Corso di Laurea Magistrale
in
Ingegneria Informatica

Big Data Analytics and Business Intelligence
Homework 2

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 secondo 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 framework *Spark* per l'elaborazione distribuita di dati relativi alla diffusione del Covid-19 in Italia.

1.1 Business Understanding

Tale elaborato ha l'obiettivo di processare 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 delle elaborazioni effettuate è quello di mettere in evidenza informazioni esplicite, correlando dati provenienti da diverse sorgenti. Le informazioni prodotte possono essere quindi visualizzate facilmente da un tradizionale tool di Business Intelligence. Il vantaggio di effettuare l'elaborazione a monte risiede nella possibilità di utilizzare uno strumento di calcolo distribuito, che consente di processare grandi moli di dati in maniera efficiente.

L'elaborazione è stata condotta mediante **Apache Spark**, noto framework impiegato per il calcolo distribuito, ampiamente utilizzato in applicazioni di Big Data. Nello specifico, è stato utilizzato **PySpark**, APIs Python che permette di scrivere applicazioni Spark. Il modulo principalmente utilizzato è **Spark SQL**, il quale permette di processare dati strutturati in modo distribuito, fornendo un'astrazione di alto livello chiamata *DataFrame*.

La potenza di calcolo è stata infine fornita dalla piattaforma cloud **Databricks** in versione community. Essa, fondata dagli stessi creatori di Apache Spark, rappresenta un ambiente web-based che fornisce gestione automatica di cluster per l'esecuzione di calcolo distribuito ed un ambiente integrato di sviluppo di notebook Python.

Di conseguenza, il codice relativo alle elaborazioni effettuate è contenuto in un file *.ipynb*, reperibile su un repository GitHub [1], ed il cui output sono una serie di tabelle.

1.2 Data Understanding

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

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 e quelle relative al piano vaccinale. Le fonti da cui sono stati raccolti i dati sono due repository Git-Hub pubblicati rispettivamente dal *Dipartimento della Protezione Civile* [2] e dal *Dipartimento per la Trasformazione Digitale* [3]. In particolare i file considerati nell'elaborazione sono:

- *dpc-covid19-ita-regioni.csv*: 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.

- *somministrazioni-vaccini-latest.csv*: 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 rispettivamente [4] e [5].

1.2.2 Dati Generali

La seconda categoria di dati comprende alcune informazioni generali utili per contestualizzare i dati della pandemia, tra cui dati relativi alla popolazione delle regioni italiane, divisa per diverse fasce anagrafiche. Tali informazioni provengono dall'*Istituto nazionale di statistica* (ISTAT).

Capitolo 2

Preprocessing

Nel capitolo corrente sono riportate tutte le operazioni di preprocessing effettuate sui dati prima dell'effettiva elaborazione.

2.1 Download e Lettura dei Dati

Dal momento che i dati vengono aggiornati con una certa frequenza, al fine di accedere sempre ad una versione aggiornata, per prima cosa si scaricano i file presentati nel Paragrafo 1.2. Per ciascun file sono stati quindi definiti l'URL di origine e il path locale in cui il file sarà memorizzato.

Download dei Dati

```
local_path_popolazione_regioni = "dbfs:/FileStore/shared_uploads/ \
                                  user/popolazione_regioni.csv"
local_path_dati_covid_regioni = "dbfs:/FileStore/shared_uploads/ \
                                  user/dati_covid_regioni.csv"
local_path_dati_vaccini_somministrazioni = "dbfs:/FileStore/shared_uploads/ \
                                  user/dati_vaccini_somministrazioni.csv"

popolazione_regioni_URL = "https://raw.githubusercontent.com/pcm-dpc/COVID-19/master/ \
              dati-statistici-riferimento/popolazione-istat-regione-range.csv"
dati_covid_regioni_URL = "https://raw.githubusercontent.com/pcm-dpc/COVID-19/master/ \
           dati-regioni/dpc-covid19-ita-regioni.csv"
dati_vaccini_somministrazioni_URL = "https://raw.githubusercontent.com/italia/ \
           covid19-opendata-vaccini/master/dati/somministrazioni-vaccini-latest.csv"

dbutils.fs.put(local_path_popolazione_regioni, \
               requests.get(popolazione_regioni_URL).text, True)
dbutils.fs.put(local_path_dati_covid_regioni, \
               requests.get(dati_covid_regioni_URL).text, True)
dbutils.fs.put(local_path_dati_vaccini_somministrazioni, \
               requests.get(dati_vaccini_somministrazioni_URL).text, True)
```

Una volta scaricati, i dati possono essere caricati all'interno di apposite strutture dati, dette *DataFrame*.

Lettura dei Dati

```
popolazione_regioni = spark.read.option("inferSchema", "true") \
                                .option("header", "true") \
                                .csv(local_path_popolazione_regioni)
dati_covid_regioni = spark.read.option("inferSchema", "true") \
                                .option("header", "true") \
                                .csv(local_path_dati_covid_regioni)
dati_vaccini_somministrazioni = spark.read.option("inferSchema", "true") \
                                          .option("header", "true") \
                                          .csv(local_path_dati_vaccini_somministrazioni)
```

A questo punto, per ciascun DataFrame sono eseguite diverse operazioni di preprocessing.

2.2 Popolazione Regioni

Di seguito sono elencate le operazione di preprocessing eseguite sul DataFrame contenente i dati relativi alla popolazione delle diverse regioni italiane.

2.2.1 Eliminazione Colonne

Per prima cosa si eliminano le colonne ritenute non di interesse per la successiva elaborazione.

Eliminazione Colonnne

```
popolazione_regioni = popolazione_regioni.drop('codice_regione', 'codice_nuts_1',
                'descrizione_nuts_1', 'sigla_regione', 'latitudine_regione',
                'longitudine_regione', 'totale_genere_maschile', 'totale_genere_femminile')
```

2.2.2 Formato Colonne

Si verifica quindi che il formato delle colonne sia corretto.

Formato Colonne

```
popolazione_regioni.dtypes
```

2.2.3 Colonne Categoriche

Si verifica la correttezza dei valori relativi alle colonne categoriche.

Colonne Categoriche

```
num_regioni = popolazione_regioni.groupBy('denominazione_regione') \
                .count() \
                .count()
print('Il numero di regioni è:', num_regioni)
print()
popolazione_regioni.groupBy('codice_nuts_2', 'denominazione_regione') \
                    .count() \
                    .drop('count') \
                    .orderBy('codice_nuts_2') \
                    .show(num_regioni)

popolazione_regioni.groupBy('range_eta') \
                    .count() \
                    .orderBy('range_eta') \
                    .show()
```

```
Il numero di regioni è: 21

+-------------+--------------------+
|codice_nuts_2|denominazione_regione|
+-------------+--------------------+
|         ITC1|            Piemonte|
|         ITC2|        Valle d'Aosta|
|         ITC3|             Liguria|
|         ITC4|           Lombardia|
|         ITF1|            Abruzzo|
|         ITF2|             Molise|
|         ITF3|            Campania|
|         ITF4|             Puglia|
|         ITF5|          Basilicata|
|         ITF6|            Calabria|
|         ITG1|             Sicilia|
|         ITG2|            Sardegna|
|         ITH1|             Bolzano|
|         ITH2|              Trento|
|         ITH3|              Veneto|
|         ITH4| Friuli Venezia Gi...|
|         ITH5|       Emilia-Romagna|
|         ITI1|             Toscana|
|         ITI2|              Umbria|
|         ITI3|              Marche|
|         ITI4|               Lazio|
+-------------+--------------------+
```

(a) Codice Nuts e Denominazione Regione

```
+---------+-----+
|range_eta|count|
+---------+-----+
|     0-15|   21|
|    16-19|   21|
|    20-29|   21|
|    30-39|   21|
|    40-49|   21|
|    50-59|   21|
|    60-69|   21|
|    70-79|   21|
|    80-89|   21|
|      90+|   21|
+---------+-----+
```

(b) Range età

Figura 2.1: Colonne Categoriche

2.2.4 Colonne Numeriche

Si verifica la correttezza dei valori relativi alle colonne numeriche.

