UNIVERSITÀ DEGLI STUDI DI NAPOLI

FEDERICO II

Scuola Politecnica e delle Scienze di Base

Corso di Laurea Magistrale in Ingegneria Informatica

Homework 1 – Big Data Engineering

Anno Accademico 2022-2023

Riccio Emanuele – M63001339

Tammaro Ferdinando – M63001380

Prof. Moscato Vincenzo

Sommario

Introduzione

Il presente report descrive l'analisi condotta su un dataset di progetti universitari utilizzando le tecnologie di analisi dei dati di Apache Spark: PySpark, Hive e Pig. Il dataset è stato fornito durante il corso e contiene informazioni dettagliate su una vasta gamma di progetti universitari, tra cui il loro titolo, gli autori, l'anno di inizio, l'anno di fine e le università coinvolte.

Di particolare interesse per queste analisi sono stati principalmente gli enti che hanno finanziato questi progetti di ricerca, la quantità di fondi ricevuta da ogni progetto e le città coinvolte negli stessi.

In particolare, le analytics effettuate sono state:

- Media investimenti relativa ai progetti (in EUR)
- Numero di enti finanziatori distinti
- Media fondi investiti per ogni ente finanziatore (in EUR)
- Media fondi annui per ogni progetto (in EUR)
- Media fondi annui spesi da ogni ente finanziatore (in EUR)
- Numero di progetti per ogni città

Al fine di eseguire le analytics specificate, il dataset è stato inizialmente filtrato tramite la libreria Pandas nel linguaggio Python; le colonne considerate sono state:

- Grant_ID (`string`)
- Funding Amount in EUR (`long int`)
- Start Year (`int`)
- End Year (`int`)
- City of Research Organization (`string`, ogni campo comprende più città separate da un punto e virgola)
- Funder (`string`)
- Funder Country (`string`)

Analytics

In questa sezione andremo ad approfondire ogni analisi effettuata osservando il codice scritto in ciascun linguaggio

PySpark

Abbiamo caricato il dataset con i seguenti comandi:

```
1. import pandas as pd
2.
3. grantDS = spark.read \
4.   .option("inferSchema", True) \
5.   .option("header", True) \
6.   .option("quote", "\"") \
7.   .option("escape", "\"") \
8.   .csv("/content/drive/MyDrive/dataset.csv", sep=';', multiLine=True)
```

Query 1

La prima query effettuata calcola la media in euro degli investimenti relativa ai progetti. È stata effettuata così:

```
1. from pyspark.sql.functions import avg,col, asc,desc, when
2.
3. grantDS.select(avg("Funding Amount in EUR")).show(100)
```

che ci ha fornito in output quanto si può vedere in Figura 1.

Query 2

La seconda query serve per calcolare il numero di enti finanziatori presenti nel dataset

```
1. grantDS.select("Funder").distinct().count()
```

In output otteniamo 31 enti differenti.

```
+-------------------------+
|avg(Funding Amount in EUR)|
+-------------------------+
|       1321146.4091249064|
+-------------------------+
```

Figura 1 - Output query 1 in Pyspark

Query 3

La terza query riguarda la media dei fondi investiti per ogni ente finanziatore

```
1. from pyspark.sql import functions as F
2.
3. grantDS.groupBy("Funder").agg(F.sum('Funding Amount in EUR'), F.count("Grant
   ID"))\
4. .withColumnRenamed("sum(Funding Amount in EUR)",
   "total").na.fill(value=0,subset=["total"])\
5. .withColumnRenamed("count(Grant ID)", "numb_proj").sort(col("total").desc())\
6. .withColumn("Ratio_per_proj", col("total") / col("numb_proj")).show(31)
```

Funder	total	numb_proj	Ratio_per_proj
European Commission	2863366132	630	4545025.6063492065
Ministry of Educa...	495857743	2049	241999.87457296244
Medical Research ...	80444483	12	6703706.916666667
Engineering and P...	31308013	18	1739334.0555555555
European Research...	25625750	19	1348723.6842105263
Biotechnology and...	13257773	11	1205252.0909090908
Telethon Foundation	11549052	65	177677.72307692308
Economic and Soci...	4896585	2	2448292.5
Swiss National Sc...	2184410	5	436882.0
Natural Environme...	1268127	5	253625.4
Science and Techn...	702280	3	234093.33333333334
United States Dep...	530536	4	132634.0
Foundation Fighti...	525869	2	262934.5
Australian Resear...	253794	1	253794.0
United States Air...	253787	1	253787.0
Arts and Humaniti...	165404	1	165404.0
Juvenile Diabetes...	125997	1	125997.0
American Heart As...	110985	1	110985.0
Fundação para a C...	101456	2	50728.0
Crohn's and Colit...	99002	1	99002.0
Brain & Behavior ...	60960	1	60960.0
Japan Society for...	36982	1	36982.0
Breast Cancer Now	20378	1	20378.0
Qatar National Re...	0	1	0.0
Belgian Federal S...	0	1	0.0
Council for Inter...	0	79	0.0
Italian Associati...	0	30	0.0
National Aeronaut...	0	3	0.0
Agricultural Rese...	0	1	0.0
Agencia Nacional ...	0	1	0.0
Deutsche Forschun...	0	12	0.0

Figura 2 - Output query 3 in Pyspark

Per realizzare la query viene effettuato prima un raggruppamento sul campo "Funder" poi utilizzare le funzioni di aggregazioni `sum` e `count`, le quali generano due nuovi campi (`total`, `numb_proj`) contenenti la somma dei fondi investiti in EUR ed il numero di progetti per ogni ente finanziatore. Si prosegue rinominando alcune colonne ed infine per ogni ente viene calcolato la somma in EUR media fornita per ogni progetto come rapporto tra i campi `total` e `numb_proj` calcolati in precedenza.

