
# UNIVERSITÀ DEGLI STUDI DI NAPOLI "FEDERICO II"

SCUOLA POLITECNICA E DELLE SCIENZE DI BASE

CORSO DI BIG DATA ENGINEERING - LAUREA MAGISTRALE IN INGEGNERIA INFORMATICA

ANNO ACCADEMICO 2022-2023


## HOMEWORK 1

## *Big Data Engineering*

*Analytics in Hive, Pig e PySpark di una raccolta di dati relative a tutte le ricerche della Federico II di Napoli*


***Professore:***

Ing. Vincenzo Moscato

***Studenti:***

Antonio Romano M63001315

Andriy Korsun M63001275

Giuseppe Riccio M63001314

Michele Cirillo M63001293

# Indice

# 1 Raccolta e Preprocessing

Contenuti



## 1.1 Raccolta dei dati

Il dataset è una raccolta di tutti gli articoli di ricerca fatti dall'Università degli Studi di Napoli "Federico II" e comprende le seguenti colonne:

- **Rank** - Identificativo dell'Università;
- **Grant ID** - Codice univoco della sovvenzione da parte dell'UE;
- **Grant Number(s)** - Numero della legge con cui è stato sovvenzionato il progetto;
- **Title** - Titolo del progetto;
- **Title translated** - Titolo del progetto tradotto in inglese;
- **Abstract** - Descrizione del progetto;
- **Abstract translated** - Descrizione del progetto tradotto in inglese;
- **Keywords** - Parole chiave con cui è possibile sintetizzare la tematica trattata nel progetto;
- **Funding Amount** - Importo finanziato per il progetto;
- **Currency** - Valuta del finanziamento;
- **Funding Amount in EUR** - Importo finanziato per il progetto in euro;
- **Start Date** - Data inizio progetto;
- **Start Year** - Anno d'inizio del progetto;
- **End Date** - Data di fine del progetto;
- **End Year** - Anno di fine del progetto;
- **Researchers** - Ricercatori coinvolti nel progetto;
- **Research Organization - original** - Lista delle Istituzioni coinvolte nel progetto;
- **Research Organization - standardized** - Lista delle Istituzioni coinvolte nel progetto con Nomenclatura Ufficiale;

- **GRID ID** - Codice univoco assegnato dalla UE per identificare il progetto;
- **City of Research organization** - Città coinvolte nel progetto;
- **State of Research organization** - Regioni coinvolte nel progetto;
- **Country of Research organization** - Paesi coinvolti nel progetto;
- **Funder** - Ente finanziatore;
- **Funder Group** - Gruppo finanziatore;
- **Funder Country** - Paese finanziatore;
- **Program** - Ambito del programma di ricerca del progetto;
- **Resulting publications** - Identificativi delle pubblicazioni con i risultati del progetto;
- **Source Linkout** - URL della pagina del progetto;
- **Dimensions URL** - URL del progetto sulla piattaforma Dimensions;
- **Fields of Research (ANZSRC 2020)** - TOPIC del progetto, Standard Australiani;
- **RCDC Categories** - TOPIC sintetici del progetto, ricerche medicali nel sistema Americano;
- **HRCS HC Categories** - TOPIC sintetici del progetto, ricerche medicali nel sistema UK;
- **HRCS RAC Categories** - TOPIC sintetici del progetto, ricerche medicali nel sistema UK;
- **Cancer Types** - Tipo di tumore studiato nel progetto;
- **CSO Categories** - (Common Scientific Outline) rappresenta il tipo di cura studiata per trattare il cancro;
- **Units of Assessment** - Dipartimenti (Corsi di Laurea) di ricerca coinvolti nel progetto;
- **Sustainable Development Goals** - obbiettivo di sostenibilità perseguito dal progetto.

Il dataset presenta **37 colonne e 2964 righe**.

## 1.2 Preprocessing dei dati

Una fase critica della pipeline di qualsiasi applicazione di Big Data è quella di preprocessing dei dati, in cui vengono identificati e rimossi i dati errati, incompleti o duplicati, colonne costanti e/o irrilevanti ai fini della data analysis. Nel caso in esame, il preprocessing ha portato alla seguente analisi:

- **Colonne eliminate**
  - **Rank** - valori costanti
  - **Grant ID** - è un codice identificativo
  - **Grant Number(s)** - è un numero identificativo
  - **Title** - è il titolo del progetto, viene mantenuto quello tradotto
  - **Abstract** - solo una descrizione del progetto

  - **Abstract translated** - solo una descrizione del progetto
  - **Keywords** - valori spesso nulli
  - **Funding Amount** - viene mantenuto quello in EUR
  - **Currency** - non è di particolare interesse la valuta perchè la maggior parte è EUR
  - **Researchers** - spesso non presenti
  - **GRID ID** - è un codice identificativo
  - **State of Research organization** - per molti paesi il valore è nullo
  - **City of Research organization** - non usato nell'analisi
  - **Program** - sigla del programma all'interno del quale è sovvenzionato il progetto
  - **Resulting publications** - è un codice identificativo
  - **Source Linkout** - è un URL, non esprime nessuna informazione utile
  - **Dimensions URL** - è un URL, non esprime nessuna informazione utile
  - **HRCS HC Categories** - spesso non presenti
  - **HRCS RAC Categories** - spesso non presenti

- **Colonne con valori multipli, NON eliminate**
  - **Research Organization - original (standardized)**
  - **Country of Research organization**
  - **Fields of Research (ANZSRC 2020)**
  - **RCDC Categories**
  - **Cancer Types**
  - **CSO Categories**
  - **Units of Assessment**
  - **Sustainable Development Goals**

Si è fatta quest'analisi per ridurre le dimensioni del dataset in esame e per renderlo più facilmente gestibile in seguito. In particolare il dataset dopo il preprocessing presenta **18 colonne**.

**ATTENZIONE**: Non è stata effettuata alcuna **normalizzazione** sui dati in quanto non è richiesta nessuna predizione, regressione o inferenza.

**SI NOTI**: Il dataset sotto esame, come enunciato precedentemente, non deve essere necessariamente **partizionato e distribuito** in quanto le sue dimensioni sono considerevolmente piccole. Pertanto, si lavorerà con piattaforme Big Data nonostante non siano necessarie per il dataset fornito.

## 1.3 Tecnologie utilizzate

### 1.3.1 Pig

Pig è un framework open-source sviluppato principalmente per l'analisi di grandi dataset in modo semplice ed efficiente, grazie al linguaggio di alto livello Pig Latin, utilizzando il modello di programmazione MapReduce. L'architettura di Pig si basa su diversi componenti principali:

- **Pig Latin**: linguaggio di programmazione di alto livello basato su script che consente di scrivere query SQL, combinando il paradigma di MapReduce e fornendo un'astrazione ad alto livello;
- **Pig Engine**: interprete che esegue i programmi Pig Latin e converte le query in una serie di lavori MapReduce che vengono eseguiti su Hadoop;
- **Parser**: responsabile dell'analisi sintattica e semantica delle query Pig Latin che verifica la correttezza del codice e genera un piano logico che rappresenta le operazioni richieste dall'utente;
- **Optimizer**: l'optimizer riceve il piano logico dal parser e lo ottimizza per migliorare le prestazioni delle query, ad esempio con la rimozione di operazioni ridondanti;
- **Compiler**: prende il piano logico ottimizzato e lo compila in un piano fisico che può essere eseguito dal motore di esecuzione sottostante (MapReduce);
- **Runtime**: componente che effettivamente esegue i lavori generati dal compiler.

Pig viene utilizzato in combinazione con **Hadoop**, framework open-source per lo storage e l'elaborazione di grandi dataset su cluster.
Pig può essere eseguito sia in modalità locale che in modalità MapReduce:

- In modalità locale, Pig esegue le operazioni sul file system locale;
- In modalità MapReduce, Pig esegue le operazioni su un cluster Hadoop.

Nel caso in esame, per utilizzare Pig, viene utilizzato GRUNT, la shell di Pig su cui eseguiremo lo script creato per l'occasione.
Nel dettaglio, per l'esecuzione viene avviato WSL e si eseguono le seguenti operazioni:

- **bin/hdfs namenode -format**: il namenode è il repository centrale dei metadati per HDFS, dove sono immagazzinati i file e directory salvati nel file system. Con questo comando viene formattato il file system del namenode cancellando dati e configurazioni e verrà creato un nuovo file system image. Questo comando è usato solitamente per inizializzare un nuovo cluster Hadoop;
- **sbin/start-dfs.sh**: viene avviato il file system distribuito di Hadoop (HDFS);
- **pig -x local**: viene lanciato Apache Pig in modalità locale, il che significa che tutte le operazioni verranno eseguite sul file system locale senza utilizzare un cluster Hadoop, quindi le operazioni precedenti potrebbero anche essere omesse.

Di seguito la configurazione di Pig - Hadoop in esame:

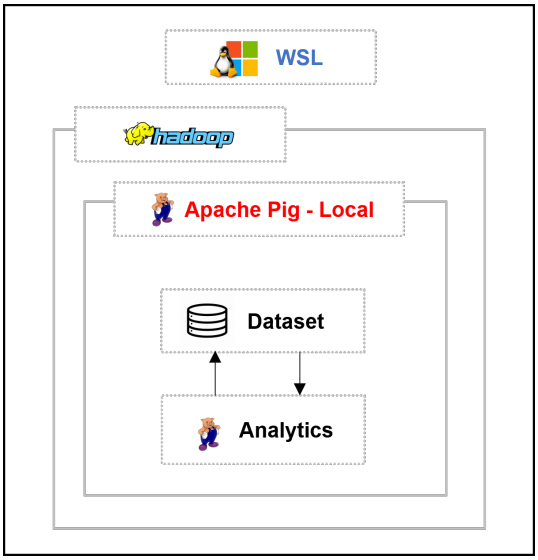


**Figura 1.1:** Pig: la configurazione

#### 1.3.1.1 Workflow: Raccolta dati → WSL → Hadoop → Pig

Pertanto il workflow seguito è così suddiviso:

- **In locale**
  - Raccolta dati;
  - Preprocessing dei dati su Excel.
- **WSL**
  - Configurazione Java, Hadoop e variabili di ambiente;
  - Configurazione Pig;
  - Upload dati;
  - Analytics con Pig Latin.

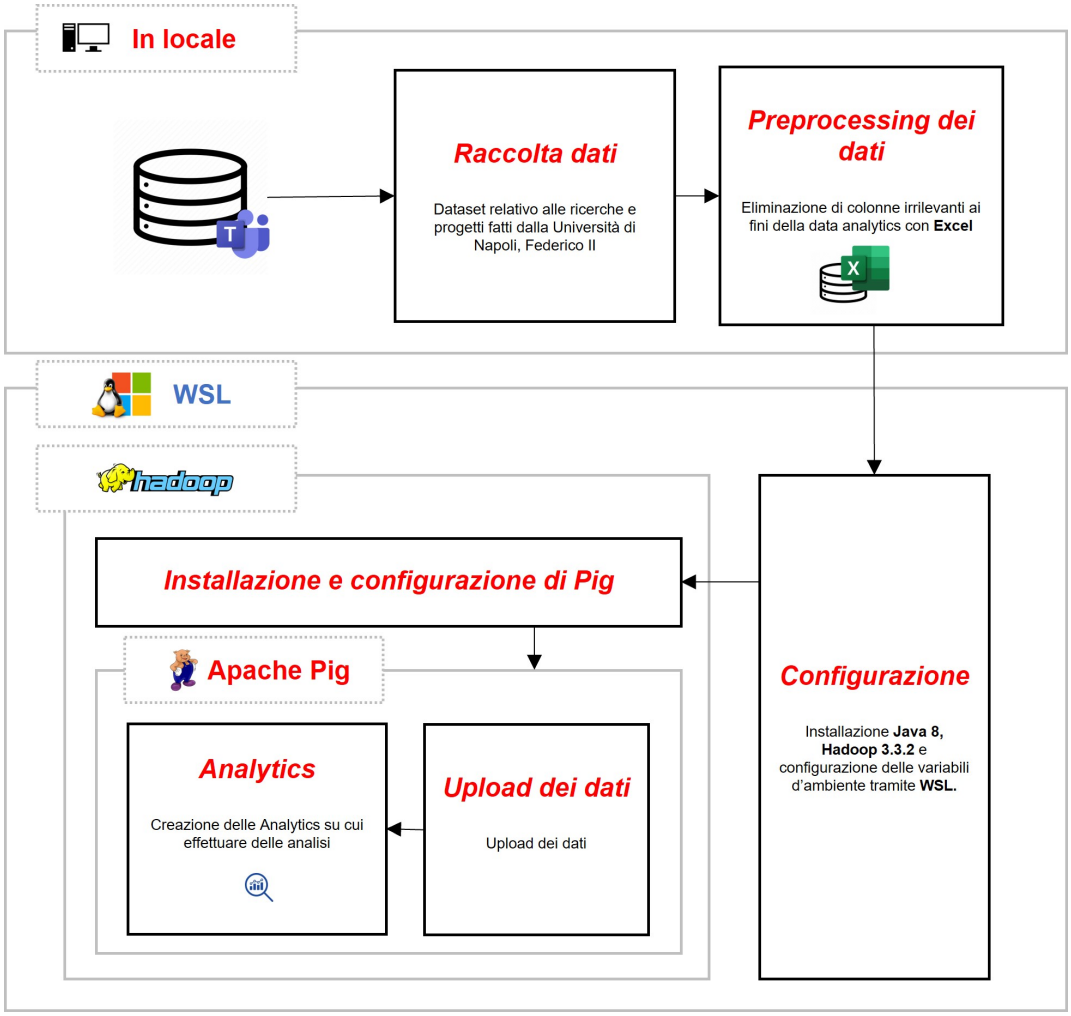


**Figura 1.2:** Workflow - Local to WSL

Come si può notare dal workflow 1.2, si è seguita una configurazione locale per gli stessi motivi descritti in precedenza.

