LAPORAN PRAKTIKUM
PRAKTIKUM DATABASE PERTEMUAN 11

Disusun untuk Memenuhi Matakuliah Praktikum {Nama Matakuliah}
Dibimbing oleh : Sulaibatul Aslamiyah, M.Kom

Oleh:

RAFLI RAHMAN.EFENDY

1124102162

PROGRAM STUDI S1 TEKNIK INFORMATIKA
SEKOLAH TINGGI ILMU KOMPUTER PGRI BANYUWANGI
2025

LEMBAR PENGESAHAN

LAPORAN PRAKTIKUM

Matakuliah : Praktikum Database

Oleh : Rafli Rahman.Efendy

NIM : 1124102162

Telah disahkan pada

Hari : RABU

Tanggal : 11/06/2025

Mengetahui/ Menyetujui :

Dosen Pengampu Mata Kuliah

Sulaibatul Aslamiyah, M.Kom

NIDN. 0712058304

KATA PENGANTAR

Berisikan kata pengantar yang dituliskan oleh mahasiswa untuk laporan praktikum matakuliah yang diikuti.

MODUL PRAKTIKUM 1

1.1 Capaian Praktikum Pertemuan AGREGATION Operation 1

Fitur dalam SQL yang dapat digunakan untuk melakukan kalkulasi data field table untuk mendapatkan sebuah value.

Dalam MongoDB, Aggregation (agregasi) adalah proses mengolah data dari beberapa dokumen dan menggabungkannya untuk menghasilkan hasil yang diringkas. Agregasi sangat berguna ketika kamu ingin melakukan analisis data seperti mengelompokkan, menghitung jumlah, menjumlahkan nilai, mengurutkan, atau memfilter berdasarkan kondisi tertentu.

MongoDB menyediakan beberapa cara untuk melakukan agregasi, tapi yang paling umum adalah menggunakan aggregation pipeline.

1.2 Landasan Teori

1. **Pengertian MongoDB**

   sebuah sistem manajemen basis data NoSQL yang berbasis dokumen dan bersifat open-source. Tidak seperti sistem basis data relasional (RDBMS) yang menggunakan tabel dan baris, MongoDB menyimpan data dalam format dokumen BSON (Binary JSON). Hal ini memungkinkan MongoDB untuk menyimpan data yang lebih fleksibel dan kompleks.

2. **Definisi Agregasi**

   konteks basis data, agregasi adalah proses pengolahan sekumpulan data untuk mendapatkan informasi yang diringkas, seperti jumlah total, rata-rata, maksimum, minimum, atau pengelompokan data berdasarkan kategori tertentu.

3. **Aggregation dalam MongoDB**

   Aggregation dalam MongoDB adalah teknik untuk mengolah dan merangkum data dari satu atau lebih dokumen dalam koleksi (collection). MongoDB menyediakan fitur aggregation pipeline, yang memungkinkan pengguna melakukan operasi agregasi secara berurutan dan efisien.

4. **Aggregation Pipeline**

   Aggregation pipeline adalah fitur utama dalam MongoDB untuk melakukan agregasi data. Pipeline ini terdiri dari beberapa tahap (stages) yang masing-masing bertugas untuk memproses dan mentransformasi data. Setiap tahap menerima masukan dari tahap sebelumnya dan memberikan keluaran ke tahap berikutnya.

- Beberapa tahapan penting dalam aggregation pipeline antara lain:
- $match – Menyaring dokumen berdasarkan kondisi tertentu (seperti WHERE dalam SQL)
- $group – Mengelompokkan data berdasarkan satu field dan menghitung agregat seperti jumlah atau rata-rata.
- $project – Memilih dan memodifikasi field yang akan ditampilkan.
- $sort – Mengurutkan hasil berdasarkan field tertentu.
- $limit dan $skip – Membatasi jumlah hasil atau melewatkan sejumlah hasil.

5. Fungsi-Fungsi Agregasi Umum

MongoDB menyediakan operator agregasi yang umum digunakan, seperti:

- $sum – Menjumlahkan nilai.
- $avg – Menghitung nilai rata-rata
- $min dan $max – Mengambil nilai minimum dan maksimum.
- $count – Menghitung jumlah dokumen.

6. Keunggulan Aggregation Pipeline

- Lebih fleksibel dan ekspresif dibandingkan metode tradisional seperti map-reduce.
- Efisien dan cepat, karena bisa berjalan langsung di server MongoDB tanpa perlu pengolahan eksternal.
- Mendukung transformasi data yang kompleks dengan kombinasi banyak tahap.

