Corso di Laurea Magistrale
in
Ingegneria Informatica

Big Data Analytics and Business Intelligence
Homework 3

prof. Giancarlo Sperlì

M63000989 Fabio d'Andrea
M63000986 Guido Di Chiara

Università degli Studi di Napoli Federico II

Scuola Politecnica e delle Scienze di Base

Anno Accademico 2020/2021
Secondo Semestre

Indice

Capitolo 1

Introduzione

Il seguente elaborato illustra il lavoro svolto per il terzo homework del corso di *Big Data Analytics and Business Intelligence* presso l'*Università degli Studi di Napoli Federico II*. L'homework è incentrato sull'utilizzo del database NoSQL *MongoDB* per la gestione dei dati relativi alla diffusione del Covid-19 in Italia.

1.1 Business Understanding

Tale elaborato ha l'obiettivo di sviluppare un database NoSQL per memorizzare un insieme di dati relativi all'epidemia di Covid-19 sul territorio nazionale e al piano vaccinale condotto dal Ministero della Salute. I dati relativi alla pandemia provengono dal *Dipartimento della Protezione Civile* mentre quelli inerenti alle vaccinazioni sono forniti dal *Dipartimento per la Trasformazione Digitale*.

Lo scopo dell'elaborato è quello di definire ed implementare un modello dei dati in un database NoSQL, mettendo in evidenza il motivo della scelta del modello e della tecnologia. Inoltre, per valutare i vantaggi che ne derivano sono state realizzate diverse query sul database.

Per la memorizzazione dei dati si è scelto di utilizzare **MongoDB**, un database NoSQL orientato ai documenti. Il database è stato installato localmente ed è stato gestito attraverso l'APIs Python **PyMongo**, la quale permette, tra le altre cose, di creare un database, definire delle *collection* di documenti, realizzare query e operazioni CRUD.

Il codice relativo alle elaborazioni effettuate è contenuto in un file *.ipynb*, reperibile su un repository GitHub [1].

1.2 Data Understanding

Nella sezione corrente si vogliono descrivere con maggiore precisione i dati elaborati, i quali possono essere divisi in due categorie.

È bene sottolineare che, dovendo implementare un database documentale, i dati utilizzati per popolarlo provengono da file in formato *.json*.

1.2.1 Dati Covid-19

La prima categoria di dati include tutte le informazioni relative alla diffusione dell'epidemia di Covid-19 in Italia. La fonte da cui sono stati raccolti i dati è un repository Git-Hub pubblicato dal *Dipartimento della Protezione Civile* [2].

In particolare, il file considerato nell'elaborazione è: *dpc-covid19-ita-regioni.json* e contiene i dati di principale interesse per ciascuna regione italiana dall'inizio della pandemia ad oggi. Tra le altre cose, sono riportate informazioni relative a nuovi contagi, attuali positivi, decessi, guariti e ricoverati. Per ulteriori informazioni sui dati appena citati e sul loro formato è possibile consultare [3].

1.2.2 Dati Vaccini

La seconda categoria di dati include tutte le informazioni relative al piano vaccinale italiano. La fonte da cui sono stati raccolti i dati è un repository Git-Hub pubblicato dal *Dipartimento per la Trasformazione Digitale* [4].

In particolare, il file considerato nell'elaborazione è: *somministrazioni-vaccini-latest.json* e contiene i dati relativi ai vaccini somministrati alla popolazione di ciascuna regione, distinguendo tra le diverse categorie definite dal Ministero della Salute e fasce di età. Per ulteriori informazioni sui dati appena citati e sul loro formato è possibile consultare [5].

Capitolo 2

Data Model

Nel capitolo corrente sono riportate le operazioni di manipolazione effettuate sui dati per la definizione di un modello dei dati che permetta una semplice fruizione da parte degli utenti finali, sulla base di quelle che si ritengono essere le operazioni più frequenti.