### 1.3.2 Hive

Hive è un framework per l'elaborazione distribuita di grandi quantità di dati. È stato ideato per semplificare lo sviluppo di task di tipo MapReduce su HDFS.

Si ricorda brevemente che Hadoop è un framework software open source che consente l'elaborazione distribuita di grandi set di dati su cluster, ed il suo core è basato sui componenti principali, quali:

1. **Hadoop Distributed File System (HDFS)**: Rappresenta il file system distribuito che archivia grandi quantità di dati sui nodi del cluster;
2. **Hadoop MapReduce**: Modello di programmazione per l'elaborazione di grandi volumi di dati in parallelo. Funziona dividendo un set di dati in piccole parti che possono essere elaborate in parallelo su diversi nodi del cluster:
   - **Fase Map**: Viene preso un set di dati di input e convertito in un altro set di dati, dove gli elementi individuali sono suddivisi in coppie chiave/valore. In questa fase, i dati vengono filtrati e ordinati;
   - **Fase Reduce**: L'output della fase Map viene ulteriormente elaborato, prendendo le coppie chiave/-valore dalla fase Map e combinate in un set più piccolo di coppie chiave/valore. L'output della fase Reduce non è altro che un set di dati che ha subito una sorta di aggregazione o sintesi rispetto all'input originale, costituito da coppie chiave/valore (proprio come l'output della fase Map) ma con un numero molto ridotto di chiavi uniche.

Riprendendo con la descrizione di Hive, verranno elencate di seguito le principali proprietà:

- **Elaborazione distribuita**: Hive è progettato per l'elaborazione distribuita di grandi quantità di dati su cluster di macchine. Utilizza il framework MapReduce di Hadoop per eseguire operazioni parallele e scalabili;
- **SQL-Like**: Hive fornisce un linguaggio di query simile a SQL chiamato HiveQL. Questo permette di scrivere query sui dati strutturati ed effettuare analisi dei dati con un linguaggio di facile uso;
- **Metastore**: Hive utilizza un metastore per archiviare i metadati del sistema, come la definizione delle tabelle, lo schema, le partizioni e le informazioni sui dati. Il metastore è essenziale per l'elaborazione efficiente delle query Hive;
- **Integrazione con Hadoop**: Hive è strettamente integrato con l'ecosistema Hadoop. Utilizza il file system distribuito HDFS per memorizzare i dati e sfrutta il framework MapReduce per l'elaborazione distribuita;
- **Integrazione con altre tecnologie**: Hive può essere integrato con altre tecnologie, come Apache Spark, per fornire funzionalità avanzate di analisi e Machine Learning.

**Nel caso in esame**, per effettuare Analytics in Hive, sarà necessario utilizzare la piattaforma **Databricks** [1]. Essa infatti supporta Apache Hive consentendo di usare e connettersi con il metastore [2] Hive come area di lavoro.
Si crea un cluster su Databricks per gestire grandi volumi di dati e fornire un ambiente di elaborazione distribuita altamente efficiente, scalabile e sicuro per l'esecuzione di analisi di dati.
Su di esso viene caricato il dataset o flusso di dati e un notebook utile per effettuare le analisi del caso con HiveQL.
Nella Figura 1.3 si nota la configurazione Dataset-Notebook Hive.

[1] Per utilizzare Apache Hive è possibile anche installare Apache Hadoop e su di esso usare HiveQL;
[2] Il metastore di Hive è un database che contiene i metadati del sistema Hive, come ad esempio le definizioni delle tabelle Hive, lo schema, i partizionamenti e le informazioni sui dati. Il metastore di Hive viene utilizzato dal server di Hive per tradurre le query SQL in operazioni a basso livello che vengono eseguite sui dati sottostanti. I metadati memorizzati nel metastore sono essenziali per l'elaborazione efficiente delle query Hive.

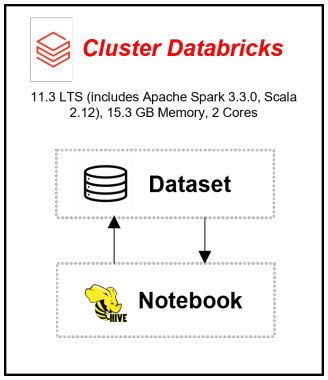


**Figura 1.3:** Databricks: la configurazione

#### 1.3.2.1 Workflow: Raccolta dati → Databricks → Hive

Pertanto il workflow seguito è così suddiviso:

- **In locale**
  - Raccolta dati;
  - Preprocessing dei dati su Excel.
- **Databricks**
  - Creazione Cluster;
  - Caricamento Dataset;
  - Analytics con HiveQL.

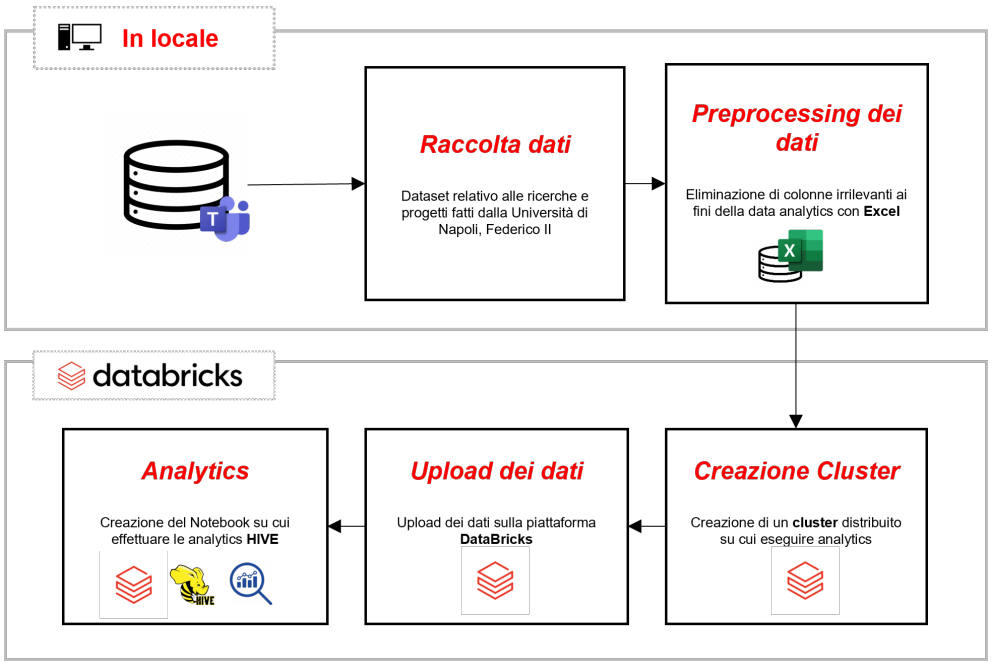


**Figura 1.4:** Workflow - Local to Databricks

### 1.3.3 Spark

Apache Spark è un framework open-source di elaborazione di dati distribuiti, che fornisce un'interfaccia per programmare cluster con API parallele e richiede dati distribuiti. È stato sviluppato per essere veloce e generale, consentendo di sviluppare applicazioni su larga scala per l'elaborazione ed il processamento di flussi di dati.
Nel caso in esame si è utilizzato **PySpark**, ovvero l'interfaccia Python per Spark che fornisce un'API per scrivere applicazioni Spark usando Python. Spark viene utilizzato per diversi motivi, tra cui:

- **Scalabilità**: gestione di grandi volumi di dati distribuiti su un cluster di nodi, sfruttando la potenza di calcolo di computer per elaborare e analizzare dati;
- **Velocità**: grazie alla sua architettura di memoria in-cache, Spark è in grado di elaborare dati molto più velocemente rispetto ad altri framework di elaborazione di dati distribuiti come Hadoop MapReduce;
- **Flessibilità**: supporto a diversi linguaggi di programmazione (Scala, Java, Python e R); offre diverse librerie per l'analisi di dati, come MLlib per il Machine Learning, GraphX per il calcolo su grafi e Spark Streaming per l'elaborazione di flussi di dati in tempo reale;
- **Facilità d'uso**: le API di Spark sono facili da usare e consentono di scrivere codice più pulito e conciso.

Il concetto alla base del modello di programmazione di Spark è quello di creare una pipeline di operazioni per la manipolazione degli RDD:

- **RDD** (Resilient Distributed Dataset): è l'astrazione principale di Spark, che rappresenta un insieme di dati distribuiti e immutabili. Gli RDD possono essere creati, trasformati e oggetto di azioni attraverso le API di Spark.

Tali operazioni possono essere:

- **Trasformazione**: se l'operazione prende in input un RDD e restituisce in output un altro RDD;
- **Azione**: se l'operazione prende come input un RDD e come output restituisce un oggetto generico.

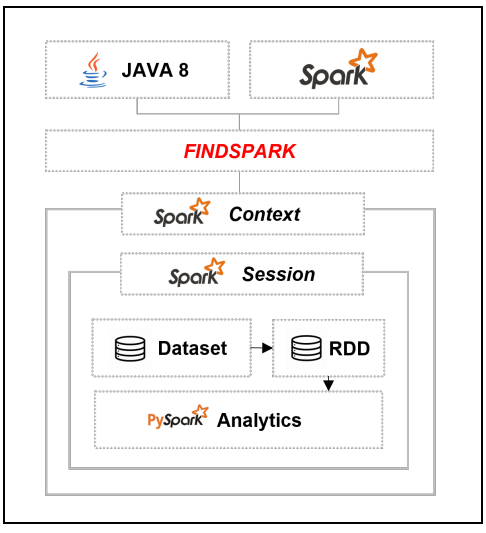


**Figura 1.5:** Spark: la configurazione

Anche in questo caso non è stato necessario configurare un cluster distribuito su cui estendere il potenziale dataset partizionato, sempre per le ragioni descritte in precedenza.

#### 1.3.3.1 Workflow: Raccolta dati → Colab → PySpark

Pertanto il workflow seguito è così suddiviso:

- **In locale**
  1. Raccolta dati;
  2. Preprocessing dei dati su Excel;
- **Google Colab**
  1. Configurazione;
  2. Upload dei dati;
  3. Analytics.

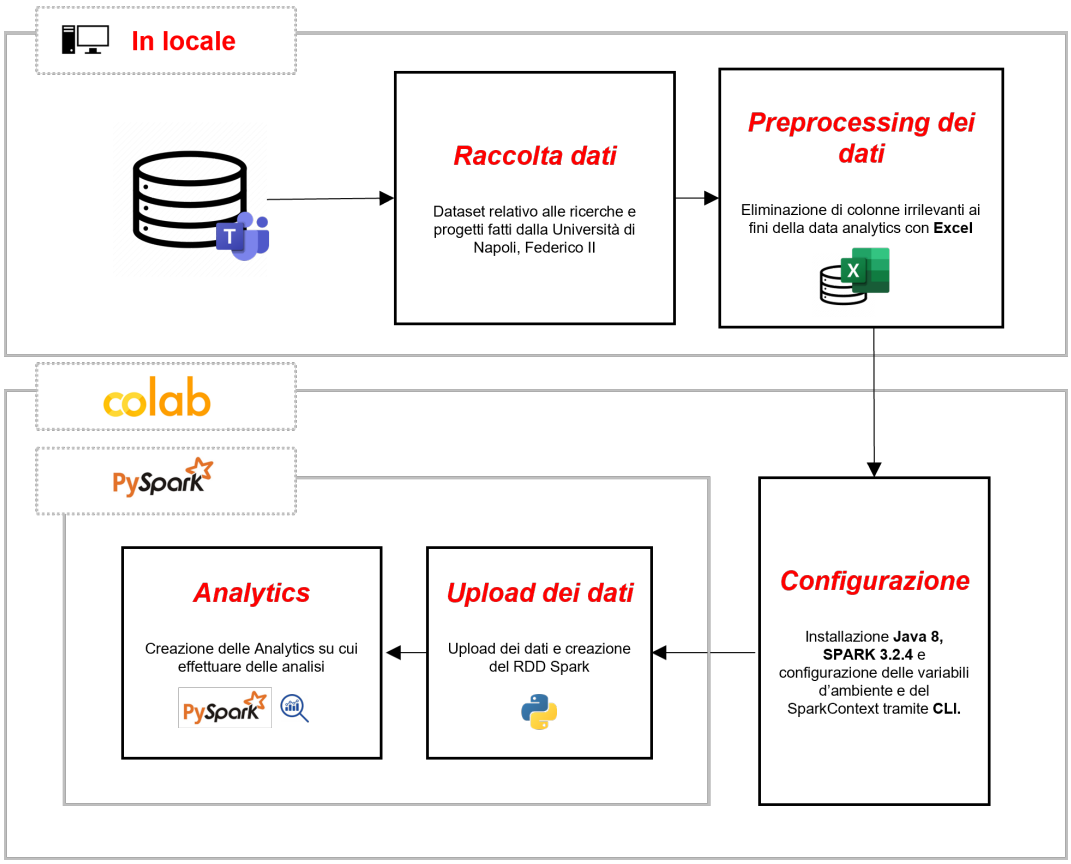


**Figura 1.6:** Workflow - Local to Colab

# 2 Analytics

Contenuti



Attraverso l'analisi di dati relativi alla somma finanziata, le collaborazioni e le tematiche affrontate nei progetti e ricerche, è possibile ottenere un quadro dettagliato dell'**impegno dell'Università** nel corso degli anni verso la **comunità scientifica mondiale**. In questa raccolta di query, verranno selezionate le analisi più significative che permetteranno di comprendere la portata e la varietà delle attività di ricerca condotte dalla **Federico II**.

## 2.1 Somma finanziata tra i 5 dipartimenti con più progetti nella Federico II

L'obbiettivo della prima Analytic, è quello di analizzare il dataset di progetti di ricerca ed identificare i primi 5 dipartimenti in termini di numero di progetti finanziati. Inoltre, la query calcola la somma totale del finanziamento ricevuto da ciascuno dei dipartimenti e ordina i risultati in base a tale somma in ordine decrescente.