1.3 Pelaksanaan Praktikum

Model data embedded cocok untuk digunakan ketika : Antar document saling ketergantungan Kita jarang melakukan perubahan isi data pada embedded document Embedded document selalu dibutuhkan ketika mengambil data document

1.4.1 Percobaan Pertama

Latihan Pratikum Data Base MongoDB

Accumulator dalam Aggregate

Accumulator	Keterangan
$sum	Menjumlahkan nilai
$avg	Menghitung nilai rata-rata
$min	Mengambil nilai terkecil
$max	Mengambil nilai terbesar
$push	Memasukkan nilai ke dalam array
$addToSet	Memasukkan nilai unik ke array
$first	Mengambil nilai pertama
$last	Mengambil nilai terakhir

1. Percobaan Pertama Contoh Praktikum Non Database

```
test> show dbs
PertemuanTujuh  184.00 KiB
admin            40.00 KiB
biodata          80.00 KiB
bulkwrite        80.00 KiB
config           72.00 KiB
local            72.00 KiB
mahasiswa        40.00 KiB
mahasiwa         72.00 KiB
matakuliah       80.00 KiB
soallatihan     184.00 KiB
test> use NonRelationalPTM11
switched to db NonRelationalPTM11
NonRelationalPTM11> db.createCollection("nilaiMHS")
{ ok: 1 }
NonRelationalPTM11>
```

Buat data base nya:

use NonRelationalPTM11

1.Penjelasan :

- Ini berpindah (switch) ke database bernama NonRelationalPTM11.
- Jika database belum ada, MongoDB akan otomatis membuatnya saat kamu membuat collection atau insert dokumen pertama.

db.createCollection("nilaiMHS")

2.Penjelasan :

- Ini membuat collection bernama nilaiMHS secara eksplisit di database NonRelationalPTM11.
- Output { ok: 1 } artinya perintah berhasil.

Tambahan :

Sebenarnya, kamu tidak harus menggunakan createCollection( ) karena MongoDB secara otomatis akan membuat collection saat kamu pertama kali melakukan insert.

Contoh nya : db.nilaiMHS.insertOne({ nama: "Ali", mk: "Algoritma", nilai: 90 })

2. Percobaan Pertama Contoh Praktikum Non Database

Setelah membuat data dan Collection nya membuat isi di dalam Collection nya dengan db.insertMany atau db.insertOne.

Code nya MongoDB nya :

```
]
NonRelationalPTM11> db.nilaiMHS.insertMany([
...   { nama: "Ali", mk: "Basis Data", nilai: 88 },
...   { nama: "Budi", mk: "Algoritma", nilai: 92 },
...   { nama: "Citra", mk: "Jaringan", nilai: 75 },
...   { nama: "Dina", mk: "Basis Data", nilai: 78 },
...   { nama: "Eka", mk: "Algoritma", nilai: 85 },
...   { nama: "Fani", mk: "Jaringan", nilai: 80 },
...   { nama: "Gilang", mk: "Basis Data", nilai: 95 },
...   { nama: "Hani", mk: "Algoritma", nilai: 90 },
...   { nama: "Indra", mk: "Jaringan", nilai: 88 }
... ])
...
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('6848e302ea4747aaa6b71240'),
    '1': ObjectId('6848e302ea4747aaa6b71241'),
    '2': ObjectId('6848e302ea4747aaa6b71242'),
    '3': ObjectId('6848e302ea4747aaa6b71243'),
    '4': ObjectId('6848e302ea4747aaa6b71244'),
    '5': ObjectId('6848e302ea4747aaa6b71245'),
    '6': ObjectId('6848e302ea4747aaa6b71246'),
    '7': ObjectId('6848e302ea4747aaa6b71247'),
    '8': ObjectId('6848e302ea4747aaa6b71248')
  }
}
```

```
NonRelationalPTM11> db.nilaiMHS.insertMany([
...   { nama: "Sindy", mk: "Matematika", nilai: 90 },
...   { nama: "Ratna", mk: "Alpro", nilai: 85 },
...   { nama: "Rara", mk: "Agama", nilai: 95 },
...   { nama: "Sany", mk: "Matematika", nilai: 90 },
...   { nama: "Sinta", mk: "Alpro", nilai: 70 },
...   { nama: "Rara", mk: "Alpro", nilai: 85 },
...   { nama: "Sinta", mk: "Matematika", nilai: 95 },
...   { nama: "Ratna", mk: "Agama", nilai: 90 },
...   { nama: "Sany", mk: "Agama", nilai: 70 },
...   { nama: "Sindy", mk: "Alpro", nilai: 80 }
... ])
...
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('6848e112ea4747aaa6b71236'),
    '1': ObjectId('6848e112ea4747aaa6b71237'),
    '2': ObjectId('6848e112ea4747aaa6b71238'),
    '3': ObjectId('6848e112ea4747aaa6b71239'),
    '4': ObjectId('6848e112ea4747aaa6b7123a'),
    '5': ObjectId('6848e112ea4747aaa6b7123b'),
    '6': ObjectId('6848e112ea4747aaa6b7123c'),
    '7': ObjectId('6848e112ea4747aaa6b7123d'),
    '8': ObjectId('6848e112ea4747aaa6b7123e'),
    '9': ObjectId('6848e112ea4747aaa6b7123f')
  }
}
NonRelationalPTM11>
```