2.1 MongoDB

Osservando la struttura dei dati a disposizione si è scelto di utilizzare il database NoSQL *MongoDB*. La struttura dei dati si presta infatti alla memorizzazione in un database documentale. Tutte le informazioni associate ad una singola data dall'inizio della pandemia ad oggi (positivi, tamponi, etc.) possono essere modellate in un documento. Lo stesso vale per le informazioni relative ai vaccini. Non esistono quindi particolari relazioni tra i singoli documenti.

2.2 Creazione del Database

Attraverso l'API *PyMongo* è possibile instanziare un **client** MongoDB, connettersi al server e creare un nuovo **database**. Si creano quindi due **collection** all'interno del database, che conterranno rispettivamente i documenti relativi ai dati della diffusione del Covid-19 e ai dati del piano vaccinale. Una volta create le collection, il database può essere popolato inserendo i dati contenuti nei file *.json* descritti nel Paragrafo 1.2.

Creazione database e collection

```
client = pymongo.MongoClient('localhost', 27017)

db = client['homework3']

collection_covid = db.dati_covid
collection_vaccini = db.dati_vaccini

collection_covid.insert_many(dati_covid)
collection_vaccini.insert_many(dati_vaccini)
```

Sebbene MongoDB sia un database *schema-less*, i documenti memorizzati nelle collection presentano una struttura fissa.

```
_id: ObjectId("60a60b011330a563fbcf4423")
data: 2020-02-24T18:00:00.000+00:00
stato: "ITA"
codice_regione: 13
denominazione_regione: "Abruzzo"
lat: 42.35122196
long: 13.39843823
ricoverati_con_sintomi: 0
terapia_intensiva: 0
totale_ospedalizzati: 0
isolamento_domiciliare: 0
totale_positivi: 0
variazione_totale_positivi: 0
nuovi_positivi: 0
dimessi_guariti: 0
deceduti: 0
casi_da_sospetto_diagnostico: null
totale_casi: 0
tamponi: 5
casi_testati: null
note: null
ingressi_terapia_intensiva: null
note_test: null
note_casi: null
totale_positivi_test_molecolare: null
totale_positivi_test_antigenico_rapido: null
tamponi_test_molecolare: null
tamponi_test_antigenico_rapido: null
codice_nuts_1: null
codice_nuts_2: null
```

(a) Schema collection dati Covid-19

```
_id: ObjectId("60a60b021330a563fbcf68b9")
index: 0
data_somministrazione: 2020-12-27T00:00:00.000+00:00
fornitore: "Pfizer/BioNTech"
area: "ABR"
fascia_anagrafica: "20-29"
sesso_maschile: 1
sesso_femminile: 0
categoria_operatori_sanitari_sociosanitari: 1
categoria_personale_non_sanitario: 0
categoria_ospiti_rsa: 0
categoria_60_69: 0
categoria_70_79: 0
categoria_over80: 0
categoria_forze_armate: 0
categoria_personale_scolastico: 0
categoria_soggetti_fragili: 0
categoria_altro: 0
prima_dose: 1
seconda_dose: 0
codice_NUTS1: "ITF"
codice_NUTS2: "ITF1"
codice_regione_ISTAT: 13
nome_area: "Abruzzo"
```

(b) Schema collection dati vaccini

Figura 2.1: Schemi collection

2.3 Preprocessing

Prima di procedere, si converte il formato delle date presenti nei documenti di entrambe le collection dal tipo `string` al tipo `date`. In questo modo, si facilitano le interrogazioni effettuate da parte dell'utente. Per effettuare la conversione si utilizza il metodo `update_many` offerto da PyMongo, che permette di aggiornare più documenti presenti in una determinata collection.