### 2.1.1 Implementazione

Nell'implementazione:

- Vengono estratti i dipartimenti dalla colonna "UnitsofAssessment" e il relativo finanziamento;
- Vengono filtrati i dipartimenti eliminando quelli nulli e raggruppando i risultati per dipartimento, contando il numero di progetti per ciascuno;
- Vengono ordinati i risultati in base al conteggio dei progetti in ordine decrescente e seleziona i primi 5 dipartimenti;
- Viene poi calcolata la somma del finanziamento per ciascuno dei top 5 dipartimenti e ordina i risultati per tale somma in ordine decrescente.

Le seguenti implementazioni in **Hive, Spark, Pig**:

**Hive**

```
SELECT Dipartimento, Somma_Finanziata_EUR
FROM (SELECT trim(Dipartimento) AS Dipartimento, count(*) AS
    Num_progetti, SUM(Funding_Amount_in_EUR) AS Somma_Finanziata_EUR
      FROM (SELECT explode(split(Units_of_Assessment,'; ')) AS
          Dipartimento, Funding_Amount_in_EUR
            FROM dataset_unina_research)
      WHERE Dipartimento IS NOT NULL AND Funding_Amount_in_EUR IS NOT
          NULL
      GROUP BY Dipartimento
      ORDER BY Num_progetti DESC
      LIMIT 5)
ORDER BY int(Somma_Finanziata_EUR) DESC;
```

**Tabella 2.1:** Somma finanziata tra i 5 dipartimenti con più progetti nella Federico II - **Hive**

**Spark**

```
Top5Dip = projectsDS.select(explode(split(projectsDS
["Units_of_Assessment"], "; ")).alias("Dipartimento"),
    projectsDS["Funding_Amount_in_EUR"]) \
                .filter(col("Dipartimento").isNotNull()) \
                .groupBy(col("Dipartimento")) \
                .count() \
                .orderBy(col("count").desc()) \
                .limit(5)

Top5DipSommaProgetti = projectsDS.select(explode(split(projectsDS
["Units_of_Assessment"], "; ")).alias("Dipartimento"),
    projectsDS["Funding_Amount_in_EUR"]) \
                .filter(col("Dipartimento").isNotNull()) \
                .join(Top5Dip, "Dipartimento") \
                .groupBy(col("Dipartimento")) \
                .agg(F.sum("Funding_Amount_in_EUR").alias("Somma
                    Finanziata (EUR)")) \
                .orderBy(col("Somma Finanziata (EUR)").desc())
```

**Tabella 2.2:** Somma finanziata tra i 5 dipartimenti con più progetti nella Federico II - **Spark**

**Pig**

```
dipartimenti = FOREACH dataset_unina_research GENERATE
    FLATTEN(TOKENIZE(TRIM(Units_of_Assessment), ';')) AS Dipartimento,
    Funding_Amount_in_EUR;
dipartimenti_filtrati = FILTER dipartimenti BY Dipartimento IS NOT NULL
    AND Funding_Amount_in_EUR IS NOT NULL;
dipartimenti_contati = GROUP dipartimenti_filtrati BY Dipartimento;
dipartimenti_contati_final = FOREACH dipartimenti_contati GENERATE group
    AS Dipartimento, COUNT(dipartimenti_filtrati) AS Numero_Progetti,
    SUM(dipartimenti_filtrati.Funding_Amount_in_EUR) AS
    Somma_Finanziata_EUR;
dipartimenti_ordinati = ORDER dipartimenti_contati_final BY
    Numero_Progetti DESC;
top_5_dipartimenti = LIMIT dipartimenti_ordinati 5;
top_5_dipartimenti_finanziati = FOREACH top_5_dipartimenti GENERATE
    Dipartimento AS Dipartimento, Somma_Finanziata_EUR AS
    Somma_Finanziata_EUR;
top_5_dipartimenti_finanziati_final = ORDER
    top_5_dipartimenti_finanziati BY Somma_Finanziata_EUR DESC;
```

**Tabella 2.3:** Somma finanziata tra i 5 dipartimenti con più progetti nella Federico II - **Pig**

### 2.1.2 Risultati

Nella Figura 2.1, vengono mostrate le somme finanziate tra i cinque dipartimenti universitari della Federico II con più progetti. Ogni riga rappresenta un dipartimento e i valori corrispondenti alla somma totale dei finanziamenti ricevuti in euro. La Tabella evidenzia una notevole differenza tra i finanziamenti ricevuti dai vari dipartimenti.
Il Dipartimento di Ingegneria (B12 Engineering) risulta essere il più finanziato con oltre un miliardo di euro, seguito dal Dipartimento di Informatica (B11 Computer Science and Informatics) con quasi mezzo miliardo di euro. Gli altri tre dipartimenti mostrano somme di finanziamenti nettamente inferiori. Si può perfettamente dedurre come la Federico II sia molto afferente ai dipartimenti di Ingegneria, anche perché la stessa sede presenta un numero di facoltà di Ingegneria differenti molto alto.

| Dipartimento | Somma_Finanziata_EUR |
|---|---|
| B12 Engineering | 1101742946 |
| B11 Computer Science and Informatics | 488879107 |
| A06 Agriculture, Veterinary and Food Science | 190442745 |
| A01 Clinical Medicine | 177342410 |
| A03 Allied Health Professions, Dentistry, Nursing and Pharmacy | 87311094 |

**Tabella 2.4:** Risultati della query: Somma finanziata tra i 5 dipartimenti con più progetti nella Federico II

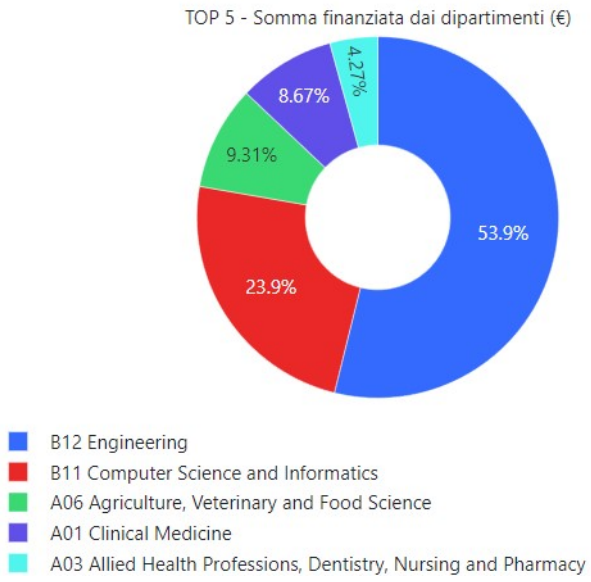


**Figura 2.1:** Pie View - Somma finanziata tra i 5 dipartimenti con più progetti nella Federico II

## 2.2 Le 10 tematiche più trattate negli ultimi 10 anni dalla Federico II

L'obbiettivo della seconda Analytic, è quello di analizzare il dataset di progetti di ricerca e identificare i primi 10 ambiti di ricerca per numero di progetti finanziati, considerando solo i progetti con data di fine a partire dal 2012.

### 2.2.1 Implementazione

Nell'implementazione:

- Vengono estratti gli ambiti di ricerca dalla colonna ”FieldsofResearchANZSRC2020” e l’anno di fine del progetto;
- Vengono filtrati i progetti con data di fine anno a partire dal 2012 e gli ambiti di ricerca non nulli;
- Vengono raggruppati i risultati per ambito di ricerca e contati il numero di progetti per ciascun ambito;
- Vengono ordinati i risultati in base al conteggio dei progetti in ordine decrescente e selezionati i primi 10 ambiti di ricerca.

Le seguenti implementazioni in **Hive, Spark, Pig**:

**Hive**

```
SELECT trim(Ambito_di_Ricerca) AS Ambito_di_Ricerca, count(*) AS
    Numero_Progetti
FROM (SELECT explode(split(Fields_of_Research_ANZSRC_2020,'; ')) AS
    Ambito_di_Ricerca, End_Year
      FROM dataset_unina_research)
WHERE Ambito_di_Ricerca IS NOT NULL AND End_Year >= '2012'
GROUP BY Ambito_di_Ricerca
ORDER BY Numero_Progetti DESC
LIMIT 10;
```

**Tabella 2.5:** Le 10 tematiche più trattate negli ultimi 10 anni dalla Federico II - **Hive**

**Spark**

```
Tem_Top10 = projectsDS.select(explode(split(projectsDS
           ["Fields_of_Research_ANZSRC_2020"], "; ")).alias("Ambito di
               Ricerca"), "End_Year") \
               .filter(col("End_Year") >= 2012 ) \
               .filter(col("Ambito di Ricerca").isNotNull()) \
               .groupBy(col("Ambito di Ricerca")) \
               .count() \
               .select("Ambito di Ricerca", col("count").alias("Numero
                   progetti")) \
               .orderBy(col("Numero progetti").desc()) \
               .limit(10)
```

**Tabella 2.6:** Le 10 tematiche più trattate negli ultimi 10 anni dalla Federico II - **Spark**

**Pig**

```
ambiti_di_ricerca = FOREACH dataset_unina_research GENERATE
    FLATTEN(TOKENIZE(TRIM(Fields_of_Research_ANZSRC_2020), ';')) AS
    Ambito_di_Ricerca, End_Year;
ambiti_di_ricerca_filtrati = FILTER ambiti_di_ricerca BY
    Ambito_di_Ricerca IS NOT NULL AND End_Year >= '2012';
conta_progetti = GROUP ambiti_di_ricerca_filtrati BY Ambito_di_Ricerca;
numero_progetti = FOREACH conta_progetti GENERATE group AS
    Ambito_di_Ricerca, COUNT(ambiti_di_ricerca_filtrati) AS
    Numero_Progetti;
top_10_ambiti = ORDER numero_progetti BY Numero_Progetti DESC,
    Ambito_di_Ricerca ASC;
limite_top_10_ambiti = LIMIT top_10_ambiti 10;
ambiti_di_ricerca_e_numero_progetti = FOREACH limite_top_10_ambiti
    GENERATE Ambito_di_Ricerca AS Ambito_di_Ricerca, Numero_Progetti AS
    Numero_Progetti;
```

**Tabella 2.7:** Le 10 tematiche più trattate negli ultimi 10 anni dalla Federico II - **Pig**

### 2.2.2 Risultati

Come si può notare in Tabella ed in Figura 2.2 sottostante, vengono presentati i dati relativi al numero di progetti di ricerca condotti in diverse tematiche. La Tabella evidenzia che l'ambito (32) Biomedical and Clinical Sciences, è quello con il maggior numero di progetti, con un totale di 187. Seguono Engineerings (40) con 150 progetti e le Biological Sciences (31) con 143 progetti.

| Ambito_di_Ricerca | Numero_Progetti |
|---|---|
| 32 Biomedical and Clinical Sciences | 187 |
| 40 Engineerings | 150 |
| 31 Biological Sciences | 143 |
| 34 Chemical Sciences | 78 |
| 46 Information and Computing Sciences | 68 |
| 3101 Biochemistry and Cell Biology | 52 |
| 3211 Oncology and Carcinogenesis | 46 |
| 30 Agricultural, Veterinary and Food Sciences | 45 |
| 44 Human Society | 40 |
| 37 Earth Sciences | 38 |

**Tabella 2.8:** Risultati della query: Le 10 tematiche più trattate negli ultimi 10 anni dalla Federico II

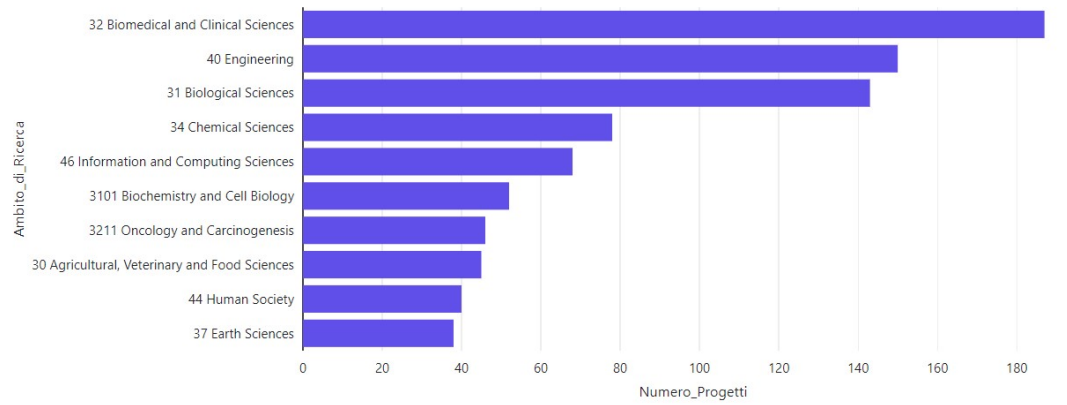


**Figura 2.2:** Bar View - Le 10 tematiche più trattate negli ultimi 10 anni dalla Federico II

## 2.3 Le 100 istituzioni con cui ha collaborato di più la Federico II

In questa terza Analytic viene, ancora una volta, analizzato il dataset di progetti di ricerca e vengono identificate le prime 100 istituzioni per numero di collaborazioni, escludendo l'Università degli Studi di Napoli Federico II.

### 2.3.1 Implementazione

Nell'implementazione:

- Vengono estratte delle istituzioni dalla colonna "ResearchOrganizationOriginal" e filtrate quelle diverse dall'Università degli Studi di Napoli Federico II;
- Vengono raggruppati i risultati per istituzione e contati il numero di collaborazioni per ciascuna istituzione;
- Vengono ordinati i risultati in base al numero di collaborazioni in ordine decrescente e seleziona le prime 100 istituzioni.

