Homework BDE 1

Raffaele Russo, Alessandro Vanacore, Giuliano Di Giuseppe

1 Traccia

Eseguire almeno 5 diverse analytics di tipo descrittivo sul dataset relativo ai progetti di ricerca della Federico II. Ogni analytics dovrà essere implementata attraverso Pig, HIVE e PySpark e presentata attraverso apposita reportistica.

1.1 Dashboard

È stata realizzata una dashboard interattiva con streamlit, visibile in figura 1, e accessibile con il link `https://raffaelerusso-dashboard-hw1-app-z09emx.streamlit.app`.

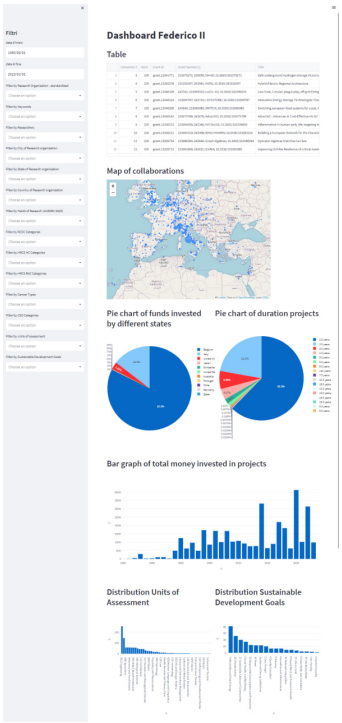

Figura 1: Dashboard

Figura 2: Dataset

Figura 3: Tabelle per ricercatori, università e stato dell'organizzazione

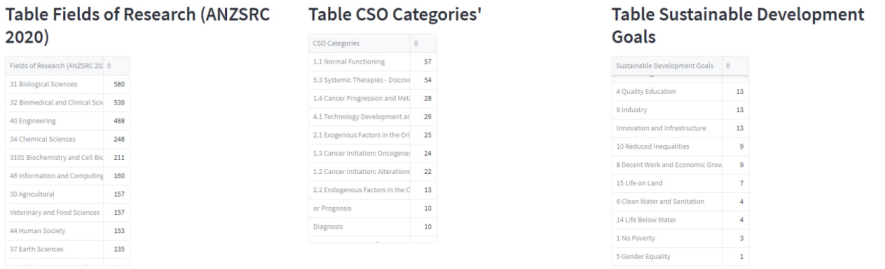

Figura 4: Tabelle per campi di ricerca, categorie CSO e obiettivi di sviluppo sostenibili

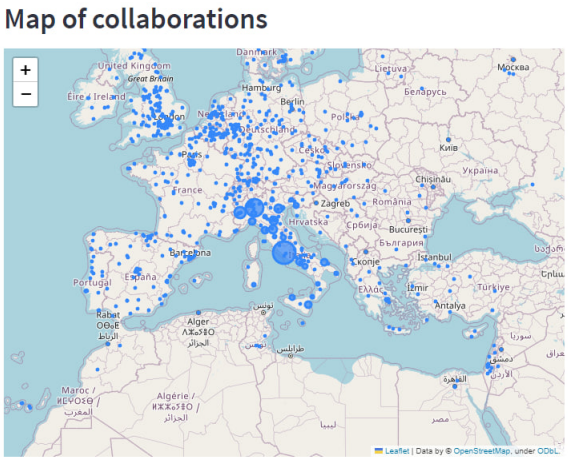

Figura 5: Mappa delle collaborazioni

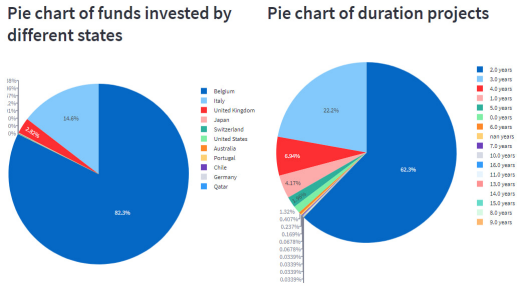

Figura 6: Pie charts fondi investiti per paese e durata dei progetti

Figura 7: Soldi investiti in progetti

Figura 8: Distribuzione unità di valutazione e obiettivi sostenibili di sviluppo

1.2 Analisi del dataset

La prima cosa che abbiamo fatto, dopo aver caricato il dataset fornitoci per l'homework, è stata quella di svolgere una breve analisi di quella che è la sua composizione andando a comprendere al meglio le sue caratteristiche. Il dataset originale contiene 2964 righe e 37 colonne così strutturate:

- Rank: costante e pari a 100 per ogni riga
- GrantID : ID associato a ogni pubblicazione, proprio di ogni riga
- Grant Number : numero del grant con 2885 valori non nulli
- Title : titolo della pubblicazione, 2905 differenti, sono presenti 59 duplicati
- Title traslate: si lavorerà eventualmente su questa versione tradotta del titolo
- Abstract: è una breve sintesi del testo , sono presenti 70 record con lo stesso abstract, mentre 228 record con l'abstract Nan
- Abstract translate: sono presenti 72 record con lo stesso abstract, mentre 158 record con l'abstract Nan
- Keywords: sono parole chiavi per identificare il progetto, solo 86 valori non sono nulli e abbiamo 53 valori unici
- Funding Amount: si riferisce al finanziamento ricevuto, non sono presenti valori nulli
- Currency: la valuta del finanziamento , sono presenti 2836 valori non nulli, abbiamo 7 tipi di valute