Formato date

```
filter = {}

update = [{"$set": {
            'data_somministrazione': {
                "$toDate": "$data_somministrazione"
                }
            }
         }]

collection_vaccini.update_many(filter, update)


filter = {}

update = [{"$set": {
            'data': {
```

```
                "$toDate": "$data"
                }
            }
        }]

collection_covid.update_many(filter, update)
```

2.4 Creazione delle Views

Sulla base delle interrogazioni più frequenti che si ritiene vengano effettuate sui dati sono state definite diverse **view** sulle due collection. Una view non è altro che il risultato read-only di una **aggregation pipeline** eseguita su una collection. Le aggregation pipeline sono strumenti offerti da MongoDB che permettono di eseguire una serie di operazioni aggregate in grado di trasformare i documenti presenti in una collection. Le viste definite permettono quindi di accedere facilmente ai risultati delle aggregazioni più frequenti, senza richiedere che l'utente definisca la pipeline.

2.4.1 Dati Covid-19

2.4.1.1 Totale Casi e Deceduti ad ogni Mese

La prima view trasforma i dati in modo da restituire documenti contenenti il numero di casi e deceduti al termine di ogni mese dall'inizio della pandemia.

Totale casi e deceduti ad ogni mese

```
pipeline = [
    { "$group" : {
        "_id": {
            "anno" : {"$year": "$data"},
            "mese" : {"$month": "$data"},
            "regione" : "$denominazione_regione"
        },
        "deceduti": { "$max": "$deceduti" },
        "casi": { "$max": "$totale_casi" }
    }},

    { "$group" : {
        "_id": {
            "anno" : "$_id.anno",
            "mese" : "$_id.mese",
        },
        "deceduti": { "$sum": "$deceduti" },
        "casi": { "$sum": "$casi" }
    }},

    { "$project": {
        "anno": "$_id.anno",
        "mese": "$_id.mese",
        "deceduti": 1,
        "casi": 1,
        "_id": 0
    }},

    { "$sort" : {"anno": -1, "mese": -1} }
]

db.create_collection(
    'dati_covid_mensili',
    viewOn='dati_covid',
    pipeline=pipeline
)
```

Per mostrare la struttura di un documento della vista di seguito se ne riporta un esempio.

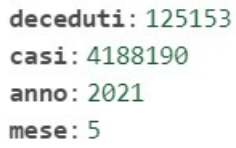

Figura 2.2: Schema documenti vista

A partire da questa vista, l'utente può effettuare query filtrando i dati su mesi e/o anni di interesse. Ad esempio, è possibile consultare i casi e i deceduti al termine di ogni mese dell'anno 2021.

Totale casi e deceduti ad ogni mese nel 2021

```
result = db.dati_covid_mensili.find(filter={'anno': 2021})
result_list = list(result)

df = pd.DataFrame(result_list)
df
```

	deceduti	casi	anno	mese
0	125153	4188190	2021	5
1	120807	4022653	2021	4
2	109346	3583444	2021	3
3	97699	2925265	2021	2
4	88516	2553032	2021	1

Figura 2.3: Totale casi e deceduti ad ogni mese nel 2021

Per mostrare i risultati delle query in maniera opportuna, i documenti restituiti vengono inseriti all'interno di una struttura dati *DataFrame* offerta dalla libreria **Pandas**.

2.4.2 Dati Vaccini

2.4.2.1 Prima e Seconda Dose per Regione e Fascia Anagrafica

La seconda view trasforma i dati in modo da restituire documenti contenenti il numero di dosi di vaccino somministrate in ogni regione, per ciascuna fascia anagrafica, distinguendo tra prima e seconda dose.