Query 4

La quarta query riguarda la media dei fondi annui investiti per ogni progetto, si vuole calcolare questa quantità come rapporto dell'investimento per il progetto e la sua durata in anni:

```
1. grantDS2 = grantDS.withColumn('Duration',(col("End Year") - col("Start
   Year"))+1)
2. grantDS2 = grantDS2.withColumn("Duration", when(col("Duration") < 0,
   0).otherwise(col("Duration")))
3.
4. grantDS2 = grantDS2.withColumn('Ratio',col("Funding Amount in EUR") /
   col("Duration")).na.fill(value=0,subset=["Ratio"])
```

Per prima cosa si genera un nuovo campo "Duration" calcolato come differenza tra l'anno di fine ed inizio del progetto, gestendo il caso in cui non sia presente l'anno di fine in quel caso si assegna durata pari a zero. Infine, si calcola il rapporto tra i campi "`Funding Amount in EUR`" e "`Duration`", rinominando il campo generato come "Ratio", si gestisce il caso in cui siano presenti valori NULL sostituendoli con zero. L'output è riportato in figura 3.

Grant ID	Funding Amount in EUR	Start Year	End Year	City of Research Organization	Funder	Funder Country	Duration	Ratio
grant.13242069	10632000	2023	2027	The Hague; Bergen...	European Commission	Belgium	5	2126400.0
grant.13046237	10815148	2023	2028	Paris; Vienna; Am...	European Commission	Belgium	6	1802524.6666666667
grant.12941744	2062404	2023	2027	Hamburg; Manchest...	European Commission	Belgium	5	412480.8
grant.12941771	2136946	2023	2027	Naples; Rome; Mad...	European Commission	Belgium	5	427389.2
grant.13242078	34979440	2023	2026	Toulouse; Getafe;...	European Commission	Belgium	4	8744860.0
grant.13046328	6193128	2023	2026	Paris; Brilon; Na...	European Commission	Belgium	4	1548282.0
grant.13046324	8098455	2023	2025	Munich; Porto; Na...	European Commission	Belgium	3	2699485.0
grant.13046266	10427094	2023	2026	Naples; Gothenbur...	European Commission	Belgium	4	2606773.5
grant.13046143	3242373	2023	2025	Mondragón; Basili...	European Commission	Belgium	3	1080791.0
grant.13030153	6858523	2023	2028	Turku; Linköping;...	European Commission	Belgium	6	1143087.1666666667
grant.13030121	11502963	2023	2027	Wageningen; Trond...	European Commission	Belgium	5	2300592.6
grant.13029734	496800	2023	2026	Warsaw; Warsaw; L...	European Commission	Belgium	4	124200.0
grant.13029713	2294146	2023	2025	Barcelona; Exeter...	European Commission	Belgium	3	764715.3333333334
grant.13029460	331200	2023	2026	Porto; Changchun;...	European Commission	Belgium	4	82800.0
grant.13029389	4939558	2023	2026	Coimbra; Naples; ...	European Commission	Belgium	4	1234889.5
grant.12959784	2645172	2023	2026	Brussels; Louvain...	European Commission	Belgium	4	661293.0
grant.12941770	2357093	2023	2026	Trento; Mancheste...	European Commission	Belgium	4	589273.25
grant.12940668	1993750	2023	2027	Naples; Rome	European Commission	Belgium	5	398750.0
grant.13046367	8499999	2022	2025	Trondheim; Aarhus...	European Commission	Belgium	4	2124999.75
grant.13029966	9997594	2022	2027	Bogotá; Siena; Is...	European Commission	Belgium	6	1666265.6666666667
grant.12961031	1317750	2022	2025	Jülich; Tartu; Be...	European Commission	Belgium	4	329437.5
grant.12960989	13923476	2022	2026	Potsdam; Reykjavi...	European Commission	Belgium	5	2784695.2
grant.12960660	3587827	2022	2025	Braunschweig; Ott...	European Commission	Belgium	4	896956.75
grant.12960278	6766959	2022	2025	Zoetermeer; Brisb...	European Commission	Belgium	4	1691739.75
grant.12959780	4000000	2022	2026	Rome; Pamplona; N...	European Commission	Belgium	5	800000.0
grant.12941853	2201655	2022	2025	Barcelona; Naples...	European Commission	Belgium	4	550413.75
grant.12940315	1954308	2022	2027	Naples	European Commission	Belgium	6	325718.0

Figura 3 - Estratto dell'output della query 4 in Pyspark

Query 5

La quinta query riguarda la Media fondi annui spesi da ogni ente finanziatore (in EUR).

```
1. grantDS3 = grantDS2.groupBy("Funder").sum("Funding Amount in EUR",
   "Duration").withColumnRenamed("sum(Funding Amount in EUR)", "total") \
2. .withColumnRenamed("sum(Duration)", "total_d")
3.
4. grantDS3 = grantDS3.withColumn('Ratio2',(col("total") / col("total_d")))
5. grantDS3 =
   grantDS3.na.fill(value=0,subset=["Ratio2"]).sort(col("Ratio2").desc())
6. grantDS3.show(31)
```

Partendo dalle colonne precedentemente create, si fa un raggruppamento sulla colonna "`Funder`" e una somma sulle colonne "`Funding Amount in EUR`" e "`Duration`", per calcolare ogni ente finanziatore quanti anni totali di ricerca ha finanziato e qual è stata la sua spesa. Questi campi sono poi utilizzati per calcolare una nuova colonna, da noi nominata "`Ratio2`", definita come il totale di soldi spesi diviso il totale degli anni finanziati, per capire in media quanto ogni ente abbia speso per ogni anno di ricerca. Abbiamo poi riempito i valori NULL con degli zeri e infine visualizzato la tabella risultante, come si può vedere in figura 4.