- Funding Amount in EUR: fondi ricevuti in euro, sono 2679 record non nulle
- Start Date: giorno d'inizio
- Start Year: anno d'inizio, 53 anni diversi
- End Date: giorno di fine
- End Year: anno di fine, 56 anni diversi
- Researchers: nome dei ricercatori che hanno partecipato al progetto, sono presenti 2334 valori non nulli
- Research Organization - original: sono presenti 2592 organizzazioni di ricerca
- Research Organization - standardized: sono presenti 2592 organizzazioni di ricerca standardizzate
- GRID ID: Global Research Identifier Database (GRID) non sono presenti valori nulli e vengono utilizzate 2581 DB diversi
- City of Research organization
- State of Research organization
- Country of Research organization: paese che ha ricevuto il finanziamento
- Funder: finanziatore
- Funder Group: gruppo finanziatore, che comprende 31 finanziatori diversi
- Funder Country: paese finanziario, comprende 11 gruppi finanziari diversi
- Program: comprende 400 programmi diversi
- Resulting publications
- Source Linkout: sono presenti 900 link
- Dimensions URL: non sono presenti valori nulli e sono tutti diversi tra loro
- Fields of Research (ANZSRC 2020): contiene 2528 paper che hanno ricevuto un premio, sono stati assegnati 856 premi
- RCDC Categories: Research, Condition, and Disease Categorization, contiene 532 valori non nulli
- HRCS HC Categories Health Research Classification System Health, contiene 523 valori non nulli

- HRCS RAC Categories: Health Research Classification System Research Activity contiene 480 valori non nulli
- Cancer Types: sono presenti 256 tuple sul cancro
- CSO Categories: 212 categorie
- Units of Assessment: sono presenti 937 criteri di valutazione
- Sustainable Development Goals: sono presenti 299 progetti con obiettivi sostenibili e 27 obiettivi

1.3 Lettura del dataset PIG

Viene riportata l'istruzione necessaria al caricamento del dataset su PIG:

```
data = LOAD './dataset.csv'
USING PigStorage('|') AS (
Rank: int,
Grant_ID: chararray,
Grant_Number: chararray,
Title: chararray,
Title_translated: chararray,
Abstract: chararray,
Abstract_translated: chararray,
Keywords: chararray,
Funding_Amount: float,
Currency: chararray,
Funding_Amount_in_EUR: float,
Start_Date: chararray,
Start_Year: int,
End_Date: int,
End_Year: int,
Researchers: chararray,
Research_Organization_original: chararray,
Research_Organization_standardized: chararray,
GRID_ID: chararray,
City_of_Research_organization: chararray,
State_of_Research_organization: chararray,
Country_of_Research_organization: chararray,
Funder: chararray,
Funder_Group: chararray,
Funder_Country: chararray,
Program: chararray,
Resulting_publications: chararray,
Source_Linkout: chararray,
Dimensions_URL: chararray,
Fields_of_Research_ANZSRC_2020: chararray,
RCDC_Categories: chararray,
HRCS_HC_Categories: chararray,
HRCS_RAC_Categories: chararray,
Cancer_Types: chararray,
CSO_Categories: chararray,
Units_of_Assessment: chararray,
Sustainable_Development_Goals: chararray
);
```

2 Query base

2.1 Visualizzazione : Progetti con piu Fondi

La prima query sulla quale abbiamo posto la notra attenzione è stata quella che ci ha permesso di identificare quelli che sono stati i progetti che hanno goduto di più fondi durante il loro sviluppo.

2.1.1 PIG

```
sorted_fund = ORDER data BY Funding_Amount DESC;
output_res = FOREACH sorted_fund
GENERATE Grant_ID, Title_translated, Funding_Amount;
```

```
(grant.12961001,Partnership for the Assessment of Risks from Chemicals,2.0E8)
(grant.7676812,AIRFRAME ITD,1.6097488E8)
(grant.9244722,Human Brain Project Specific Grant Agreement 3,1.5E8)
(grant.8104228,Human Brain Project Specific Grant Agreement 2,8.8E7)
(grant.7834124,Fast Rotorcraft,8.5997328E7)
(grant.9414031,AIRFRAME ITD,7.5307408E7)
(grant.9662426,JTI-Clean Sky 1-Green Regional
Aircraft-Integrated Technology Demonstrator,6.913204E7)
(grant.7676805,Regional Aircraft,5.0224604E7)
(grant.6503807,European Human Biomonitoring Initiative,4.9933776E7)
(grant.9065967,Fast Rotorcraft,4.8242712E7)
(grant.3740016,Technology application to the near term business goals
and objectives of the aerospace industry,4.2631312E7)
(grant.3766989,More Open Electrical Technologies,3.7756232E7)
(grant.3789484,Enabling Grids for E-sciencE-II,3.6971364E7)
(grant.13242078,Hybrid-Electric Regional Architecture,3.497944E7)
(grant.3789652,Smart Intelligent Aircraft Structures,3.2434312E7)
(grant.3774746,Enabling Grids for E-sciencE III,3.2E7)
(grant.3760701,Enabling grids For E-science,3.1867E7)
(grant.9065971,REGIONAL AIRCRAFT 2020-2023,2.9969458E7)
```

2.1.2 PySpark

```
df_sorted = df.rdd.sortBy(lambda x: x['Funding_Amount'])
result1 = df_sorted.collect()
```

```
df.createOrReplaceTempView("temp_table")
result1 = spark.sql("SELECT * FROM temp_table
ORDER BY Funding_Amount").collect()
```

2.1.3 HIVE

```
SELECT *
FROM dataset
ORDER BY 'Funding Amount' ASC
```

2.2 Visualizzazione: Per ogni founder country calcolare i soldi investiti totali

Successivamente abbiamo voluto mettere in mostra quali fossero i paesi che avessero investito maggiormente nella ricerca.