Prima e seconda dose per regione e fascia anagrafica

```
pipeline = [
    { "$group": {
        "_id": {
            "regione": "$nome_area",
            "fascia_anagrafica": "$fascia_anagrafica"
        },
        "prima_dose": { "$sum": "$prima_dose" },
        "seconda_dose" : { "$sum": "$seconda_dose"}
    }},

    { "$project": {
        "regione": "$_id.regione",
        "fascia_anagrafica": "$_id.fascia_anagrafica",
        "prima_dose": 1,
        "seconda_dose": 1,
        "_id": 0
    }},

    { "$sort" : { "regione": 1,  "fascia_anagrafica": 1} }
]

db.create_collection(
    'dati_vaccini_regioni_fasce',
    viewOn='dati_vaccini',
    pipeline=pipeline
)
```

Per mostrare la struttura di un documento della vista di seguito se ne riporta un esempio.

```
prima_dose: 1108
seconda_dose: 521
regione: "Abruzzo"
fascia_anagrafica: "16-19"
```

Figura 2.4: Schema documenti vista

A partire da questa vista, l'utente può effettuare query filtrando i dati sulla regione di interesse o aggregando rispetto ad un campo di interesse. Ad esempio, è possibile consultare le dosi somministrate in ciascuna regione aggregando ulteriormente sulla vista rispetto al campo `regione`.

Prima e seconda dose per regione

```
result = db.dati_vaccini_regioni_fasce.aggregate([
    { "$group": {
        "_id": "$regione",
        "prima_dose": { "$sum": "$prima_dose" },
        "seconda_dose" : { "$sum": "$seconda_dose"}
    }},

    { "$sort" : { "prima_dose": -1 } }
])

result_list = list(result)

df = pd.DataFrame(result_list)
df
```

	_id	prima_dose	seconda_dose
0	Lombardia	3609626	1655555
1	Lazio	2041929	895460
2	Campania	2034829	814263
3	Veneto	1747921	768710
4	Emilia-Romagna	1563154	783585
5	Sicilia	1490197	719623
6	Piemonte	1477529	767950
7	Puglia	1430250	637612
8	Toscana	1230896	630477
9	Calabria	624727	276600
10	Sardegna	559951	222128
11	Liguria	555896	298546
12	Marche	526885	273997
13	Abruzzo	462091	226233
14	Friuli-Venezia Giulia	438011	196538
15	Umbria	304560	151378
16	Provincia Autonoma Trento	207046	58765
17	Basilicata	199231	95228
18	Provincia Autonoma Bolzano / Bozen	198447	80884
19	Molise	117674	51321
20	Valle d'Aosta / Vallée d'Aoste	43190	21731

Figura 2.5: Prima e seconda dose per regione

Analogamente, è possibile consultare le dosi somministrate per ciascuna fascia anagrafica aggregando ulteriormente sulla vista rispetto al campo `fascia_anagrafica`.

Prima e seconda dose per fascia anagrafica

```
result = db.dati_vaccini_regioni_fasce.aggregate([
    { "$group": {
        "_id": "$fascia_anagrafica",
        "prima_dose": { "$sum": "$prima_dose" },
        "seconda_dose" : { "$sum": "$seconda_dose"}
    }},

    { "$sort" : { "prima_dose": -1 } }
])

result_list = list(result)

df = pd.DataFrame(result_list)
df
```

	_id	prima_dose	seconda_dose
0	60-69	4863518	1323231
1	70-79	4809466	1612266
2	50-59	3402991	1206718
3	80-89	3311967	2974452
4	40-49	1767644	842068
5	30-39	1083518	579357
6	20-29	799809	412516
7	90+	757214	651397
8	16-19	67913	24579

Figura 2.6: Prima e seconda dose per fascia anagrafica

2.4.2.2 Dosi Somministrate per Regione e Categoria

La terza view trasforma i dati in modo da restituire documenti contenenti il numero di dosi di vaccino somministrate in ogni regione, per ognuna delle categorie definite dal Ministero della Salute.

