BAB VI
INTEGRASI SPARK DAN MONGODB DALAM PENGELOLAAN DATA BESAR

6.1. Tujuan Pembelajaran

Setelah mengikuti praktikum yang dijelaskan didalam modul praktikum ini, mahasiswa diharapkan mampu:

1. Memahami konsep integrasi PySpark dengan MongoDB dalam pengelolaan data besar.
2. Mengkonfigurasi koneksi Spark dengan MongoDB menggunakan connector.
3. Melakukan operasi *read* dan *write* data antara MongoDB dan Spark *DataFrame*.
4. Melakukan transformasi dan analisis sederhana pada data besar yang tersimpan di MongoDB.

6.2. Pendahuluan

Perkembangan teknologi informasi yang pesat telah mendorong munculnya fenomena Big Data, yaitu data dengan *volume* besar, kecepatan tinggi, dan beragam format yang sulit dikelola dengan sistem basis data tradisional. Dalam menghadapi tantangan ini, dibutuhkan teknologi yang mampu melakukan pengolahan data secara cepat, efisien, serta mendukung penyimpanan terdistribusi. Dua di antara teknologi yang banyak digunakan adalah Apache Spark dan MongoDB.

Apache Spark merupakan kerangka kerja komputasi terdistribusi yang dirancang untuk memproses data dalam skala besar secara paralel di berbagai node. Spark menyediakan beragam API untuk analisis data, mulai dari pemrosesan *batch*, pemrosesan *real-time*, hingga pembelajaran mesin. Keunggulan Spark terletak pada kemampuannya mengelola data dalam memori, sehingga meningkatkan performa komputasi dibandingkan sistem tradisional berbasis disk.

Di sisi lain, MongoDB adalah sistem basis data NoSQL berbasis dokumen yang fleksibel dan mampu menyimpan data tidak terstruktur maupun semi-terstruktur. MongoDB banyak digunakan dalam skenario Big Data karena mendukung skema yang dinamis, skalabilitas horizontal, serta integrasi yang baik dengan berbagai platform analitik.

Integrasi antara Spark dan MongoDB menjadi solusi yang kuat dalam pengelolaan data besar. MongoDB dapat digunakan sebagai penyimpanan utama maupun sumber data, sedangkan Spark bertugas mengolah dan menganalisis data tersebut secara paralel. Melalui

integrasi ini, pengguna dapat memanfaatkan kecepatan pemrosesan Spark dan fleksibilitas MongoDB untuk mendukung proses analisis data yang kompleks.

6.3. Persiapan Lingkungan

Sebelum melakukan Praktikum mengenai Integrasi Spark dan MongoDB dalam pengelolaan data besar, pastikan perangkat sudah memenuhi syarat minimum berikut:

1. **Perangkat Keras**
   - Sistem Operasi Windows 10/11
   - Laptop/PC dengan minimal RAM 8 GB (lebih disarankan 16 GB untuk simulasi dataset besar).
   - Prosesor multi-core (disarankan ≥ 4 core).
   - Penyimpanan 20 GB kosong untuk instalasi Spark, MongoDB dan dataset.
   - Koneksi internet untuk mengunduh dataset dan library.
2. **Perangkat Lunak**
   - JDK 11 (Rekomendasi)
   - Python 3.11 atau lebih baru.
   - Apache Spark 3.x (3.5.6 atau lebih)
   - MongoDB Compass 8.0.13
   - IDE/Editor: Jupyter Notebook, VSCode, atau PyCharm.
   - Instal Library

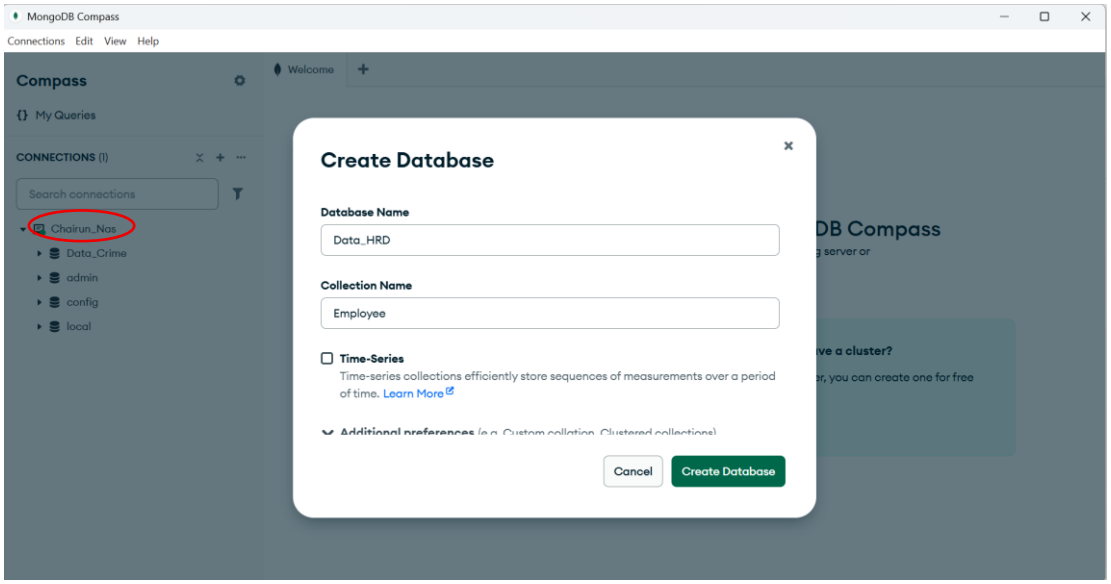
     ```
     pip install pymongo
     ```
3. **Dataset**
   - Dataset yaitu HR_DATA_MNC (ukuran > 200MB) : https://s.id/dataset_HR_MNC