2.2.1 PIG

```
Grpd = GROUP data BY Funder_Country;
Smmd = FOREACH Grpd GENERATE group AS
Funder_Country, SUM(data.Funding_Amount) AS sum_fund;
Srtd = ORDER Smmd BY sum_fund DESC;
output_res = FOREACH Srtd GENERATE Funder_Country, sum_fund;
```

```
(Belgium,2.888991882E9)
(Italy,5.07406795E8)
(United Kingdom,9.8143326E7)
(Japan,4810000.0)
(Switzerland,2388849.0)
(United States,1939309.0)
(Australia,388648.0)
(Portugal,101456.0)
(Germany,0.0)
(Qatar,0.0)
(Chile,0.0)
```

2.2.2 Hive

```
SELECT
   Funder_Country AS Funder_Country,
   SUM(Funding_Amount) AS sum_fund
FROM
   dataset
GROUP BY
   Funder_Country
ORDER BY
   sum_fund DESC;
```

2.2.3 PySpark

```
funder_country_sum = df.rdd.map(lambda x:
(x['Funder_Country'], x['Funding_Amount']))
funder_country_sum = funder_country_sum.reduceByKey(lambda x, y: x + y)
funder_country_sum_sorted = funder_country_sum.map(lambda x:
(x[1], x[0])).sortByKey(False)
result2 = funder_country_sum_sorted.collect()
```

```
df.createOrReplaceTempView("temp_table")
funder_country_sum = spark.sql("SELECT Funder_Country, SUM(Funding_Amount) AS
Total_Funding FROM temp_table GROUP BY Funder_Country")
funder_country_sum_sorted =
funder_country_sum.orderBy(funder_country_sum.Total_Funding.desc())
result2 = funder_country_sum_sorted.collect()
```

2.3 Visualizzazione : Per ogni anno calcolare i soldi investiti

La realizzazione di questa query ci ha permesso di estrapolare le quantità di soldi investititi durante gli anni nella ricerca.

2.3.1 PIG

```
data_filtered = FILTER data BY (Start_Year is not null);
Grpd = GROUP data_filtered BY Start_Year;
Smmd = FOREACH Grpd GENERATE group AS
Start_Year, SUM(data_filtered.Funding_Amount) AS sum_fund;
Srtd = ORDER Smmd BY sum_fund DESC;
output_res = FOREACH Srtd GENERATE Start_Year, sum_fund;
DUMP output_res;
```

```
(2020,4.12821195E8)
(2014,3.31822494E8)
(2022,3.140682E8)
(2017,2.21172667E8)
(2018,1.84555616E8)
(2004,1.72466365E8)
(2006,1.67975657E8)
(2008,1.67228516E8)
(2000,1.24884986E8)
(2023,1.22006193E8)
(2010,1.08335575E8)
(2009,1.03489804E8)
(2021,1.02085214E8)
(2007,1.01106737E8)
(2002,9.7124361E7)
(2011,9.2427091E7)
(2016,9.0076472E7)
(2005,8.5702511E7)
(2013,7.7062417E7)
(2012,7.6025069E7)
(2015,6.4836112E7)
(2019,6.3026461E7)
(2001,6.2467011E7)
(1999,4.9228101E7)
(2003,4.8762244E7)
(1993,2.9638922E7)
(1997,1.0611469E7)
(1996,1.033968E7)
(1992,5024139.0)
```

```
(1995,3445140.0)
(1994,2261026.0)
(1998,2029820.0)
(1991,63000.0)
(1990,0.0)
(1989,0.0)
(1988,0.0)
(1987,0.0)
(1986,0.0)
(1985,0.0)
(1983,0.0)
(1982,0.0)
(1980,0.0)
(1979,0.0)
(1978,0.0)
(1977,0.0)
(1975,0.0)
(1974,0.0)
(1973,0.0)
(1972,0.0)
(1967,0.0)
(1966,0.0)
(1965,0.0)
```

2.3.2 Hive

```
SELECT
   Start_Year,
   SUM(Funding_Amount) AS sum_fund
FROM
   dataset
GROUP BY
   Start_Year
ORDER BY
   sum_fund DESC;
```

2.3.3 PySpark

```
start_year_sum = df.rdd.map(lambda x: (x['Start_Year'], x['Funding_Amount']))
start_year_sum = start_year_sum.reduceByKey(lambda x, y: x + y)
start_year_sum_sorted = start_year_sum.map(lambda x:
(x[1], x[0])).sortByKey(False)
result3 = start_year_sum_sorted.collect()
```

```
df.createOrReplaceTempView("temp_table")
start_year_sum = spark.sql("SELECT Start_Year, SUM(Funding_Amount) AS
Total_Funding FROM temp_table GROUP BY Start_Year")
start_year_sum_sorted =
start_year_sum.orderBy(start_year_sum.Total_Funding.desc())
result3 = start_year_sum_sorted.collect()
```

2.4 Visualizzazione: Per ogni anno calcolare il numero dei progetti

L'obiettivo di questa query è andare a mettere in luce quelli che sono i progetti realizzati di anno in anno, per evidenziare gli anni in cui maggiormente ci si è dedicati alla ricerca.