Dosi somministrate per regione e categoria

```
pipeline = [
    { "$group": {
        "_id": "$nome_area",
        "personale_scolastico": { "$sum": "$categoria_personale_scolastico" },
        "soggetti_fragili": { "$sum": "$categoria_soggetti_fragili" },
        "operatori_sanitari_sociosanitari": {
                "$sum": "$categoria_operatori_sanitari_sociosanitari" },
        "personale_non_sanitario": { "$sum": "$categoria_personale_non_sanitario" },
        "60_69": { "$sum": "$categoria_60_69" },
        "over80": { "$sum": "$categoria_over80" },
        "altro": { "$sum": "$categoria_altro" },
        "ospiti_rsa": { "$sum": "$categoria_ospiti_rsa" },
        "70_79": { "$sum": "$categoria_70_79" },
        "forze_armate": { "$sum": "$categoria_forze_armate" }
    }},

    { "$project": {
        "_id": 0,
        "regione": "$_id",
        "personale_scolastico": 1,
        "soggetti_fragili": 1,
        "operatori_sanitari_sociosanitari": 1,
        "personale_non_sanitario": 1,
        "60_69": 1,
        "over80": 1,
        "altro": 1,
        "ospiti_rsa": 1,
        "70_79": 1,
        "forze_armate": 1
    }}
]

db.create_collection(
    'dati_vaccini_regioni_categorie',
    viewOn='dati_vaccini',
    pipeline=pipeline
)
```

Per mostrare la struttura di un documento della vista di seguito se ne riporta un esempio.


```
personale_scolastico: 44318
soggetti_fragili: 183133
operatori_sanitari_sociosanitari: 82285
personale_non_sanitario: 27390
60_69: 85774
over80: 182557
altro: 22991
ospiti_rsa: 14952
70_79: 145270
forze_armate: 12212
regione: "Marche"
```


Figura 2.7: Schema documenti vista

A partire da questa vista, l'utente può effettuare query filtrando i dati sulla regione di interesse o aggregando ulteriormente. Ad esempio, è possibile consultare le dosi somministrate in tutto il Paese per ciascuna categoria aggregando ulteriormente sulla vista.

Dosi somministrate per categoria

```
result = db.dati_vaccini_regioni_categorie.aggregate([
    { "$group": {
        "_id": "null",
        "personale_scolastico": { "$sum": "$personale_scolastico" },
        "soggetti_fragili": { "$sum": "$soggetti_fragili" },
        "operatori_sanitari_sociosanitari": { "$sum": "$operatori_sanitari_sociosanitari" },
        "personale_non_sanitario": { "$sum": "$personale_non_sanitario" },
        "60_69": { "$sum": "$60_69" },
        "over80": { "$sum": "$80" },
        "altro": { "$sum": "$altro" },
        "ospiti_rsa": { "$sum": "$ospiti_rsa" },
        "70_79": { "$sum": "$70_79" },
        "forze_armate": { "$sum": "$forze_armate" }
    }},

    { "$project": {
        "_id": 0
    }}
])

result_list = list(result)

df = pd.DataFrame(result_list)
df
```

	personale_scolastico	soggetti_fragili	operatori_sanitari_sociosanitari	personale_non_sanitario	60_69	over80	altro	ospiti_rsa	70_79	forze_armate
0	1557866	6422747	3427335	988025	3483209	0	1970840	693699	4551183	480331

Figura 2.8: Dosi somministrate per categoria

2.4.2.3 Dosi Somministrate per Fascia Anagrafica e Fornitore

La quarta view trasforma i dati in modo da restituire documenti contenenti il numero di dosi di vaccino di ciascun fornitore somministrate ad ogni fascia anagrafica.

Dosi somministrate per fascia anagrafica e fornitore

```
pipeline = [
    { "$group" : {
        "_id": {
            "fascia_anagrafica": "$fascia_anagrafica",
            "fornitore": "$fornitore"
        },
        "prima_dose": { "$sum": "$prima_dose" },
        "seconda_dose" : { "$sum": "$seconda_dose"}
    }},

    { "$addFields": {
        "totale_dosi": { "$sum": ["$prima_dose", "$seconda_dose"]}
    }},

    { "$project": {
        "fascia_anagrafica": "$_id.fascia_anagrafica",
        "fornitore": "$_id.fornitore",
        "totale_dosi": 1,
        "_id": 0
    }},

    { "$sort" : {
        "fascia_anagrafica": -1,
        "totale_dosi": -1
    }}
]

db.create_collection(
    'dati_vaccini_fasce_fornitori',
    viewOn='dati_vaccini',
    pipeline=pipeline
)
```

Per mostrare la struttura di un documento della vista di seguito se ne riporta un esempio.

