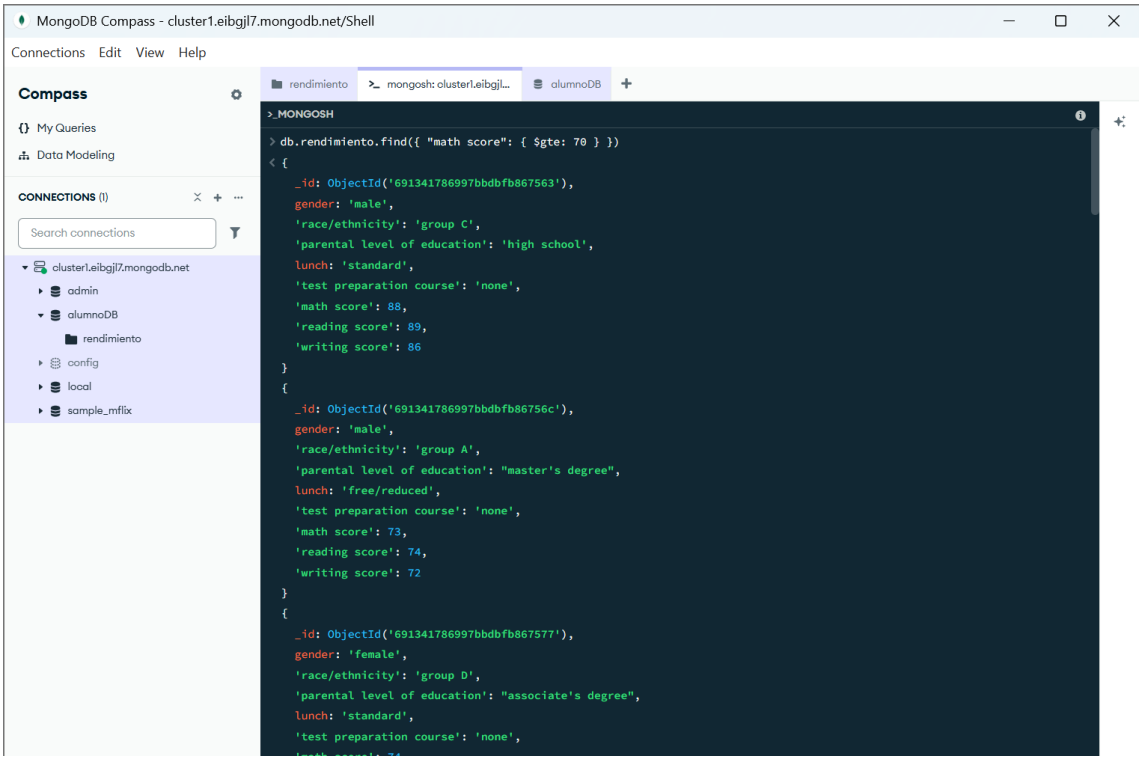


CONSULTA CON FILTROS

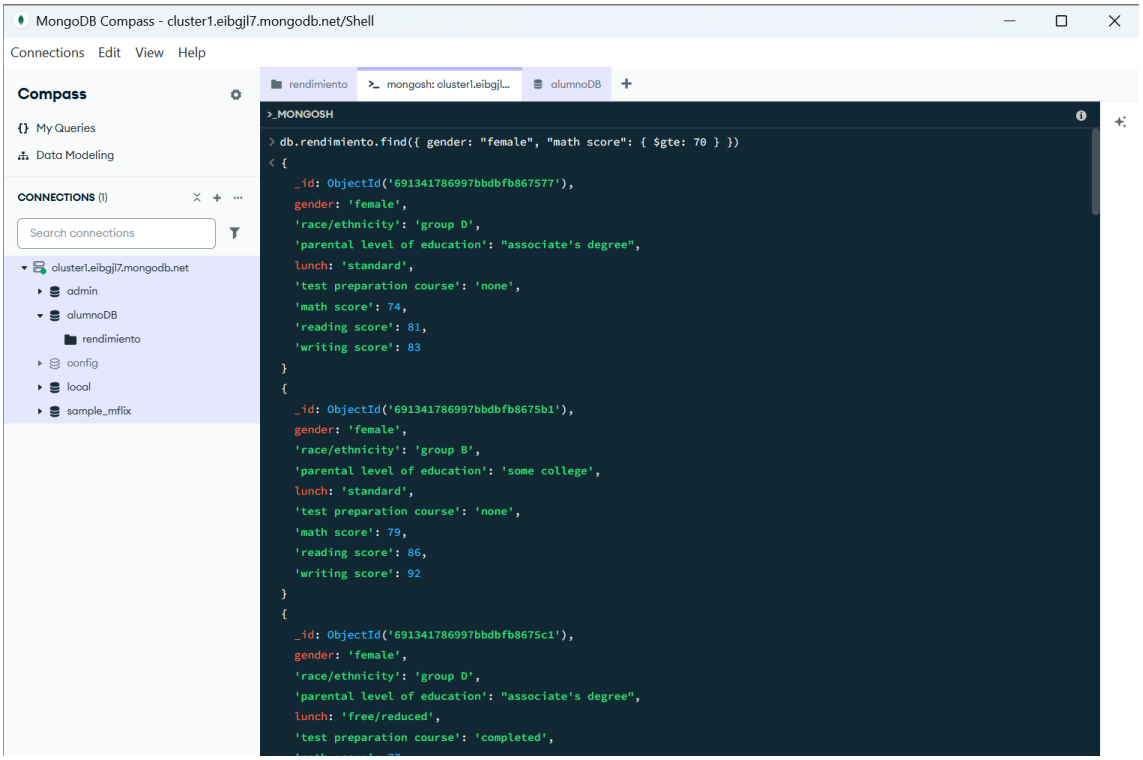
- Todos los estudiantes femeninos



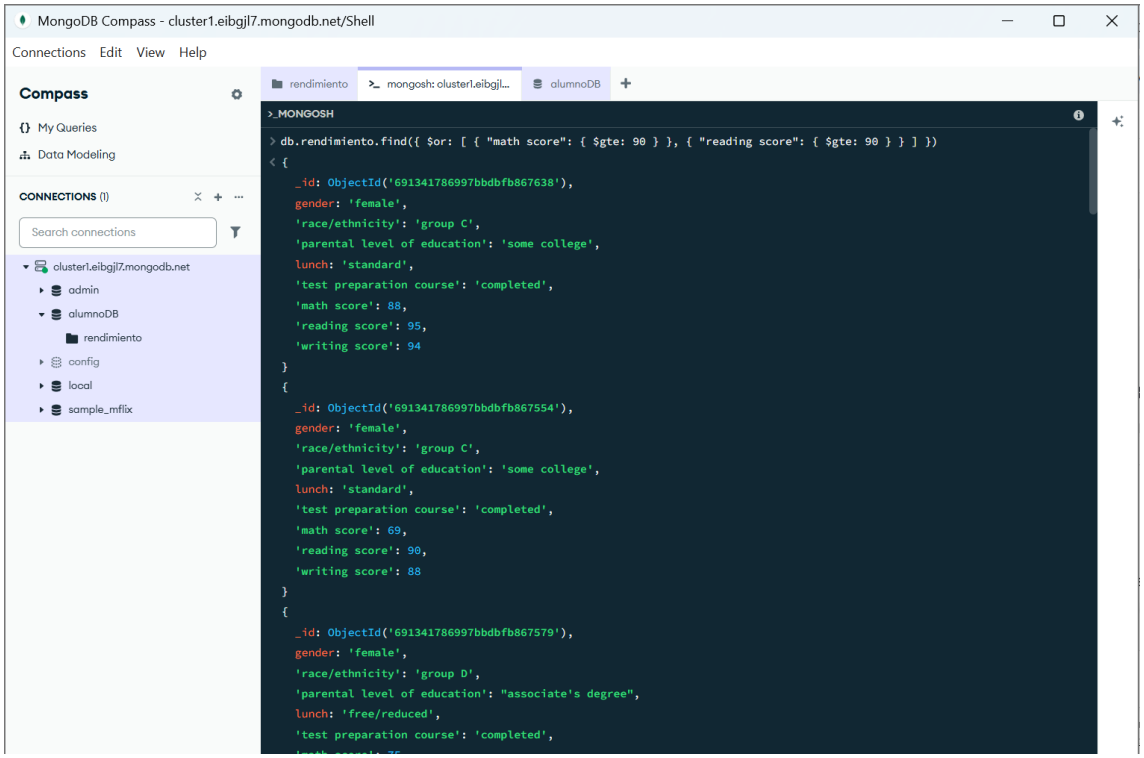