Colonne Numeriche

```
popolazione_regioni.describe('totale_generale') \
                   .show()
```

```
+-------+-----------------+
|summary|  totale_generale|
+-------+-----------------+
|  count|              210|
|   mean|284007.0857142857|
| stddev|303588.4857882117|
|    min|             1764|
|    max|          1592109|
+-------+-----------------+
```

Figura 2.2: Colonne Numeriche

2.2.5 Valori Mancanti

Infine, si verifica la presenza di valori mancanti.

Valori Mancanti

```
popolazione_regioni.select([count(when(isnull(c), c)).alias(c) \
                            for c in popolazione_regioni.columns]).show()
```

```
+-------------+--------------------+---------+---------------+
|codice_nuts_2|denominazione_regione|range_eta|totale_generale|
+-------------+--------------------+---------+---------------+
|            0|                   0|        0|              0|
+-------------+--------------------+---------+---------------+
```

Figura 2.3: Valori Mancanti

2.3 Dati Covid 19

Di seguito sono elencate le operazione di preprocessing eseguite sul DataFrame contenente i dati relativi alla diffusione del Covid-19 nelle diverse regioni italiane.

2.3.1 Eliminazione Colonne

Per prima cosa si eliminano le colonne ritenute non di interesse per la successiva elaborazione.

Eliminazione Colonnne

```
dati_covid_regioni = dati_covid_regioni.drop('stato', 'codice_nuts_1', 'codice_regione',
            'lat', 'long', 'note_test', 'note_casi', 'note', 'casi_da_sospetto_diagnostico',
            'casi_da_screening','casi_testati', 'ingressi_terapia_intensiva',
            'totale_positivi_test_molecolare', 'totale_positivi_test_antigenico_rapido',
            'tamponi_test_molecolare', 'tamponi_test_antigenico_rapido',
            'isolamento_domiciliare', 'variazione_totale_positivi',
            'dimessi_guariti', 'totale_casi')
```

2.3.2 Formato Colonne

Si verifica quindi che il formato delle colonne sia corretto.

Formato Colonne

```
dati_covid_regioni.dtypes

dati_covid_regioni = dati_covid_regioni.withColumn('data',
                        date_format(dati_covid_regioni.data, 'yyyy-MM-dd').cast('date'))
```

In particolare, si osserva che il formato della colonna data è di tipo `timestamp` e si realizza un cast al tipo `date`, impostando inoltre la formattazione desiderata.

2.3.3 Colonne Categoriche

Si verifica la correttezza dei valori relativi alle colonne categoriche.

Colonne Categoriche

```
num_regioni = dati_covid_regioni.groupBy('denominazione_regione') \
                .count() \
                .count()

print('Il numero di regioni è:', num_regioni)
print()
dati_covid_regioni.groupBy('codice_nuts_2', 'denominazione_regione') \
                    .count() \
                    .drop('count') \
                    .orderBy('codice_nuts_2') \
                    .show(5)
```

```
Il numero di regioni è: 21

+------------+--------------------+
|codice_nuts_2|denominazione_regione|
+------------+--------------------+
|        null|             Abruzzo|
|        null|           Lombardia|
|        null|            Piemonte|
|        null|      Emilia-Romagna|
|        null|            Calabria|
+------------+--------------------+
only showing top 5 rows
```

Figura 2.4: Colonne Categoriche

Con riferimento ai dati delle regioni è possibile fare alcune osservazioni:

- Per alcune regioni, nello specifico Trento e Bolzano, il valore della colonna `denominazione_regione` non coincide con il valore della colonna omonima della tabella `popolazione_regioni`.
- Per alcune entry mancano i valori della colonna `codice_nuts_2`.
- Per alcune entry i valori della colonna `codice_nuts_2` sono errati. Tale problema è dovuto alla lettura scorretta del file, causata dalla presenza di virgole nelle note, che vengono erroneamente considerate come separatore di colonna.

Per verificare che effettivamente la denominazione di alcune regioni non coincide, si effettua un *anti-join* tre le tabelle `popolazione_regioni` e `dati_covid_regioni` sulla colonna `denominazione_regione`, operazione che restituisce le entry per cui i valori della colonna indicata non coincidono. A questo punto è possibile sostituire opportunamente le denominazioni delle regioni.

Correzione Denominazione Regioni

```
join1 = popolazione_regioni.join(dati_covid_regioni,
                                 on='denominazione_regione', how='anti') \
                    .groupBy('codice_nuts_2', 'denominazione_regione') \
                    .count() \
                    .drop('count') \
                    .withColumnRenamed('denominazione_regione', \
                            'popolazione_regioni_denominazione_regione')

join2 = dati_covid_regioni.join(popolazione_regioni,
                                 on='denominazione_regione', how='anti') \
                    .groupBy('codice_nuts_2', 'denominazione_regione') \
                    .count() \
                    .drop('count') \
```

```
                    .withColumnRenamed('denominazione_regione', \
                            'dati_covid_regioni_denominazione_regione')

join1.join(join2, on='codice_nuts_2').show()

dati_covid_regioni = dati_covid_regioni.replace('P.A. Trento',
                        value='Trento', subset='denominazione_regione')
dati_covid_regioni = dati_covid_regioni.replace('P.A. Bolzano',
                        value='Bolzano', subset='denominazione_regione')
```

```
+-------------+----------------------------------------+----------------------------------------+
|codice_nuts_2|popolazione_regioni_denominazione_regione|dati_covid_regioni_denominazione_regione|
+-------------+----------------------------------------+----------------------------------------+
|         ITH1|                                Bolzano|                            P.A. Bolzano|
|         ITH2|                                 Trento|                             P.A. Trento|
+-------------+----------------------------------------+----------------------------------------+
```

Figura 2.5: Correzione Denominazione Regioni

Per risolvere i problemi legati ai codici NUTS 2, dopo aver eliminato la colonna scorretta dalla tabella `dati_covid_regioni`, si effettua un *join* tra `popolazione_regioni` e `dati_covid_regioni` sulla colonna `denominazione_regione`. In questo modo, ad ogni regione viene associato il codice NUTS 2 corretto, fornito dalla tabella `popolazione_regioni`.

Correzione Codici NUTS 2

```
dati_covid_regioni = dati_covid_regioni.drop('codice_nuts_2')

nuts_2 = popolazione_regioni.groupBy('codice_nuts_2', 'denominazione_regione') \
                            .count() \
                            .drop('count')

dati_covid_regioni = dati_covid_regioni.join(nuts_2, on='denominazione_regione')

dati_covid_regioni.groupBy('codice_nuts_2', 'denominazione_regione') \
                  .count() \
                  .drop('count') \
                  .orderBy('codice_nuts_2').show(num_regioni)
```

2.3.4 Colonne Numeriche

Si verifica la correttezza dei valori relativi alle colonne numeriche.

Colonne Numeriche

```
dati_covid_regioni.describe(['ricoverati_con_sintomi', 'terapia_intensiva',
                    'totale_ospedalizzati', 'totale_positivi', 'nuovi_positivi',
                    'deceduti', 'tamponi']).show()
```

```
+-------+---------------------+-----------------+--------------------+------------------+-----------------+------------------+------------------+
|summary|ricoverati_con_sintomi|terapia_intensiva|totale_ospedalizzati|   totale_positivi|   nuovi_positivi|          deceduti|           tamponi|
+-------+---------------------+-----------------+--------------------+------------------+-----------------+------------------+------------------+
|  count|                 9072|             9072|                9072|              9072|             9072|              9072|              9072|
|   mean|  712.4712301587301|81.71847442680776|   794.1897045855379|12468.846560846561|442.7234347442681|2470.1935626102295| 848671.4441137566|
| stddev|  1289.9373804295726|143.0112526568166|  1428.1841519339673| 22237.61213787529| 853.939588083822| 4695.206250550821|1338368.1129099485|
|    min|                    0|                0|                   0|                 0|             -229|                 0|                 0|
|    max|                12077|             1381|               13328|            164406|            11489|             32870|           9475296|
+-------+---------------------+-----------------+--------------------+------------------+-----------------+------------------+------------------+
```

Figura 2.6: Colonne Numeriche

Dalla Figura 2.6 è possibile osservare che il valore minimo della colonna `nuovi_positivi` è negativo. Tale colonna rappresenta il numero di nuovi positivi al Covid-19 che si sono verificati in un dato giorno, per cui i suoi valori dovrebbero essere sempre non negativi.