Penjelasan code nya :

1. **db.nilaiMHS.insertMany([**

```
NonRelationalPTM11> db.nilaiMHS.insertMany([
...   { nama: "Sindy", mk: "Matematika", nilai: 90 },
...   { nama: "Ratna", mk: "Alpro", nilai: 85 },
...   { nama: "Rara", mk: "Agama", nilai: 95 },
...   { nama: "Sany", mk: "Matematika", nilai: 90 },
...   { nama: "Sinta", mk: "Alpro", nilai: 70 },
...   { nama: "Rara", mk: "Alpro", nilai: 85 },
...   { nama: "Sinta", mk: "Matematika", nilai: 95 },
...   { nama: "Ratna", mk: "Agama", nilai: 90 },
...   { nama: "Sany", mk: "Agama", nilai: 70 },
...   { nama: "Sindy", mk: "Alpro", nilai: 80 }
... ])
...
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('6848e112ea4747aaa6b71236'),
    '1': ObjectId('6848e112ea4747aaa6b71237'),
    '2': ObjectId('6848e112ea4747aaa6b71238'),
    '3': ObjectId('6848e112ea4747aaa6b71239'),
    '4': ObjectId('6848e112ea4747aaa6b7123a'),
    '5': ObjectId('6848e112ea4747aaa6b7123b'),
    '6': ObjectId('6848e112ea4747aaa6b7123c'),
    '7': ObjectId('6848e112ea4747aaa6b7123d'),
    '8': ObjectId('6848e112ea4747aaa6b7123e'),
    '9': ObjectId('6848e112ea4747aaa6b7123f')
  }
}
```

2. **db.nilaiMHS.insertMany([**

```
NonRelationalPTM11> db.nilaiMHS.insertMany([
...   { nama: "Ali", mk: "Basis Data", nilai: 88 },
...   { nama: "Budi", mk: "Algoritma", nilai: 92 },
...   { nama: "Citra", mk: "Jaringan", nilai: 75 },
...   { nama: "Dina", mk: "Basis Data", nilai: 78 },
...   { nama: "Eka", mk: "Algoritma", nilai: 85 },
...   { nama: "Fani", mk: "Jaringan", nilai: 80 },
...   { nama: "Gilang", mk: "Basis Data", nilai: 95 },
...   { nama: "Hani", mk: "Algoritma", nilai: 90 },
...   { nama: "Indra", mk: "Jaringan", nilai: 88 }
... ])
...
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('6848e302ea4747aaa6b71240'),
    '1': ObjectId('6848e302ea4747aaa6b71241'),
```

```
    '2': ObjectId('6848e302ea4747aaa6b71242'),
    '3': ObjectId('6848e302ea4747aaa6b71243'),
    '4': ObjectId('6848e302ea4747aaa6b71244'),
    '5': ObjectId('6848e302ea4747aaa6b71245'),
    '6': ObjectId('6848e302ea4747aaa6b71246'),
    '7': ObjectId('6848e302ea4747aaa6b71247'),
    '8': ObjectId('6848e302ea4747aaa6b71248')
  }
}
```

Penjelsan Code MongoDB :

Fungsi insertMany():

- Digunakan untuk menambahkan banyak dokumen sekaligus ke dalam collection (nilaiMHS).
- Setiap objek di dalam array { ... } adalah satu dokumen mahasiswa.

Contoh Hasil:

Jika berhasil, MongoDB akan menampilkan :

```
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("665xxxx..."),
    '1': ObjectId("665xxxx..."),
    '2': ObjectId("665xxxx..."),
    ...
  }
}
```

Penjelasan Output :

acknowledged: true	MongoDB mengonfirmasi bahwa data telah berhasil dimasukkan.
insertedIds	Menampilkan ID unik (ObjectId) yang dibuat MongoDB untuk setiap dokumen yang disisipkan. Key '0', '1', dst. mewakili urutan data dalam array.