- Estudiantes con math score >= 70



- Varios filtros: género y math

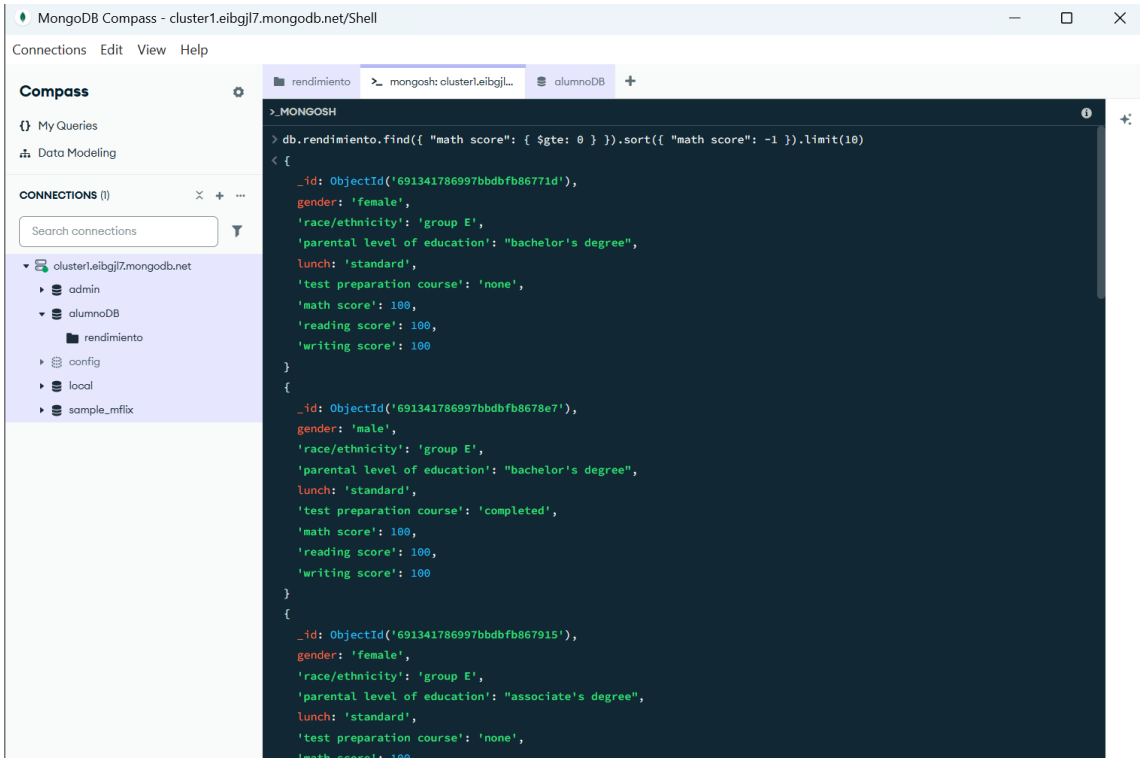


- Filtro con or