Funder	total	total_d	Ratio2
European Commission	2863366132	2572	1113283.8771384137
Medical Research ...	80444483	126	638448.2777777778
Economic and Soci...	4896585	11	445144.0909090909
Engineering and P...	31308013	81	386518.6790123457
Biotechnology and...	13257773	53	250146.6603773585
European Research...	25625750	116	220911.6379310345
Swiss National Sc...	2184410	22	99291.36363636363
Ministry of Educa...	495857743	6498	76309.2863958141
United States Dep...	530536	7	75790.85714285714
Foundation Fighti...	525869	8	65733.625
Australian Resear...	253794	4	63448.5
United States Air...	253787	4	63446.75
Natural Environme...	1268127	21	60387.0
Telethon Foundation	11549052	201	57457.97014925373
American Heart As...	110985	2	55492.5
Arts and Humaniti...	165404	3	55134.666666666664
Science and Techn...	702280	14	50162.857142857145
Crohn's and Colit...	99002	2	49501.0
Juvenile Diabetes...	125997	3	41999.0
Brain & Behavior ...	60960	3	20320.0
Fundação para a C...	101456	8	12682.0
Breast Cancer Now	20378	2	10189.0
Japan Society for...	36982	4	9245.5
Qatar National Re...	null	5	0.0
Belgian Federal S...	null	3	0.0
Council for Inter...	null	131	0.0
Italian Associati...	null	91	0.0
National Aeronaut...	null	15	0.0
Agricultural Rese...	null	3	0.0

Figura 4 - Estratto dell'output della query 5 in Pyspark

Query 6

La sesta query consiste nel contare in quanti progetti è coinvolta ogni città presente nel dataset.

```
1. from pyspark.sql.functions import split, explode
2. grantDS4 = grantDS.withColumn("City of Research organization",
   split(grantDS["City of Research organization"], "; "))\
3. .withColumn("city", explode("City of Research organization"))
4. grantDS_citycount = grantDS4.groupBy("city").count().sort(col("count").desc())
5. num = grantDS_citycount.count()
6. grantDS_citycount.show(num)
```

Siccome ogni riga del campo “City of Research organization” conteneva più città, abbiamo come prima cosa effettuato uno split su questo campo, seguito dal comando explode in modo da ottenere per ogni riga un numero di copie pari al numero di città presenti nel campo “City of Research organization”, quindi più righe aventi tutti i campi uguali tranne quello “city”. Fatto ciò, è bastato fare un’operazione di count sui dati raggruppati secondo la colonna delle città. Un estratto dell’output è visualizzato in figura 5.

city	count
Naples	3042
Rome	1205
Milan	898
Turin	489
London	437
Florence	371
Pisa	334
Paris	330
Padova	324
Bologna	318
Bari	284
Caserta	280
Genoa	254
Catania	245
Palermo	242
Fisciano	228
Pavia	183
Perugia	181
Trieste	177
Madrid	173
Munich	168
Siena	155
Messina	149
Rende	140
Cagliari	122
Cambridge	121
Parma	119
Barcelona	117
Udine	111
Lecce	109
Venice	107

Figura 5 - Estratto dell'output della query 6 in Pyspark

Hive

In Hive sono state effettuate le stesse query eseguite in Pyspark, adattando la sintassi a quella di Hive. Per usare Hive ci siamo serviti di un Notebook tramite la community edition di Databricks.

Per importare il dataset è stato usato il seguente codice:

```
# File location and type
file_location = "/FileStore/tables/dataset.csv"
file_type = "csv"

# CSV options
infer_schema = "false"
first_row_is_header = "true"
delimiter = ";"

# The applied options are for CSV files. For other file types, these will be ignored.
df = spark.read.format(file_type) \
  .option("inferSchema", infer_schema) \
  .option("header", first_row_is_header) \
  .option("sep", delimiter) \
  .option("quote", "\"") \
  .option("escape", "\"") \
  .load(file_location)

df = df.withColumnRenamed("Fields of Research (ANZSRC 2020)","Fields of Research")

import re
schema1 = [re.sub("[^a-zA-Z0-9,]", "", i) for i in df.columns]
df = df.toDF(*schema1)

display(df)

temp_table_name = "grantDS"
df.createOrReplaceTempView(temp_table_name)
```

Che per la maggior parte è stato automaticamente generato da Databricks una volta importato il file `dataset.csv`. Tuttavia, una volta importato il dataset ci siamo accorti che alcuni nomi di colonne creavano problemi, dovuti al fatto che presentavano caratteri anomali, tra cui spazi. Abbiamo quindi provveduto a rimuovere gli spazi e a rinominare le colonne che non andavano bene.