Data Akhir Setelah Kedua insertMany():

19 dokumen dalam collection nilaiMHS :

- 10 data pertama dari matkul: Matematika, Alpro, Agama
- 9 data kedua dari matkul: Basis Data, Algoritma, Jaringan

3. **Percobaan Pertama Contoh Praktikum Non Database**

Sertalah membaut isi data dari collection nya tampilkan hasil dengan db,nama collection nya.find( )

Dan hasil nya akan muncul dengan semua hasil yang telah yang di buat.

Ini yang saya ambil SS nya cumin Sebagian karenana kebanyakan

```
}
NonRelationalPTM11> db.nilaiMHS.find()
[
  {
    _id: ObjectId('6848e112ea4747aaa6b71236'),
    nama: 'Sindy',
    mk: 'Matematika',
    nilai: 90
  },
  {
    _id: ObjectId('6848e112ea4747aaa6b71237'),
    nama: 'Ratna',
    mk: 'Alpro',
    nilai: 85
  },
  {
    _id: ObjectId('6848e112ea4747aaa6b71238'),
    nama: 'Rara',
    mk: 'Agama',
    nilai: 95
  },
  {
    _id: ObjectId('6848e112ea4747aaa6b71239'),
    nama: 'Sany',
    mk: 'Matematika',
    nilai: 90
  },
  {
    _id: ObjectId('6848e112ea4747aaa6b7123a'),
    nama: 'Sinta',
    mk: 'Alpro',
    nilai: 70
  },
  {
    _id: ObjectId('6848e112ea4747aaa6b7123b'),
    nama: 'Rara',
    mk: 'Alpro',
    nilai: 85
  },
  {
    _id: ObjectId('6848e112ea4747aaa6b7123c'),
    nama: 'Sinta',
    mk: 'Matematika',
    nilai: 95
  },
  {
    _id: ObjectId('6848e112ea4747aaa6b7123d'),
    nama: 'Ratna',
    mk: 'Agama',
    nilai: 90
  },
  {
    _id: ObjectId('6848e112ea4747aaa6b7123e'),
    nama: 'Sany',
    mk: 'Agama',
    nilai: 70
  },
  {
    _id: ObjectId('6848e112ea4747aaa6b7123f'),
    nama: 'Sindy',
    mk: 'Alpro',
    nilai: 80
  }
]
NonRelationalPTM11>
```

Code MongoDB nya :

```
NonRelationalPTM11> db.nilaiMHS.find()
[
  {
    _id: ObjectId('6848e112ea4747aaa6b71236'),
    nama: 'Sindy',
    mk: 'Matematika',
    nilai: 90
  },
  {
    _id: ObjectId('6848e112ea4747aaa6b71237'),
    nama: 'Ratna',
    mk: 'Alpro',
    nilai: 85
  },
  {
    _id: ObjectId('6848e112ea4747aaa6b71238'),
    nama: 'Rara',
    mk: 'Agama',
    nilai: 95
  },
  {
    _id: ObjectId('6848e112ea4747aaa6b71239'),
    nama: 'Sany',
    mk: 'Matematika',
    nilai: 90
  },
  {
    _id: ObjectId('6848e112ea4747aaa6b7123a'),
    nama: 'Sinta',
    mk: 'Alpro',
    nilai: 70
  },
  {
    _id: ObjectId('6848e112ea4747aaa6b7123b'),
    nama: 'Rara',
    mk: 'Alpro',
    nilai: 85
  },
  {
    _id: ObjectId('6848e112ea4747aaa6b7123c'),
    nama: 'Sinta',
    mk: 'Matematika',
    nilai: 95
  },
  {
    _id: ObjectId('6848e112ea4747aaa6b7123d'),
    nama: 'Ratna',
    mk: 'Agama',
    nilai: 90
  },
  {
```

```
  _id: ObjectId('6848e112ea4747aaa6b7123e'),
  nama: 'Sany',
  mk: 'Agama',
  nilai: 70
 },
 {
  _id: ObjectId('6848e112ea4747aaa6b7123f'),
  nama: 'Sindy',
  mk: 'Alpro',
  nilai: 80
 },
 {
  _id: ObjectId('6848e302ea4747aaa6b71240'),
  nama: 'Ali',
  mk: 'Basis Data',
  nilai: 88
 },
 {
  _id: ObjectId('6848e302ea4747aaa6b71241'),
  nama: 'Budi',
  mk: 'Algoritma',
  nilai: 92
 },
 {
  _id: ObjectId('6848e302ea4747aaa6b71242'),
  nama: 'Citra',
  mk: 'Jaringan',
  nilai: 75
 },
 {
  _id: ObjectId('6848e302ea4747aaa6b71243'),
  nama: 'Dina',
  mk: 'Basis Data',
  nilai: 78
 },
 {
  _id: ObjectId('6848e302ea4747aaa6b71244'),
  nama: 'Eka',
  mk: 'Algoritma',
  nilai: 85
 },
 {
  _id: ObjectId('6848e302ea4747aaa6b71245'),
  nama: 'Fani',
  mk: 'Jaringan',
  nilai: 80
 },
 {
  _id: ObjectId('6848e302ea4747aaa6b71246'),
  nama: 'Gilang',
  mk: 'Basis Data',
  nilai: 95
 },
```

```
  {
    _id: ObjectId('6848e302ea4747aaa6b71247'),
    nama: 'Hani',
    mk: 'Algoritma',
    nilai: 90
  },
  {
    _id: ObjectId('6848e302ea4747aaa6b71248'),
    nama: 'Indra',
    mk: 'Jaringan',
    nilai: 88
  }
]
```