6.4. Langkah Praktikum

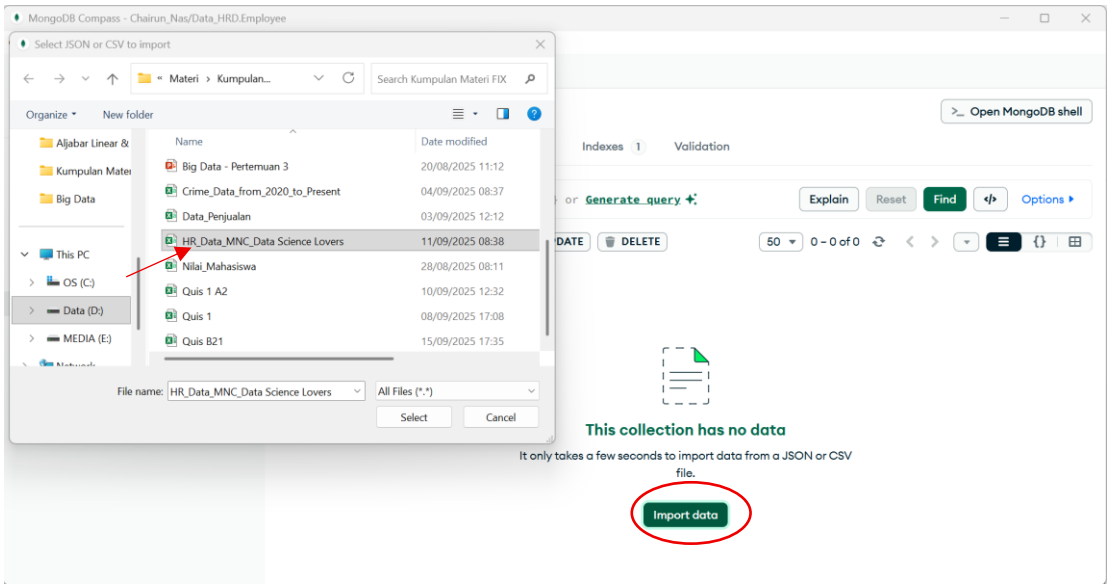
a. *Database* dan *Collection* pada MongoDB

Dalam MongoDB, *Database* adalah wadah utama untuk menyimpan data, yang dapat berisi satu atau lebih *Collection* sebagai unit penyimpanan di dalamnya. Setiap *database* memiliki ruang lingkup tersendiri dan dipisahkan dari *database* lain, sehingga memungkinkan pengelolaan data yang lebih terorganisir. Sementara itu, *Collection* dapat diibaratkan seperti tabel dalam basis data relasional, tetapi lebih fleksibel karena berisi kumpulan dokumen dalam format BSON (*Binary JSON*) yang tidak harus memiliki struktur atau skema yang sama. Dengan demikian, *database*

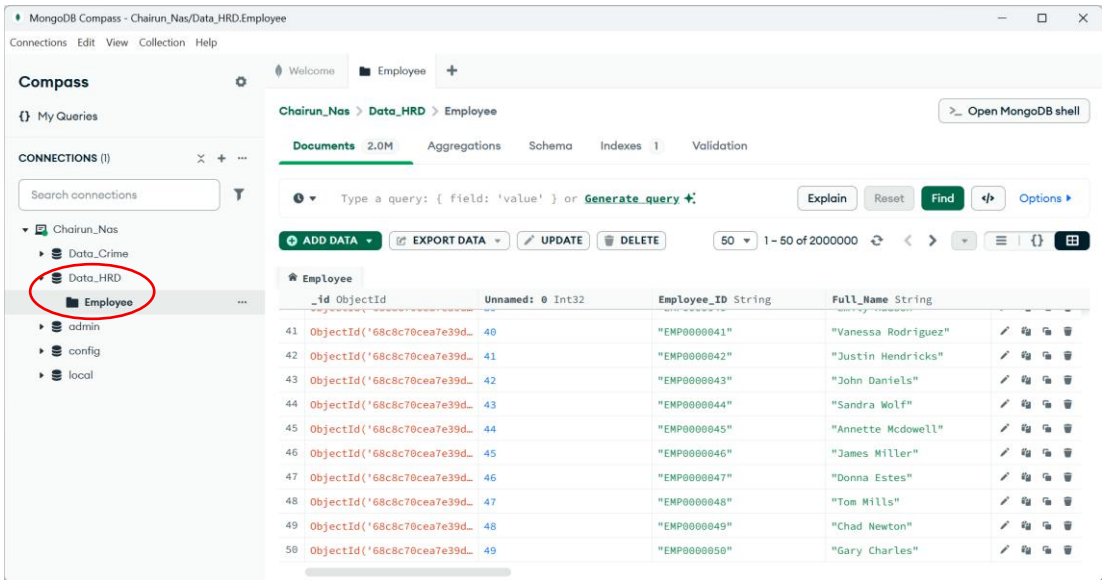
berfungsi sebagai level tertinggi untuk pengelompokan data, sedangkan *collection* menjadi tempat utama menyimpan dan mengelola dokumen yang saling berkaitan. Adapun untuk membuat *Database* dan *Collection*, maka klik symbol (+) pada nama Connection anda. Buat nama database "**Data_HRD**" dan nama Collection "**Employee**".