2.4.1 PIG

```
data_filtered = FILTER data BY (Start_Year is not null);
Grpd = GROUP data_filtered BY Start_Year;
Smmd = FOREACH Grpd GENERATE group AS
Start_Year, COUNT(data_filtered.Funding_Amount) AS num_proj;
Srtd = ORDER Smmd BY num_proj DESC;
output_res = FOREACH Srtd GENERATE Start_Year, num_proj;
output_res = FOREACH Srtd GENERATE Start_Year, num_proj;
```

```
(2004,237)
(2002,219)
(2005,196)
(2017,195)
(2000,189)
(2003,186)
(2001,170)
(2008,169)
(2007,164)
(1999,152)
(2006,141)
(2020,107)
(2009,106)
(2010,86)
(2015,78)
(2012,48)
(1996,40)
(2022,37)
(1998,36)
(2021,36)
(1994,31)
(2016,31)
(2011,31)
(1997,31)
(1993,31)
(2018,26)
(2013,25)
(2014,25)
(2019,20)
(2023,18)
(1995,15)
(1992,11)
(1986,9)
(1990,8)
(1988,8)
(1987,7)
(1989,6)
(1973,5)
(1991,4)
(1972,4)
(1980,2)
(1982,2)
(1978,2)
(1977,2)
(1975,2)
(1974,2)
(1965,1)
(1983,1)
(1979,1)
(1985,1)
(1967,1)
(1966,1)
```

2.4.2 Hive

```
SELECT
   Start_Year,
   COUNT(*) AS cnt_fund
FROM
   dataset
GROUP BY
   Start_Year
ORDER BY
   cnt_fund DESC;
```

2.4.3 PySpark

```
start_year_count = df.rdd.map(lambda x: (x['Start_Year'], 1))
start_year_count = start_year_count.reduceByKey(lambda x, y: x + y)
start_year_count_sorted = start_year_count.map(lambda x:
(x[1], x[0])).sortByKey(False)
result4 = start_year_count_sorted.collect()
```

```
df.createOrReplaceTempView("temp_table")
start_year_count = spark.sql("SELECT Start_Year,
COUNT(*) AS Count FROM temp_table GROUP BY Start_Year")
start_year_count_sorted = start_year_count.orderBy
(start_year_count.Count.desc())
result4 = start_year_count_sorted.collect()
```

2.5 Visualizzazione: Distribuzione durata dei progetti

L'ultima query realizzata ci da informazione relativamente alla durata dei singoli progrtti.

2.5.1 PIG

```
data_filtered = FILTER data BY (Start_Year is not null)
AND (End_Year is not null);
data_len = FOREACH data_filtered GENERATE *,
(End_Year - Start_Year) AS Len;
Grpd = GROUP data_len BY Len;
Smmd = FOREACH Grpd GENERATE group AS Len,
COUNT(data_len.Funding_Amount) AS num_proj;
Srtd = ORDER Smmd BY Len ASC;
output_res = FOREACH Srtd GENERATE Len, num_proj;
```

```
(0,39)
(1,124)
(2,1838)
(3,655)
(4,207)
(5,59)
(6,12)
(7,5)
(8,1)
(9,1)
(10,2)
(11,1)
(13,1)
(14,1)
(15,1)
(16,2)
```

2.5.2 Hive

```
SELECT (End_Year - Start_Year) AS Len, COUNT(Funding_Amount) AS num_proj
FROM dataset
WHERE Start_Year IS NOT NULL AND End_Year IS NOT NULL
GROUP BY Len
ORDER BY Len ASC;
```

2.5.3 PySpark

```
#Query 5
df = df.withColumn("Start_Year", df["Start_Year"].cast("int"))
df = df.withColumn("End_Year", df["End_Year"].cast("int"))
len_count = df.rdd.filter(lambda x: x['Start_Year'] is not None and
x['End_Year'] is not None)
len_count = len_count.map(lambda x: ((x['End_Year'] - x['Start_Year']), 1))
len_count = len_count.reduceByKey(lambda x, y: x + y)
len_count_sorted = len_count.map(lambda x: (x[0], x[1])).sortByKey()
result5 = len_count_sorted.collect()
```

```
df.createOrReplaceTempView("temp_table")
len_count = spark.sql("SELECT
(CAST(End_Year AS INT) - CAST(Start_Year AS INT))
AS Length, COUNT(*) AS Count
FROM temp_table WHERE Start_Year IS NOT NULL
AND End_Year IS NOT NULL GROUP BY Length")
len_count_sorted = len_count.orderBy(len_count.Length.asc())
result5 = len_count_sorted.collect()
```

2.6 Visualizzazione: Keywords più presenti tra i progetti

Le successive 9 query si focalizzano sulla distribuzione di alcuni campi, ritenuti da noi più significativi, tra quelli che sono i progetti svolti dalla Federico II durante gli anni.

```
split_data = FOREACH data GENERATE FLATTEN(TOKENIZE(Keywords, ';')) AS
keyword:chararray;
trimmed_data = FOREACH split_data GENERATE TRIM(keyword) AS keyword;
filtered_data = FILTER trimmed_data BY keyword IS NOT NULL AND keyword != '';
counted_data = GROUP filtered_data BY keyword;
counted_data = FOREACH counted_data GENERATE group
AS keyword, COUNT(filtered_data) AS count;
ordered_data = ORDER counted_data BY count DESC;

(Research,72)
(Lecturer,30)
(Engineering,14)
(Biology,12)
(Medical Sciences,11)
(Chemistry,4)
(Linguistics,4)
(American Literature,4)
(History,4)
(Architecture,4)
...
```