Penjelasan Code MongoDB nya :

```
db.nilaiMHS.find()
```

Penjelasan Fungsi:

- db merujuk ke database aktif saat ini, yaitu NonRelationalPTM11.
- nilaiMHS nama collection tempat data disimpan (seperti nama tabel).
- .find( ) digunakan untuk mengambil semua dokumen (baris data) dari collection tersebut.

Code Output nya:

Setelah kamu menjalankan insertMany( ) sebelumnya, hasil dari db.nilaiMHS.find() akan menampilkan seluruh dokumen yang ada.

```
[
  { _id: ObjectId("..."), nama: "Sindy", mk: "Matematika", nilai: 90 },
  { _id: ObjectId("..."), nama: "Ratna", mk: "Alpro", nilai: 85 },
  { _id: ObjectId("..."), nama: "Rara", mk: "Agama", nilai: 95 },
  { _id: ObjectId("..."), nama: "Sany", mk: "Matematika", nilai: 90 },
  ...
  { _id: ObjectId("..."), nama: "Indra", mk: "Jaringan", nilai: 88 }
]
```

Penjelasan Output:

- Setiap objek dalam array mewakili 1 baris dokumen dalam database.
- Field _id otomatis dibuat oleh MongoDB sebagai unique identifier.
- Field nama, mk, dan nilai sesuai dengan data yang kamu masukkan sebelumnya.

3. **Percobaan Pertama Contoh Praktikum Non Database**

1. **Uji Pertama**

```
db.nilaiMHS.countDocuments({ mk: "Matematika" })
```

Penjelasan :

- db → merujuk pada database aktif saat ini (NonRelationalPTM11).
- nilaiMHS → nama collection (seperti tabel) yang berisi data nilai mahasiswa.
- .countDocuments() → fungsi MongoDB untuk menghitung jumlah dokumen (baris data) yang memenuhi kriteria filter.
- { mk: "Matematika" } → adalah filter kondisi, artinya hanya cari data di mana mata kuliah (mk) bernilai "Matematika".

Code Output nya :

Artinya: Ada 3 mahasiswa yang mengambil mata kuliah Matematika di collection nilaiMHS

2. **Uji Pertama**

```
db.nilaiMHS.countDocuments({ mk: "Alpro" })
```

Penjelasan :

- Db : Mengacu ke database aktif saat ini, yaitu NonRelationalPTM11.
- NilaiMHS : Nama collection tempat data nilai mahasiswa disimpan (seperti tabel).
- countDocuments() : Fungsi untuk menghitung jumlah dokumen (baris data) di dalam collection yang sesuai dengan kriteria tertentu.
- { mk: "Alpro" } : Ini adalah filter pencarian: hanya hitung dokumen yang memiliki nilai mk (mata kuliah) sama dengan "Alpro".
- Mencari semua dokumen di collection nilaiMHS yang punya field mk bernilai "Alpro".
- Kemudian menghitung totalnya.

Code Output nya :

Ada 4 dokumen (data mahasiswa) yang mengambil mata kuliah Alpro dalam collection nilaiMHS.

3. **Uji Pertama**

```
NonRelationalPTM11> db.nilaiMHS.countDocuments({ mk: "Agama" })
```

Penjelasan :

- Db : Merujuk pada database aktif saat ini, yaitu NonRelationalPTM11.
- nilaiMHS : Nama collection (seperti tabel) tempat menyimpan data nilai mahasiswa.
- countDocuments() : Fungsi MongoDB untuk menghitung jumlah dokumen yang sesuai dengan kondisi.
- { mk: "Agama" }Filter : hanya dokumen yang memiliki mk (mata kuliah) bernilai "Agama" yang dihitung.