Query 1

La prima query è realizzata nel seguente modo

```
1. SELECT AVG(`FundingAmountinEUR`) FROM grantDS;
```

L'output è visualizzato nella figura 6, come possiamo notare è identico a quanto già visto in Pyspark. Per questo motivo in questa sezione non saranno riportate tutte le tabelle di output ottenute in Hive.

Query 2

La seconda query è semplicemente realizzata facendo un COUNT sui Funder distinti presenti nella colonna

```
1. SELECT COUNT(DISTINCT Funder) FROM grantDS;
```

Query 3

La terza query è realizzata in maniera analoga a quanto già visto in Pyspark

```
1. SELECT Funder, SUM(`FundingAmountinEUR`) AS total, COUNT(`GrantID`) AS
   numb_proj, total/numb_proj AS Ratio_per_proj
2. FROM grantDS
3. GROUP BY Funder
4. ORDER BY total DESC
```

Possiamo osservare l'output nella figura 7, ancora una volta identico a quanto già visto in Pyspark.

	avg(FundingAmountinEUR)
1	1321146.4091249064

Figura 6 - Output della query 1 in Hive

	Funder	total	numb_proj	Ratio_per_proj
1	European Commission	2863366132	630	4545025.6063492065
2	Ministry of Education, Universities and Research	495857743	2049	241999.87457296244
3	Medical Research Council	80444483	12	6703706.916666667
4	Engineering and Physical Sciences Research Council	31308013	18	1739334.0555555555
5	European Research Council	25625750	19	1348723.6842105263
6	Biotechnology and Biological Sciences Research Council	13257773	11	1205252.0909090908
7	Telethon Foundation	11549052	65	177677.72307692308

31 righe | runtime 2,98 secondi

Figura 7 - Output della query 2 in Hive

Query 4

La query 4 è stata così eseguita:

```
1. DROP TABLE IF EXISTS grantDS2;
2. CREATE TABLE grantDS2 AS
3. SELECT GrantID, FundingAmountinEUR, StartYear,
   EndYear,CityofResearchorganization, Funder, FunderCountry,
4.   CASE WHEN (`EndYear` - `StartYear`) + 1 < 0 THEN 0 ELSE (`EndYear` -
   `StartYear`) + 1 END AS Duration,
5.   CASE WHEN (`FundingAmountinEUR` IS NULL OR (`EndYear` - `StartYear`) + 1 <
   0) THEN 0 ELSE `FundingAmountinEUR` / ((`EndYear` - `StartYear`) + 1) END AS
   Ratio
6. FROM grantDS;
7.
8. SELECT * FROM grantDS2;
```

Query 5

La query 5 è stata così eseguita

```
1. SELECT Funder, SUM(`FundingAmountinEUR`) AS total, SUM(`Duration`) AS total_d,
   total/total_d AS Ratio2
2. FROM grantDS2
3. GROUP BY Funder
4. ORDER BY Ratio2 DESC
```

E possiamo osservarne l'output nella figura 8

	Funder	total	total_d	Ratio2
1	European Commission	2863366132	2572	1113283.8771384137
2	Medical Research Council	80444483	126	638448.2777777778
3	Economic and Social Research Council	4896585	11	445144.0909090909
4	Engineering and Physical Sciences Research Council	31308013	81	386518.6790123457
5	Biotechnology and Biological Sciences Research Council	13257773	53	250146.6603773585
6	European Research Council	25625750	116	220911.6379310345
7	Swiss National Science Foundation	2184410	22	99291.36363636363

31 righe | runtime 5,88 secondi

Figura 8 - Estratto dell'output della query 5 in Hive

Query 6

La query 6 è stata così eseguita:

```
1. SELECT city, COUNT(*) AS COUNT
2. FROM (
3.   SELECT EXPLODE(SPLIT(`CityofResearchorganization`, '; ')) AS city
4.   FROM grantDS
5. ) t
6. GROUP BY city
7. ORDER BY COUNT DESC
```

E possiamo osservarne l'output nella figura 9

	city	count
1	Naples	3042
2	Rome	1205
3	Milan	898
4	Turin	489
5	London	437
6	Florence	371
7	Pisa	334

1.136 righe | runtime 1,67 secondi

Figura 9 - Estratto dell'output della query 6 in Hive

Pig

Abbiamo usato Hadoop e Pig attraverso una macchina virtuale Ubuntu 20.04 tramite il programma Multipass. Dopo non pochi problemi per inizializzare la macchina virtuale, abbiamo caricato il dataset utilizzando questo codice:

```
1. grantDS = LOAD '/home/ubuntu/hadoop/pig/dataOtt.csv' USING PigStorage(',')
2. AS (col1:chararray, grant_id:chararray, funding:long, s_year:INT, e_year:INT,
   city:chararray, funder:chararray,
3. funder_country:chararray);
```

Dove abbiamo manualmente specificato l'header e il tipo per ogni colonna del dataframe importato.

L'interfaccia di Pig è completamente a linea di comando, per questo motivo non riporteremo tutti gli output delle query sotto forma di immagine ma soltanto un estratto di quelli più importanti.