Figura 2.9: Schema documenti vista

A partire da questa vista, l'utente può effettuare query filtrando o aggregando i dati ulteriormente sui campi di interesse. Ad esempio, è possibile individuare il principale fornitore di ciascuna fascia anagrafica in termini di dosi somministrate.

Principale fornitore per fascia anagrafica

```
result = db.dati_vaccini_fasce_fornitori.aggregate([
    { "$group" : {
        "_id": "$fascia_anagrafica",
        "totale_dosi": {"$first": "$totale_dosi"},
        "fornitore": {"$first": "$fornitore" }
    }},

    { "$sort" : {
        "_id": -1,
    }},
])

result_list = list(result)

df = pd.DataFrame(result_list)
df
```

	_id	totale_dosi	fornitore
0	90+	1166414	Pfizer/BioNTech
1	80-89	5548385	Pfizer/BioNTech
2	70-79	3655678	Pfizer/BioNTech
3	60-69	3674368	Pfizer/BioNTech
4	50-59	3284392	Pfizer/BioNTech
5	40-49	1734724	Pfizer/BioNTech
6	30-39	1144658	Pfizer/BioNTech
7	20-29	888774	Pfizer/BioNTech
8	16-19	80181	Pfizer/BioNTech

Figura 2.10: Principale fornitore per fascia anagrafica

2.4.2.4 Dosi Somministrate per Fascia Anagrafica ogni mese

La quinta view trasforma i dati in modo da restituire documenti contenenti il numero di dosi di vaccino di somministrate a ciascuna fascia anagrafica ogni mese.

Dosi somministrate per fascia anagrafica ogni mese

```
pipeline = [
    { "$group" : {
        "_id": {
            "anno" : {"$year": "$data_somministrazione"},
            "mese" : {"$month": "$data_somministrazione"},
            "fascia_anagrafica" : "$fascia_anagrafica"
        },
        "prima_dose": { "$sum": "$prima_dose" },
        "seconda_dose" : { "$sum": "$seconda_dose"}
    }},

    { "$addFields": {
        "totale_dosi": { "$sum": ["$prima_dose", "$seconda_dose"]}
    }},

    { "$project": {
        "anno": "$_id.anno",
        "mese": "$_id.mese",
        "fascia_anagrafica": "$_id.fascia_anagrafica",
        "totale_dosi": 1,
        "_id": 0
    }},

    { "$sort" : {
        "anno": -1,
        "mese": -1,
        "totale_dosi": -1
    }}
]

db.create_collection(
    'dati_vaccini_fasce_mensili',
    viewOn='dati_vaccini',
    pipeline=pipeline
)
```

Per mostrare la struttura di un documento della vista di seguito se ne riporta un esempio.

```
totale_dosi: 3309114
anno: 2021
mese: 5
fascia_anagrafica: "60-69"
```

Figura 2.11: Schema documenti vista

A partire da questa vista, l'utente può effettuare query filtrando o aggregando i dati ulteriormente sui campi di interesse. Ad esempio, è possibile individuare la fascia anagrafica a cui sono stati somministrati più vaccini ogni mese.