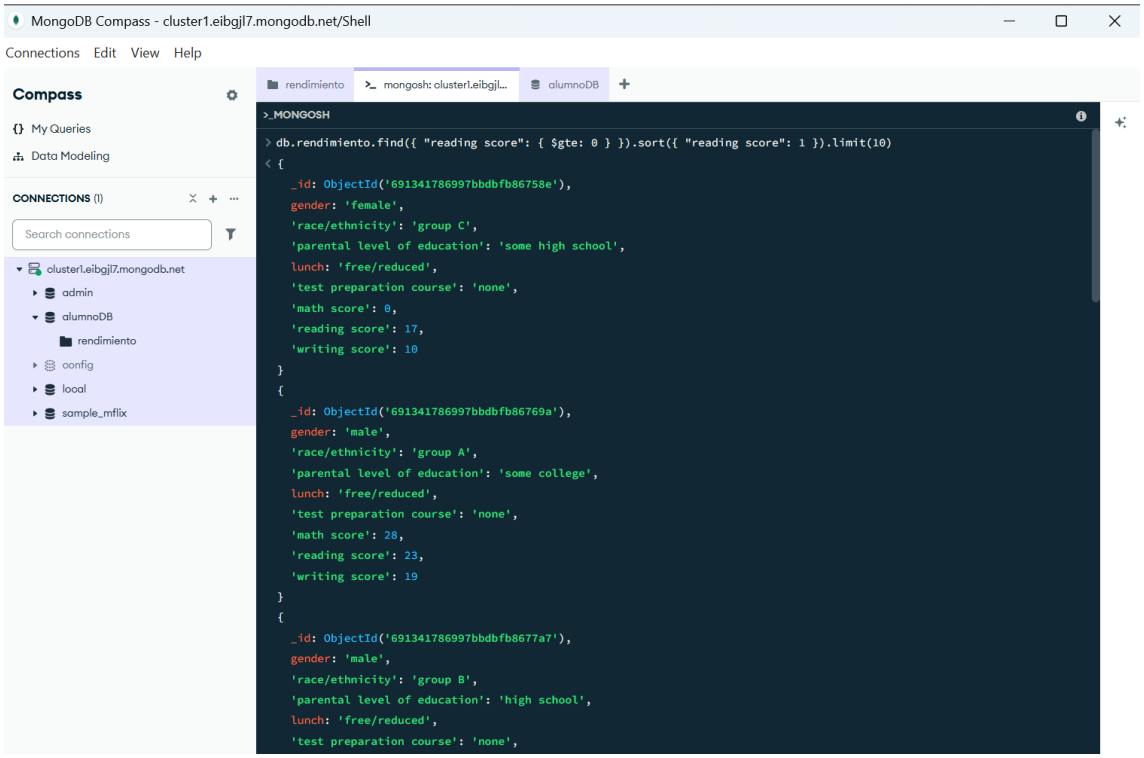


CONSULTA CON FILTROS Y ORDENACIÓN

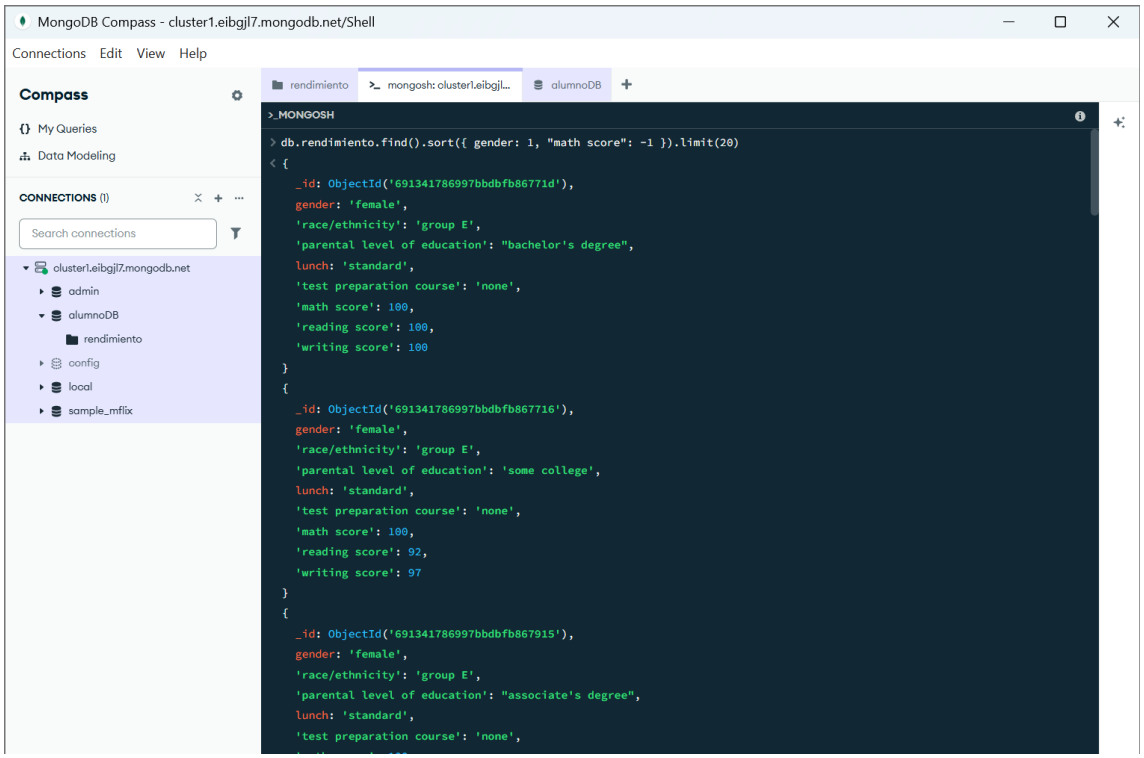
- Orden descendente por math score



- Orden ascendente por reading score