Query 1

Per eseguire questa query abbiamo dovuto fare un'operazione di GROUP ALL, in modo da permettere a Pig di lavorare con le bag. Questa operazione crea un nuovo `register` suddiviso in due parti: la prima parte contiene tutti i valori raggruppati, la seconda contiene una bag che contiene tutti i valori del gruppo. Abbiamo poi generato la media della colonna funding ed eseguito il DUMP dei dati ottenuti.

```
1. K = GROUP grantDS ALL;
2. B = FOREACH K GENERATE AVG(grantDS.funding);
3. DUMP B;
```

Query *2*

La query 2 è stata così eseguita:

```
1. funders =  FOREACH grantDS GENERATE funder;
2. uniq_funders = DISTINCT funders;
3. group_funders = GROUP uniq_funders ALL;
4. count_funders = FOREACH group_funders GENERATE COUNT(uniq_funders);
5. DUMP count_funders;
```

Prima di tutto si estrae la sola colonna dei finanziatori e utilizza il comando DISTINCT per individuare i singoli finanziatori. Infine si utilizza GROUP ALL per generare una bag con cui Pig sappia lavorare e si esegue il comando Count per ottenere il numero di finanziatori presenti.

Query *3*

La query 3 è stata così eseguita:

```
1. funderGrp = GROUP grantDS BY funder;
2. fundingCount = FOREACH funderGrp GENERATE GROUP AS funder,
3.                         SUM(grantDS.funding) AS total,
   COUNT(grantDS.grant_id) AS numb_proj;
4.
5.
6. filled = FOREACH fundingCount GENERATE funder, (total IS NULL ? 0 : total) AS
   total, (numb_proj IS NULL ? 0 : numb_proj) AS numb_proj;
7.
8. sorted = FOREACH (ORDER filled BY total DESC) GENERATE funder, total,
   numb_proj, (numb_proj == 0 ? 0 : total / numb_proj) AS Ratio_per_proj;
9.
10.DUMP sorted;
```

Possiamo osservarne l'output all'interno della figura 10

```
2023-05-11 18:14:52,373 [main] INFO  org.apache.pig.backend.hadoop.executionengine
(European Commission,2863366132,630,4545025)
(Ministry of Education,495857743,2049,241999)
(Medical Research Council,80444483,12,6703706)
(Engineering and Physical Sciences Research Council,31308013,18,1739334)
(European Research Council,25625750,19,1348723)
(Biotechnology and Biological Sciences Research Council,13257773,11,1205252)
(Telethon Foundation,11549052,65,177677)
(Economic and Social Research Council,4896585,2,2448292)
(Swiss National Science Foundation,2184410,5,436882)
(Natural Environment Research Council,1268127,5,253625)
(Science and Technology Facilities Council,702280,3,234093)
(United States Department of the Navy,530536,4,132634)
(Foundation Fighting Blindness,525869,2,262934)
(Australian Research Council,253794,1,253794)
(United States Air Force,253787,1,253787)
(Arts and Humanities Research Council,165404,1,165404)
(Juvenile Diabetes Research Foundation,125997,1,125997)
(American Heart Association,110985,1,110985)
(Fundação para a Ciência e Tecnologia,101456,2,50728)
(Crohn's and Colitis Foundation,99002,1,99002)
(Brain & Behavior Research Foundation,60960,1,60960)
(Japan Society for the Promotion of Science,36982,1,36982)
(Breast Cancer Now,20378,1,20378)
(Qatar National Research Fund,0,1,0)
(Agricultural Research Service,0,1,0)
(Belgian Federal Science Policy Office,0,1,0)
(Deutsche Forschungsgemeinschaft,0,12,0)
(Italian Association for Cancer Research,0,30,0)
(National Aeronautics and Space Administration,0,3,0)
```

Figura 10 - Estratto dell'output della query 3 in Pig

Query 4

La query 4 è stata così eseguita:

```
1. grantDS2 = FOREACH grantDS GENERATE col1, grant_id, funding, s_year, e_year,
   city, funder, funder_country, (e_year - s_year) + 1 AS duration;
2.
3. grantDS2 = FOREACH grantDS2 GENERATE col1, grant_id, funding, s_year, e_year,
   city, funder, funder_country, (duration < 0 ? 0 : duration) AS duration;
4.
5. grantDS2 = FOREACH grantDS2 GENERATE col1, grant_id, funding, s_year, e_year,
   city, funder, funder_country, duration, (funding IS NULL OR duration IS NULL
   OR duration == 0 ? 0 : funding / duration) AS ratio;
6.
7. grantDS3 = FOREACH grantDS2 GENERATE col1, grant_id, funding, s_year, e_year,
   funder, funder_country, duration, ratio;
8. OUT = LIMIT grantDS3 10;
9. dump OUT;
```

Possiamo osservarne un estratto dell'output nella figura 11

```
2023-05-11 18:16:34,861 [main] INFO  org.apache.pig.backend.hadoop.executione
(0,grant.13242069,10632000,2023,2027,European Commission,Belgium,5,2126400)
(1,grant.13046237,10815148,2023,2028,European Commission,Belgium,6,1802524)
(2,grant.12941744,2062404,2023,2027,European Commission,Belgium,5,412480)
(3,grant.12941771,2136946,2023,2027,European Commission,Belgium,5,427389)
(4,grant.13242078,34979440,2023,2026,European Commission,Belgium,4,8744860)
(5,grant.13046328,6193128,2023,2026,European Commission,Belgium,4,1548282)
(6,grant.13046324,8098455,2023,2025,European Commission,Belgium,3,2699485)
(7,grant.13046266,10427094,2023,2026,European Commission,Belgium,4,2606773)
(8,grant.13046143,3242373,2023,2025,European Commission,Belgium,3,1080791)
(9,grant.13030153,6858523,2023,2028,European Commission,Belgium,6,1143087)
```