Selanjutnya lakukan *import* dataset kedalam database dengan dengan cara mengklik *Import Data* dan pilih dataset yang di inginkan.



Maka selanjutnya dataset akan masuk kedalam database "Data_HRD" dan Collection "Employee".



b. Integrasi Spark dengan MongoDB

Menghubungkan Apache Spark dengan MongoDB memungkinkan integrasi antara kemampuan analitik big data Spark dan fleksibilitas penyimpanan dokumen MongoDB. Koneksi ini biasanya dilakukan menggunakan MongoDB Spark Connector, yang berfungsi sebagai jembatan untuk membaca dan menulis data secara langsung antara Spark DataFrame/RDD dengan koleksi MongoDB. Adapun MongoDB Spark Connector pada pyspark dapat dilihat sebagai berikut :

```
from pymongo import MongoClient

# Koneksi ke MongoDB lokal
client = MongoClient("mongodb://localhost:27017/")

# Pilih database
db = client["Data_HRD"]

# Pilih collection
collection = db["Employee"]

print("MongoDB Terkoneksi")
```

Selanjutnya untuk membaca data dari mongodb melalui spark, maka dapat menggunakan perintah-perintah dasar MongoDB. Adapun beberapa perintah yang dapat dilakukan dapat dilihat sebagai berikut :

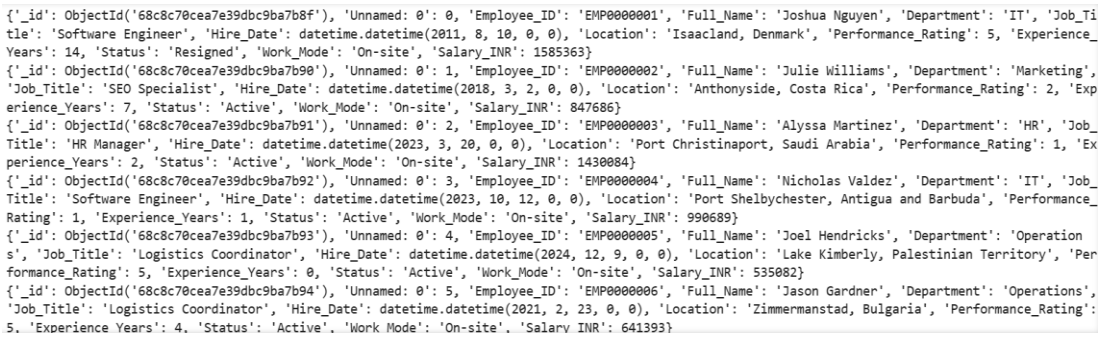
- Mengambil Data (Fungsi **find**)

```
# Mengambil satu data
emp = collection.find_one({"Employee_ID": "EMP0000002"})
print("Data Karyawan:", emp)
```

```
Data Karyawan: {'_id': ObjectId('68c8c70cea7e39dbc9ba7b90'), 'Unnamed: 0': 1, 'Employee_ID': 'EMP0000002', 'Full_Name': 'Julie Williams', 'Department': 'Marketing', 'Job_Title': 'SEO Specialist', 'Hire_Date': datetime.datetime(2018, 3, 2, 0, 0), 'Location': 'Anthonyside, Costa Rica', 'Performance_Rating': 2, 'Experience_Years': 7, 'Status': 'Active', 'Work_Mode': 'On-site', 'Salary_INR': 847686}
```

```
# Ambil semua data
for emp in collection.find():
    print(emp)
```