Le seguenti implementazioni in **Hive, Spark, Pig**:

**Hive**

```
SELECT trim(Istituzione) AS Istituzione, count(*) AS
    Numero_Collaborazioni
FROM (SELECT explode(split(Research_Organization_original, "; ")) AS
    Istituzione
      FROM dataset_unina_research)
WHERE Istituzione != "University of Naples Federico II"
GROUP BY Istituzione
ORDER BY Numero_Collaborazioni DESC
LIMIT 100;
```

**Tabella 2.9:** Le 100 istituzioni con cui ha collaborato di più la Federico II - **Hive**

**Spark**

```
Istituzioni = projectsDS.select(explode(split(projectsDS
          ["Research_Organization_original"], ";
              ")).alias("Istituzione")) \
              .filter(col("Istituzione") != 'University of Naples
                  Federico II') \
              .groupBy(trim("Istituzione").alias("Istituzione")) \
              .count() \
              .select("Istituzione",
                  col("count").alias("Numero_Collaborazioni")) \
              .orderBy(col("Numero_Collaborazioni").desc()) \
              .limit(100)
```

**Tabella 2.10:** Le 100 istituzioni con cui ha collaborato di più la Federico II - **Spark**

**Pig**

```
istituzioni = FOREACH dataset_unina_research GENERATE
    FLATTEN(TOKENIZE(TRIM(Research_Organization_original), ';')) AS
    Istituzione;
istituzioni_filtered = FILTER istituzioni BY Istituzione != 'University
    of Naples Federico II';
istituzioni_grouped = GROUP istituzioni_filtered BY Istituzione;
collaborazioni = FOREACH istituzioni_grouped GENERATE group AS
    Istituzione, COUNT(istituzioni_filtered) AS Numero_Collaborazioni;
istituzioni_ordered = ORDER collaborazioni BY Numero_Collaborazioni DESC;
top100istituzioni = LIMIT istituzioni_ordered 100;
```

**Tabella 2.11:** Le 100 istituzioni con cui ha collaborato di più la Federico II - **Pig**

### 2.3.2 Risultati

Nella Tabella sottostante, vengono presentate le prime 10 di 100 istituzioni con cui ha collaborato di più la Federico II. La Tabella mostra che la Sapienza (Università di Roma) è l'istituzione con il maggior numero di collaborazioni, con un totale di 476. Seguono l'Università degli Studi di Milano con 350 collaborazioni, l'Università degli Studi di Firenze con 346, l'Università degli Studi di Padova con 322 e l'Università di Bologna con 317. Altre istituzioni con un numero significativo di collaborazioni includono l'Università degli Studi di Torino, l'Università di Pisa, l'Università degli Studi della Campania Luigi Vanvitelli.

| Istituzione | Numero_Collaborazioni |
|---|---|
| Sapienza University of Rome | 476 |
| University of Milan | 350 |
| University of Florence | 346 |
| University of Padua | 322 |
| University of Bologna | 317 |
| University of Turin | 290 |
| University of Pisa | 290 |
| University of Campania Luigi Vanvitelli | 280 |
| National Research Council | 255 |
| University of Catania | 243 |
| ... | ... |

**Tabella 2.12:** Risultati della query: Le 100 istituzioni con cui ha collaborato di più la Federico II

## 2.4 Top 10 dipartimenti con progetti ancora in corso

L'obbiettivo della quarta Analytic è quello di analizzare il dataset di progetti di ricerca e identificare i primi 10 dipartimenti con il maggior numero di progetti in corso, ovvero i progetti la cui data di fine è successiva alla data corrente.

### 2.4.1 Implementazione

Nell'implementazione:

- Vengono filtrati progetti in base alla data di fine e selezionati solo quelli con una data di fine successiva alla data corrente;
- Viene raggruppata la colonna "UnitsofAssessment" per ottenere un elenco di dipartimenti e contati il numero di progetti in corso per ciascun dipartimento;
- Vengono ordinati i risultati in base al numero di progetti in corso in ordine decrescente e seleziona i primi 10 dipartimenti.

Le seguenti implementazioni in **Hive, Spark, Pig**:

**Hive**

```
SELECT trim(Dipartimento) AS Dipartimento, count(*) AS
    Numero_Progetti_In_Corso
FROM (SELECT explode(split(Units_of_Assessment, '; ')) AS Dipartimento
      FROM dataset_unina_research
      WHERE End_Date > CURRENT_DATE())
GROUP BY Dipartimento
ORDER BY Numero_Progetti_In_Corso DESC
LIMIT 10;
```

**Tabella 2.13:** TOP 10 dipartimenti con progetti ancora in corso - **Hive**

**Spark**

```
Dipartimenti = projectsDS.filter(col("End_Date") > current_date()) \
               .select(explode(split("Units_of_Assessment", ";
                   ")).alias("Dipartimento")) \
               .groupBy(trim("Dipartimento").alias("Dipartimento")) \
               .count() \
               .select("Dipartimento",
                   col("count").alias("Numero_Progetti_In_Corso")) \
               .orderBy(col("Numero_Progetti_In_Corso").desc()) \
               .limit(10)
```

**Tabella 2.14:** TOP 10 dipartimenti con progetti ancora in corso - **Spark**

**Pig**

```
dipartimenti = FOREACH dataset_unina_research GENERATE
    FLATTEN(TOKENIZE(TRIM(Units_of_Assessment), ';')) AS Dipartimento,
    End_Date;
dipartimenti_filtrati = FILTER dipartimenti BY End_Date >
    ToString(CurrentTime());
dipartimenti_contati = GROUP dipartimenti_filtrati BY Dipartimento;
dipartimenti_contati_final = FOREACH dipartimenti_contati GENERATE group
    AS Dipartimento, COUNT(dipartimenti_filtrati) AS
    Numero_Progetti_In_Corso;
dipartimenti_ordinati = ORDER dipartimenti_contati_final BY
    Numero_Progetti_In_Corso DESC, Dipartimento DESC;
dipartimenti_finali = LIMIT dipartimenti_ordinati 10;
```

**Tabella 2.15:** TOP 10 dipartimenti con progetti ancora in corso - **Pig**

### 2.4.2 Risultati

In Tabella ed in Figura 2.3, vengono mostrati i 10 dipartimenti con progetti ancora in corso, mostrando il numero di progetti di ricerca nei diversi dipartimenti universitari. La Tabella mostra che il Dipartimento di Ingegneria (B12) ha il maggior numero di progetti di ricerca attualmente in corso, con un totale di 28 progetti (in accordo con i risultati precedenti). Seguono il Dipartimento di Informatica e Scienze dell'Informazione (B11) con 14 progetti, la Medicina Clinica (A01) con 12, l'Agricoltura, Medicina Veterinaria e Scienze Alimentari (A06) con 11 e la Geografia e Studi Ambientali (C14) con 10 progetti e così via.

| Dipartimento | Numero_Progetti_In_Corso |
|---|---|
| B12 Engineering | 28 |
| B11 Computer Science and Informatics | 14 |
| A01 Clinical Medicine | 12 |
| A06 Agriculture, Veterinary and Food Science | 11 |
| C14 Geography and Environmental Studies | 10 |
| A03 Allied Health Professions, Dentistry, Nursing and Pharmacy | 7 |
| C17 Business and Management Studies | 6 |
| B07 Earth Systems and Environmental Sciences | 3 |
| A05 Biological Sciences | 3 |
| A05 Biological Sciences | 3 |

**Tabella 2.16:** Risultati della query: TOP 10 dipartimenti con progetti ancora in corso

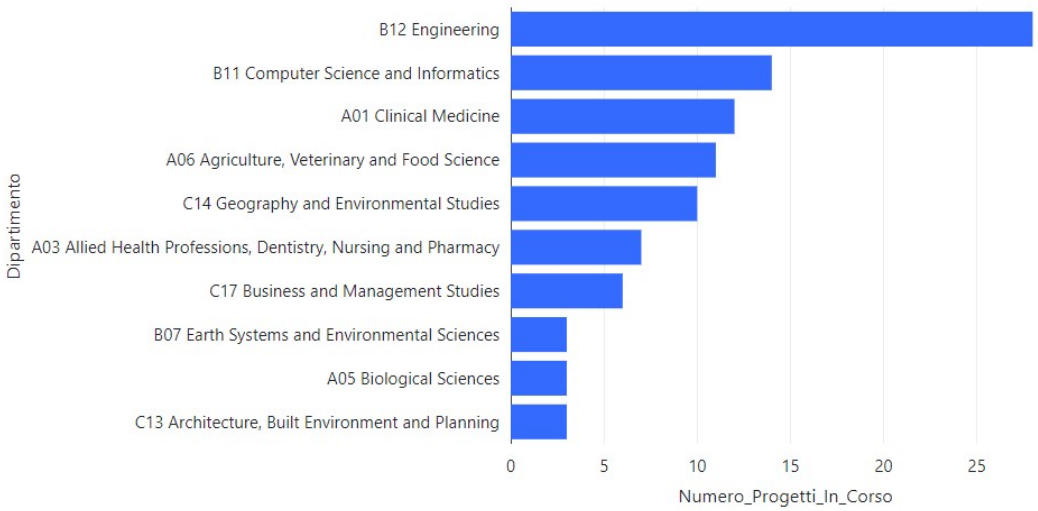


**Figura 2.3:** Bar View - TOP 10 dipartimenti con progetti ancora in corso

## 2.5 Somma finanziata dalla Federico II nelle ricerche negli ultimi 20 anni

L'obbiettivo della quinta Analytic è quello di analizzare il dataset di progetti di ricerca e calcolare la somma dei finanziamenti ricevuti per le ricerche per ogni anno a partire dal 2002.

### 2.5.1 Implementazione

Nell'implementazione:

- Vengono filtrati i progetti in base all'anno di inizio, selezionando solo quelli con un anno di inizio successivo al 2001;
- Vengono raggruppati i progetti per "StartYear" e viene calcolata la somma dei finanziamenti ricevuti in euro ("FundingAmountinEUR") per ciascun anno;
- Vengono ordinati i risultati in base all'anno di inizio in ordine decrescente e mostrati i risultati.

Le seguenti implementazioni in **Hive, Spark, Pig**:

**Hive**

```
SELECT Start_Year AS Anno_Inizio, SUM(Funding_Amount_in_EUR) AS
    Somma_Finanziata_EUR
FROM dataset_unina_research
WHERE Start_Year > '2001'
GROUP BY Start_Year
ORDER BY Start_Year DESC;
```

**Tabella 2.17:** Somma finanziata dalla Federico II nelle ricerche negli ultimi 20 anni - **Hive**

**Spark**

```
SommaFinanziata = projectsDS.filter(col("Start_Year") > '2001') \
                .groupBy(col("Start_Year")) \
                .agg(F.sum("Funding_Amount_in_EUR")
                    .alias("Somma_Finanziata_EUR")) \
                .orderBy(col("Start_Year").desc())
```

**Tabella 2.18:** Somma finanziata dalla Federico II nelle ricerche negli ultimi 20 anni - **Spark**

**Pig**

```
ricerche = FOREACH dataset_unina_research GENERATE Start_Year,
    Funding_Amount_in_EUR;
ricerche_filtrate = FILTER ricerche BY Start_Year > '2001';
fondi = GROUP ricerche_filtrate BY Start_Year;
fondi_stanziati = FOREACH fondi GENERATE group AS Anno,
    SUM(ricerche_filtrate.Funding_Amount_in_EUR) AS Somma_Finanziata_EUR;
fondi_stanziati_ordinati = ORDER fondi_stanziati BY Anno DESC;
```

**Tabella 2.19:** Somma finanziata dalla Federico II nelle ricerche negli ultimi 20 anni - **Pig**

### 2.5.2 Risultati

In Tabella ed in Figura 2.4, vengono presentati i dati relativi ai finanziamenti ricevuti per la ricerca nel corso degli ultimi 20 anni. Si nota come i finanziamenti per la ricerca hanno subito variazioni significative nel corso degli anni. L'anno con il più alto importo di finanziamenti è il 2020, con oltre 412 milioni di euro, seguito dal 2022 con oltre 314 milioni di euro e dal 2014 con oltre 327 milioni di euro.

**Si noti** come dall'avvento dell'Euro la somma media annuale finanziata sui progetti dalla Federico II sono stati più bassi rispetto alla media annuale finanziata negli ultimi 10 anni. Uno dei possibili motivi può essere dovuta dalla evoluzione tecnologica e alla strumentazione all'avanguardia che negli ultimi anni richiedono una richiesta di denaro molto più alta oltre anche all'aumento del valore dell'Euro negli ultimi 10 anni.

| Anno_Inizio | Somma_Finanziata_EUR |
|---|---|
| 2023 | 122006193 |
| 2022 | 314043199 |
| 2021 | 102775511 |
| 2020 | 412702370 |
| 2019 | 63059973 |
| 2018 | 184665016 |
| 2017 | 222371781 |
| 2016 | 91142273 |
| 2015 | 65034430 |
| 2014 | 327815115 |
| ... | ... |

**Tabella 2.20:** Risultati della query: Somma finanziata dalla Federico II nelle ricerche negli ultimi 20 anni

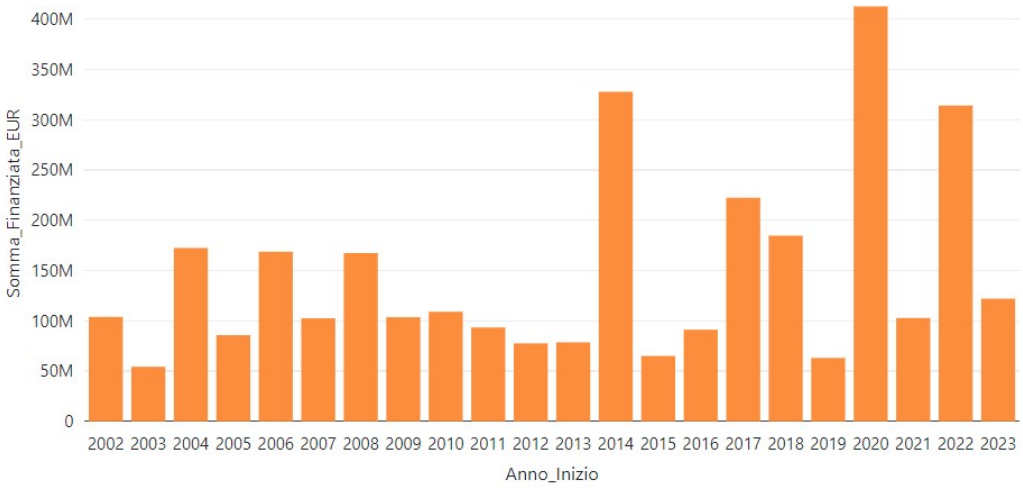


**Figura 2.4:** Bar View - Somma finanziata dalla Federico II nelle ricerche negli ultimi 20 anni

## 2.6 Somma e numero di progetti della Federico II nel 2020

La sesta Analytic è finalizzata ad analizzare come sono stati investiti i fondi da parte dell'Università nei diversi campi di ricerca, e quanti progetti in ognuno di essi sono stati avviati durante il 2020, l'anno che mostra il picco di investimenti negli ultimi 20 anni. In particolare, la somma in Euro è stata divisa rispetto ai diversi settori di ricerca, ordinando il risultato finale in base al numero di progetti iniziati durante l'anno.