Incongruenza nei Nuovi Positivi

```
dati_covid_regioni.filter(condition='nuovi_positivi < 0').show()
```

```
+--------------------+----------+---------------------+----------------+--------------------+---------------+---------------+--------+-------+-------------+
|denominazione_regione|      data|ricoverati_con_sintomi|terapia_intensiva|totale_ospedalizzati|totale_positivi|nuovi_positivi|deceduti|tamponi|codice_nuts_2|
+--------------------+----------+---------------------+----------------+--------------------+---------------+---------------+--------+-------+-------------+
|            Piemonte|2020-02-27|                    2|               0|                   2|              2|             -1|       0|    156|         ITC1|
|             Liguria|2020-03-01|                   12|               1|                  13|             21|            -17|       0|    121|         ITC3|
|             Liguria|2020-03-02|                   12|               1|                  13|             18|             -3|       0|    121|         ITC3|
|             Sicilia|2020-03-02|                    2|               0|                   2|              5|             -2|       0|    307|         ITG1|
|            Piemonte|2020-03-09|                  222|              50|                 272|            337|            -10|      13|   1681|         ITC1|
|            Calabria|2020-04-17|                  154|               7|                 161|            819|            -18|      73|  21657|         ITF6|
|           Sardegna|2020-05-04|                   91|               9|                 100|            653|             -2|     119|  28052|         ITG2|
|          Basilicata|2020-05-07|                   48|               2|                  50|            155|            -16|      26|  16272|         ITF5|
|          Basilicata|2020-05-08|                   48|               2|                  50|            152|             -1|      26|  16777|         ITF5|
|              Marche|2020-05-19|                  144|              17|                 161|           2128|             -3|     986|  89985|         ITI3|
|           Sardegna|2020-05-25|                   49|               3|                  52|            231|             -2|     129|  51073|         ITG2|
|           Sardegna|2020-06-09|                    9|               1|                  10|             54|             -1|     131|  64272|         ITG2|
|            Campania|2020-06-12|                   61|               2|                  63|            346|           -229|     430| 230551|         ITF3|
+--------------------+----------+---------------------+----------------+--------------------+---------------+---------------+--------+-------+-------------+
```

Figura 2.7: Incongruenza nei Nuovi Positivi

Il fatto che in alcuni casi la colonna `nuovi_positivi` assume valori negativi potrebbe essere dovuto ad errori di rilevamento di casi Covid-19 avvenuti nei giorni precedenti. Valori negativi possono essere interpretati come una sorta di correzione di tali errori, specie se si osserva che le entry fanno riferimento principalmente ai primi periodi della pandemia. Per questo motivo, non avendo ulteriori informazioni a riguardo, si è scelto di lasciare inalterate tali entry.

2.3.5 Valori Mancanti

Infine, si verifica che non vi sia la presenza di valori mancanti.

Valori Mancanti

```
dati_covid_regioni.select([count(when(isnull(c), c)).alias(c) \
                           for c in dati_covid_regioni.drop('data').columns]).show()
```

Si verifica inoltre che non vi sia la presenza di valori mancanti nella serie temporale che va dal primo all'ultimo giorno della colonna `data`.

Valori Mancanti nella Serie Temporale

```
min_data = dati_covid_regioni.agg(min('data').alias('minimo'))
max_data = dati_covid_regioni.agg(max('data').alias('massimo'))

date = min_data.join(max_data)

time_series = date.withColumn('data',
                    explode(expr('sequence(minimo, massimo, interval 1 day)'))) \
                  .select('data')

date_covid = dati_covid_regioni.select('data') \
                               .distinct() \
                               .orderBy('data')

date_covid.join(time_series, on='data', how='anti').show()
```

2.4 Dati Vaccini

Di seguito sono elencate le operazioni di preprocessing eseguite sul DataFrame contenente i dati relativi alle vaccinazioni eseguite nelle diverse regioni italiane.

2.4.1 Eliminazione Colonne

Per prima cosa si eliminano le colonne ritenute non di interesse per la successiva elaborazione.

Eliminazione Colonnne

```
dati_vaccini_somministrazioni = dati_vaccini_somministrazioni.drop('area',
        'codice_regione_ISTAT', 'sesso_maschile', 'sesso_femminile' 'codice_NUTS1')
```

2.4.2 Formato Colonne

Si verifica quindi che il formato delle colonne sia corretto.

Formato Colonne

```
dati_vaccini_somministrazioni.dtypes

dati_vaccini_somministrazioni = \
        dati_vaccini_somministrazioni.withColumn('data_somministrazione',
        date_format(dati_vaccini_somministrazioni.data_somministrazione, 'yyyy-MM-dd') \
                                        .cast('date'))
```

In particolare, si osserva che il formato della colonna data è di tipo `timestamp` e si realizza un cast al tipo `date`, impostando inoltre la formattazione desiderata.

2.4.3 Colonne Categoriche

Si verifica la correttezza dei valori relativi alle colonne categoriche.

Colonne Categoriche

```
num_regioni = dati_vaccini_somministrazioni.groupBy('nome_area') \
                                           .count() \
                                           .count()
print('Il numero di regioni è:', num_regioni)
print()
dati_vaccini_somministrazioni.groupBy('codice_NUTS2', 'nome_area') \
                             .count() \
                             .drop('count') \
                             .orderBy('codice_NUTS2') \
                             .show(num_regioni)

dati_vaccini_somministrazioni.groupBy('fascia_anagrafica') \
                     .count() \
                     .drop('count') \
                     .orderBy('fascia_anagrafica') \
                     .show()
```

```
Il numero di regioni è: 21

+------------+--------------------+
|codice_NUTS2|          nome_area|
+------------+--------------------+
|        ITC1|            Piemonte|
|        ITC2|Valle d'Aosta / V...|
|        ITC3|             Liguria|
|        ITC4|           Lombardia|
|        ITF1|             Abruzzo|
|        ITF2|              Molise|
|        ITF3|            Campania|
|        ITF4|              Puglia|
|        ITF5|          Basilicata|
|        ITF6|            Calabria|
|        ITG1|             Sicilia|
|        ITG2|            Sardegna|
|        ITH1|Provincia Autonom...|
|        ITH2|Provincia Autonom...|
|        ITH3|              Veneto|
|        ITH4|Friuli-Venezia Gi...|
|        ITH5|      Emilia-Romagna|
|        ITI1|             Toscana|
|        ITI2|              Umbria|
|        ITI3|              Marche|
|        ITI4|               Lazio|
+------------+--------------------+
```

(a) Codice Nuts e Denominazione Regione

```
+-----------------+
|fascia_anagrafica|
+-----------------+
|            16-19|
|            20-29|
|            30-39|
|            40-49|
|            50-59|
|            60-69|
|            70-79|
|            80-89|
|              90+|
+-----------------+
```

(b) Fascia Anagrafica

Figura 2.8: Colonne Categoriche

Con riferimento alle informazioni relative alle regioni è possibile effettuare alcune osservazioni:

- Per alcune regioni, nello specifico Trento, Bolzano, Valle d'Aosta e Friuli Venezia Giulia, il valore della colonna `nome_area` non coincide con il valore della colonna `denominazione_regione` della tabella `popolazione_regioni`.
- Nella colonna `fascia_anagrafica` manca il valore `0-15`, presente nella colonna `range_eta` della tabella `popolazione_regioni`.