2.6.1 Hive

```
SELECT Keywords, COUNT(*) AS count
FROM (
    SELECT TRIM(type) AS Keywords
    FROM (
        SELECT explode(split(`Keywords`, ';')) AS type
        FROM `dataset_header_csv`
        WHERE `Keywords` IS NOT NULL AND `Keywords` != ''
      )
) t
GROUP BY Keywords
ORDER BY count DESC;
```

2.6.2 PySpark

```
split_data =
df.filter("Keywords IS NOT NULL AND Keywords !=
'"").select(split(col("Keywords"), "\\|").alias("split_types"))
exploded_data =
split_data.select(explode(col("split_types")).alias("Keywords"))
unique_data =
exploded_data.select(trim(col("Keywords")).alias("Keywords")).distinct()
df.createOrReplaceTempView("Keywords_data")
split_data = spark.sql("SELECT explode(split(Keywords, ';'))
as Keywords FROM Keywords_data")

counts = (
    split_data
      .select(trim(col("Keywords")).alias("Keywords"))
      .groupBy("Keywords")
      .count()
      .orderBy(desc("count"))
)

counts.show(truncate=False)
```

2.7 Visualizzazione: Distribuzione fields of research

2.7.1 PIG

```
split_data = FOREACH data
GENERATE FLATTEN(TOKENIZE(Fields_of_Research_ANZSRC_2020,';'))
AS field:chararray;
trimmed_data = FOREACH split_data GENERATE TRIM(field) AS field;
filtered_data = FILTER trimmed_data BY field IS NOT NULL AND field != '';
counted_data = GROUP filtered_data BY field;
counted_data = FOREACH counted_data GENERATE group AS
field, COUNT(filtered_data) AS count;
ordered_data = ORDER counted_data BY count DESC;

(31 Biological Sciences,581)
(32 Biomedical and Clinical Sciences,540)
(40 Engineering,489)
(34 Chemical Sciences,248)
```

```
(3101 Biochemistry and Cell Biology,211)
(46 Information and Computing Sciences,164)
(30 Agricultural, Veterinary and Food Sciences,157)
(44 Human Society,153)
(37 Earth Sciences,135)
(33 Built Environment and Design,129)
...
```

2.7.2 Hive

```
SELECT Fields_of_Research_ANZSRC_2020, COUNT(*) AS count
FROM (
    SELECT TRIM(type) AS Fields_of_Research_ANZSRC_2020
    FROM (
        SELECT explode(split(`Fields_of_Research_ANZSRC_2020`, ';')) AS type
        FROM `dataset_header_csv`
        WHERE `Fields_of_Research_ANZSRC_2020` IS NOT NULL
        AND `Fields_of_Research_ANZSRC_2020` != ''
      )
) t
GROUP BY Fields_of_Research_ANZSRC_2020
ORDER BY count DESC;
```

2.7.3 PySpark

```
split_data = df.filter("Fields_of_Research_ANZSRC_2020 IS NOT NULL
AND Fields_of_Research_ANZSRC_2020 != ''").select(split(col("Fields_of_Research_ANZSRC_2020"),
"\\|").alias("split_types"))
exploded_data = split_data.select(explode(col("split_types")).alias("Fields_of_Research_ANZSRC_2020"))
unique_data =
exploded_data.select(trim(col("Fields_of_Research_ANZSRC_2020"))
.alias("Fields_of_Research_ANZSRC_2020")).distinct()
df.createOrReplaceTempView("Fields_of_Research_ANZSRC_2020_data")
split_data = spark.sql("SELECT explode(split(Fields_of_Research_ANZSRC_2020, ';'))
as Fields_of_Research_ANZSRC_2020 FROM Fields_of_Research_ANZSRC_2020_data")

counts = (
    split_data
      .select(trim(col("Fields_of_Research_ANZSRC_2020")).alias("Fields_of_Research_ANZSRC_2020"))
      .groupBy("Fields_of_Research_ANZSRC_2020")
      .count()
      .orderBy(desc("count"))
)

counts.show(truncate=False)
```

2.8 Visualizzazione: Distribuzione Sustainable Development Goals

2.8.1 PIG

```
split_data = FOREACH data
GENERATE FLATTEN(TOKENIZE(Sustainable_Development_Goals,';'))
AS goal:chararray;
trimmed_data = FOREACH split_data GENERATE TRIM(goal) AS goal;
filtered_data = FILTER trimmed_data BY goal IS NOT NULL AND goal != '';
counted_data = GROUP filtered_data BY goal;
counted_data = FOREACH counted_data GENERATE group
AS goal, COUNT(filtered_data) AS count;
ordered_data = ORDER counted_data BY count DESC;

(7 Affordable and Clean Energy,82)
(13 Climate Action,53)
(11 Sustainable Cities and Communities,40)
(3 Good Health and Well Being,34)
(12 Responsible Consumption and Production,25)
(16 Peace, Justice and Strong Institutions,23)
(2 Zero Hunger,21)
(4 Quality Education,15)
(9 Industry, Innovation and Infrastructure,13)
(10 Reduced Inequalities,9)
(8 Decent Work and Economic Growth,9)
(15 Life on Land,7)
(14 Life Below Water,4)
(6 Clean Water and Sanitation,4)
```

```
(1 No Poverty,3)
(5 Gender Equality,1)
```