Menghasilkan output :



```
{'_id': ObjectId('68c8c70cea7e39dbc9ba7b8f'), 'Unnamed: 0': 0, 'Employee_ID': 'EMP0000001', 'Full_Name': 'Joshua Nguyen', 'Department': 'IT', 'Job_Title': 'Software Engineer', 'Hire_Date': datetime.datetime(2011, 8, 10, 0, 0), 'Location': 'Isaacland, Denmark', 'Performance_Rating': 5, 'Experience_Years': 14, 'Status': 'Resigned', 'Work_Mode': 'On-site', 'Salary_INR': 1585363}
{'_id': ObjectId('68c8c70cea7e39dbc9ba7b90'), 'Unnamed: 0': 1, 'Employee_ID': 'EMP0000002', 'Full_Name': 'Julie Williams', 'Department': 'Marketing', 'Job_Title': 'SEO Specialist', 'Hire_Date': datetime.datetime(2018, 3, 2, 0, 0), 'Location': 'Anthonyside, Costa Rica', 'Performance_Rating': 2, 'Experience_Years': 7, 'Status': 'Active', 'Work_Mode': 'On-site', 'Salary_INR': 847686}
{'_id': ObjectId('68c8c70cea7e39dbc9ba7b91'), 'Unnamed: 0': 2, 'Employee_ID': 'EMP0000003', 'Full_Name': 'Alyssa Martinez', 'Department': 'HR', 'Job_Title': 'HR Manager', 'Hire_Date': datetime.datetime(2023, 3, 20, 0, 0), 'Location': 'Port Christinaport, Saudi Arabia', 'Performance_Rating': 1, 'Experience_Years': 2, 'Status': 'Active', 'Work_Mode': 'On-site', 'Salary_INR': 1430084}
{'_id': ObjectId('68c8c70cea7e39dbc9ba7b92'), 'Unnamed: 0': 3, 'Employee_ID': 'EMP0000004', 'Full_Name': 'Nicholas Valdez', 'Department': 'IT', 'Job_Title': 'Software Engineer', 'Hire_Date': datetime.datetime(2023, 10, 12, 0, 0), 'Location': 'Port Shelbychester, Antigua and Barbuda', 'Performance_Rating': 1, 'Experience_Years': 1, 'Status': 'Active', 'Work_Mode': 'On-site', 'Salary_INR': 990689}
{'_id': ObjectId('68c8c70cea7e39dbc9ba7b93'), 'Unnamed: 0': 4, 'Employee_ID': 'EMP0000005', 'Full_Name': 'Joel Hendricks', 'Department': 'Operations', 'Job_Title': 'Logistics Coordinator', 'Hire_Date': datetime.datetime(2024, 12, 9, 0, 0), 'Location': 'Lake Kimberly, Palestinian Territory', 'Performance_Rating': 5, 'Experience_Years': 0, 'Status': 'Active', 'Work_Mode': 'On-site', 'Salary_INR': 535082}
{'_id': ObjectId('68c8c70cea7e39dbc9ba7b94'), 'Unnamed: 0': 5, 'Employee_ID': 'EMP0000006', 'Full_Name': 'Jason Gardner', 'Department': 'Operations', 'Job_Title': 'Logistics Coordinator', 'Hire_Date': datetime.datetime(2021, 2, 23, 0, 0), 'Location': 'Zimmermanstad, Bulgaria', 'Performance_Rating': 5, 'Experience_Years': 4, 'Status': 'Active', 'Work_Mode': 'On-site', 'Salary_INR': 641393}
```

```
# Ambil Data dengan filter
for emp in collection.find({"Department": "IT"}):
    print("Karyawan dengan Departemen IT :", emp)
```

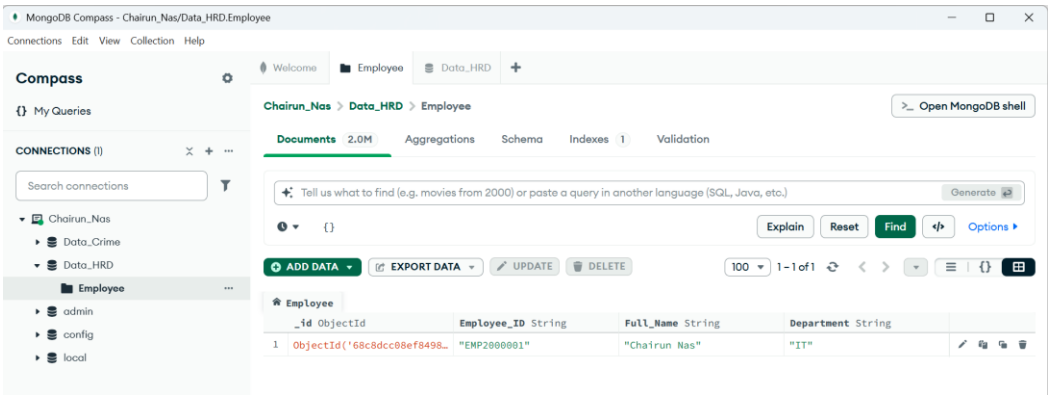
Menghasilkan output :

```
Karyawan dengan Departemen IT : {'_id': ObjectId('68c8c70cea7e39dbc9ba7b8f'), 'Unnamed: 0': 0, 'Employee_ID': 'EMP0000001', 'Full_Name': 'Joshua Nguyen', 'Department': 'IT', 'Job_Title': 'Software Engineer', 'Hire_Date': datetime.datetime(2011, 8, 10, 0, 0), 'Location': 'Isaacland, Denmark', 'Performance_Rating': 5, 'Experience_Years': 14, 'Status': 'Resigned', 'Work_Mode': 'On-site', 'Salary_INR': 1585363}
Karyawan dengan Departemen IT : {'_id': ObjectId('68c8c70cea7e39dbc9ba7b92'), 'Unnamed: 0': 3, 'Employee_ID': 'EMP0000004', 'Full_Name': 'Nicholas Valdez', 'Department': 'IT', 'Job_Title': 'Software Engineer', 'Hire_Date': datetime.datetime(2023, 10, 12, 0, 0), 'Location': 'Port Shelbychester, Antigua and Barbuda', 'Performance_Rating': 1, 'Experience_Years': 1, 'Status': 'Active', 'Work_Mode': 'On-site', 'Salary_INR': 990689}
Karyawan dengan Departemen IT : {'_id': ObjectId('68c8c70cea7e39dbc9ba7b98'), 'Unnamed: 0': 9, 'Employee_ID': 'EMP0000010', 'Full_Name': 'Maria Yu MD', 'Department': 'IT', 'Job_Title': 'Software Engineer', 'Hire_Date': datetime.datetime(2015, 10, 8, 0, 0), 'Location': 'Brownport, Yemen', 'Performance_Rating': 4, 'Experience_Years': 9, 'Status': 'Active', 'Work_Mode': 'Remote', 'Salary_INR': 1543102}
Karyawan dengan Departemen IT : {'_id': ObjectId('68c8c70cea7e39dbc9ba7ba1'), 'Unnamed: 0': 18, 'Employee_ID': 'EMP0000019', 'Full_Name': 'Samuel Ferguson', 'Department': 'IT', 'Job_Title': 'DevOps Engineer', 'Hire_Date': datetime.datetime(2025, 4, 5, 0, 0), 'Location': 'Matthewsville, Ghana', 'Performance_Rating': 2, 'Experience_Years': 0, 'Status': 'Active', 'Work_Mode': 'On-site', 'Salary_INR': 631273}
Karyawan dengan Departemen IT : {'_id': ObjectId('68c8c70cea7e39dbc9ba7ba4'), 'Unnamed: 0': 21, 'Employee_ID': 'EMP0000022', 'Full_Name': 'Kylie May', 'Department': 'IT', 'Job_Title': 'DevOps Engineer', 'Hire_Date': datetime.datetime(2015, 7, 7, 0, 0), 'Location': 'Houstonland, Qatar', 'Performance_Rating': 2, 'Experience_Years': 10, 'Status': 'Active', 'Work_Mode': 'On-site', 'Salary_INR': 723820}
Karyawan dengan Departemen IT : {'_id': ObjectId('68c8c70cea7e39dbc9ba7ba7'), 'Unnamed: 0': 24, 'Employee_ID': 'EMP0000025', 'Full_Name': 'Amanda Miller', 'Department': 'IT', 'Job_Title': 'IT Manager', 'Hire_Date': datetime.datetime(2021, 7, 31, 0, 0), 'Location': 'Tammyview, Barbados', 'Performance_Rating': 2, 'Experience_Years': 4, 'Status': 'Active', 'Work_Mode': 'Remote', 'Salary_INR': 2153098}
```