Figura 11 - Estratto dell'output della query 4 in Pig

Query 5

La query 5 è stata così eseguita:

```
1. funderGrp = GROUP grantDS2 BY funder;
2. fundingDurationSum = FOREACH funderGrp GENERATE GROUP AS funder,
   SUM(grantDS2.funding) AS total, SUM(grantDS2.duration) AS total_d;
3. filled = FOREACH fundingDurationSum GENERATE funder, (total IS NULL ? 0 :
   total) AS total, (total_d IS NULL ? 0 : total_d) AS total_d, total / total_d
   AS Ratio2;
4. sorted = FOREACH (ORDER filled BY Ratio2 DESC) GENERATE funder, Ratio2;
```

Possiamo osservarne un estratto dell'output nella figura 12

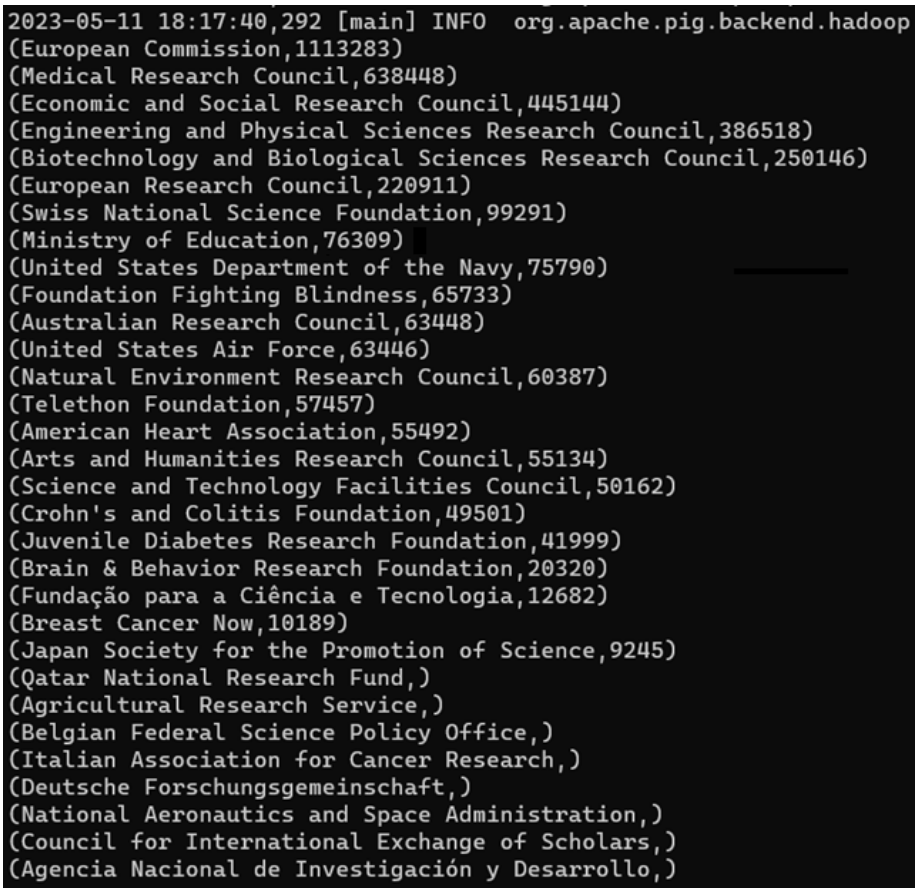

Figura 6 - Estratto dell'output della query 5 in Pig

Query 6

La query 6 è stata così eseguita:

```
1. grantDS2 = FOREACH grantDS GENERATE col1, grant_id, funding, s_year, e_year,
   FLATTEN(TOKENIZE(city, '; ')) AS city, funder, funder_country;
2. cityGrp = GROUP grantDS2 BY city;
3. cityCounts = FOREACH cityGrp GENERATE GROUP AS city, COUNT(grantDS2) AS COUNT;
4. sortedCounts = ORDER cityCounts BY COUNT DESC;
5. DUMP sortedCounts;
```

Possiamo osservarne un estratto dell'output nella figura 13

```
2023-05-11 18:18:21,670
(Naples,3042)
(Rome,1205)
(Milan,898)
(Turin,489)
(London,438)
(Florence,371)
(Pisa,334)
(Paris,330)
(Padova,324)
(Bologna,318)
(Bari,284)
(Caserta,280)
(Genoa,254)
(Catania,245)
(Palermo,242)
(Fisciano,228)
(Pavia,183)
(Perugia,181)
(Trieste,177)
(Madrid,173)
(Munich,168)
(Siena,155)
(Messina,149)
(Rende,140)
(Cagliari,122)
(Cambridge,121)
(Parma,119)
(Barcelona,117)
(Udine,111)
(Lecce,109)
(Trento,107)
(Venice,107)
(Modena,102)
(Ferrara,100)
(Athens,100)
(Catanzaro,99)
(Lisbon,98)
```

Figura 7 - Estratto dell'output della query 6 in Pig

Grafici

Abbiamo plottato dei grafici per le query che risultavano più interessanti, in particolare le query 3, 5 e 6.