Code Output nya :

Di dalam collection nilaiMHS, ada 3 dokumen (baris data) yang memiliki nilai mk: "Agama".

4. **Percobaan Pertama Contoh Praktikum Non Database**

```
db.nilaiMHS.aggregate([
 {
  $group: {
   _id: "$mk",
   peserta: { $sum: 1 },
   nilaitertinggi: { $max: "$nilai" },
   nilaiterendah: { $min: "$nilai" },
   rata: { $avg: "$nilai" }
  }
 }
]).pretty()
```

Penjelasan Code MongoDB :

- aggregate([...])Menjalankan pipeline agregasi, yaitu proses analisis dan pengelompokan data.
- $group: { _id: "$mk" ... }Mengelompokkan data berdasarkan field mk (mata kuliah).
- peserta: { $sum: 1 } Menghitung jumlah mahasiswa (peserta) pada setiap mata kuliah.
- nilaitertinggi: { $max: "$nilai" } Mengambil nilai tertinggi di setiap kelompok mata kuliah.
- nilaiterendah: { $min: "$nilai" } Mengambil nilai terendah di setiap kelompok mata kuliah.
- rata: { $avg: "$nilai" } Menghitung nilai rata-rata pada setiap mata kuliah.
- .pretty() Membuat tampilan hasil menjadi lebih rapi dan terbaca.

Latihan Soal Pertemuan 11 AGREGATION

Soal Latihan Pratikum Non Database

- Berapa jumlah customer yang membeli tiap produk?
- Berapa jumlah penjualan tertinggi tiap produk?
- Berapa jumlah penjualan terendah tiap produk?
- Berapa Jumlah rata-rata penjualan tiap produk?

1. **Soal Pertama Berapa jumlah customer yang membeli tiap produk?**

```
]
NonRelationalPTM11> db.Barang11.aggregate([
...   {
...     $group: {
...       _id: "$Produk",
...       uniqueCustomers: { $addToSet: "$Customer" }
...     }
...   },
...   {
...     $project: {
...       _id: 0,
...       Produk: "$_id",
...       jumlahCustomer: { $size: "$uniqueCustomers" }
...     }
...   }
... ])
...
[
  { Produk: 'Pizza', jumlahCustomer: 3 },
  { Produk: 'Laptop', jumlahCustomer: 2 },
  { Produk: 'Keyboard', jumlahCustomer: 2 },
  { Produk: 'Headset', jumlahCustomer: 2 },
  { Produk: 'Toothbrush', jumlahCustomer: 3 },
  { Produk: 'Monitor', jumlahCustomer: 2 },
  { Produk: 'Mouse', jumlahCustomer: 2 },
  { Produk: 'Milk', jumlahCustomer: 2 },
  { Produk: 'Guitar', jumlahCustomer: 2 }
]
NonRelationalPTM11>
```

```
db.Barang11.aggregate([
 {
  $group: {
   _id: "$Produk",
   uniqueCustomers: { $addToSet: "$Customer" }
  }
 },
 {
  $project: {
   _id: 0,
   Produk: "$_id",
   jumlahCustomer: { $size: "$uniqueCustomers" }
  }
 }
])
```

Penjelasan Code MongoDB nya :

- _id: "$Produk"
- Mengelompokkan dokumen berdasarkan field Produk (artinya satu grup untuk setiap nama produk).
- uniqueCustomers: { $addToSet: "$Customer" }
- Mengumpulkan customer yang berbeda (unik) dalam array. $addToSet memastikan tidak ada duplikat.

Penjelasan Code MongoDB nya :

- _id: 0
- Tidak menampilkan field _id di hasil akhir.
- Produk: "$_id"
- Menyalin nilai _id dari hasil $group (yang berisi nama produk) ke field Produk.
- jumlahCustomer: { $size: "$uniqueCustomers" }
- Menghitung jumlah elemen dalam array uniqueCustomers, alias menghitung jumlah customer unik untuk produk itu.