- Mengisi Data (Fungsi **Insert**)

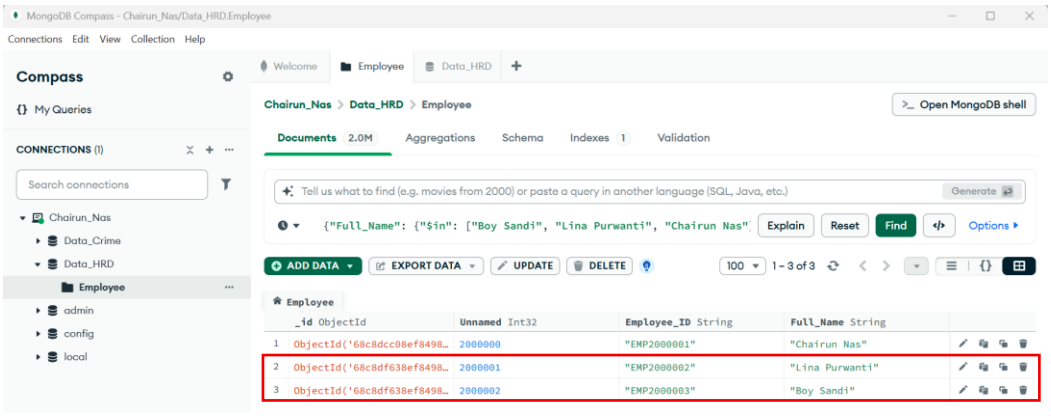
```
employee = {
    "Unnamed": 2000000,
    "Employee_ID": "EMP2000001",
    "Full_Name": "Chairun Nas",
    "Department": "IT",
    "Job_Title": "IT Manager",
    "Hire_Date": datetime(2022, 10, 8, 0, 0, 0),
    "Location": "Brownport, Yemen",
    "Performance_Rating": 4,
    "Experience_Years": 9,
    "Status": "Active",
    "Work_Mode": "Remote",
    "Salary_INR": 1552102
}
collection.insert_one(employee)
```

Menghasilkan output :




```
# Insert banyak data
from datetime import datetime
employees = [
    {"Unnamed": 2000001, "Employee_ID": "EMP2000002",
"Full_Name": "Lina Purwanti", "Department": "Finance",
"Job_Title": "Finance", "Hire_Date": datetime(2024, 8, 23, 0, 0,
0), "Location": "Jakarta, Indonesia", "Performance_Rating": 4,
"Experience_Years": 3, "Status": "Active", "Work_Mode": "On-
site", "Salary_INR": 1052102},

    {"Unnamed": 2000002, "Employee_ID": "EMP2000003",
"Full_Name": "Boy Sandi", "Department": "Sales", "Job_Title":
"Marketing", "Hire_Date": datetime(2025, 12, 8, 0, 0, 0),
"Location": "Riau, Indonesia", "Performance_Rating": 3,
"Experience_Years": 2, "Status": "Active", "Work_Mode": "On-
site", "Salary_INR": 552102},
]
collection.insert_many(employees)
```