### 2.6.1 Implementazione

Nell'implementazione:

- Vengono estratti dal dataset gli ambiti di ricerca dalla colonna "FieldsofResearchANZSRC2020" e "FundingAmountinEUR", la colonna relativa agli investimenti in euro;
- Vengono filtrati i progetti con l'anno di inizio pari al 2020;
- Vengono raggruppati i risultati per ambito di ricerca, contati il numero di progetti per ciascun ambito e infine viene fatta la somma in merito ad ogni gruppo realizzato;
- I risultati finali vengono ordinati in base al conteggio dei progetti in ordine decrescente.

Le seguenti implementazioni in **Hive, Spark, Pig**:

**Hive**

```
SELECT Campi_di_ricerca, count(Campi_di_ricerca) AS Numero_progetti,
    SUM(Funding_Amount_in_EUR) AS Somma_Finanziata_Totale
FROM (SELECT explode(split(Fields_of_Research_ANZSRC_2020, "; ")) AS
    Campi_di_ricerca, Funding_Amount_in_EUR
      FROM dataset_unina_research
      WHERE Start_Year = '2020')
GROUP BY Campi_di_ricerca
ORDER BY Numero_progetti DESC;
```

**Tabella 2.21:** Somma e numero di progetti della Federico II nel 2020 - **Hive**

**Spark**

```
ProgFondi2020 = projectsDS.filter(col("Start_Year") == '2020') \
              .select(explode(split("Fields_of_Research_ANZSRC_2020",
                  "; ")).alias("Campi_di_ricerca"),
                  "Funding_Amount_in_EUR") \
              .groupBy(col("Campi_di_ricerca")) \
              .agg(count("Campi_di_ricerca").alias("Numero_progetti"),
                  F.sum("Funding_Amount_in_EUR")
                      .alias("Somma_Finanziata_Totale")) \
              .orderBy(col("Numero_progetti").desc())
```

**Tabella 2.22:** Somma e numero di progetti della Federico II nel 2020 - **Spark**

**Pig**

```
campi = FOREACH dataset_unina_research GENERATE
    FLATTEN(TOKENIZE(TRIM(Fields_of_Research_ANZSRC_2020), ';')) AS
    Campi_di_ricerca, Funding_Amount_in_EUR, Start_Year;
campi_filtrati = FILTER campi BY Start_Year = '2020';
campi_raggruppati = GROUP campi_filtrati BY Campi_di_ricerca;
progetti = FOREACH campi_raggruppati GENERATE group AS Campi_di_ricerca,
    SUM(campi_filtrati.Funding_Amount_in_EUR) AS
    Somma_Finanziata_Totale, COUNT(campi_filtrati) AS Numero_Progetti;
progetti_ordinati = ORDER progetti BY Numero_Progetti DESC;
```

**Tabella 2.23:** Somma e numero di progetti della Federico II nel 2020 - **Pig**

### 2.6.2 Risultati

Dai risultati si può notare che durante l'anno 2020 l'Università ha avviato il maggior numero di progetti nel settore medico, vedendo rispettivamente "Biomedical and Clinical Sciences", "Biological Sciences" e "Oncology and Carcinogenesis" occupare le prime 3 posizioni per numero di progetti.
Segue poi "Engineering", che oltre alla presenza di 16 progetti è anche il campo che ha ricevuto la maggiore somma in termini di finanziamenti. Da questo punto di vista, tematiche come "Information and Computing Sciences" e diversi settori dell'ingegneria occupano i primi posti, nonostante un più basso numero di progetti. Un riassunto con i primi 10 risultati restituiti dalla query è mostrato nella seguente Tabella 2.24:

| Campi_di_ricerca | Numero_progetti | Somma_Finanziata_Totale |
|---|---|---|
| 32 Biomedical and Clinical Sciences | 32 | 29903187 |
| 31 Biological Sciences | 25 | 15639113 |
| 3211 Oncology and Carcinogenesis | 17 | 15737406 |
| 40 Engineering | 16 | 172178921 |
| 34 Chemical Sciences | 10 | 7570303 |
| 3101 Biochemistry and Cell Biology | 9 | 2883287 |
| 46 Information and Computing Sciences | 7 | 165248297 |
| 44 Human Society | 6 | 10151596 |
| 33 Built Environment and Design | 6 | 9864166 |
| 51 Physical Sciences | 6 | 4676452 |
| ... | ... | ... |

**Tabella 2.24:** Risultati della query: Somma e numero di progetti della Federico II nel 2020

Nella seguente Figura 2.5, vengono riportati gli altri campi di ricerca interessati in progetti nel 2020, tramite un istogramma.

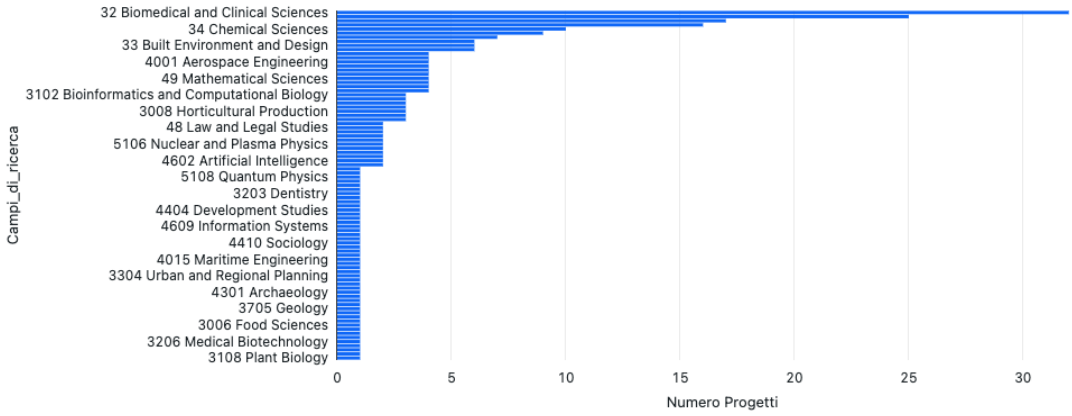


**Figura 2.5:** Bar View - Somma e numero di progetti della Federico II nel 2020

## 2.7 Somma finanziata dalla Federico II nelle ricerche sul cancro negli ultimi 20 anni

L'obbiettivo della settima Analytic è analizzare la distribuzione dei fondi in euro, da parte dell'Università, negli ultimi 20 anni per quanto concerne la ricerca sul cancro, realizzandone una vista disposta in ordine cronologico.

### 2.7.1 Implementazione

Nell'implementazione:

- Vengono estratti l'anno di inizio "StartYear" e la somma ("FundingAmountinEUR") dal dataset a disposizione;
- Vengono filtrati i progetti con la colonna "CancerTypes" non nulla e gli anni di inizio presenti a partire dal 2002;
- I risultati vengono raggruppati e ordinati per anno in ordine decrescente.

Le seguenti implementazioni in **Hive, Spark, Pig**:

**Hive**

```
SELECT Start_Year AS Anno_Inizio, SUM(Funding_Amount_in_EUR) AS
    Somma_Finanziata_Cancro_EUR
FROM dataset_unina_research
WHERE Cancer_Types IS NOT NULL AND (Start_Year IS NOT NULL AND
    Start_Year > '2001')
GROUP BY Start_Year
ORDER BY Start_Year DESC;
```

**Tabella 2.25:** Somma finanziata dalla Federico II nelle ricerche sul cancro negli ultimi 20 anni - **Hive**

**Spark**

```
SommaFinanziataCancro =
    projectsDS.filter(col("Cancer_Types").isNotNull()) \
              .filter(col("Start_Year") > '2001') \
              .groupBy(col("Start_Year")) \
              .agg(F.sum("Funding_Amount_in_EUR") \
              .alias("Somma_Finanziata_Cancro_EUR")) \
              .orderBy(col("Start_Year").desc())
```

**Tabella 2.26:** Somma finanziata dalla Federico II nelle ricerche sul cancro negli ultimi 20 anni - **Spark**

**Pig**

```
ricerche_cancro = FOREACH dataset_unina_research GENERATE Cancer_Types,
    Start_Year, Funding_Amount_in_EUR;
ricerche_cancro_filtrate = FILTER ricerche_cancro BY Cancer_Types != ''
    AND Start_Year > '2001';
fondi_cancro = GROUP ricerche_cancro_filtrate BY Start_Year;
fondi_stanziati = FOREACH fondi_cancro GENERATE group AS Anno,
    SUM(ricerche_cancro_filtrate.Funding_Amount_in_EUR) AS
    Somma_Finanziata_Cancro_EUR;
fondi_stanziati_ordinati = ORDER fondi_stanziati BY Anno DESC;
```

**Tabella 2.27:** Somma finanziata dalla Federico II nelle ricerche sul cancro negli ultimi 20 anni - **Pig**

### 2.7.2 Risultati

Negli ultimi 20 anni la Federico II ha costantemente finanziato progetti relativi alla ricerca sul cancro e continua a farlo tuttora, come si evince dalla Figura 2.6.
Con i dati a disposizione, si può notare che negli anni più recenti, in particolare nel 2017 e nel 2020, sono state investite somme superiori ai 15 milioni, mentre nel 2007 è stata anche superata la soglia dei 20 milioni.

| Anno_Inizio | Somma_Finanziata_Cancro_EUR |
|---|---|
| 2023 | 8852273 |
| 2022 | 4099002 |
| 2021 | 4595975 |
| 2020 | 17067493 |
| 2017 | 17031248 |
| 2016 | 998050 |
| 2015 | 4707851 |
| 2014 | 759100 |
| ... | ... |

**Tabella 2.28:** Risultati della query: Somma finanziata dalla Federico II nelle ricerche sul cancro negli ultimi 20 anni

**Si noti** come, negli anni tra il 2008 ed il 2014 i fondi stanziati per i progetti sulla ricerca abbiano risentito di un drastico decremento riconducibile in parte alla concomitante crisi finanziaria attraversata dal paese in quegli stessi anni.
**Si noti**, inoltre, come la Federico II abbia investito in media molto dal 2002 al 2007 in rapporto con gli investimenti totali nei rispettivi anni, dimostrazione del fatto che in quegli anni l'Università era intenta a consolidarsi tra le migliori Università nella ricerca di soluzioni imminenti al Cancro.

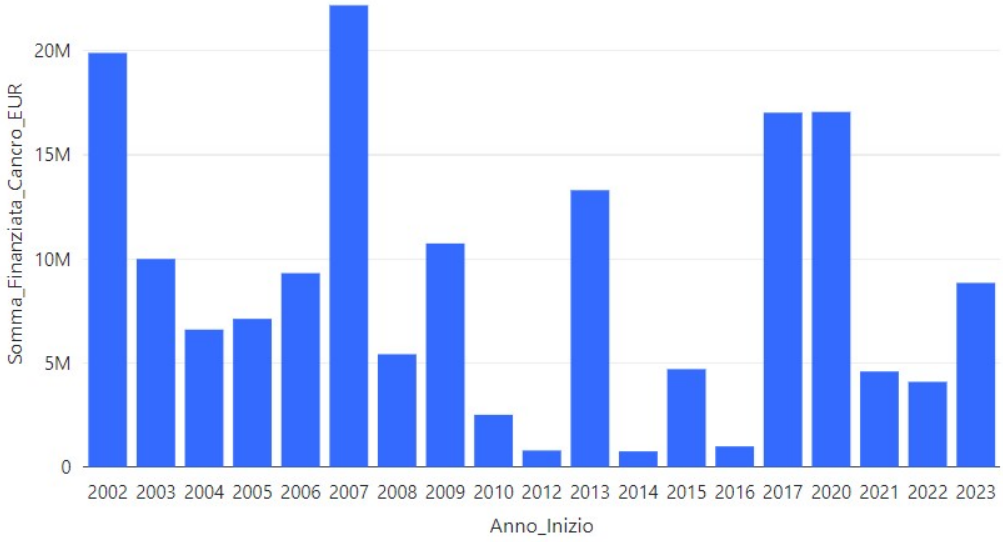


**Figura 2.6:** Bar View - Somma finanziata dalla Federico II nelle ricerche sul cancro negli ultimi 20 anni

## 2.8 Le parole chiave più utilizzate nei titoli dei progetti

Con l'ottava Analytic verranno valutate se, nei titoli dei progetti, terminati ed in corso, son presenti dei termini più utilizzati di altri, in modo da individuare rapidamente le tematiche maggiormente trattate. Verrà verificato in tal caso il numero di ripetizione delle parole per poi ordinarle.