Per risolvere il problema relativo alla denominazione delle regioni si ripetono le operazioni viste nel Paragrafo 2.3.3.

Correzione Denominazione Regioni

```
join1 = popolazione_regioni.join(dati_vaccini_somministrazioni,
                                 popolazione_regioni.denominazione_regione ==
                                 dati_vaccini_somministrazioni.nome_area,
                                 how='anti') \
                           .groupBy('codice_nuts_2', 'denominazione_regione') \
                           .count() \
                           .drop('count')

join2 = dati_vaccini_somministrazioni.join(popolazione_regioni,
                                    popolazione_regioni.denominazione_regione ==
                                    dati_vaccini_somministrazioni.nome_area,
                                    how='anti') \
                                   .groupBy('codice_NUTS2', 'nome_area') \
                                   .count() \
                                   .drop('count')

join1.join(join2, popolazione_regioni.codice_nuts_2 ==
                  dati_vaccini_somministrazioni.codice_NUTS2)
     .show()

dati_vaccini_somministrazioni = dati_vaccini_somministrazioni.replace(
                                'Provincia Autonoma Trento', value='Trento',
                                subset='nome_area')
dati_vaccini_somministrazioni = dati_vaccini_somministrazioni.replace(
                                'Provincia Autonoma Bolzano / Bozen', value='Bolzano',
                                subset='nome_area')
dati_vaccini_somministrazioni = dati_vaccini_somministrazioni.replace(
                                'Friuli-Venezia Giulia', value='Friuli Venezia Giulia',
                                subset='nome_area')
dati_vaccini_somministrazioni = dati_vaccini_somministrazioni.replace(
                                'Valle d\'Aosta / Vallée d\'Aoste', value='Valle d\'Aosta',
                                subset='nome_area')
```

```
+-------------+--------------------+------------+--------------------+
|codice_nuts_2|denominazione_regione|codice_NUTS2|          nome_area|
+-------------+--------------------+------------+--------------------+
|         ITC2|        Valle d'Aosta|        ITC2|Valle d'Aosta / V...|
|         ITH2|              Trento|        ITH2|Provincia Autonom...|
|         ITH4| Friuli Venezia Gi...|        ITH4|Friuli-Venezia Gi...|
|         ITH1|             Bolzano|        ITH1|Provincia Autonom...|
+-------------+--------------------+------------+--------------------+
```

Figura 2.9: Correzione Denominazione Regione

2.4.4 Colonne Numeriche

Si verifica la correttezza dei valori relativi alle colonne numeriche.

Colonne Numeriche

```
dati_vaccini_somministrazioni.describe(['prima_dose', 'seconda_dose']) \
                             .show()
```

```
+-------+-----------------+-----------------+
|summary|       prima_dose|      seconda_dose|
+-------+-----------------+-----------------+
|  count|            42748|            42748|
|   mean|328.72164779638814|137.9450266791428|
| stddev|1000.9215678340463| 558.9414535757198|
|    min|                0|                0|
|    max|            29498|            19243|
+-------+-----------------+-----------------+
```

```
+-------+----------------------------------------+-----------------------------+------------------+------------------+------------------+
|summary|categoria_operatori_sanitari_sociosanitari|categoria_personale_non_sanitario|categoria_ospiti_rsa|  categoria_60_69|  categoria_70_79|
+-------+----------------------------------------+-----------------------------+------------------+------------------+------------------+
|  count|                                   42748|                        42748|             42748|             42748|             42748|
|   mean|                      75.81898568354075|             20.82904463366707|  15.316506035370075| 25.2064424066623|69.76822307476372|
| stddev|                      226.9536664572629|             66.15868493516085|  61.102737050086844|325.9967518509442|679.4798418900963|
|    min|                                     0|                            0|                 0|                 0|                 0|
|    max|                                  4662|                         2303|              2317|             20021|             28033|
+-------+----------------------------------------+-----------------------------+------------------+------------------+------------------+
```

```
+-------+------------------+--------------------+-------------------------------+--------------------------+------------------+
|summary|  categoria_over80|categoria_forze_armate|categoria_personale_scolastico|categoria_soggetti_fragili|  categoria_altro|
+-------+------------------+--------------------+-------------------------------+--------------------------+------------------+
|  count|             42748|               42748|                          42748|                     42748|             42748|
|   mean|144.24272480583886|   7.601548610461308|             27.258023767193787|        72.07352390755123|8.551651539253298|
| stddev| 911.4114589394183|   42.6170651430661|              151.6780634008098|        312.05108674408626|39.96799868320293|
|    min|                 0|                   0|                              0|                         0|                 0|
|    max|             31064|                1214|                           3583|                     10910|              2008|
+-------+------------------+--------------------+-------------------------------+--------------------------+------------------+
```

Figura 2.10: Colonne Numeriche

2.4.5 Valori Mancanti

Infine, si verifica che non vi sia la presenza di valori mancanti.

Valori Mancanti

```
dati_vaccini_somministrazioni.select([count(when(isnull(c), c)).alias(c)
                    for c in dati_vaccini_somministrazioni.drop('data').columns]).show()
```

Si verifica inoltre che non vi sia la presenza di valori mancanti nella serie temporale che va dal primo all'ultimo giorno della colonna `data_somministrazione`.

Valori Mancanti nella Serie Temporale

```
min_data = dati_vaccini_somministrazioni.agg(min('data_somministrazione').alias('minimo'))
max_data = dati_vaccini_somministrazioni.agg(max('data_somministrazione').alias('massimo'))

date = min_data.join(max_data)

time_series = date.withColumn('data_somministrazione',
                    explode(expr('sequence(minimo, massimo, interval 1 day)'))) \
                  .select('data_somministrazione')

date_covid = dati_vaccini_somministrazioni.select('data_somministrazione') \
                        .withColumn('data_somministrazione',
                          date_format(dati_vaccini_somministrazioni.data_somministrazione,
                          'yyyy-MM-dd').cast('date')) \
                        .distinct() \
                        .orderBy('data_somministrazione')

date_covid.join(time_series, on='data_somministrazione', how='anti') \
          .show()
```

Capitolo 3

Elaborazione

In questo capitolo si riportano le elaborazioni effettuate sui dati risultanti dalla precedente fase di preprocessing.

3.1 Dati Covid-19

3.1.1 Attuali Positivi per Regione

Si evidenzia il numero di persone attualmente positive al Covid-19 in ciascuna regione. Per fornire un'informazione più completa, il numero di attuali positivi viene contestualizzato rispetto al numero di abitanti di ogni regione, calcolando il numero di positivi ogni 100 mila abitanti.