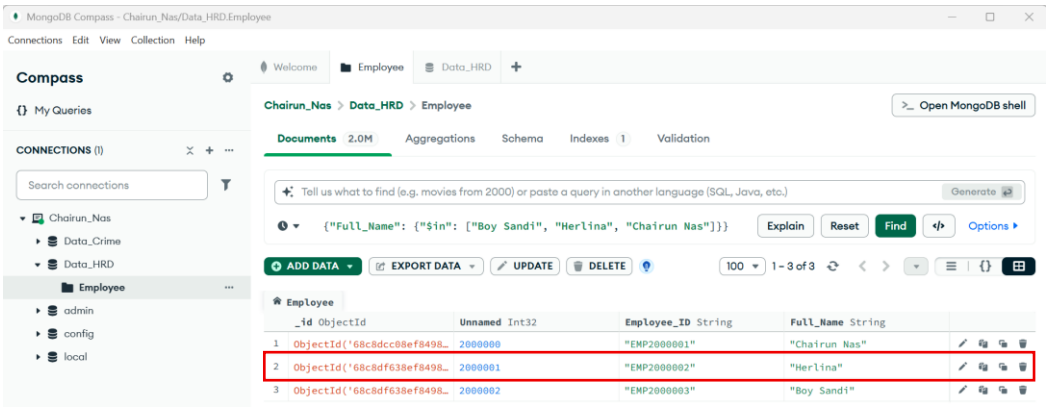
Menghasilkan output :



- Ubah dan Hapus Data (Fungsi Update dan Delete)

```
# Update data
collection.update_one({"Employee_ID": "EMP2000002"}, {"$set":
{"Full_Name": "Herlina"}})
```

Menghasilkan Output




```
# Hapus Data
collection.delete_one({"Employee_ID": "EMP2000002"})
```

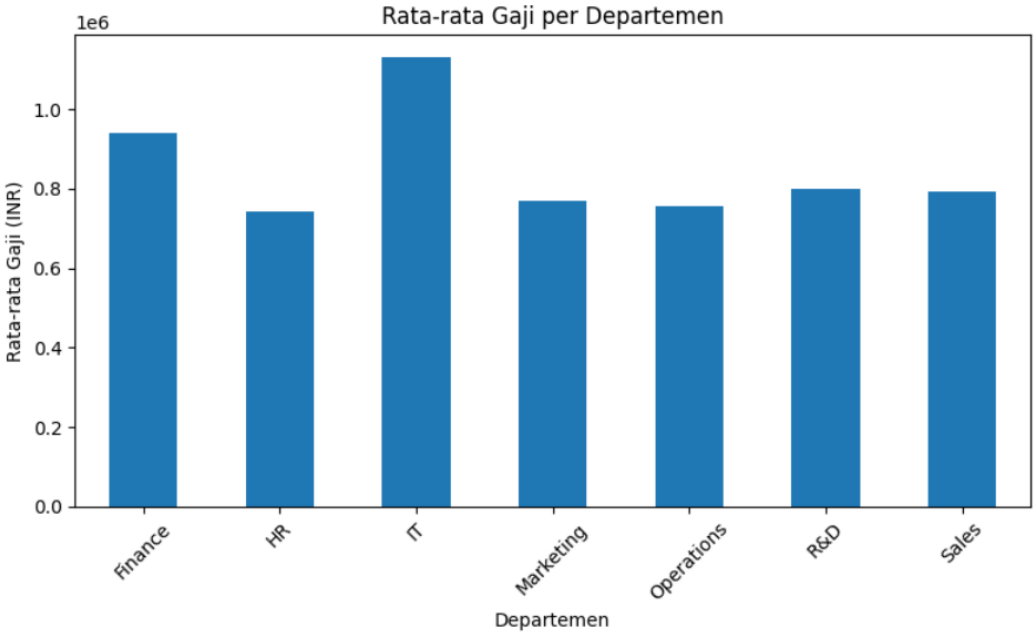
Selanjutnya sebagai contoh beberapa analisis data yang dapat dilakukan dapat dicontohkan sebagai berikut :

- Analisis Data rata-rata gaji karyawan per departemen

```
from pymongo import MongoClient
import pandas as pd
import matplotlib.pyplot as plt

# Koneksi MongoDB
client = MongoClient("mongodb://localhost:27017/")
db = client["Data_HRD"]
collection = db["Employee"]

# Ambil semua data dari MongoDB ke DataFrame
df = pd.DataFrame(list(collection.find()))

# Analisis sederhana: rata-rata gaji per Departemen
rata_gaji =
df.groupby("Department")["Salary_INR"].mean().round()

# Tampilkan hasil dengan format Rupiah
for dept, salary in rata_gaji.items():
    print(f"{dept}: Rp {salary:,.0f}".replace(",", "."))

# --- Visualisasi ---
rata_gaji.plot(kind="bar", figsize=(8,5), title="Rata-rata Gaji
per Departemen")
plt.ylabel("Rata-rata Gaji (INR)")
plt.xlabel("Departemen")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

Output kode menghasilkan :