### 2.8.1 Implementazione

Nell'implementazione:

- Vengono estratti tutti i titoli dalla colonna "Titletranslated", ed effettuate operazioni di conversione, portando tutta la riga in lowercase, rimuovendo caratteri di controllo e segni di punteggiatura, ed infine, viene divisa ogni riga in singole parole;
- Vengono filtrate le stringhe vuote e le parole che appartengono ad un file di stopwords;
- I risultati sono raggruppati per parola, facendo un conteggio su ogni termine;
- I primi 100 risultati sono infine ordinati per conteggio in modo decrescente e per parola in ordine crescente per gestire i casi di pareggio;

Le seguenti implementazioni in **Hive, Spark, Pig**:

**Hive**

```
SELECT trim(Topic_Titolo) AS Topic_Titolo, count(*) AS Conteggio
FROM (SELECT explode(split(regexp_replace(lower(Title_translated),
    '[\\p{Punct},\\p{Cntrl}]', ''), ' ')) AS Topic_Titolo
      FROM dataset_unina_research)
WHERE Topic_Titolo NOT IN (SELECT * FROM stopwords) AND Topic_Titolo !=
    ""
GROUP BY Topic_Titolo
ORDER BY Conteggio DESC, Topic_Titolo ASC
LIMIT 100;
```

**Tabella 2.29:** Le parole chiave più utilizzate nei titoli dei progetti - **Hive**

**Spark**

```
stopwords_rdd = sc.textFile('/content/stopwords.txt')
stopwords_df = stopwords_rdd.map(lambda x: (x, )).toDF(['word'])
stopwords = stopwords_df.select(col('word'))

parolechiavi = projectsDS.select(explode(split(regexp_replace(lower(
       col("Title_translated")), '[\\p{Punct},\\p{Cntrl}]', ''), '
           ')).alias("Topic_Titolo")) \
               .join(stopwords, col("Topic_Titolo") == col("word"),
                   "left_anti") \
               .filter(col("Topic_Titolo") != '') \
               .groupBy(trim("Topic_Titolo").alias("Topic_Titolo")) \
               .count() \
               .select("Topic_Titolo", col("count").alias("Conteggio")) \
               .orderBy(col("Conteggio").desc(),
                   col("Topic_Titolo").asc()) \
               .limit(100)
```

**Tabella 2.30:** Le parole chiave più utilizzate nei titoli dei progetti - **Spark**

**Pig**

```
TopicTitolo = FOREACH dataset_unina_research GENERATE
    FLATTEN(TOKENIZE(REPLACE(LOWER(TRIM(Title_translated)),
'[\\p{Punct},\\p{Cntrl}]',''))) AS Topic;
TopicTitolo_uniti = JOIN TopicTitolo BY Topic LEFT, stoplist BY stop;
TopicTitolo_puliti = FILTER TopicTitolo_uniti BY stoplist::stop IS NULL;
TopicTitolo_grp = GROUP TopicTitolo_puliti BY $0;
TopicTitolo_contati = FOREACH TopicTitolo_grp GENERATE $0, COUNT($1);
TopicTitolo_final = ORDER TopicTitolo_contati BY $1 DESC, $0 ASC;
Top100_Topic = LIMIT TopicTitolo_final 100;
```

**Tabella 2.31:** Le parole chiave più utilizzate nei titoli dei progetti - **Pig**

### 2.8.2 Risultati

I risultati ottenuti riflettono il carattere innovativo della Federico II, con parole come ”development” e ”innovative” presenti tra le prime 10 parole più usate nei titoli dei progetti.
Con la presenza del termine ”european”, presente quasi 100 volte, si conferma la centralità dell'Università in numerosi progetti in collaborazioni con istituzioni da tutta Europa.
Infine, è facile notare dai risultati che ”molecular” è la parola più presente di tutte, dato il grande numero di progetti in campo medico e scientifico in cui è coinvolta la Federico II.

| Topic_Titolo | Conteggio |
|---|---|
| molecular | 185 |
| systems | 172 |
| development | 137 |
| study | 129 |
| mechanisms | 114 |
| control | 104 |
| innovative | 102 |
| european | 99 |
| ... | ... |

**Tabella 2.32:** Risultati della query: Le parole chiave più utilizzate nei titoli dei progetti

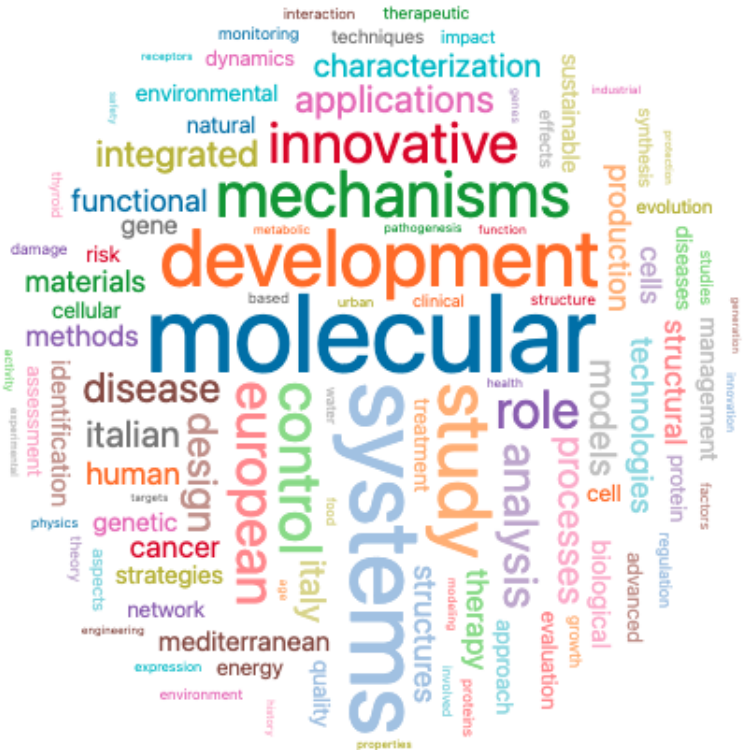


**Figura 2.7:** Word Cloud - Le parole chiave più utilizzate nei titoli dei progetti

## 2.9 I 10 Paesi esteri con cui ha maggiormente collaborato l'Università

L'obbiettivo della nona Analytic è quello di analizzare il dataset di progetti di ricerca ed identificare i primi 10 paesi con il maggior numero di collaborazioni nei diversi progetti, escludendo l'Italia.

### 2.9.1 Implementazione

Nell'implementazione:

- Viene splittata la colonna "CountryofResearchOrganization";
- Vengono selezionate le righe con il paese diverso dall'Italia;
- Vengono raggruppati i risultati per paese, contate le occorrenze, ordinati i risultati per conteggio decrescente;
- Vengono limitati i risultati ai primi 10.

Le seguenti implementazioni in **Hive, Spark, Pig**:

**Hive**

```
SELECT Paese, count(Paese) AS Numero_Collaborazioni
FROM (SELECT DISTINCT
    explode(split(trim(Country_of_Research_organization), "; ")) AS
    Paese, Title_translated
      FROM dataset_unina_research)
WHERE Paese != "Italy"
GROUP BY Paese
ORDER BY Numero_Collaborazioni DESC
LIMIT 10;
```

**Tabella 2.33:** I 10 Paesi esteri con cui ha maggiormente collaborato l'Università - **Hive**

**Spark**

```
top10paesi = projectsDS.select(explode(split(projectsDS
           ["Country_of_Research_organization"], "; ")).alias("Paese"),
               "Title_translated") \
               .distinct() \
               .filter(col("Paese") != 'Italy') \
               .groupBy(trim("Paese").alias("Paese")) \
               .count() \
               .select("Paese",
                   col("count").alias("Numero_Collaborazioni")) \
               .orderBy(col("Numero_Collaborazioni").desc()) \
               .limit(10)
```

**Tabella 2.34:** I 10 Paesi esteri con cui ha maggiormente collaborato l'Università - **Spark**

**Pig**

```
paesi = FOREACH dataset_unina_research GENERATE
    FLATTEN(TOKENIZE(TRIM(Country_of_Research_organization), ';')) AS
    Paese, Title_translated;
paesi_distinct = DISTINCT paesi;
paesi_filtrati = FILTER paesi_distinct BY Paese != 'Italy';
paesi_grp = GROUP paesi_filtrati BY Paese;
paese_contati = FOREACH paesi_grp GENERATE group AS Paese,
    COUNT(paesi_filtrati) AS Numero_Collaborazioni;
paese_ordinati = ORDER paese_contati BY Numero_Collaborazioni DESC;
top_10_paesi = LIMIT paese_ordinati 10;
```

**Tabella 2.35:** I 10 Paesi esteri con cui ha maggiormente collaborato l'Università - **Pig**

### 2.9.2 Risultati

Come è possibile notare dalla Tabella 2.36, si evince che l'Università "Federico II" ha collaborato principalmente con istituzioni del Regno Unito. In particolare, sono stati effettuati ben 414 progetti con tale paese.
A seguire, si trovano la Germania, la Francia e la Spagna con rispettivamente 385, 332 e 262 progetti, da cui si può dedurre l'importanza dell'Università, non solo a livello nazionale ma anche internazionale nella ricerca.

| Paese | Numero_Collaborazioni |
|---|---|
| United Kingdom | 414 |
| Germany | 385 |
| France | 332 |
| Spain | 262 |
| Netherlands | 241 |
| Belgium | 193 |
| Sweden | 149 |
| Greece | 138 |
| United States | 135 |
| Switzerland | 124 |

**Tabella 2.36:** Risultati della query: I 10 Paesi esteri con cui ha maggiormente collaborato l'Università

Inoltre, dalla Mappa 2.8, si nota che l'Università non collabora solo con paesi europei ma anche con paesi di altri continenti come gli Stati Uniti con cui ha collaborato per ben 135 progetti.

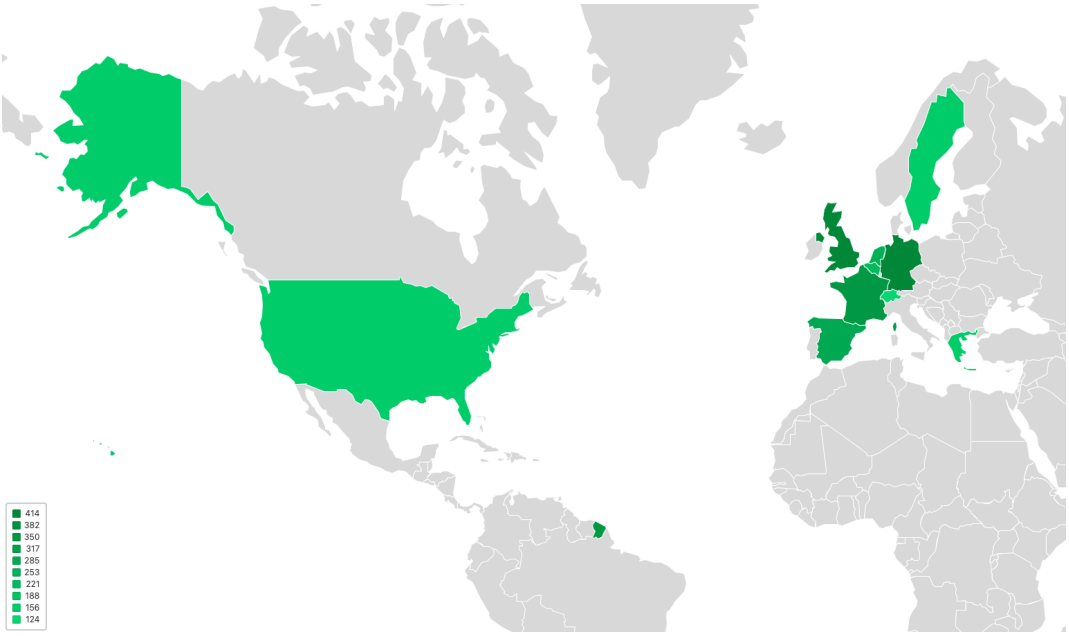


**Figura 2.8:** Map View - I 10 Paesi esteri con cui ha maggiormente collaborato l'Università

## 2.10 Gli ambiti di ricerca con progetti di maggiore durata

L'obbiettivo della decima Analytic è quello di analizzare il dataset di progetti di ricerca e calcolare la durata in anni dei progetti di ricerca per ciascun dipartimento, restituendo i primi 10 progetti con la durata maggiore.

### 2.10.1 Implementazione

Nell'implementazione:

- Viene splittata la colonna "UnitsofAssessment";
- Vengono raggruppati i risultati per "TitoloProgetto", "Dipartimento" e "DurataProgettoinAnni";
- Viene calcolata la durata del progetto sottraendo l'anno di inizio dall'anno di fine;
- Vengono ordinati i risultati per durata decrescente, limitando i risultati a 10.