Attuali Positivi per Regione

```
attuali_positivi_regione = dati_covid_regioni
        .where(col("data") >= current_date() - expr("INTERVAL 1 days")) \
        .select('denominazione_regione', 'totale_positivi') \
        .withColumnRenamed('denominazione_regione', 'regione') \
        .withColumnRenamed('totale_positivi', 'attuali_positivi')

popolazione = popolazione_regioni.groupBy('denominazione_regione') \
                                 .sum('totale_generale')

cond = [popolazione.denominazione_regione == attuali_positivi_regione.regione]
attuali_positivi_regione = attuali_positivi_regione.join(popolazione, cond, how='left') \
                           .drop('denominazione_regione') \
                           .withColumnRenamed('sum(totale_generale)', 'popolazione') \
                           .orderBy('regione')

attuali_positivi_regione = attuali_positivi_regione \
        .withColumn('positivi_per_100k', (attuali_positivi_regione.attuali_positivi /
            attuali_positivi_regione.popolazione * 100000).cast('int')) \
        .orderBy('positivi_per_100k', ascending=False)

attuali_positivi_regione.show(21)
```

regione	attuali_positivi	popolazione	positivi_per_100k
Campania	91388	5712143	1599
Puglia	47592	3953305	1203
Basilicata	6230	553254	1126
Sardegna	17093	1611621	1060
Emilia-Romagna	44356	4464119	993
Calabria	15214	1894110	803
Lazio	44696	5755700	776
Abruzzo	8383	1293941	647
Friuli Venezia Gi...	7578	1206216	628
Valle d'Aosta	749	125034	599
Toscana	20669	3692555	559
Lombardia	52485	10027602	523
Sicilia	24896	4875290	510
Veneto	21892	4879133	448
Marche	6202	1512672	410
Piemonte	15699	4311217	364
Liguria	5471	1524826	358
Umbria	2933	870165	337
Molise	629	300516	209
Trento	1076	545425	197
Bolzano	1039	532644	195

Figura 3.1: Attuali Positivi per Regione

3.1.2 Casi Covid-19 per Mese

Si evidenzia il numero di casi registrati ogni mese dall'inizio della pandemia. Per fornire un'informazione più completa, il numero di casi viene contestualizzato rispetto al numero di tamponi effettuati ogni mese, calcolando la percentuale di tamponi positivi.

Casi Covid-19 per Mese

```
casi_mensili = dati_covid_regioni.select(date_format('data','yyyy-MM').alias('mese'),
                    'nuovi_positivi') \
                .groupby('mese') \
                .sum('nuovi_positivi') \
                .withColumnRenamed('sum(nuovi_positivi)', 'casi')


tamponi_mensili = dati_covid_regioni.select(date_format('data','yyyy-MM').alias('mese'),
                                        'denominazione_regione', 'tamponi') \
                                    .groupby('mese', 'denominazione_regione') \
                                    .max('tamponi') \
                                    .groupby('mese') \
                                    .sum('max(tamponi)') \
                                    .withColumnRenamed('sum(max(tamponi))', 'tamponi')

my_window = Window.partitionBy().orderBy('mese')

tamponi_mensili = tamponi_mensili.withColumn('prev_value', lag(tamponi_mensili.tamponi) \
                                .over(my_window))
tamponi_mensili = tamponi_mensili.withColumn('diff', when(isnull(tamponi_mensili.tamponi -
tamponi_mensili.prev_value), tamponi_mensili.tamponi).otherwise(tamponi_mensili.tamponi -
                                    tamponi_mensili.prev_value)) \
                                .drop('tamponi', 'prev_value') \
                                .withColumnRenamed('diff', 'tamponi')

panoramica_mensile = casi_mensili.join(tamponi_mensili, on='mese') \
                                .orderBy('mese', ascending=False)

panoramica_mensile = panoramica_mensile.withColumn('percentuale_tamponi_positivi',
                        panoramica_mensile.casi / panoramica_mensile.tamponi * 100)

panoramica_mensile.show()
```

```
+-------+------+-------+--------------------------+
|   mese|  casi|tamponi|percentuale_tamponi_positivi|
+-------+------+-------+--------------------------+
|2021-04|439446|8677255|          5.064343504944825|
|2021-03|648200|9769610|          6.634860552263601|
|2021-02|372503|7466161|          4.989217350121435|
|2021-01|445585|5948709|         7.4904487679595695|
|2020-12|508914|4772918|         10.662533904835575|
|2020-11|922206|6160638|         14.969326228874348|
|2020-10|364607|4450539|          8.192423434554781|
|2020-09| 45623|2689063|         1.6966132812805055|
|2020-08| 21702|1831746|         1.1847712510358968|
|2020-07|  7006|1423003|         0.4923390885331935|
|2020-06|  7584|1511371|         0.5017960513996894|
|2020-05| 27556|1899522|         1.4506807502097896|
|2020-04| 99671|1472249|          6.769982523336745|
|2020-03|104664| 488307|         21.434056853577772|
|2020-02|  1120|  18661|          6.001821981673007|
+-------+------+-------+--------------------------+
```

Figura 3.2: Casi Covid-19 per Mese

3.1.3 Dati Critici per Mese

Si evidenzia il numero di casi registrati e di decessi dovuti al Covid-19, insieme al numero medio di persone in terapia intensiva ogni mese dall'inizio della pandemia.

Dati Critici per Mese

```
decessi_mensili = dati_covid_regioni.select(date_format('data','yyyy-MM').alias('mese'),
                                            'denominazione_regione', 'deceduti') \
                                    .groupby('mese', 'denominazione_regione') \
                                    .max('deceduti') \
                                    .groupby('mese') \
                                    .sum('max(deceduti)') \
                                    .withColumnRenamed('sum(max(deceduti))', 'deceduti')

my_window = Window.partitionBy().orderBy('mese')

decessi_mensili = decessi_mensili.withColumn('prev_value', lag(decessi_mensili.deceduti) \
                                 .over(my_window))
decessi_mensili = decessi_mensili.withColumn('diff', when(isnull(decessi_mensili.deceduti -
decessi_mensili.prev_value), decessi_mensili.deceduti).otherwise(decessi_mensili.deceduti -
                                    decessi_mensili.prev_value)) \
                                 .drop('deceduti', 'prev_value') \
                                 .withColumnRenamed('diff', 'deceduti')

terapie_intensive_mensili = dati_covid_regioni.select('data', 'terapia_intensiva') \
                  .groupby('data') \
                  .sum('terapia_intensiva') \
                  .select(date_format('data','yyyy-MM').alias('mese'),
                                            'sum(terapia_intensiva)') \
                  .groupby('mese') \
                  .mean('sum(terapia_intensiva)') \
                  .withColumnRenamed('avg(sum(terapia_intensiva))',
                                            'media_terapia_intensiva')

terapie_intensive_mensili = terapie_intensive_mensili.withColumn('media_terapia_intensiva',
                            terapie_intensive_mensili.media_terapia_intensiva.cast('int'))


dati_critici_mensili = decessi_mensili.join(casi_mensili, on='mese') \
                                      .join(terapie_intensive_mensili, on='mese') \
                                      .orderBy('mese')

dati_critici_mensili.show()
```

mese	deceduti	casi	media_terapia_intensiva
2020-02	29	1120	53
2020-03	12399	104664	1985
2020-04	15539	99671	2975
2020-05	5458	27556	882
2020-06	1406	7584	207
2020-07	311	7006	57
2020-08	341	21702	58
2020-09	411	45623	194
2020-10	2724	364607	785
2020-11	16958	922206	3231
2020-12	18583	508914	3005
2021-01	14357	445585	2483
2021-02	9183	372503	2128
2021-03	11647	648200	3127
2021-04	11461	439446	3306

Figura 3.3: Dati Critici per Mese

3.2 Dati Vaccini

3.2.1 Dosi Somministrate per Regione

Si evidenzia il numero di dosi somministrate in ciascuna regione, distinguendo tra prima e seconda dose. Per fornire un'informazione più completa, il numero di dosi somministrate viene contestualizzato rispetto al numero di abitanti di ogni regione, calcolando la percentuale di popolazione vaccinata con prima e seconda dose.