2.8.2 Hive

```
SELECT Sustainable_Development_Goals, COUNT(*) AS count
FROM (
    SELECT TRIM(type) AS Sustainable_Development_Goals
    FROM (
        SELECT explode(split(`Sustainable_Development_Goals`, ';')) AS type
        FROM `dataset_header_csv`
        WHERE `Sustainable_Development_Goals` IS NOT NULL AND `Sustainable_Development_Goals` != ''
      )
) t
GROUP BY Sustainable_Development_Goals
ORDER BY count DESC;
```

2.8.3 PySpark

```
split_data = df.filter("Sustainable_Development_Goals IS NOT NULL
AND Sustainable_Development_Goals != ''").select(split(col("Sustainable_Development_Goals"),
"\\|").alias("split_types"))
exploded_data = split_data.select(explode(col("split_types")).alias("Sustainable_Development_Goals"))
unique_data =
exploded_data.select(trim(col("Sustainable_Development_Goals"))
.alias("Sustainable_Development_Goals")).distinct()
df.createOrReplaceTempView("Sustainable_Development_Goals_data")
split_data = spark.sql("SELECT explode(split(Sustainable_Development_Goals, ';'))
as Sustainable_Development_Goals FROM Sustainable_Development_Goals_data")

counts = (
    split_data
      .select(trim(col("Sustainable_Development_Goals")).alias("Sustainable_Development_Goals"))
      .groupBy("Sustainable_Development_Goals")
      .count()
      .orderBy(desc("count"))
)

counts.show(truncate=False)
```

2.9 Visualizzazione: Distribuzione RCDC Categories

2.9.1 PIG

```
split_data = FOREACH data
GENERATE FLATTEN(TOKENIZE(RCDC_Categories,';')) AS category:chararray;
trimmed_data = FOREACH split_data GENERATE TRIM(category) AS category;
filtered_data = FILTER trimmed_data BY category IS NOT NULL AND category != '';
counted_data = GROUP filtered_data BY category;
counted_data = FOREACH counted_data GENERATE group
AS category, COUNT(filtered_data) AS count;
ordered_data = ORDER counted_data BY count DESC;

RCDC_Categories,
(Genetics,146)
(Biotechnology,99)
(Clinical Research,76)
(Cancer,74)
(Rare Diseases,72)
(Neurosciences,60)
(Prevention,55)
(Brain Disorders,45)
(Bioengineering,44)
(Digestive Diseases,41)
...
```

2.9.2 Hive

```
SELECT RCDC_Categories, COUNT(*) AS count
FROM (
    SELECT TRIM(type) AS RCDC_Categories
    FROM (
        SELECT explode(split(`RCDC_Categories`, ';')) AS type
        FROM `dataset_header_csv`
        WHERE `RCDC_Categories` IS NOT NULL AND `RCDC_Categories` != ''
    )
) t
GROUP BY RCDC_Categories
ORDER BY count DESC;
```

2.9.3 PySpark

```
split_data = df.filter("RCDC_Categories IS NOT NULL
AND RCDC_Categories != ''").select(split(col("RCDC_Categories"), "\\|").alias("split_types"))
exploded_data = split_data.select(explode(col("split_types")).alias("RCDC_Categories"))
unique_data =
exploded_data.select(trim(col("RCDC_Categories")).alias("RCDC_Categories")).distinct()
df.createOrReplaceTempView("RCDC_Categories_data")
split_data = spark.sql("SELECT explode(split(RCDC_Categories, ';')) as RCDC_Categories FROM RCDC_Categories_data")

counts = (
    split_data
      .select(trim(col("RCDC_Categories")).alias("RCDC_Categories"))
      .groupBy("RCDC_Categories")
      .count()
      .orderBy(desc("count"))
)

counts.show(truncate=False)
```

2.10 Visualizzazione: Distribuzione HRCS HC Categories

2.10.1 PIG

```
split_data = FOREACH data
GENERATE FLATTEN(TOKENIZE(HRCS_HC_Categories,';')) AS category:chararray;
trimmed_data = FOREACH split_data GENERATE TRIM(category) AS category;
filtered_data = FILTER trimmed_data BY category IS NOT NULL AND category != '';
counted_data = GROUP filtered_data BY category;
counted_data = FOREACH counted_data GENERATE group
AS category, COUNT(filtered_data) AS count;

(Generic health relevance,118)
(Cancer,114)
(Cardiovascular,84)
(Neurological,55)
(Infection,50)
(Inflammatory and immune system,48)
(Metabolic and endocrine,41)
(Oral and gastrointestinal,27)
(Musculoskeletal,17)
(Mental health,14)
(Reproductive health and childbirth,12)
(Congenital,12)
(Stroke,12)
(Eye,8)
(Respiratory,6)
(Renal and urogenital,4)
(Ear,1)
(Injuries and accidents,1)
(Skin,1)
```

2.10.2 Hive

```
SELECT HRCS_HC_Categories, COUNT(*) AS count
FROM (
    SELECT TRIM(type) AS HRCS_HC_Categories
    FROM (
        SELECT explode(split(`HRCS_HC_Categories`, ';')) AS type
        FROM `dataset_header_csv`
        WHERE `HRCS_HC_Categories` IS NOT NULL AND `HRCS_HC_Categories` != ''
    )
) t
GROUP BY HRCS_HC_Categories
ORDER BY count DESC;
```