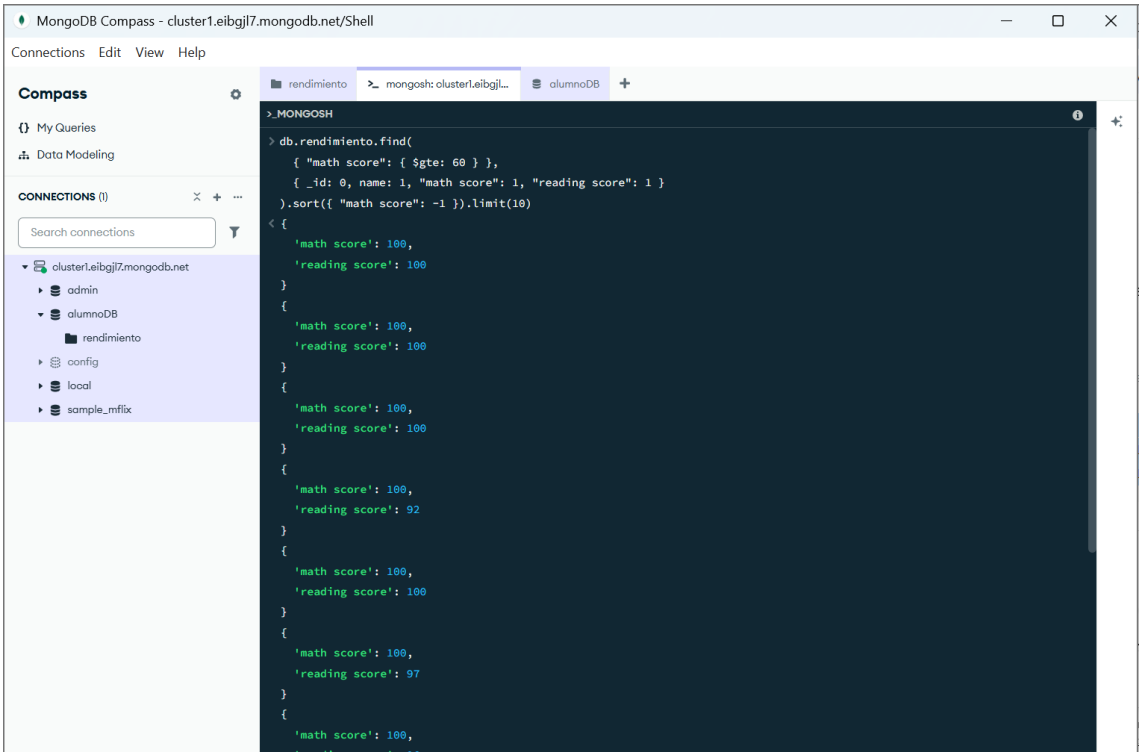


- Orden por varias claves



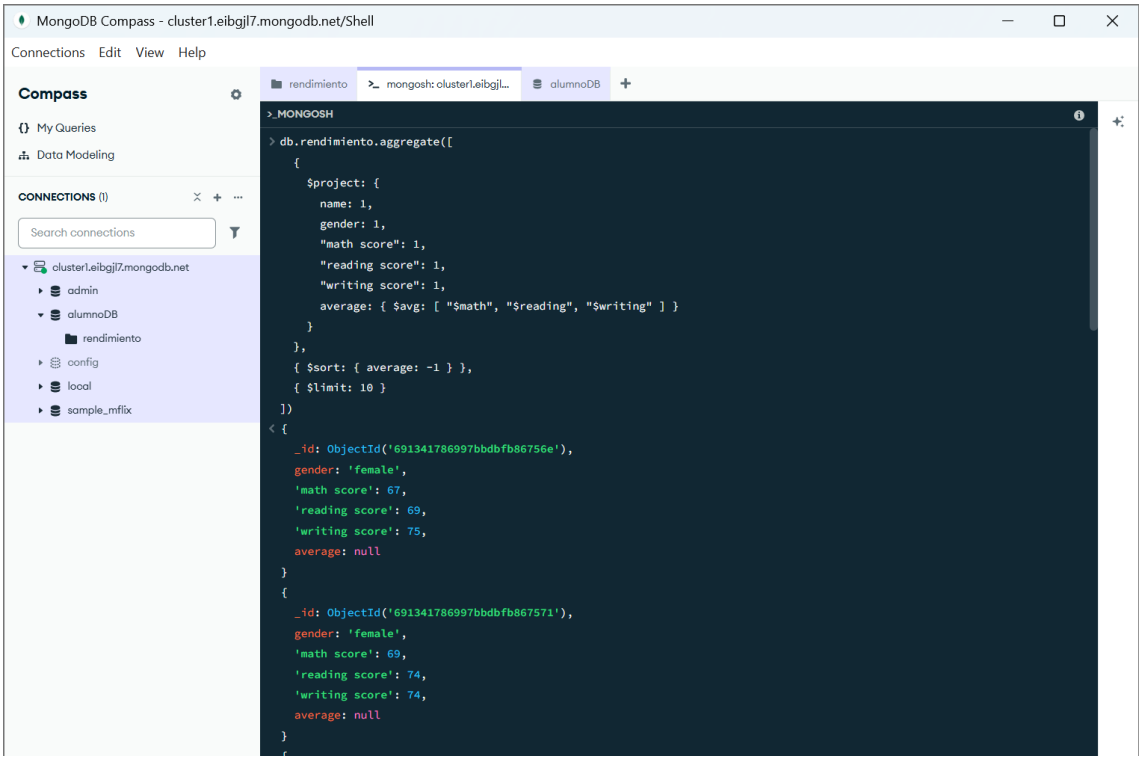
CONSULTA CON FILTROS, ORDENACIÓN Y PROYECCIÓN