Dosi Somministrate per Regione

```
dosi_somministrate_regione = dati_vaccini_somministrazioni.groupBy('nome_area') \
                        .sum('prima_dose', 'seconda_dose') \
                        .orderBy('nome_area') \
                        .withColumnRenamed('nome_area', 'regione') \
                        .withColumnRenamed('sum(prima_dose)', 'vaccinati_prima_dose') \
                        .withColumnRenamed('sum(seconda_dose)', 'vaccinati_seconda_dose')

popolazione = popolazione_regioni.groupBy('denominazione_regione').sum('totale_generale')

cond = [popolazione.denominazione_regione == dosi_somministrate_regione.regione]
dosi_somministrate_regione = dosi_somministrate_regione \
                                .join(popolazione, cond, how='left') \
                                .drop('denominazione_regione') \
                                .withColumnRenamed('sum(totale_generale)', 'popolazione') \
                                .orderBy('regione')

dosi_somministrate_regione = dosi_somministrate_regione \
    .withColumn('percentuale_prima_dose', dosi_somministrate_regione.vaccinati_prima_dose /
                                          dosi_somministrate_regione.popolazione * 100)
dosi_somministrate_regione = dosi_somministrate_regione \
    .withColumn('percentuale_seconda_dose',
                dosi_somministrate_regione.vaccinati_seconda_dose /
                dosi_somministrate_regione.popolazione * 100)

dosi_somministrate_regione.show(21)
```

regione	vaccinati_prima_dose	vaccinati_seconda_dose	popolazione	percentuale_prima_dose	percentuale_seconda_dose
Abruzzo	315821	126627	1293941	24.407681648545026	9.786149445762984
Basilicata	131193	54155	553254	23.71297812577948	9.788451597277199
Bolzano	136324	51672	532644	25.593830025307714	9.701038592380652
Calabria	379280	161775	1894110	20.024180221845615	8.540950631167146
Campania	1237505	477110	5712143	21.664461131312716	8.352557000061097
Emilia-Romagna	1110377	532111	4464119	24.87337367126638	11.919731530454273
Friuli Venezia Gi...	283762	135626	1206216	23.52497396817817	11.243923144776723
Lazio	1324507	590965	5755700	23.012092360616432	10.267473982313186
Liguria	423584	190865	1524826	27.77916955770691	12.517165893026483
Lombardia	2419930	955480	10027602	24.132688951954815	9.528499435857148
Marche	400176	153045	1512672	26.454908929364727	10.11752713079901
Molise	75377	40223	300516	25.08252472414114	13.384645077133998
Piemonte	1088512	460401	4311217	25.248369543913007	10.679142339622432
Puglia	968258	317182	3953305	24.49236777835254	8.02321095893183
Sardegna	354594	152565	1611621	22.002319403879696	9.466555722468248
Sicilia	986820	445884	4875290	20.24125744314697	9.145794404025196
Toscana	878416	390248	3692555	23.78883997665573	10.568508796754552
Trento	135223	42070	545425	24.792226245588303	7.713251134436448
Umbria	215876	90102	870165	24.80862824866548	10.354587922980125
Valle d'Aosta	33188	12352	125034	26.54318025497065	9.878912935681495
Veneto	1153470	516416	4879133	23.640880459704626	10.584175508230663

Figura 3.4: Dosi Somministrate per Regione

3.2.2 Dosi Somministrate per Fascia Anagrafica

Si evidenzia il numero di dosi somministrate per ciascuna fascia anagrafica, distinguendo tra prima e seconda dose. Per fornire un'informazione più completa, il numero di dosi somministrate viene contestualizzato rispetto al numero di persone di ogni fascia anagrafica, calcolando la percentuale di persone vaccinate con prima e seconda dose.

Dosi Somministrate per Fascia Anagrafica

```
dosi_somministrate_fascia = dati_vaccini_somministrazioni.groupBy('fascia_anagrafica') \
                            .sum('prima_dose', 'seconda_dose') \
                            .orderBy('fascia_anagrafica') \
                            .withColumnRenamed('sum(prima_dose)', 'vaccinati_prima_dose') \
                            .withColumnRenamed('sum(seconda_dose)',
                                               'vaccinati_seconda_dose')

popolazione = popolazione_regioni.groupBy('range_eta').sum('totale_generale')

cond = [popolazione.range_eta == dosi_somministrate_fascia.fascia_anagrafica]
dosi_somministrate_fascia = dosi_somministrate_fascia.join(popolazione, cond, how='left') \
                            .withColumnRenamed('sum(totale_generale)', 'popolazione') \
                            .drop('range_eta') \
                            .orderBy('fascia_anagrafica')

dosi_somministrate_fascia = dosi_somministrate_fascia.withColumn('percentuale_prima_dose',
dosi_somministrate_fascia.vaccinati_prima_dose/dosi_somministrate_fascia.popolazione*100)
dosi_somministrate_fascia = dosi_somministrate_fascia.withColumn('percentuale_seconda_dose',
dosi_somministrate_fascia.vaccinati_seconda_dose/dosi_somministrate_fascia.popolazione*100)

dosi_somministrate_fascia.show(9)
```

fascia_anagrafica	vaccinati_prima_dose	vaccinati_seconda_dose	popolazione	percentuale_prima_dose	percentuale_seconda_dose
16-19	28153	9250	2298846	1.2246579370692947	0.40237580072784346
20-29	567939	286377	6084382	9.334374468927164	4.70675575596667
30-39	820538	382754	6854632	11.970562387594258	5.583873795121313
40-49	1203438	494618	8937229	13.46544885444918	5.534355223526219
50-59	1678332	679190	9414195	17.827674060288746	7.214530822869082
60-69	2287448	570961	7364364	31.06103935112387	7.7530252442709235
70-79	3603044	538469	5968373	60.368948120367136	9.022040009898845
80-89	3157181	2413368	3628160	87.01879189451402	66.51768389486682
90+	706120	521887	791543	89.20804049811571	65.93286782903772

Figura 3.5: Dosi Somministrate per Fascia Anagrafica

3.2.3 Dosi Somministrate per Regione e Fascia Anagrafica

Si evidenzia il numero di dosi di vaccino somministrate in ogni regione per ciascuna fascia anagrafica, distinguendo tra prima e seconda dose. Per fornire un'informazione più completa, il numero di dosi somministrate viene contestualizzato rispetto al numero di abitanti di ciascuna fascia anagrafica in ogni regione, calcolando la percentuale di popolazione vaccinata con prima e seconda dose.

Dosi Somministrate per Fascia Anagrafica in Campania

```
dosi_somministrate = dati_vaccini_somministrazioni \
                        .groupBy('nome_area', 'fascia_anagrafica') \
                        .sum('prima_dose', 'seconda_dose') \
                        .orderBy('nome_area', 'fascia_anagrafica') \
                        .withColumnRenamed('nome_area', 'regione') \
                        .withColumnRenamed('sum(prima_dose)', 'vaccinati_prima_dose') \
                        .withColumnRenamed('sum(seconda_dose)', 'vaccinati_seconda_dose')

cond = [popolazione_regioni.denominazione_regione == dosi_somministrate.regione,
        popolazione_regioni.range_eta == dosi_somministrate.fascia_anagrafica]

dosi_somministrate = dosi_somministrate \
                     .join(popolazione_regioni \
                         .select('denominazione_regione', 'range_eta', 'totale_generale'),
                         cond, how='left') \
                     .drop('denominazione_regione', 'range_eta') \
                     .withColumnRenamed('totale_generale', 'popolazione') \
                     .orderBy('regione', 'fascia_anagrafica')

dosi_somministrate = dosi_somministrate.withColumn('percentuale_prima_dose',
    dosi_somministrate.vaccinati_prima_dose / dosi_somministrate.popolazione * 100)
dosi_somministrate = dosi_somministrate.withColumn('percentuale_seconda_dose',
    dosi_somministrate.vaccinati_seconda_dose / dosi_somministrate.popolazione * 100)

dosi_somministrate.where('regione == "Campania"').show(9)
```

Per semplicità si riportano i risultati dell'elaborazione per la sola regione Campania, filtrano opportunamente sulla colonna `regione`.

regione	fascia_anagrafica	vaccinati_prima_dose	vaccinati_seconda_dose	popolazione	percentuale_prima_dose	percentuale_seconda_dose
Campania	16-19	3551	1198	261694	1.3569283208632983	0.45778657516030175
Campania	20-29	53102	24637	693479	7.65733353136865	3.552667059853291
Campania	30-39	80894	36263	715258	11.309765147680977	5.069918826493377
Campania	40-49	117968	43183	835597	14.117810379884082	5.167921857067462
Campania	50-59	178825	61567	868684	20.5857365854557	7.087387358348951
Campania	60-69	276850	69851	670867	41.26749415308847	10.412048885993796
Campania	70-79	292292	55668	484380	60.34353193773484	11.492629753499319
Campania	80-89	198475	160737	255273	77.75009499633725	62.96670623215146
Campania	90+	35548	24006	49044	72.48185302993231	48.9478835331539

Figura 3.6: Dosi Somministrate per Fascia Anagrafica in Campania

3.2.4 Dosi Somministrate per Fascia Anagrafica e Fornitore

Si evidenzia il numero di dosi di ogni fornitore somministrate a ciascuna fascia anagrafica dall'inizio della pandemia. Per fornire un'informazione più chiara, il numero di dosi di ogni fornitore somministrate ad una fascia anagrafica viene mostrato come percentuale rispetto al totale delle dosi somministrate a quella fascia anagrafica.