```
Finance: Rp 940.412
HR: Rp 743.854
IT: Rp 1.129.859
Marketing: Rp 769.936
Operations: Rp 754.626
R&D: Rp 800.377
Sales: Rp 792.957
```

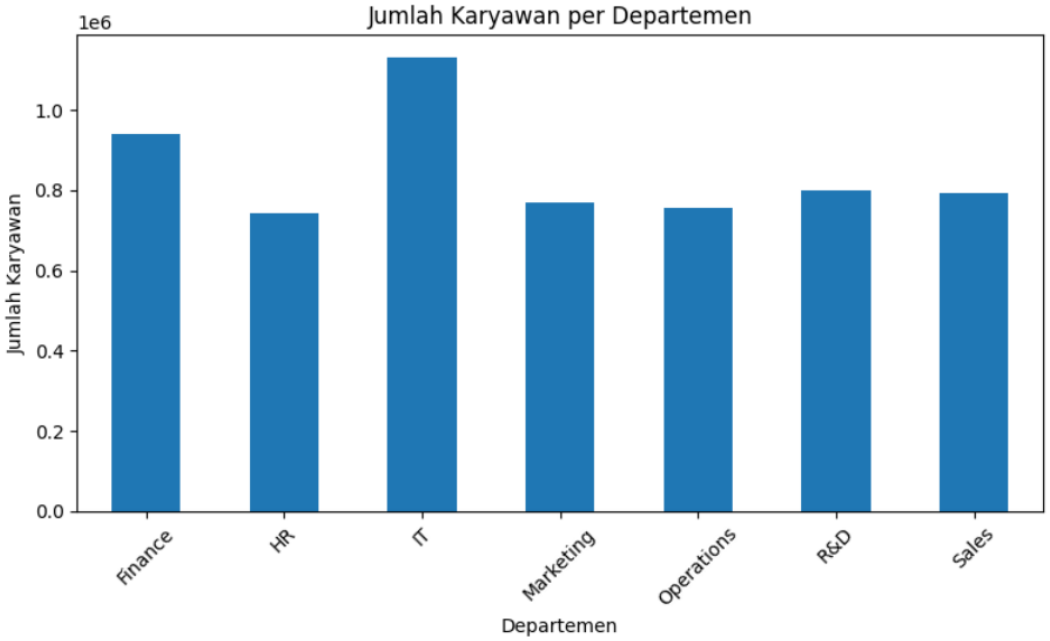


- Analisis Data jumlah karyawan di setiap departemen

```python
from pymongo import MongoClient
import pandas as pd
import matplotlib.pyplot as plt

# Koneksi MongoDB
client = MongoClient("mongodb://localhost:27017/")
db = client["Data_HRD"]
collection = db["Employee"]

# Ambil semua data dari MongoDB ke DataFrame
df = pd.DataFrame(list(collection.find()))

# Analisis sederhana: jumlah karyawan per Departemen
jum_ky = df.groupby("Department")["Full_Name"].count()
print(jum_ky)

# --- Visualisasi ---
rata_gaji.plot(kind="bar", figsize=(8,5), title="Jumlah Karyawan
per Departemen")
plt.ylabel("Jumlah Karyawan")
plt.xlabel("Departemen")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

Output kode menghasilkan :

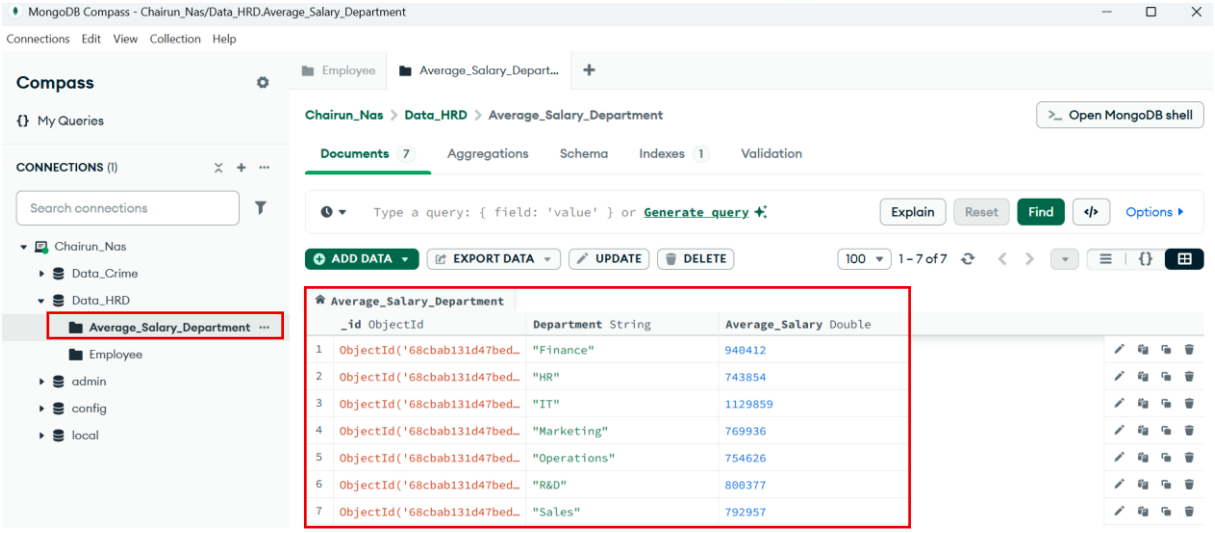
```
Department
Finance       199873
HR            159119
IT            601043
Marketing     240081
Operations    300095
R&D            99759
Sales         400032
Name: Full_Name, dtype: int64
```