2. **Soal Kedua Berapa jumlah penjualan tertinggi tiap produk?**

```
NonRelationalPTM11> db.Barang11.aggregate([
...   {
...     $group: {
...       _id: "$Produk",
...       maxTotal: { $max: "$total" }
...     }
...   },
...   {
...     $project: {
...       _id: 0,
...       Produk: "$_id",
...       PenjualanTertinggi: "$maxTotal"
...     }
...   }
... ])
...
[
  { Produk: 'Monitor', PenjualanTertinggi: 100 },
  { Produk: 'Toothbrush', PenjualanTertinggi: 20 },
  { Produk: 'Keyboard', PenjualanTertinggi: 123 },
  { Produk: 'Headset', PenjualanTertinggi: 45 },
  { Produk: 'Pizza', PenjualanTertinggi: 20 },
  { Produk: 'Mouse', PenjualanTertinggi: 100 },
  { Produk: 'Milk', PenjualanTertinggi: 20 },
  { Produk: 'Guitar', PenjualanTertinggi: 17 },
  { Produk: 'Laptop', PenjualanTertinggi: 123 }
]
NonRelationalPTM11>
```

```
db.Barang11.aggregate([
 {
  $group: {
   _id: "$Produk",
   maxTotal: { $max: "$total" }
```

```
   }
  },
  {
   $project: {
     _id: 0,
     Produk: "$_id",
     PenjualanTertinggi: "$maxTotal"
   }
  }
])
```

Penjelasan Code MongoDB nya :

- _id: "$Produk"
- Kelompokkan data berdasarkan nama produk. Setiap jenis produk akan menjadi satu grup.
- maxTotal: { $max: "$total" }
- Ambil nilai maksimum dari field total dalam tiap grup.
- Field total di sini diasumsikan merepresentasikan jumlah penjualan dalam suatu transaksi.

Penjelasan Code MongoDB nya :

- _id: 0
- Hilangkan field _id dari output.
- Produk: "$_id"
- Salin nilai produk dari _id (hasil dari $group) ke field baru bernama Produk.
- PenjualanTertinggi: "$maxTotal"
- Tampilkan nilai maksimum penjualan (maxTotal) sebagai PenjualanTertinggi.

3. Soal Ketiga Berapa jumlah penjualan terendah tiap produk?

```
NonRelationalPTM11> db.Barang11.aggregate([
...   {
...     $group: {
...       _id: "$Produk",
...       minTotal: { $min: "$total" }
...     }
...   },
...   {
...     $project: {
...       _id: 0,
...       Produk: "$_id",
...       PenjualanTerendah: "$minTotal"
...     }
...   }
... ])
...
[
  { Produk: 'Toothbrush', PenjualanTerendah: 10 },
  { Produk: 'Pizza', PenjualanTerendah: 7 },
  { Produk: 'Milk', PenjualanTerendah: 10 },
  { Produk: 'Guitar', PenjualanTerendah: 5 },
  { Produk: 'Laptop', PenjualanTerendah: 78 },
  { Produk: 'Keyboard', PenjualanTerendah: 78 },
  { Produk: 'Mouse', PenjualanTerendah: 44 },
  { Produk: 'Monitor', PenjualanTerendah: 45 },
  { Produk: 'Headset', PenjualanTerendah: 44 }
]
NonRelationalPTM11>
```

```
db.Barang11.aggregate([
 {
  $group: {
    _id: "$Produk",
```

```
    minTotal: { $min: "$total" }
   }
  },
  {
   $project: {
    _id: 0,
    Produk: "$_id",
    PenjualanTerendah: "$minTotal"
   }
  }
])
```

Penjelasan Code MongoDB nya :

1. **db.Barang11.aggregate([...])**
   - Fungsi ini menjalankan pipeline agregasi, yang memproses data secara bertahap dari koleksi Barang11.
2. **Fungsi:**
   - Mengelompokkan data berdasarkan field Produk.
   - Untuk setiap kelompok produk, cari nilai terendah dari field total.
3. **Artinya:**
   - Jika produk "Kopi" punya transaksi dengan total: 15000, 10000, 20000, maka hasilnya:
4. **Rinciannya:**
   - _id: 0 → menyembunyikan _id bawaan.
   - Produk: "$_id" → menampilkan nama produk dari hasil $group.
   - PenjualanTerendah: "$minTotal" → menampilkan nilai minimum total penjualan.