Dosi Somministrate per Fascia Anagrafica e Fornitore

```
dosi_fascia_fornitore = dati_vaccini_somministrazioni \
                                .groupBy('fascia_anagrafica', 'fornitore') \
                                .sum('prima_dose', 'seconda_dose') \
                                .orderBy('fascia_anagrafica', 'fornitore')

totale_dosi_somministrate_fascia = dosi_somministrate_fascia.select('fascia_anagrafica',
                                    'vaccinati_prima_dose', 'vaccinati_seconda_dose') \
                                    .withColumn('totale_dosi',
                                        dosi_somministrate_fascia.vaccinati_prima_dose +
                                        dosi_somministrate_fascia.vaccinati_seconda_dose) \
                                    .drop('vaccinati_prima_dose', 'vaccinati_seconda_dose')

dosi_fascia_fornitore = dosi_fascia_fornitore.join(totale_dosi_somministrate_fascia,
                                                   on='fascia_anagrafica')

dosi_fascia_fornitore = dosi_fascia_fornitore.withColumn('percentuale_dosi',
                            round((dosi_fascia_fornitore['sum(prima_dose)'] +
                            dosi_fascia_fornitore['sum(seconda_dose)']) *100 /
                            dosi_fascia_fornitore.totale_dosi, 3)) \
                        .drop('sum(prima_dose)', 'sum(seconda_dose)', 'totale_dosi') \
                        .orderBy('fascia_anagrafica', 'fornitore')

dosi_fascia_fornitore.show(100)
```

fascia_anagrafica	fornitore	percentuale_dosi
16-19	Janssen	0.032
16-19	Moderna	7.398
16-19	Pfizer/BioNTech	85.806
16-19	Vaxzevria (AstraZ...	6.764
20-29	Janssen	0.176
20-29	Moderna	4.492
20-29	Pfizer/BioNTech	74.67
20-29	Vaxzevria (AstraZ...	20.662
30-39	Janssen	0.16
30-39	Moderna	4.464
30-39	Pfizer/BioNTech	70.325
30-39	Vaxzevria (AstraZ...	25.052
40-49	Janssen	0.118
40-49	Moderna	4.81
40-49	Pfizer/BioNTech	67.071
40-49	Vaxzevria (AstraZ...	28.001
50-59	Janssen	0.086
50-59	Moderna	5.273
50-59	Pfizer/BioNTech	69.584
50-59	Vaxzevria (AstraZ...	25.057
60-69	Janssen	0.711
60-69	Moderna	6.375
60-69	Pfizer/BioNTech	61.828
60-69	Vaxzevria (AstraZ...	31.086
70-79	Janssen	0.734
70-79	Moderna	7.18
70-79	Pfizer/BioNTech	48.78
70-79	Vaxzevria (AstraZ...	43.306
80-89	Janssen	0.01
80-89	Moderna	9.496
80-89	Pfizer/BioNTech	89.352
80-89	Vaxzevria (AstraZ...	1.142
90+	Janssen	0.007
90+	Moderna	13.803
90+	Pfizer/BioNTech	85.511
90+	Vaxzevria (AstraZ...	0.678

Figura 3.7: Dosi Somministrate per Fascia Anagrafica e Fornitore

3.2.5 Dosi Somministrate per Mese e Categoria

Si evidenzia il numero di dosi somministrate ogni mese alle diverse categorie definite dal Ministero della Salute. Per fornire un'informazione più chiara, il numero di dosi somministrate ad ogni categoria in un mese viene mostrato come percentuale rispetto al totale delle dosi somministrate in quel mese.

Dosi Somministrate per Mese e Categoria

```
dosi_somministrate_categorie_mensili = dati_vaccini_somministrazioni \
    .select(date_format('data_somministrazione','yyyy-MM').alias('mese'),
            'categoria_operatori_sanitari_sociosanitari',
            'categoria_personale_non_sanitario', 'categoria_ospiti_rsa',
            'categoria_60_69', 'categoria_70_79', 'categoria_over80',
            'categoria_forze_armate', 'categoria_personale_scolastico',
            'categoria_soggetti_fragili', 'categoria_altro')
    .groupby('mese') \
    .agg(sum('categoria_operatori_sanitari_sociosanitari') \
              .alias('operatori_sanitari_sociosanitari'), \
         sum('categoria_personale_non_sanitario').alias('personale_non_sanitario'), \
         sum('categoria_ospiti_rsa').alias('ospiti_rsa'), \
         sum('categoria_60_69').alias('60_69'), \
         sum('categoria_70_79').alias('70_79'), \
```

```
            sum('categoria_over80').alias('over80'), \
            sum('categoria_forze_armate').alias('forze_armate'), \
            sum('categoria_personale_scolastico').alias('personale_scolastico'), \
            sum('categoria_soggetti_fragili').alias('soggetti_fragili'), \
            sum('categoria_altro').alias('altro')) \
        .orderBy('mese')

dosi_somministrate_mensili = dati_vaccini_somministrazioni \
    .select(date_format('data_somministrazione','yyyy-MM').alias('mese'),
            'prima_dose', 'seconda_dose') \
    .groupby('mese') \
    .sum('prima_dose', 'seconda_dose')

dosi_somministrate_mensili = dosi_somministrate_mensili \
    .withColumnRenamed('sum(prima_dose)', 'vaccinati_prima_dose') \
    .withColumnRenamed('sum(seconda_dose)', 'vaccinati_seconda_dose')


totale_dosi_somministrate_mensili = dosi_somministrate_mensili.withColumn('totale_dosi',
    dosi_somministrate_mensili.vaccinati_prima_dose +
    dosi_somministrate_mensili.vaccinati_seconda_dose)

percentuale_dosi_categorie = totale_dosi_somministrate_mensili \
    .join(dosi_somministrate_categorie_mensili, on='mese') \
    .drop('vaccinati_prima_dose', 'vaccinati_seconda_dose') \
    .withColumn('operatori_sanitari_sociosanitari',
        round(dosi_somministrate_categorie_mensili.operatori_sanitari_sociosanitari /
        totale_dosi_somministrate_mensili.totale_dosi *100, 3)) \
    .withColumn('personale_non_sanitario',
        round(dosi_somministrate_categorie_mensili.personale_non_sanitario /
        totale_dosi_somministrate_mensili.totale_dosi *100, 3)) \
     .withColumn('ospiti_rsa', round(dosi_somministrate_categorie_mensili.ospiti_rsa /
        totale_dosi_somministrate_mensili.totale_dosi *100, 3)) \
    .withColumn('60_69', round(dosi_somministrate_categorie_mensili['60_69'] /
        totale_dosi_somministrate_mensili.totale_dosi *100, 3)) \
    .withColumn('70_79', round(dosi_somministrate_categorie_mensili['70_79'] /
        totale_dosi_somministrate_mensili.totale_dosi *100, 3)) \
    .withColumn('over80', round(dosi_somministrate_categorie_mensili.over80 /
        totale_dosi_somministrate_mensili.totale_dosi *100, 3)) \
    .withColumn('forze_armate', round(dosi_somministrate_categorie_mensili.forze_armate /
        totale_dosi_somministrate_mensili.totale_dosi *100, 3)) \
    .withColumn('personale_scolastico',
        round(dosi_somministrate_categorie_mensili.personale_scolastico /
        totale_dosi_somministrate_mensili.totale_dosi *100, 3)) \
    .withColumn('soggetti_fragili',
        round(dosi_somministrate_categorie_mensili.soggetti_fragili /
        totale_dosi_somministrate_mensili.totale_dosi *100, 3)) \
    .withColumn('altro', round(dosi_somministrate_categorie_mensili.altro /
        totale_dosi_somministrate_mensili.totale_dosi *100, 3)) \
    .drop('totale_dosi') \
    .orderBy('mese')

percentuale_dosi_categorie.show()
```

```
+-------+-------------------------------+----------------------+----------+-----+-----+-----+------------+-------------------+----------------+-----+
|   mese|operatori_sanitari_sociosanitari|personale_non_sanitario|ospiti_rsa| 60_69| 70_79|over80|forze_armate|personale_scolastico|soggetti_fragili|altro|
+-------+-------------------------------+----------------------+----------+-----+-----+-----+------------+-------------------+----------------+-----+
|2020-12|                         85.275|                  3.75|     8.193| 0.409| 0.105| 0.128|         0.0|               0.02|           0.003|2.117|
|2021-01|                         73.676|                11.605|    10.148| 0.603| 0.226| 0.898|       0.009|              0.016|           0.007|2.811|
|2021-02|                          38.03|                11.388|     8.972| 0.608| 0.353|27.002|       2.845|              7.462|            0.11| 3.23|
|2021-03|                         10.036|                 4.709|     2.594| 1.457| 6.731|47.339|       3.611|             14.207|           7.361|1.955|
|2021-04|                          2.594|                  1.13|     0.868|10.048|26.763|27.981|       0.438|              1.466|          27.504|1.208|
+-------+-------------------------------+----------------------+----------+-----+-----+-----+------------+-------------------+----------------+-----+
```