- Mostrar sólo campos relevantes sin id

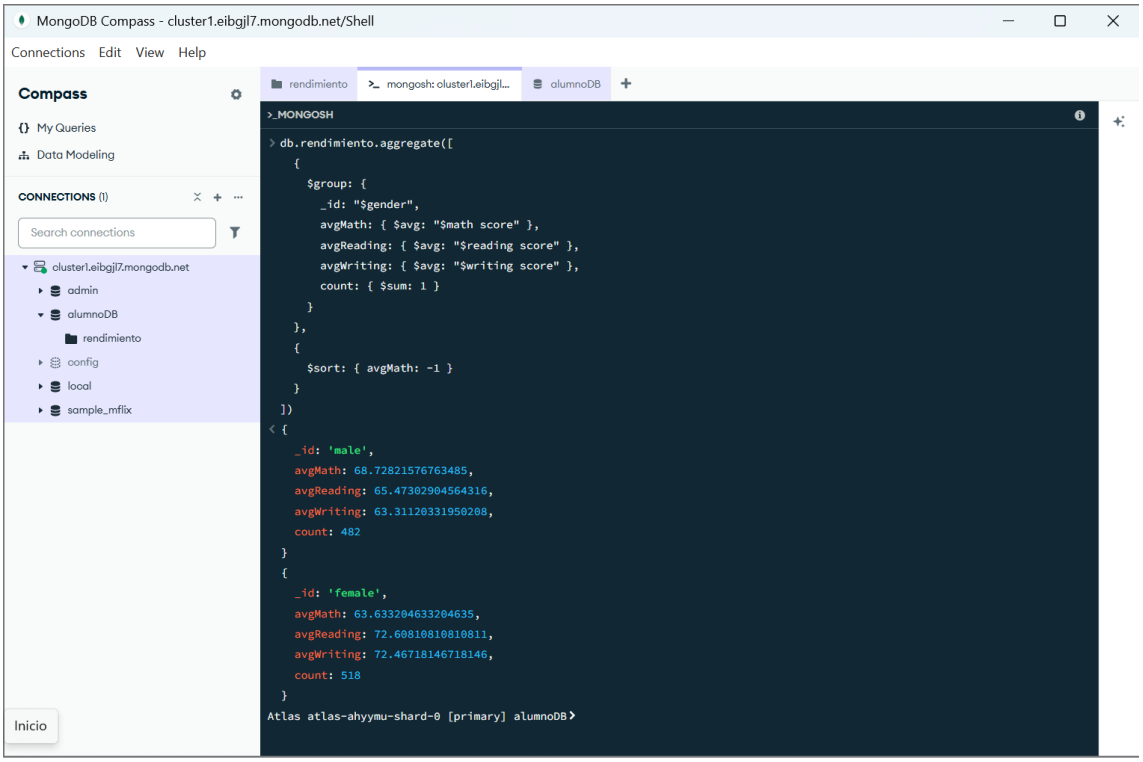


INVESTIGACIONES Y CONSULTAS DE AGREGACIÓN

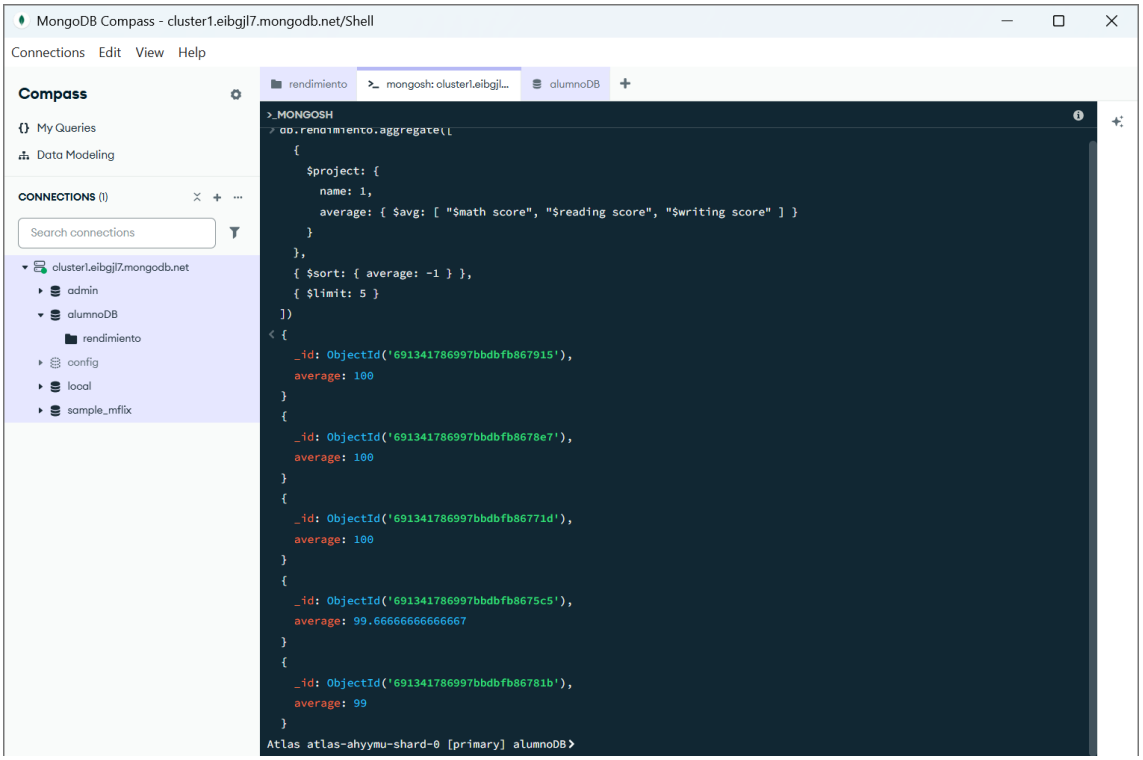
- Cálculo del promedio por estudiante



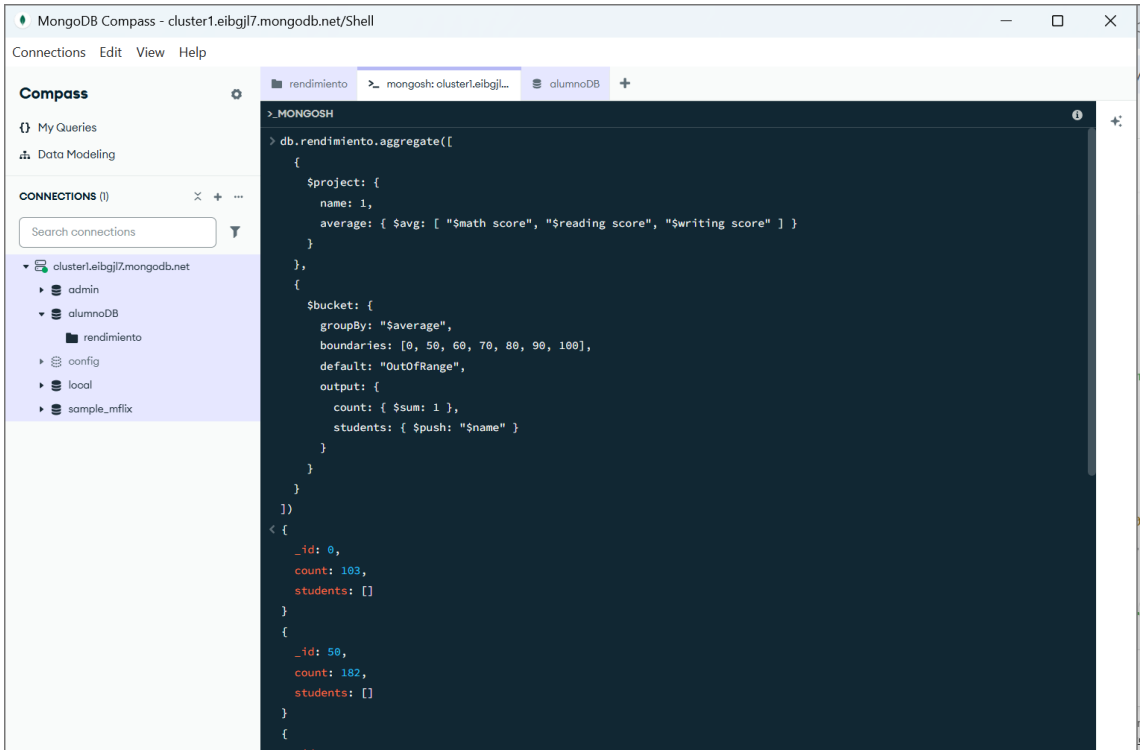
- Media de puntuaciones por género



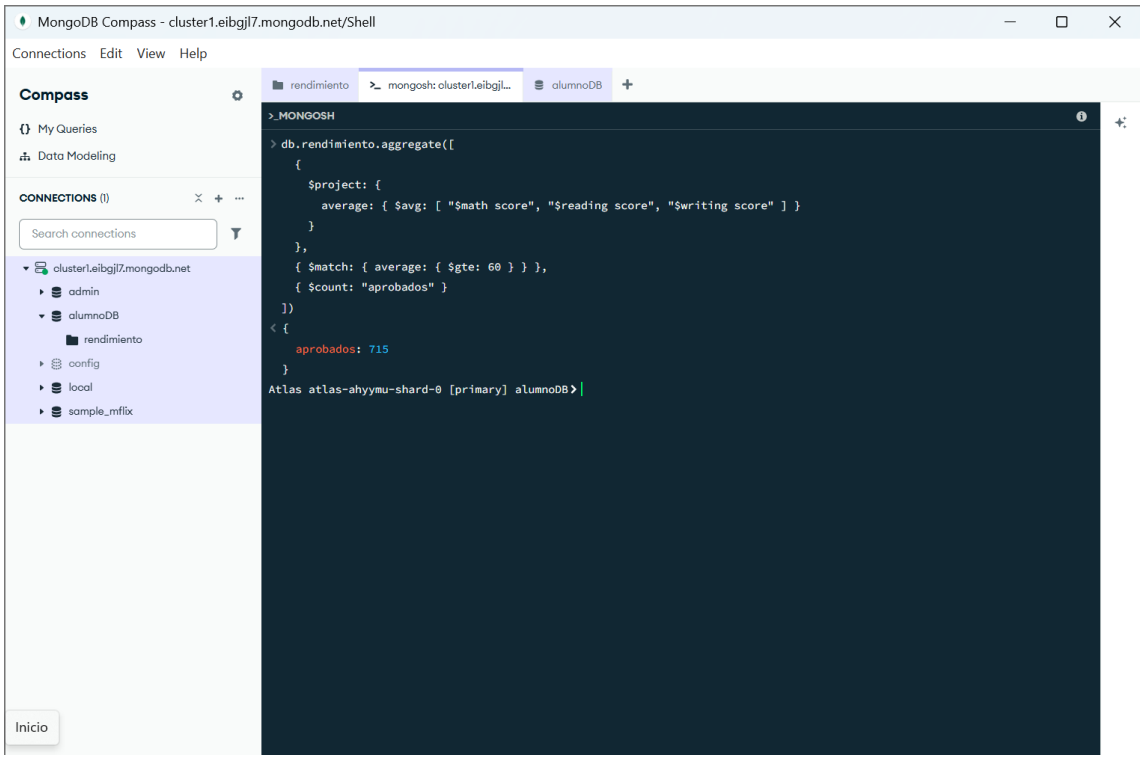
- Top 5 alumnos por promedio



- Distribución por rangos de nota media

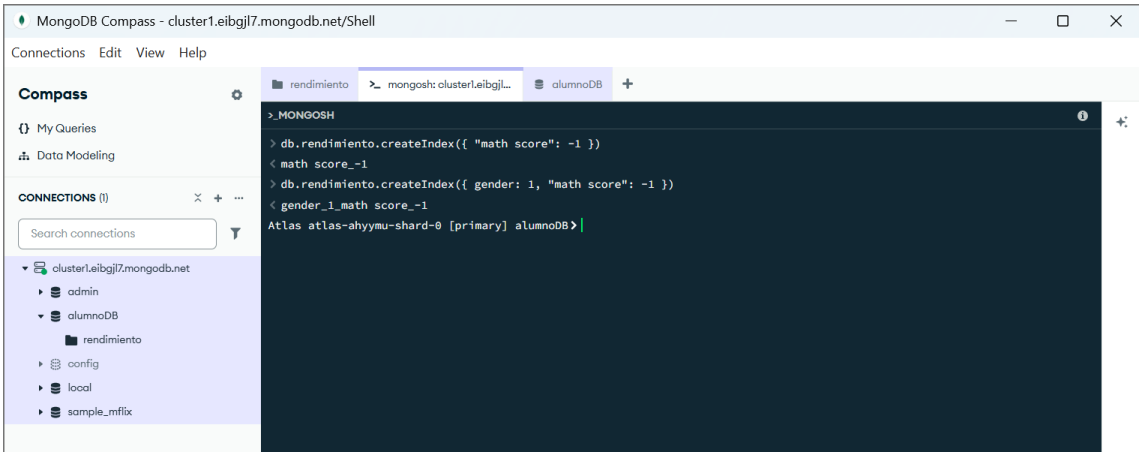


- Contar cuántos aprueban

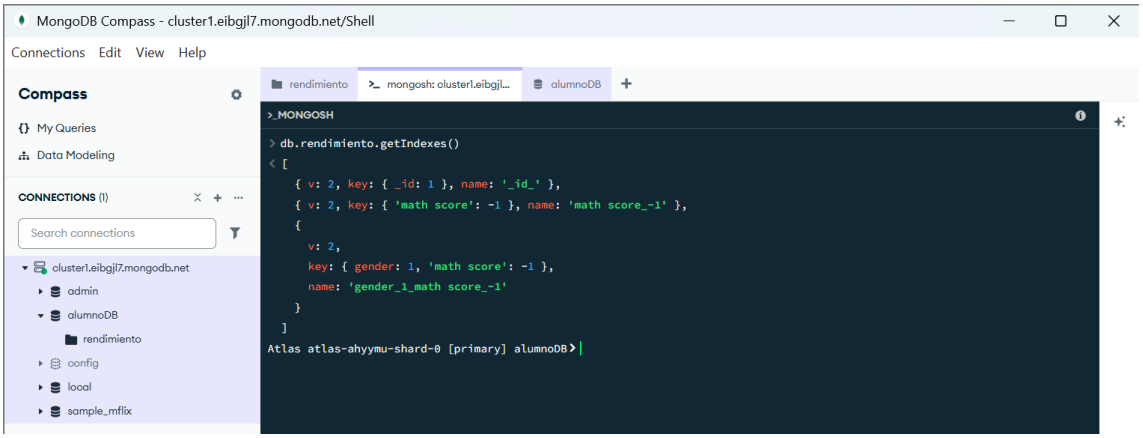


ÍNDICES PARA OPTIMIZACIÓN

- Crear índices para mejorar el rendimiento de las consultas

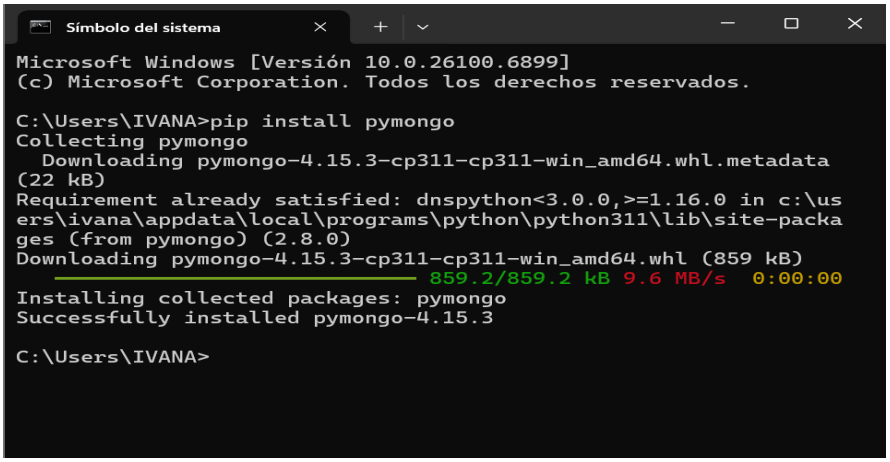


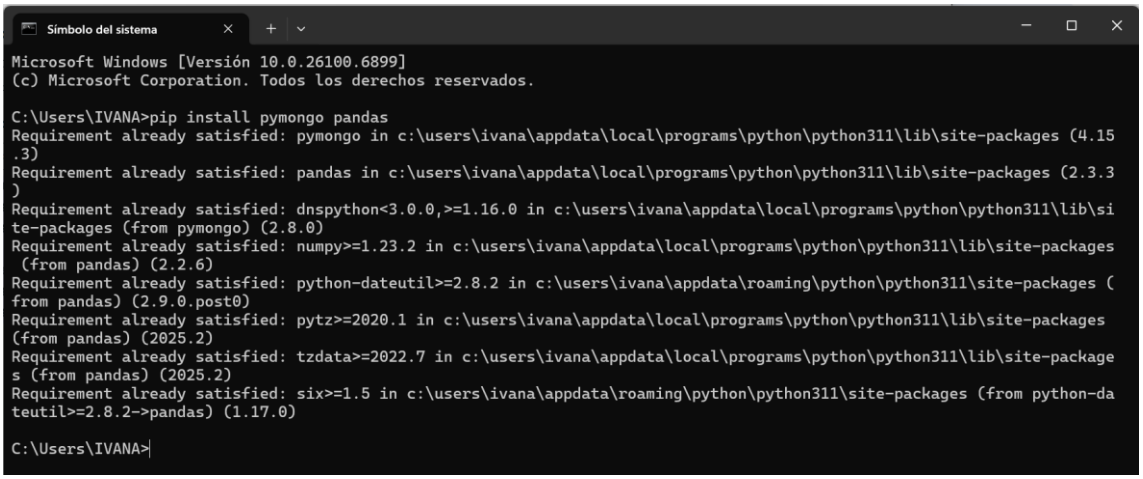
- Comprobar índices



INTEGRACIÓN CON PYTHON (PYMONGO + PANDAS)

Para scripts y análisis en Python instalamos la librería pymongo