Fascia anagrafica più vaccinata ogni mese

```
result = db.dati_vaccini_fasce_mensili.aggregate([
    { "$group" : {
        "_id": {
            "anno": "$anno",
            "mese": "$mese"
        },
        "totale_dosi": {"$first": "$totale_dosi"},
        "fascia_anagrafica": {"$first": "$fascia_anagrafica" }
    }},

    { "$sort" : {
        "_id": -1,
    }},
])

result_list = list(result)

df = pd.DataFrame(result_list)
df
```

	_id	totale_dosi	fascia_anagrafica
0	{'anno': 2021, 'mese': 5}	3309114	60-69
1	{'anno': 2021, 'mese': 4}	3411580	70-79
2	{'anno': 2021, 'mese': 3}	2425520	80-89
3	{'anno': 2021, 'mese': 2}	613806	80-89
4	{'anno': 2021, 'mese': 1}	523279	50-59
5	{'anno': 2020, 'mese': 12}	11268	50-59

Figura 2.12: Fascia anagrafica più vaccinata ogni mese

2.5 Creazione degli Indici

Dal momento che si prevede un elevato uso di query sulla collection `dati_covid` che coinvolgono filtri e aggregazioni sul campo `data`, si è ritenuto opportuno definire un indice su tale campo. Questo consente di migliorare le prestazioni di query di questo tipo, in termini di tempo di esecuzione.

Creazione dell'indice

```
collection_covid.create_index([("data", ASCENDING)],
                              name="date_index",
                              background=True,
                              unique=False,
                              sparse=False)
```

2.5.1 Attuali Positivi e Terapie Intensive per Regione

Di seguito si propone un esempio di query che richiede un filtro sulla data, eseguita prima e dopo la creazione dell'indice. Nello specifico, la query restituisce il numero di attuali positivi e ricoverati in terapia intensiva in ogni regione nella data corrente.

Attuali positivi e terapie intensive per regione

```
days_ago = 1
date = datetime.today() - timedelta(days=days_ago)

start = date.replace(hour=0, minute=0, second=0)
end = date.replace(hour=23, minute=59, second=59)

filter = {
    'data': {
        '$gt': start,
        '$lt': end
    }
}

project = {
    'data': 1,
    'denominazione_regione': 1,
    'totale_positivi': 1,
    'terapia_intensiva': 1,
    '_id': 0
}

sort = list({
    'totale_positivi': -1
}.items())

result = collection_covid.find(
    filter=filter,
    projection=project,
    sort=sort
)

result_list = list(result)

df = pd.DataFrame(result_list)

explain_no_ind = result.explain()
explain_no_ind = explain_no_ind['executionStats']

print('executionTimeMillis:', explain_no_ind['executionTimeMillis'])
print('nReturned:', explain_no_ind['nReturned'])
print('totalDocsExamined:', explain_no_ind['totalDocsExamined'])
```

	data	denominazione_regione	terapia_intensiva	totale_positivi
0	2021-05-22 17:00:00	Campania	82	72531
1	2021-05-22 17:00:00	Lombardia	298	35416
2	2021-05-22 17:00:00	Puglia	100	32794
3	2021-05-22 17:00:00	Lazio	195	28334
4	2021-05-22 17:00:00	Emilia-Romagna	126	19338
5	2021-05-22 17:00:00	Sardegna	32	13484
6	2021-05-22 17:00:00	Sicilia	104	13267
7	2021-05-22 17:00:00	Toscana	144	11552
8	2021-05-22 17:00:00	Veneto	72	11296
9	2021-05-22 17:00:00	Calabria	24	11173
10	2021-05-22 17:00:00	Piemonte	105	8209
11	2021-05-22 17:00:00	Abruzzo	17	5901
12	2021-05-22 17:00:00	Friuli Venezia Giulia	10	5288
13	2021-05-22 17:00:00	Basilicata	5	4542
14	2021-05-22 17:00:00	Marche	39	4530
15	2021-05-22 17:00:00	Liguria	43	2048
16	2021-05-22 17:00:00	Umbria	12	2030
17	2021-05-22 17:00:00	P.A. Bolzano	4	862
18	2021-05-22 17:00:00	P.A. Trento	13	611
19	2021-05-22 17:00:00	Valle d'Aosta	0	315
20	2021-05-22 17:00:00	Molise	5	223