4. **Soal Ketiga Berapa jumlah penjualan terendah tiap produk?**

```
NonRelationalPTM11> db.Barang11.aggregate([
...   {
...     $group: {
...       _id: "$Produk",
...       avgTotal: { $avg: "$total" }
...     }
...   },
...   {
...     $project: {
...       _id: 0,
...       Produk: "$_id",
...       RataRataPenjualan: { $round: ["$avgTotal", 2] }
...     }
...   }
... ])
...
[
  { Produk: 'Monitor', RataRataPenjualan: 72.5 },
  { Produk: 'Toothbrush', RataRataPenjualan: 15 },
  { Produk: 'Keyboard', RataRataPenjualan: 100.5 },
  { Produk: 'Headset', RataRataPenjualan: 44.5 },
  { Produk: 'Pizza', RataRataPenjualan: 15.67 },
  { Produk: 'Mouse', RataRataPenjualan: 72 },
  { Produk: 'Milk', RataRataPenjualan: 15 },
  { Produk: 'Guitar', RataRataPenjualan: 11 },
  { Produk: 'Laptop', RataRataPenjualan: 100.5 }
]
NonRelationalPTM11>
```

```
db.Barang11.aggregate([
 {
  $group: {
   _id: "$Produk",
   avgTotal: { $avg: "$total" }
  }
 },
 {
  $project: {
   _id: 0,
   Produk: "$_id",
   RataRataPenjualan: { $round: ["$avgTotal", 2] }
  }
 }
])
```

Penjelasan Code MongoDB nya :

Tujuan:

Menampilkan setiap produk beserta rata-rata nilai penjualan (total), dan membulatkan hasilnya ke 2 angka di belakang koma.

1. db.Barang11.aggregate([...])

- Menjalankan proses aggregation pipeline pada koleksi Barang11.
- Pipeline ini digunakan untuk memproses data dalam beberapa tahap.

2. Tahap pertama: $group

```
          {
           $group: {
            _id: "$Produk",
            avgTotal: { $avg: "$total" }
           }
}
```

Fungsi:

- Mengelompokkan dokumen berdasarkan nama Produk.
- Menghitung rata-rata nilai dari field total pada setiap kelompok.

3. Tahap kedua: $project

```
{
 $project: {
  _id: 0,
  Produk: "$_id",
  RataRataPenjualan: { $round: ["$avgTotal", 2] }
 }
}
```

Fungsi:

- Menentukan format tampilan hasil akhir.

Rinciannya:

- _id: 0 → Menyembunyikan field _id dari output.
- Produk: "$_id" → Menampilkan nama produk dari hasil grouping.
- RataRataPenjualan: { $round: ["$avgTotal", 2] } → Membulatkan nilai avgTotal ke 2 angka di belakang koma.

Kesimpulan Kode ini digunakan untuk:

Menghitung rata-rata nilai penjualan (total) untuk setiap produk, lalu menampilkan hasilnya dengan pembulatan 2 angka di belakang koma.

Kesimpulan

Kesimpulan Percobaan 1

Dengan study kasus ini Saya bisa membuat Relasi One-to-One adalah relasi dimana suatu baris tabel A hanya berhubungan dengan suatu baris tabel B. Kita dapat menerapkan model data Embedded Document untuk mempresentasikan relasi One-to-One tersebut pada database Non Relational.

Kesimpulan Percobaan 2

Dengan study kasus ini Saya bisa membuat Mirip dengan implementasi embedded document pada relasi One-to-One, pada kasus One-to-Many, document dengan relasi “Many” akan di-embedded kedalam document dengan relasi “One”.

Kesimpulan Percobaan 3

Dengan study kasus ini Saya bisa membuatUntuk menampilkan data kita dapat menggunakan method $lookup dengan aggregation.

Kesimpulan Percobaan 4

Dengan study kasus ini Saya bisa membuat Collection dan juga menambahkan dukumen menggunakan one to one dan many to one, dan juga query untuk menampilkan isi document pada collection

BAB 1

PENUTUP

Kesimpulan

teori MySQL adalah Structured Query Language (SQL) sebagai bahasa interaktif untuk mengelola data. MySQL adalah sistem manajemen basis data relasional (RDBMS) yang menggunakan SQL untuk menjalankan fungsinya..

Saran

Saran saya tidak ada perubahan pembelajaran sama seperti si Semester 1 dan masih masuk di praktikum kali kecuali pada pengisian laporan praktikum database ke word itu yang perlu di tanyakan dan cara pengisian laporan untuk kedepannya.

DAFTAR PUSTAKA

Tuliskan rujukan yang anda gunakan baik website maupun buku seperti contoh dibawah.

1. Tim Asisten Dosen. 2014. Modul 1 Pengenalan Sistem Operasi, Ide Visual C++, Dan Algoritma Pemrograman. Malang: Unversitas Negeri Malang.
2. Program Konversi Suhu (online) http://bondanoky.blogspot.com/2012/10/program-konversi-suhu-c.html. Di akses 8 September.

LAMPIRAN

Berisikan syntax atau gambar yang dibutuhkan dalam tiap pertemuan praktikum.