- Analisis data, dengan hasil analisis data tersimpan dalam bentuk collection didalam MongoDB

```
from pymongo import MongoClient
import pandas as pd
import matplotlib.pyplot as plt

# --- Koneksi MongoDB ---
client = MongoClient("mongodb://localhost:27017/")
db = client["Data_HRD"]
collection = db["Employee"]

# --- Ambil semua data dari MongoDB ke DataFrame ---
df = pd.DataFrame(list(collection.find()))

# --- Analisis sederhana: rata-rata gaji per Departemen ---
rata_gaji = df.groupby("Department")["Salary_INR"].mean().round()

# --- Simpan hasil analisis ke MongoDB (collection baru) ---
avg_salary_collection = db["Average_Salary_Department"]

# Ubah Series ke dict lalu insert ke MongoDB
data_to_insert = [{"Department": dept, "Average_Salary":
float(salary)}
                  for dept, salary in rata_gaji.items()]
avg_salary_collection.insert_many(data_to_insert)
```

```
# --- Tampilkan hasil dalam format Rupiah ---
print("\nRata-rata Gaji per Departemen:")
for dept, salary in rata_gaji.items():
    print(f"{dept}: Rp {salary:,.0f}".replace(",", "."))

# --- Visualisasi ---
plt.figure(figsize=(8, 5))
rata_gaji.plot(kind="bar", color="skyblue", edgecolor="black")

plt.title("Rata-rata Gaji per Departemen", fontsize=14, weight="bold")
plt.ylabel("Rata-rata Gaji (INR)", fontsize=12)
plt.xlabel("Departemen", fontsize=12)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



Contoh analisis lain mencari rata-rata gaji berdasarkan jabatan disetiap departemen dapat dianalisis dengan kode berikut :

```
from pymongo import MongoClient
import pandas as pd
import matplotlib.pyplot as plt

# --- Koneksi MongoDB ---
client = MongoClient("mongodb://localhost:27017/")
db = client["Data_HRD"]
collection = db["Employee"]

# --- Ambil semua data dari MongoDB ke DataFrame ---
df = pd.DataFrame(list(collection.find()))
```

```
# --- Analisis rata-rata gaji Berdasarkan Jabatan Setiap Departemen
df_agg = df.groupby(["Department",
"Job_Title"])["Salary_INR"].mean().reset_index()

# --- Simpan hasil analisis ke MongoDB (collection baru)
Salary_JobTitle_collection = db["Salary_JobTitle_Departmen"]


# Konversi DataFrame ke list of dict
data_to_insert = df_agg.to_dict(orient="records")

# Simpan ke MongoDB
Salary_JobTitle_collection.insert_many(data_to_insert)

# --- Tampilkan hasil dalam format Rupiah ---
print("\nRata-rata Gaji per Departemen & Jabatan:")
for _, row in df_agg.iterrows():
    print(f"{row['Department']} - {row['Job_Title']}: Rp
{row['Salary_INR']:,.0f}".replace(",", "."))

# --- Visualisasi ---
plt.figure(figsize=(10, 6))

for dept in df_agg["Department"].unique():
    subset = df_agg[df_agg["Department"] == dept]
    plt.bar(subset["Job_Title"], subset["Salary_INR"], label=dept)

plt.title("Rata-rata Gaji per Jabatan di Setiap Departemen",
fontsize=14, weight="bold")
plt.ylabel("Rata-rata Gaji (INR)", fontsize=12)
plt.xlabel("Jabatan", fontsize=12)
plt.xticks(rotation=45)
plt.legend(title="Departemen")
plt.tight_layout()
plt.show()
```

6.5. Tugas Praktikum

a. Bagaimana distribusi “Work_Mode” (On-site, Remote)?

b. Berapa jumlah karyawan di setiap departemen?

c. Berapa rata-rata gaji per Departemen?

d. Jabatan apa yang memiliki rata-rata gaji tertinggi?

e. Berapa rata-rata gaji di berbagai Departemen berdasarkan Jabatan?

f. Berapa banyak karyawan yang Mengundurkan Diri & Dipecat di setiap departemen?

g. Bagaimana variasi gaji berdasarkan tahun pengalaman?

h. Berapa rata-rata peringkat kinerja berdasarkan departemen?
i. Negara mana yang memiliki konsentrasi karyawan tertinggi?
j. Apakah ada korelasi antara peringkat kinerja dan gaji?
k. Bagaimana jumlah perekrutan berubah dari waktu ke waktu (per tahun)?
l. Bandingkan gaji karyawan Remote vs. On-site, apakah ada perbedaan yang signifikan?
m. Temukan 10 karyawan dengan gaji tertinggi di setiap departemen.
n. Identifikasi departemen dengan tingkat attrition (persentase pengunduran diri) tertinggi.

6.6. Referensi

1. Apache Spark Documentation. (2025). Apache Spark: Unified Analytics Engine for Big Data. Diakses dari https://spark.apache.org/docs/latest/
2. MongoDB Documentation. (2025). MongoDB Manual. MongoDB Inc. Diakses dari https://www.mongodb.com/docs/
3. MongoDB Spark Connector. (2025). MongoDB Connector for Apache Spark. MongoDB Inc. Diakses dari https://www.mongodb.com/docs/spark-connector/current/