2.10.3 PySpark

```
split_data = df.filter("HRCS_HC_Categories IS NOT NULL
AND HRCS_HC_Categories != ''").select(split(col("HRCS_HC_Categories"),
"\\|").alias("split_types"))
exploded_data = split_data.select(explode(col("split_types")).alias("HRCS_HC_Categories"))
unique_data =
exploded_data.select(trim(col("HRCS_HC_Categories"))
.alias("HRCS_HC_Categories")).distinct()
df.createOrReplaceTempView("HRCS_HC_Categories_data")
split_data = spark.sql("SELECT explode(split(HRCS_HC_Categories, ';'))
as HRCS_HC_Categories FROM HRCS_HC_Categories_data")

counts = (
    split_data
      .select(trim(col("HRCS_HC_Categories")).alias("HRCS_HC_Categories"))
      .groupBy("HRCS_HC_Categories")
      .count()
      .orderBy(desc("count"))
)

counts.show(truncate=False)
```

2.11 Visualizzazione: Distribuzione HRCS RAC Categories

2.11.1 PIG

```
split_data = FOREACH data
GENERATE FLATTEN(TOKENIZE(HRCS_RAC_Categories,';')) AS category:chararray;
trimmed_data = FOREACH split_data GENERATE TRIM(category) AS category;
filtered_data = FILTER trimmed_data BY category IS NOT NULL AND category != '';
counted_data = GROUP filtered_data BY category;
counted_data = FOREACH counted_data GENERATE group
AS category, COUNT(filtered_data) AS count;
ordered_data = ORDER counted_data BY count DESC;

(2.1 Biological and endogenous factors,233)
(1.1 Normal biological development and functioning,120)
(5.1 Pharmaceuticals,82)
(4.1 Discovery and preclinical testing of markers and technologies,41)
(2.2 Factors relating to the physical environment,29)
(5.2 Cellular and gene therapies,24)
(6.1 Pharmaceuticals,16)
(4.2 Evaluation of markers and technologies,10)
(3.3 Nutrition and chemoprevention,7)
(2.6 Resources and infrastructure (aetiology),6)
...
```

2.11.2 Hive

```
SELECT HRCS_RAC_Categories, COUNT(*) AS count
FROM (
    SELECT TRIM(type) AS HRCS_RAC_Categories
    FROM (
        SELECT explode(split(`HRCS_RAC_Categoriess`, ';')) AS type
        FROM `dataset_header_csv`
        WHERE `HRCS_RAC_Categories` IS NOT NULL AND `HRCS_RAC_Categories` != ''
      )
) t
GROUP BY HRCS_RAC_Categories
ORDER BY count DESC;
```

2.11.3 PySpark

```
split_data = df.filter("HRCS_RAC_Categories IS NOT NULL
AND HRCS_RAC_Categories != ''").select(split(col("HRCS_RAC_Categories"),
"\\|").alias("split_types"))
exploded_data = split_data.select(explode(col("split_types")).alias("HRCS_RAC_Categories"))
unique_data =
exploded_data.select(trim(col("HRCS_RAC_Categories"))
.alias("HRCS_RAC_Categories")).distinct()
df.createOrReplaceTempView("HRCS_RAC_Categories_data")
split_data = spark.sql("SELECT explode(split(HRCS_RAC_Categories, ';'))
```

```
as HRCS_RAC_Categories FROM HRCS_RAC_Categories_data")

counts = (
    split_data
      .select(trim(col("HRCS_RAC_Categories")).alias("HRCS_RAC_Categories"))
      .groupBy("HRCS_RAC_Categories")
      .count()
      .orderBy(desc("count"))
)

counts.show(truncate=False)
```

2.12 Visualizzazione: Distribuzione Cancer Types

2.12.1 PIG

```
split_data = FOREACH data
GENERATE FLATTEN(TOKENIZE(Cancer_Types,';')) AS type:chararray;
trimmed_data = FOREACH split_data GENERATE TRIM(type) AS type;
filtered_data = FILTER trimmed_data BY type IS NOT NULL AND type != '';
counted_data = GROUP filtered_data BY type;
counted_data = FOREACH counted_data GENERATE group
AS type, COUNT(filtered_data) AS count;
ordered_data = ORDER counted_data BY count DESC;

(Not Site-Specific Cancer,122)
(Thyroid Cancer,23)
(Breast Cancer,22)
(Liver Cancer,19)
(Colon and Rectal Cancer,14)
(Stomach Cancer,8)
(Brain Tumor,8)
(Lung Cancer,7)
(Oral Cavity and Lip Cancer,7)
(Leukemia / Leukaemia,7)
(Esophageal / Oesophageal Cancer,6)
...
```

2.12.2 Hive

```
SELECT Cancer_Type, COUNT(*) AS count
FROM (
    SELECT TRIM(type) AS Cancer_Type
    FROM (
        SELECT explode(split(`Cancer Types`, ';')) AS type
        FROM `dataset_header_csv`
        WHERE `Cancer Types` IS NOT NULL AND `Cancer Types` != ''
      )
) t
GROUP BY Cancer_Type
ORDER BY count DESC;
```

2.12.3 PySpark

```
# esegui la query per estrarre i tipi di cancro univoci
split_data = df.filter("Cancer_Types IS NOT NULL
AND Cancer_Types != ''").select(split(col("Cancer_Types"),
"\\|").alias("split_types"))
exploded_data = split_data.select(explode(col("split_types")).alias("Cancer_Type"))
unique_cancer_types =
exploded_data.select(trim(col("Cancer_Type")).alias("Cancer_Type")).distinct()


# create a temporary view with the split data
df.createOrReplaceTempView("cancer_data")
split_data = spark.sql("SELECT explode(split(Cancer_Types, ';')) as Cancer_Type
FROM cancer_data")

# group by and count the cancer types
cancer_type_counts = (
    split_data
      .select(trim(col("Cancer_Type")).alias("Cancer_Type"))
      .groupBy("Cancer_Type")
      .count()
      .orderBy(desc("count"))
)
```

```
# show the results
cancer_type_counts.show(truncate=False)
```