Query 3

Per questa query abbiamo disegnato un istogramma a 3 colonne raggruppate, dove possiamo vedere il totale di soldi spesi per ogni ente, il numero di progetti ad esso associato e il rapporto tra questi due. Il grafico è riportato in scala logaritmica per permetterne una visualizzazione migliore, in quanto il range è molto ampio.

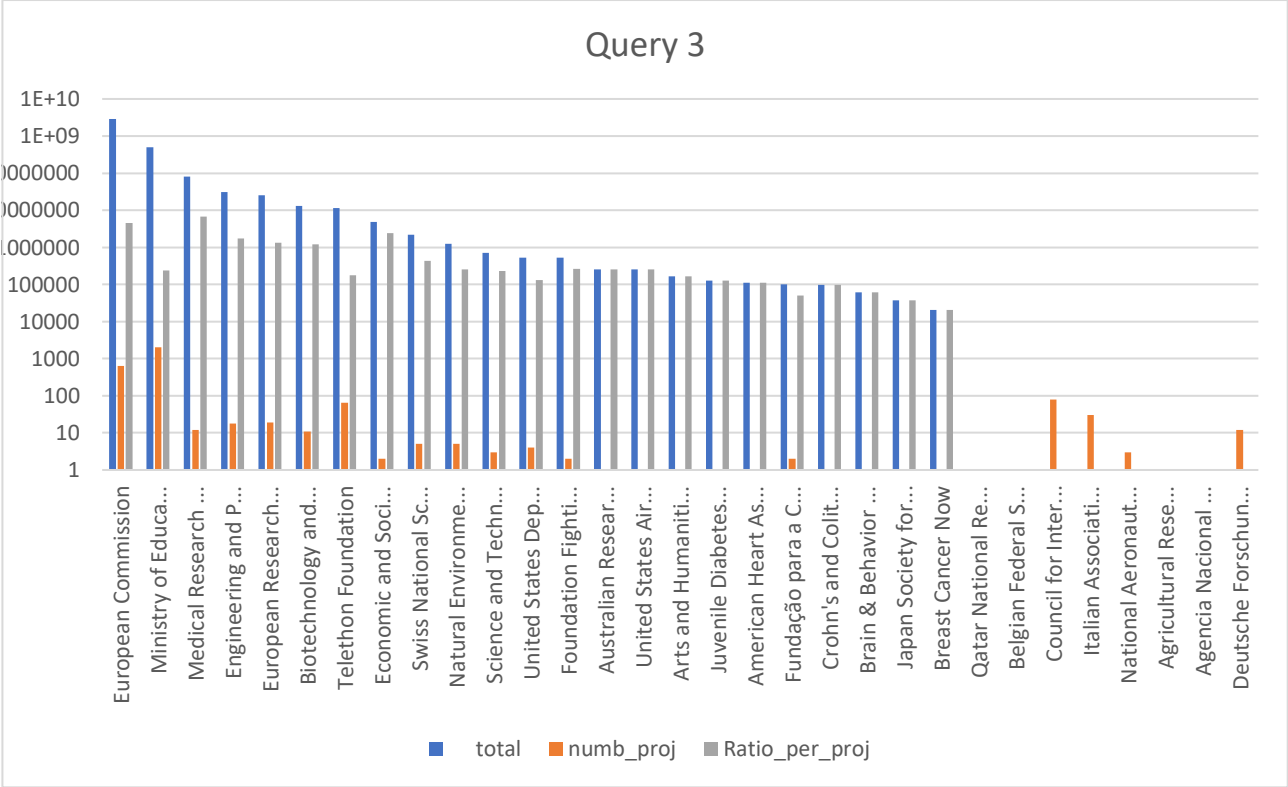

Grafico 1 - Query 3

Query 5

Anche per la query 5 abbiamo optato per un istogramma a 3 barre per rappresentare i soldi totali spesi, il numero di anni totali finanziati e il rapporto tra i due. Per lo stesso motivo di prima il grafico è rappresentato in scala logaritmica.