Figura 2.13: Attuali positivi e terapie intensive per regione

La query in assenza di indice richiede di consultare tutti i documenti della collection e, di conseguenza, viene eseguita in un tempo pari a 16ms.

```
executionTimeMillis: 16
nReturned: 21
totalDocsExamined: 9534
```

Figura 2.14: Statistiche query senza indice

A questo punto, si verifica l'esecuzione della stessa query in presenza dell'indice. In questo caso, la query richiede di consultare solo 21 documenti, quelli relativi alla data corrente, per cui viene eseguita in meno di 1ms.

```
executionTimeMillis: 0
nReturned: 21
totalDocsExamined: 21
indexName: date_index
```

Figura 2.15: Statistiche query con indice

2.5.2 Nuovi Casi e Tamponi nell'Ultima Settimana per Regione

Infine, si propone un altro esempio di query che beneficia della creazione dell'indice, in termini di riduzione del tempo di esecuzione. Nello specifico, la query restituisce il numero di nuovi casi e tamponi effettuati in ogni regione nell'ultima settimana.

Nuovi casi e tamponi nell'ultima settimana per regione

```
weeks_ago = 1
date = datetime.today() - timedelta(weeks=weeks_ago)

result = collection_covid.aggregate([
    { "$match": { "data": { "$gte": date } } },

    { "$group": {
        "_id": "$denominazione_regione",
        "nuovi_casi": { "$sum": "$nuovi_positivi" },
        "tamponi" : { "$sum": "$tamponi"}
    }},

    { "$sort" : { "nuovi_casi": -1 } }
])

result_list = list(result)

df = pd.DataFrame(result_list)
df
```

	_id	nuovi_casi	tamponi
0	Lombardia	5683	71756835
1	Campania	4620	33139697
2	Lazio	3302	46866103
3	Sicilia	3004	30165040
4	Toscana	2829	31683802
5	Piemonte	2781	30236080
6	Emilia-Romagna	2669	42205861
7	Puglia	2539	16840095
8	Veneto	1890	53830701
9	Calabria	1308	5847161
10	Marche	1028	8403926
11	Liguria	537	10878639
12	Abruzzo	482	10692541
13	Basilicata	442	2532811
14	P.A. Bolzano	375	10422129
15	Umbria	316	8968493
16	Sardegna	310	8808079
17	P.A. Trento	295	5610490
18	Friuli Venezia Giulia	265	13739567
19	Valle d'Aosta	120	873052
20	Molise	47	1573657

Figura 2.16: Nuovi casi e tamponi nell'ultima settimana per regione

È bene osservare che in questo secondo esempio la query non è una semplice `find` ma una più complessa operazione di `aggregate`. Tuttavia, anche le aggregation pipeline traggono beneficio dalla definizione dell'indice.

```
'indexName': 'date_index'
```

Figura 2.17: Indice query

Bibliografia

[1] d'Andrea Fabio; Di Chiara Guido. *Repository GitHub - Elaborazione Spark Dati Covid-19*. URL: https://github.com/fabiod20/homework-BDABI/tree/main/homework3-MongoDB.

[2] Dipartimento della Protezione Civile. *Repository GitHub - Dati Covid-19*. URL: https://github.com/pcm-dpc/COVID-19.

[3] Dipartimento della Protezione Civile. *Repository GitHub - Formato Dati Covid-19*. URL: https://github.com/pcm-dpc/COVID-19/blob/master/dati-andamento-covid19-italia.md.

[4] Dipartimento per la Trasformazione Digitale. *Repository GitHub - Dati Vaccini*. URL: https://github.com/italia/covid19-opendata-vaccini.

[5] Dipartimento per la Trasformazione Digitale. *Repository GitHub - Formato Dati Vaccini*. URL: https://github.com/italia/covid19-opendata-vaccini/blob/master/README.md.