2.13 Visualizzazione: Distribuzione CSO

2.13.1 PIG

```
split_data = FOREACH data GENERATE FLATTEN(TOKENIZE(CSO_Categories, ';')) AS category:chararray;
trimmed_data = FOREACH split_data GENERATE TRIM(category) AS category;
filtered_data = FILTER trimmed_data BY category IS NOT NULL AND category != '';
counted_data = GROUP filtered_data BY category;
counted_data = FOREACH counted_data GENERATE group AS category, COUNT(filtered_data) AS count;
ordered_data = ORDER counted_data BY count DESC;

(1.1 Normal Functioning,57)
(5.3 Systemic Therapies - Discovery and Development,54)
(1.4 Cancer Progression and Metastasis,28)
(4.1 Technology Development and/or Marker Discovery,26)
(2.1 Exogenous Factors in the Origin and Cause of Cancer,25)
(1.3 Cancer Initiation: Oncogenes and Tumor Suppressor Genes,24)
(1.2 Cancer Initiation: Alterations in Chromosomes,22)
(2.2 Endogenous Factors in the Origin and Cause of Cancer,13)
(4.4 Resources and Infrastructure Related to Detection, Diagnosis, or Prognosis,10)
(1.5 Resources and Infrastructure,9)
...
```

2.13.2 HIVE

```
SELECT CSO_Categories, COUNT(*) AS count
FROM (
    SELECT TRIM(type) AS CSO_Categories
    FROM (
        SELECT explode(split(`CSO_Categories`, ';')) AS type
        FROM `dataset_header_csv`
        WHERE `CSO_Categories` IS NOT NULL AND `CSO_Categories` != ''
      )
) t
GROUP BY CSO_Categories
ORDER BY count DESC;
```

2.13.3 PySpark

```
split_data = df.filter("CSO_Categories IS NOT NULL
AND CSO_Categories != ''").select(split(col("CSO_Categories"),
"\\|").alias("split_types"))
exploded_data = split_data.select(explode(col("split_types")).alias("CSO_Categories"))
unique_data = exploded_data.select(trim(col("CSO_Categories")).alias("CSO_Categories"))
.distinct()
df.createOrReplaceTempView("CSO_Categories_data")
split_data = spark.sql("SELECT explode(split(CSO_Categories, ';')) as CSO_Categories
FROM CSO_Categories_data")

counts = (
    split_data
      .select(trim(col("CSO_Categories")).alias("CSO_Categories"))
      .groupBy("CSO_Categories")
      .count()
      .orderBy(desc("count"))
)

counts.show(truncate=False)
```

2.14 Visualizzazione: distribuzione units of assessment

2.14.1 PIG

```
split_data = FOREACH data GENERATE FLATTEN(TOKENIZE(Units_of_Assessment, ';'))
AS unit:chararray;
trimmed_data = FOREACH split_data GENERATE TRIM(unit) AS unit;
filtered_data = FILTER trimmed_data BY unit IS NOT NULL AND unit != '';
counted_data = GROUP filtered_data BY unit;
counted_data = FOREACH counted_data GENERATE group AS unit, COUNT(filtered_data) AS count;
ordered_data = ORDER counted_data BY count DESC;

(B12 Engineering,258)
(A01 Clinical Medicine,146)
(B11 Computer Science and Informatics,60)
(A06 Agriculture, Veterinary and Food Science,59)
(A03 Allied Health Professions, Dentistry, Nursing and Pharmacy,57)
(A05 Biological Sciences,49)
(B07 Earth Systems and Environmental Sciences,44)
(C13 Architecture, Built Environment and Planning,44)
(C17 Business and Management Studies,34)
(C14 Geography and Environmental Studies,26)
...
```

2.14.2 HIVE

```
SELECT Units_of_Assessment, COUNT(*) AS count
FROM (
    SELECT TRIM(type) AS Units_of_Assessment
    FROM (
        SELECT explode(split(`Units_of_Assessment`, ';')) AS type
        FROM `dataset_header_csv`
        WHERE `Units_of_Assessment` IS NOT NULL AND `Units_of_Assessment` != ''
      )
) t
GROUP BY Units_of_Assessment
ORDER BY count DESC;
```

2.14.3 PySpark

```
split_data = df.filter("Units_of_Assessment IS NOT NULL
AND Units_of_Assessment != ''").select(split(col("Units_of_Assessment"),
"\\|").alias("split_types"))
exploded_data = split_data.select(explode(col("split_types")).alias("Units_of_Assessment"))
unique_data = exploded_data.select(trim(col("Units_of_Assessment"))
.alias("Units_of_Assessment")).distinct()
df.createOrReplaceTempView("Units_of_Assessment_data")
split_data = spark.sql("SELECT explode(split(Units_of_Assessment, ';'))
as Units_of_Assessment FROM Units_of_Assessment_data")

counts = (
    split_data
      .select(trim(col("Units_of_Assessment")).alias("Units_of_Assessment"))
      .groupBy("Units_of_Assessment")
      .count()
      .orderBy(desc("count"))
)

counts.show(truncate=False)
```