CÓDIGO Y RESULTADOS DEL SCRIPT DE PYTHON

```
# Título: Ejemplos de MongoDB
# Autor: Ivana Sánchez Pérez

# colores
azul = '\033[94m'
rosa = '\033[38;5;200m'
turquesa = '\033[38;5;44m'
reset = '\033[0m'

print(f"\n{rosa}===== Ejemplos de MongoDB ====={reset}")
print()

from pymongo import MongoClient
import pandas as pd

# Conexión
uri = "mongodb+srv://isanper504a:Peluchingordi111%40@cluster1.eibgjl7.mongodb.net/alumnoDB"
client = MongoClient(uri)
db = client["alumnoDB"]
col = db["rendimiento"]

# 1) Ver algunos documentos (sin mostrar el _id)
docs = col.find({}, {"_id": 0}).limit(5)
df = pd.DataFrame(list(docs))
print(f"{azul}Primeros registros:{reset}")
print(f"{turquesa}{df}{reset}")

# 2) Agregación: promedio por género
pipeline = [
  {
    "$group": {
      "_id": "$gender",
      "avgMath": {"$avg": "$math score"},
      "avgReading": {"$avg": "$reading score"},
```

```
        "avgWriting": {"$avg": "$writing score"},
        "count": {"$sum": 1}
      }
    },
    {"$sort": {"avgMath": -1}}
]
result = list(col.aggregate(pipeline))
print(f"{azul}\nPromedio de puntuaciones por género:{reset}")
for r in result:
    print(f"{turquesa}{r}{reset}")

# 3) Consulta: top 10 por math
top_math = list(col.find({}, {"_id":0, "name":1, "math score":1}).sort("mathscore", -1).limit(10))
print(f"\n{azul}Top math:{reset}", top_math)

# 4)Consulta con filtro y ordenación
top_math = list(col.find({}, {"_id": 0, "gender": 1, "math score": 1})
            .sort("math score", -1)
            .limit(5))
print(f"\n{azul}Top 5 alumnos con mejor nota en matemáticas: {reset}")
print(pd.DataFrame(top_math))

# 5) Estudiantes con math score >= 80
filtro = {"math score": {"$gte": 80}}
resultados = list(col.find(filtro, {"_id": 0, "gender": 1, "math score": 1}))
print(f"{azul}\nEstudiantes con math score >= 80:{reset}")
print(pd.DataFrame(resultados))

pipeline = [
    {
      "$group": {
        "_id": "$test preparation course",
        "avgMath": {"$avg": "$math score"},
        "avgReading": {"$avg": "$reading score"},
        "avgWriting": {"$avg": "$writing score"},
        "total": {"$sum": 1}
      }
    },
    {"$sort": {"avgMath": -1}}
]

result = list(col.aggregate(pipeline))
print(f"\n{azul}Promedio según haber hecho curso de preparación o no:{reset}")
for r in result:
    print(f"{turquesa}{r}{reset}")
```

RESULTADO DE LA EJECUCIÓN