Le seguenti implementazioni in **Hive, Spark, Pig**:

**Hive**

```
SELECT Title_translated AS Titolo_Progetto, trim(Dipartimento) AS
    Dipartimento, (End_Year - Start_Year) AS Durata_Progetto_in_Anni
FROM (SELECT explode(split(Units_of_Assessment, ';')) AS Dipartimento,
    End_Year, Start_Year, Title_translated
      FROM dataset_unina_research)
GROUP BY Titolo_Progetto, Dipartimento, Durata_Progetto_in_Anni
ORDER BY Durata_Progetto_in_Anni DESC
LIMIT 10;
```

**Tabella 2.37:** Gli ambiti di ricerca con progetti di maggiore durata - **Hive**

**Spark**

```
ricercadurata = projectsDS.select(explode(split(projectsDS
           ["Units_of_Assessment"], ";")).alias("Dipartimento"),
               "End_Year", "Start_Year", "Title_translated") \
               .groupBy(col("Title_translated").alias("Titolo_Progetto"),
                   trim("Dipartimento").alias("Dipartimento")) \
               .agg((F.max(col("End_Year")) - F.min(col("Start_Year")))
                   .alias("Durata_Progetto_in_Anni")) \
               .orderBy(col("Durata_Progetto_in_Anni").desc()) \
               .limit(10)
```

**Tabella 2.38:** Gli ambiti di ricerca con progetti di maggiore durata - **Spark**

**Pig**

```
dipartimenti = FOREACH dataset_unina_research GENERATE Title_translated,
    FLATTEN(TOKENIZE(TRIM(Units_of_Assessment), ';')) AS Dipartimento,
    End_Year, Start_Year;
dipartimenti_raggruppati = GROUP dipartimenti BY (Title_translated,
    Dipartimento);
dipartimenti_conta = FOREACH dipartimenti_raggruppati GENERATE group AS
    Titolo_progetto, (MAX(dipartimenti.End_Year) -
    MIN(dipartimenti.Start_Year)) AS Durata_Progetto_in_Anni;
dipartimenti_ordinati = ORDER dipartimenti_conta BY
    Durata_Progetto_in_Anni DESC, Titolo_progetto DESC;
limite_dipartimenti = LIMIT dipartimenti_ordinati 10;
```

**Tabella 2.39:** Gli ambiti di ricerca con progetti di maggiore durata - **Pig**

### 2.10.2 Risultati

In Tabella 2.40 vengono mostrati i risultati della query, da cui si può dedurre che il dipartimento di Medicina Clinica (A01) è quello che intraprende progetti con la maggior durata di ricerca. Infatti, nei primi 10 posti della classifica lo si ritrova 3 volte con ben 2 progetti con una durata di 16 anni.
Altri dipartimenti con progetti molto lunghi nel tempo sono quelli di Professioni sanitarie alleate, Odontoiatria, Infermieristica e Farmacia (A03), Scienze biologiche (A05) e Sistemi della Terra e Scienze Ambientali (B07), con progetti della durata rispettivamente di 15,14 e 11 anni.
Dalla precedente analisi, quindi, si può dedurre che l'Università "Federico II" sia molto impegnata nei settori legati all'**healthcare** nonché a quelli riguardanti **rischi ambientali (es. terremoti)** essendo legata ad un territorio come Napoli ad elevato rischio sismico.
**Si noti** che, nella successiva Tabella sono stati omessi i titoli dei progetti, presenti nella query effettuata, dalla visualizzazione in quanto sarebbe risultata difficile da leggere all'interno del documento; quindi, per maggiori dettagli si rimanda il lettore ai file sorgente.

| Dipartimento | Durata_Progetto_in_Anni |
|---|---|
| A01 Clinical Medicine | 16 |
| A01 Clinical Medicine | 16 |
| A03 Allied Health Professions, Dentistry, Nursing and Pharmacy | 15 |
| A05 Biological Sciences | 14 |
| A01 Clinical Medicine | 13 |
| B07 Earth Systems and Environmental Sciences | 11 |
| A05 Biological Sciences | 10 |
| A02 Public Health, Health Services and Primary Care | 10 |
| B12 Engineering | 9 |
| A05 Biological Sciences | 8 |

**Tabella 2.40:** Risultati della query: Gli ambiti di ricerca con progetti di maggiore durata

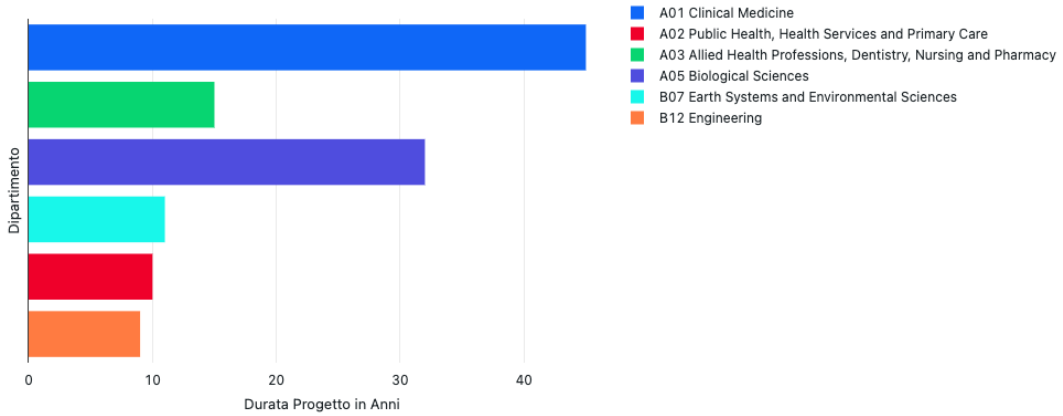


**Figura 2.9:** Bar View - Gli ambiti di ricerca con progetti di maggiore durata

## 2.11 Capacità di gestione delle risorse finanziarie dei progetti negli ultimi 20 anni

L'obbiettivo dell'undicesima Analytic è quello di analizzare il dataset di progetti di ricerca e calcolare la capacità dei progetti di ricerca e la media dei fondi per progetto in base all'anno di inizio.
In particolare, tramite il calcolo della **capacità** si vuole analizzare se i fondi finanziati nell'anno per i progetti siano stati distribuiti equamente tra i vari progetti o meno.
Mentre, con il calcolo della **media dei fondi** per progetto si cerca di stabilire in media quanto ha ricevuto/speso l'Università "Federico II" per ogni progetto iniziato in quell'anno.

### 2.11.1 Implementazione

Nell'implementazione:

- Vengono raggruppati i risultati per "StartYear", calcolando la somma dei fondi e il conteggio dei progetti;
- Vengono filtrate le righe in cui "StartYear" non è NULL;
- Vengono raggruppati i risultati per "StartYear", "SommaFinanziataEUR" e "NumeroProgetti", calcolando la capacità dei progetti (NumeroProgetti / SommaFinanziataEUR) e la media dei fondi per progetto (SommaFinanziataEUR / NumeroProgetti);
- Vengono ordinati i risultati per "StartYear" in ordine decrescente e limitando i risultati a 22 (in questo caso equivale a prendere solo i progetti con "StartYear" a partire dal 2002).

Le seguenti implementazioni in **Hive, Spark, Pig**:

**Hive**

```
SELECT Start_Year AS Anno_Inizio,
    (int(Numero_progetti)/int(Somma_Finanziata_EUR)) AS
    Capacita_Progetti, (int(Somma_Finanziata_EUR)/int(Numero_progetti))
    AS Fondi_media_Progetti
FROM (SELECT Start_Year, SUM(Funding_Amount_in_EUR) AS
    Somma_Finanziata_EUR, COUNT(*) AS Numero_Progetti
      FROM dataset_unina_research
      GROUP BY Start_Year)
WHERE Start_Year IS NOT NULL
GROUP BY Start_Year, Somma_Finanziata_EUR, Numero_Progetti
ORDER BY Start_Year DESC
LIMIT 22;
```

**Tabella 2.41:** Capacità di gestione delle risorse finanziarie dei progetti negli ultimi 20 anni - **Hive**

**Spark**

```
capacitafondi = projectsDS.groupBy("Start_Year") \
           .agg(F.sum("Funding_Amount_in_EUR")
               .alias("Somma_Finanziata_EUR"),
              F.count("*").alias("Numero_Progetti")) \
           .filter(col("Start_Year").isNotNull()) \
           .groupBy(col("Start_Year"), col("Somma_Finanziata_EUR"),
               col("Numero_Progetti")) \
           .agg((col("Numero_Progetti") / (col("Somma_Finanziata_EUR")))
               .alias("Capacita_Progetti"),
           ((col("Somma_Finanziata_EUR")) / col("Numero_Progetti"))
               .alias("Fondi_media_Progetti")) \
           .select(col("Start_Year").alias("Anno_Inizio"),
              "Capacita_Progetti", "Fondi_media_Progetti") \
           .orderBy(col("Anno_Inizio").desc()) \
           .limit(22)
```

**Tabella 2.42:** Capacità di gestione delle risorse finanziarie dei progetti negli ultimi 20 anni - **Spark**

**Pig**

```
progetti = FOREACH dataset_unina_research GENERATE Start_Year AS
    Anno_Inizio, Funding_Amount_in_EUR;
progetti_filtrati = FILTER progetti BY Anno_Inizio IS NOT NULL;
progetti_contati = GROUP progetti_filtrati BY Anno_Inizio;
progetti_contati_final = FOREACH progetti_contati GENERATE group AS
    Anno_Inizio, COUNT(progetti_filtrati) AS Numero_Progetti,
    SUM(progetti_filtrati.Funding_Amount_in_EUR) AS Somma_Finanziata_EUR;
capacita_progetti = FOREACH progetti_contati_final GENERATE Anno_Inizio,
    ((DOUBLE)Numero_Progetti/(DOUBLE)Somma_Finanziata_EUR) AS
    Capacita_Progetti,
    ((DOUBLE)Somma_Finanziata_EUR/(DOUBLE)Numero_Progetti) AS
    Fondi_Media_Progetti;
capacita_progetti_ordinati = ORDER capacita_progetti BY Anno_Inizio DESC;
capacita_progetti_finali = LIMIT capacita_progetti_ordinati 22;
```

**Tabella 2.43:** Capacità di gestione delle risorse finanziarie dei progetti negli ultimi 20 anni - **Pig**

### 2.11.2 Risultati

Nella Tabella 2.44 vengono mostrate la Capacità dei progetti e la Media dei fondi calcolati solo per gli ultimi 10 anni per questioni di compattezza della visualizzazione. Tuttavia, risulta più interessante andare ad analizzare l'andamento di questi due valori tramite la Figura 2.10.

| Anno_Inizio | Capacita_Progetti | Fondi_media_Progetti |
|---|---|---|
| 2023 | 1.4753349446777672e-7 | 6778121.833333333 |
| 2022 | 1.1781818589868586e-7 | 8487654.027027028 |
| 2021 | 3.502779956988003e-7 | 2854875.3055555555 |
| 2020 | 2.5926674470030303e-7 | 3857031.495327103 |
| 2019 | 3.1715839776842276e-7 | 3152998.65 |
| 2018 | 1.4079548234517793e-7 | 7102500.615384615 |
| 2017 | 8.769098269712558e-7 | 1140368.1076923078 |
| 2016 | 3.4012757175805785e-7 | 2940073.3225806453 |
| 2015 | 0.000001199364705741251 1 | 833774.7435897436 |
| 2014 | 7.626249936644929e-8 | 13112604.6 |
| ... | ... | ... |

**Tabella 2.44:** Risultati della query: Capacità di gestione delle risorse finanziarie dei progetti negli ultimi 20 anni

In particolare, come si può notare dall'andamento della **Capacità dei progetti** (linea blu) fino al 2010, l'Università riusciva a fare molti progetti anche con poche risorse economiche a disposizione denotando una buona capacità di gestione di quest'ultime.
Mentre, come si evince dalla **Media dei fondi** per progetto (linea rossa) dopo il 2010, si è avuto un incremento di fondi impiegati in media per ogni progetto e quindi, un peggioramento a parità di progetti di efficienza nella spesa delle risorse a disposizione.

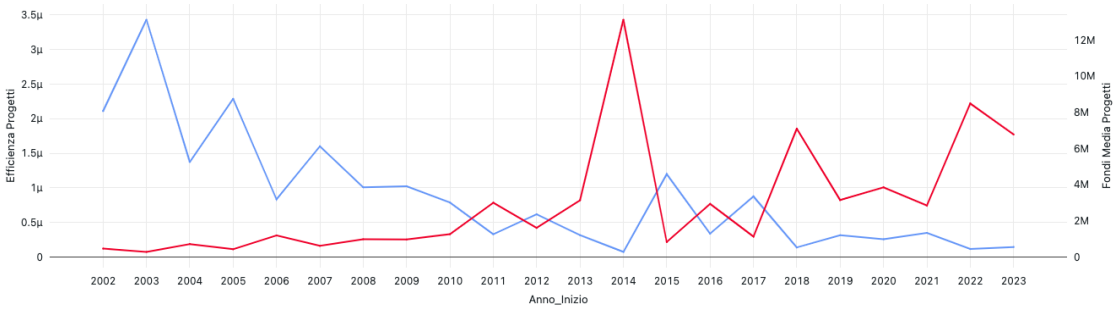


**Figura 2.10:** Line View - Capacità di gestione delle risorse finanziarie dei progetti negli ultimi 20 anni

## 2.12 Numero di progetti di ricerca per ogni obbiettivo di sviluppo sostenibile

L'obbiettivo della dodicesima Analytic è calcolare il numero di progetti finanziati e la somma dei finanziamenti per ogni obbiettivo di sviluppo sostenibile.

### 2.12.1 Implementazione

Nell'implementazione:

- Viene splittata la colonna "SustainableDevelopmentGoals";
- Vengono filtrate le righe in cui "obbiettivoSostenibilita" non è NULL;
- Vengono raggruppati i risultati per "obbiettivoSostenibilita" e calcolati il conteggio dei progetti e la somma dei finanziamenti;
- Vengono ordinati i risultati per "NumeroProgetti" in ordine decrescente.