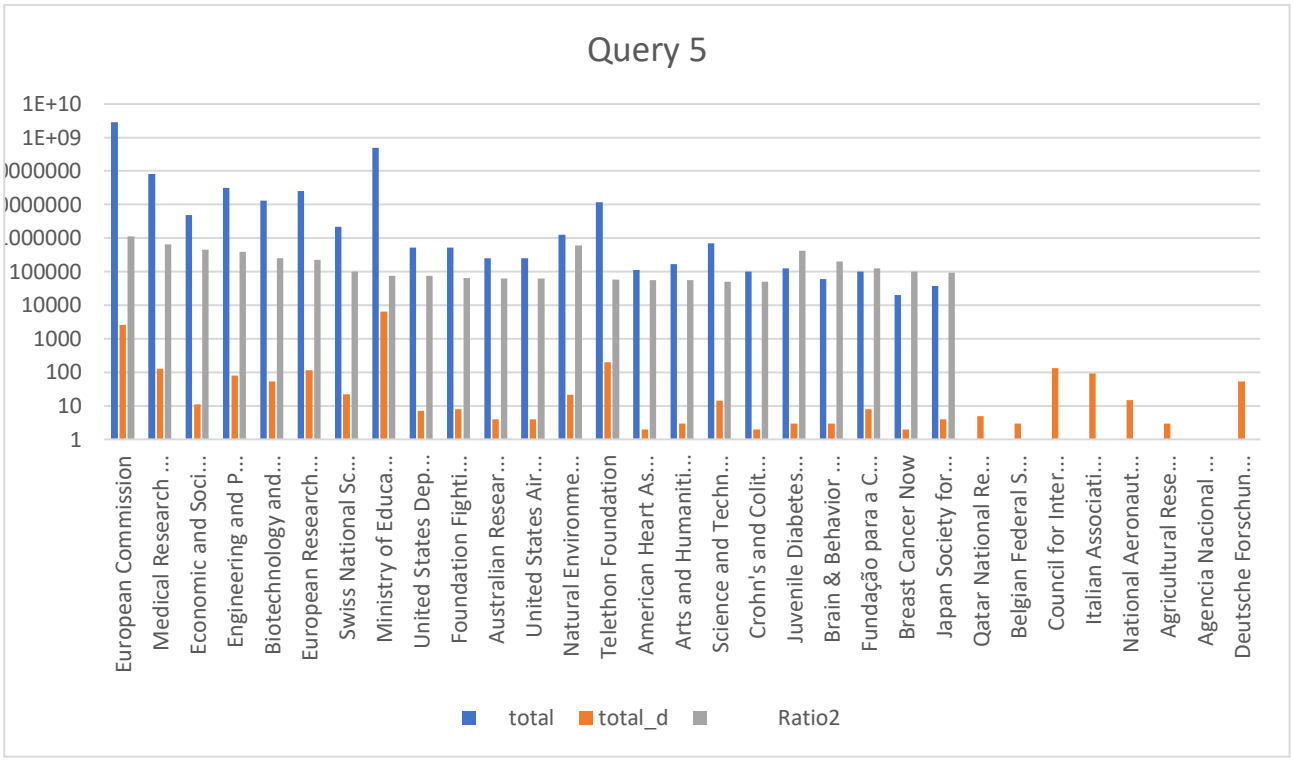

Grafico 2 - Query 5

Query 6

Questo istogramma rappresenta il numero di occorrenze di ogni città nel dataset. Essendo presenti oltre 1000 città distinte, abbiamo scelto di rappresentare solo le prime 15 in questo grafico.

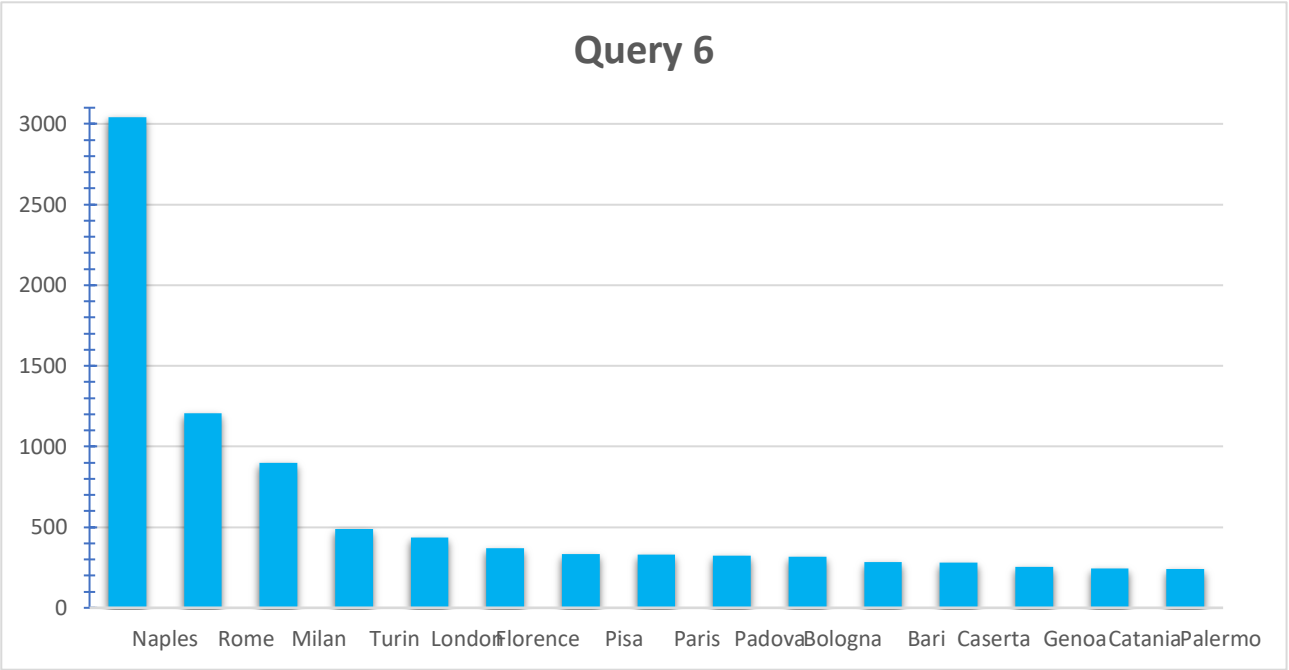

Grafico 3 - Query 6

Riferimenti

Riccio Emanuele, e Ferdinando Tammaro. *Databricks notebook per Hive.* Maggio 2023. https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/7990742217312366/2069306294869435/5560062084567770/latest.html .

—. *Notebook Colab per Pyspark.* Maggio 2023. https://colab.research.google.com/drive/1mT1DOLo9yIw-1TTjoRfD2qsfgptFRl1z?usp=sharing .