Figura 3.8: Dosi Somministrate per Mese e Categoria

3.3 Correlazione tra Dati Covid-19 e Dati Vaccini

3.3.1 Dosi Somministrate per Mese e Media Terapia Intensiva

Si evidenzia la percentuale di popolazione vaccinata al termine di ogni mese, distinguendo tra prima e seconda dose. In altre parole, le percentuali calcolate per ogni mese sono cumulative.

Ai dati relativi ai vaccini viene affiancano il numero medio di persone presenti ogni mese in terapia intensiva. In questo modo è possibile osservare come tale valore evolve ogni mese in relazione alle dosi somministrate. L'obiettivo dei vaccini è infatti quello di proteggere la popolazione dalla forma grave di malattia causata dal Covid-19.

Dosi Attualmente Somministrate per Mese e Media Terapia Intensiva

```
popolazione_totale = popolazione_regioni.groupBy().sum()

temp = dosi_somministrate_mensili.join(terapie_intensive_mensili, on='mese', how='left')

temp = temp.join(popolazione_totale) \
           .withColumnRenamed('sum(totale_generale)', 'totale') \
           .orderBy('mese')

cum_sum = temp.withColumn('cumsum_prima_dose',
                  sum(temp.vaccinati_prima_dose) \
                  .over(Window.partitionBy().orderBy().rowsBetween(-sys.maxsize, 0))) \
              .withColumn('cumsum_seconda_dose',
                  sum(temp.vaccinati_seconda_dose) \
                  .over(Window.partitionBy().orderBy().rowsBetween(-sys.maxsize, 0)))

cum_sum = cum_sum.withColumn('percentuale_prima_dose',
            cum_sum.cumsum_prima_dose / cum_sum.totale * 100)
cum_sum = cum_sum.withColumn('percentuale_seconda_dose',
            cum_sum.cumsum_seconda_dose / cum_sum.totale * 100)

dosi_somministrate_terapie_intensive = cum_sum.select('mese', 'percentuale_prima_dose',
                                    'percentuale_seconda_dose', 'media_terapia_intensiva')

dosi_somministrate_terapie_intensive.show()
```

```
+-------+---------------------+----------------------+----------------------+
|   mese|percentuale_prima_dose|percentuale_seconda_dose|media_terapia_intensiva|
+-------+---------------------+----------------------+----------------------+
|2020-12|     0.066758897765931|                   0.0|                  3005|
|2021-01|    2.3123534409470134|     1.0736184181051955|                  2483|
|2021-02|     4.975518048778394|     2.3836393887422798|                  2128|
|2021-03|    11.932136904431358|      5.427421596188211|                  3127|
|2021-04|    23.561103975138916|      9.887201338772767|                  3306|
+-------+---------------------+----------------------+----------------------+
```

Figura 3.9: Dosi Somministrate per Mese e Media Terapia Intensiva

3.3.2 Dosi Attualmente Somministrate per Regione e Attuale Terapia Intensiva

Si evidenzia la percentuale di popolazione attualmente vaccinata in ogni regione, distinguendo tra prima e seconda dose.

Ai dati relativi ai vaccini viene affiancano il numero di persone attualmente presenti in terapia intensiva in ciascuna regione. In questo modo è possibile osservare come tale valore cambia in relazione alle dosi somministrate nelle diverse regioni.

Dosi Attualmente Somministrate per Regione e Attuale Terapia Intensiva

```
attuali_terapie_intensive_regioni = dati_covid_regioni \
                        .where(col("data") >= current_date() - expr("INTERVAL 1 days")) \
                        .select('denominazione_regione', 'terapia_intensiva') \
                        .withColumnRenamed('denominazione_regione', 'regione')

attuali_dosi_somministrate_terapie_intensive = attuali_terapie_intensive_regioni \
                        .join(dosi_somministrate_regione, on='regione') \
                        .select('regione', 'percentuale_prima_dose',
                                'percentuale_seconda_dose', 'terapia_intensiva') \
```

```
                        .orderBy('regione')
attuali_dosi_somministrate_terapie_intensive.show(21)
```

regione	percentuale_prima_dose	percentuale_seconda_dose	terapia_intensiva
Abruzzo	24.407681648545026	9.786149445762984	36
Basilicata	23.71297812577948	9.788451597277199	8
Bolzano	25.593830025307714	9.701038592380652	2
Calabria	20.024180221845615	8.540950631167146	46
Campania	21.664461131312716	8.352557000061097	137
Emilia-Romagna	24.87337367126638	11.919731530454273	226
Friuli Venezia Gi...	23.52497396817817	11.243923144776723	30
Lazio	23.012092360616432	10.267473982313186	284
Liguria	27.77916955770691	12.517165893026483	65
Lombardia	24.132688951954815	9.528499435857148	550
Marche	26.454908929364727	10.11752713079901	67
Molise	25.08252472414114	13.384645077133998	8
Piemonte	25.248369543913007	10.679142339622432	210
Puglia	24.49236777835254	8.02321095893183	224
Sardegna	22.002319403879696	9.466555722468248	49
Sicilia	20.24125744314697	9.145794404025196	165
Toscana	23.78883997665573	10.568508796754552	253
Trento	24.792226245588303	7.713251134436448	20
Umbria	24.80862824866548	10.354587922980125	34
Valle d'Aosta	26.54318025497065	9.878912935681495	8
Veneto	23.640880459704626	10.584175508230663	161

Figura 3.10: Dosi Attualmente Somministrate per Regione e Attuale Terapia Intensiva

Bibliografia

[1] d'Andrea Fabio; Di Chiara Guido. *Repository GitHub - Elaborazione Spark Dati Covid-19*. URL: https://github.com/fabiod20/homework-BDABI/tree/main/homework2-Spark.

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

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

[4] 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.

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