Le seguenti implementazioni in **Hive, Spark, Pig**:

**Hive**

```
SELECT trim(obbiettivo_Sostenibilita) AS obbiettivo_Sostenibilita,
    COUNT(*) as Numero_Progetti, SUM(Funding_Amount_in_EUR) AS
    Somma_finanziata
FROM (SELECT explode(split(Sustainable_Development_Goals, "; ")) AS
    obbiettivo_Sostenibilita, Funding_Amount_in_EUR
      FROM dataset_unina_research)
WHERE obbiettivo_Sostenibilita IS NOT NULL
GROUP BY obbiettivo_Sostenibilita
ORDER BY Numero_Progetti DESC;
```

**Tabella 2.45:** Numero di progetti di ricerca per ogni obbiettivo di sviluppo sostenibile - **Hive**

**Spark**

```
ProgettiFinanziati = projectsDS.select(explode(split(projectsDS
          ["Sustainable_Development_Goals"], "; "))
              .alias("obbiettivo_Sostenibilita"),
                  "Funding_Amount_in_EUR") \
              .filter(col("obbiettivo_Sostenibilita").isNotNull()) \
              .groupBy(col("obbiettivo_Sostenibilita")) \
              .agg(F.count("*").alias("Numero_Progetti"),
                  F.sum("Funding_Amount_in_EUR")
                  .alias("Somma_finanziata")) \
              .select(F.trim("obbiettivo_Sostenibilita") \
              .alias("obbiettivo_Sostenibilita"), "Numero_Progetti",
                  "Somma_finanziata") \
              .orderBy(F.col("Numero_Progetti").desc())
```

**Tabella 2.46:** Numero di progetti di ricerca per ogni obbiettivo di sviluppo sostenibile - **Spark**

**Pig**

```
obbiettivi_sostenibilita = FOREACH dataset_unina_research GENERATE
    FLATTEN(TOKENIZE(TRIM(Sustainable_Development_Goals), ';')) AS
    obbiettivo_Sostenibilita, Funding_Amount_in_EUR;
obbiettivi_sostenibilita_filtrati = FILTER obbiettivi_sostenibilita BY
    obbiettivo_Sostenibilita IS NOT NULL;
obbiettivi_sostenibilita_contati = GROUP
    obbiettivi_sostenibilita_filtrati BY obbiettivo_Sostenibilita;
obbiettivi_sostenibilita_contati_final = FOREACH
    obbiettivi_sostenibilita_contati GENERATE group AS
    obbiettivo_Sostenibilita, COUNT(obbiettivi_sostenibilita_filtrati)
    AS Numero_Progetti,
    SUM(obbiettivi_sostenibilita_filtrati.Funding_Amount_in_EUR) AS
    Somma_Finanziata;
obbiettivi_sostenibilita_ordinati = ORDER
    obbiettivi_sostenibilita_contati_final BY Numero_Progetti DESC;
```

**Tabella 2.47:** Numero di progetti di ricerca per ogni obbiettivo di sviluppo sostenibile - **Pig**

### 2.12.2 Risultati

La Tabella 2.48 mostra la distribuzione dei progetti di sostenibilità ambientale finanziati dall'Università Federico II in base agli obbiettivi di sviluppo sostenibile dell'ONU. I tre obbiettivi con il maggior numero di progetti finanziati sono Affordable and Clean Energy, Climate Action e Sustainable Cities and Communities. In particolare, questi indicano che la Federico II ha lavorato su progetti riguardanti l'energia pulita, la lotta al cambiamento climatico e lo sviluppo sostenibile delle città e delle comunità. Tali progetti sono rilevanti, in quanto riguardano tematiche cruciali per la sostenibilità ambientale e la qualità della vita delle persone, e richiedono l'impegno di istituzioni, aziende e cittadini per raggiungere gli obbiettivi stabiliti dall'Agenda 2030 delle Nazioni Unite.
**Da Notare**: La Federico II ha finanziato circa 300mln di euro per lo sviluppo di infrastrutture resilienti, promuovendo l'industrializzazione sostenibile e fomentando l'innovazione in queste nuove tecnologie ecosostenibili.

| obbiettivo_Sostenibilita | Numero_Progetti | Somma_finanziata |
|---|---|---|
| 7 Affordable and Clean Energy | 82 | 311321685 |
| 13 Climate Action | 53 | 139135505 |
| 11 Sustainable Cities and Communities | 40 | 318092831 |
| 3 Good Health and Well Being | 34 | 98917877 |
| 12 Responsible Consumption and Production | 25 | 199551657 |
| 16 Peace, Justice and Strong Institutions | 23 | 7537352 |
| 2 Zero Hunger | 21 | 90328741 |
| 4 Quality Education | 15 | 15812081 |
| 9 Industry, Innovation and Infrastructure | 13 | 298867713 |
| 8 Decent Work and Economic Growth | 9 | 10467178 |
| 10 Reduced Inequalities | 9 | 4224395 |
| 15 Life on Land | 7 | 7203903 |
| 6 Clean Water and Sanitation | 4 | 5628593 |
| 14 Life Below Water | 4 | 6744462 |
| 1 No Poverty | 3 | 2291171 |
| 5 Gender Equality | 1 | 2200332 |

**Tabella 2.48:** Risultati della query - Numero di progetti di ricerca per ogni obbiettivo di sviluppo sostenibile

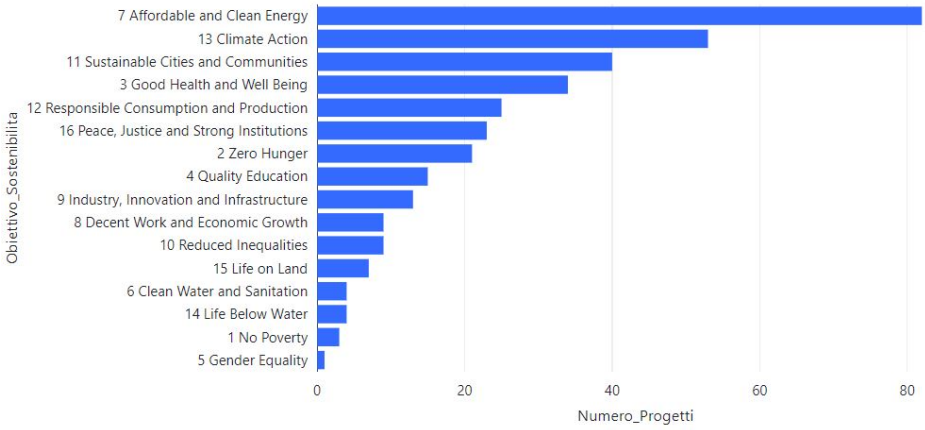


**Figura 2.11:** Bar View - Numero di progetti di ricerca per ogni obbiettivo di sviluppo sostenibile

## 2.13 Numero di progetti di ricerca per ogni tipo di cancro

L'obbiettivo della tredicesima Analytic è quello di analizzare il dataset di progetti di ricerca e calcolare il numero di progetti per ciascun tipo di cancro, escludendo quelli "Not Site-Specific Cancer", e mostrando i primi 25 tipi di cancro con il maggior numero di progetti.

### 2.13.1 Implementazione

Nell'implementazione:

- Viene selezionata e splittata la colonna "CancerTypes";
- Vengono filtrate le righe in cui "TipoCancro" non è NULL ed è diverso da "Not Site-Specific Cancer";
- Vengono raggruppati i risultati per "TipoCancro" e viene calcolato il conteggio dei progetti;
- Vengono ordinati i risultati per "NumeroProgetti" in ordine decrescente e limitando i risultati a 25 righe.

Le seguenti implementazioni in **Hive, Spark, Pig**:

**Hive**

```
SELECT trim(Tipo_Cancro) AS Tipo_Cancro, COUNT(*) as Numero_Progetti
FROM (SELECT explode(split(Cancer_types, "; ")) AS Tipo_Cancro
      FROM dataset_unina_research)
WHERE Tipo_Cancro IS NOT NULL AND Tipo_Cancro != 'Not Site-Specific
    Cancer'
GROUP BY Tipo_Cancro
ORDER BY Numero_Progetti DESC
LIMIT 25;
```

**Tabella 2.49:** Numero di progetti di ricerca per ogni tipo di cancro - **Hive**

**Spark**

```
ProgettiCancro = projectsDS.select(explode(split(projectsDS
          ["Cancer_types"], "; ")).alias("Tipo_Cancro")) \
              .filter(col("Tipo_Cancro").isNotNull()) \
              .filter(col("Tipo_Cancro")!='Not Site-Specific Cancer') \
              .groupBy(col("Tipo_Cancro")) \
              .count() \
              .select(trim("Tipo_Cancro").alias("Tipo_Cancro"),
                 col("count").alias("Numero_Progetti")) \
              .orderBy(col("Numero_Progetti").desc()) \
              .limit(25)
```

**Tabella 2.50:** Numero di progetti di ricerca per ogni tipo di cancro - **Spark**

**Pig**

```
tipi_cancro = FOREACH dataset_unina_research GENERATE
    FLATTEN(TOKENIZE(TRIM(Cancer_Types), ';')) AS Tipo_Cancro;
tipi_cancro_filtrati = FILTER tipi_cancro BY Tipo_Cancro IS NOT NULL AND
    Tipo_Cancro != 'Not Site-Specific Cancer';
tipi_cancro_contati = GROUP tipi_cancro_filtrati BY Tipo_Cancro;
tipi_cancro_contati_final = FOREACH tipi_cancro_contati GENERATE group
    AS Tipo_Cancro, COUNT(tipi_cancro_filtrati) AS Numero_Progetti;
tipi_cancro_ordinati = ORDER tipi_cancro_contati_final BY
    Numero_Progetti DESC, Tipo_Cancro ASC;
tipi_cancro_finali = LIMIT tipi_cancro_ordinati 25;
```

**Tabella 2.51:** Numero di progetti di ricerca per ogni tipo di cancro - **Pig**

### 2.13.2 Risultati

Dalla Tabella 2.52 è possibile notare che sono presenti numerose patologie diverse in cui la Federico II ha lavorato su progetti di ricerca, il che suggerisce che la sua forza potrebbe essere nell'avere un'ampia competenza nella ricerca oncologica. Inoltre, il fatto che la tiroide e il cancro al seno siano le due patologie con il maggior numero di progetti potrebbe indicare una particolare competenza in queste aree.

| Tipo_Cancro | Numero_Progetti |
|---|---|
| Thyroid Cancer | 23 |
| Breast Cancer | 22 |
| Liver Cancer | 19 |
| Colon and Rectal Cancer | 14 |
| Brain Tumor | 8 |
| Stomach Cancer | 8 |
| Oral Cavity and Lip Cancer | 7 |
| Lung Cancer | 7 |
| Leukemia / Leukaemia | 7 |
| Esophageal / Oesophageal Cancer | 6 |
| Pancreatic Cancer | 5 |
| Skin Cancer | 5 |
| Head and Neck Cancer | 4 |
| Pituitary Tumor | 4 |
| Bladder Cancer | 4 |
| Melanoma | 3 |
| Nervous System | 3 |
| Kidney Cancer | 3 |
| Gastrointestinal Tract | 3 |
| Prostate Cancer | 3 |
| Ear Cancer | 2 |
| Pharyngeal Cancer | 2 |
| Laryngeal Cancer | 2 |
| Neuroblastoma | 2 |
| Nasal Cavity and Paranasal Sinus Cancer | 2 |

**Tabella 2.52:** Risultati della query - Numero di progetti di ricerca per ogni tipo di cancro

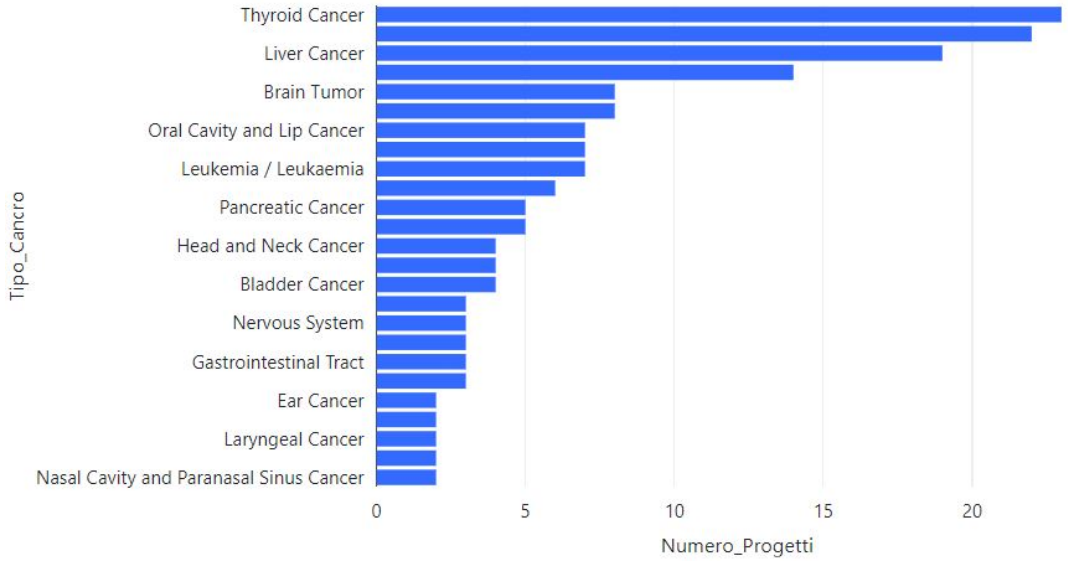


**Figura 2.12:** Bar View - Numero di progetti di ricerca per ogni tipo di cancro

## 2.14 Numero e Somma Finanziata nei settori medici di ricerca

L'obbiettivo della quattordicesima ed ultima Analytic è quello di analizzare il dataset di progetti di ricerca e calcolare il numero di progetti e la somma dei finanziamenti per ciascun settore medico (RCDC Categories), mostrando i primi 10 settori medici con la somma dei finanziamenti più alta e il maggior numero di progetti.

### 2.14.1 Implementazione

Nell'implementazione:

- Viene selezionata e splittata la colonna "RCDCCategories", includendo la colonna "FundingAmountinEUR";
- Vengono filtrate le righe in cui "CatMedica" non è NULL;
- Vengono raggruppati i risultati per "CatMedica" e vengono calcolati il conteggio dei progetti e la somma dei finanziamenti;
- Vengono ordinati i risultati per "SommaFinanziataEUR" in ordine decrescente e "NumeroProgetti" in ordine decrescente, limitando i risultati a 10 righe.

Le seguenti implementazioni in **Hive, Spark, Pig**:

**Hive**

```
SELECT trim(Cat_Medica) AS Settore_Medico, count(*) AS Numero_Progetti,
    SUM(Funding_Amount_in_EUR) AS Somma_Finanziata_EUR
FROM (SELECT explode(split(RCDC_Categories, "; ")) AS Cat_Medica,
    Funding_Amount_in_EUR
      FROM dataset_unina_research)
WHERE Cat_Medica IS NOT NULL
GROUP BY Cat_Medica
ORDER BY int(Somma_Finanziata_EUR) DESC, Numero_Progetti DESC
LIMIT 10;
```

**Tabella 2.53:** Numero e Somma Finanziata nei settori medici di ricerca - **Hive**

**Spark**

```
Settori_Medici = projectsDS.select(explode(split(
          col("RCDC_Categories"), "; ")).alias("Cat_Medica"),
              col("Funding_Amount_in_EUR")) \
              .filter(col("Cat_Medica").isNotNull()) \
              .groupBy(trim(col("Cat_Medica"))
                  .alias("Settore_Medico")) \
              .agg(F.count("*").alias("Numero_Progetti"),
                 F.sum("Funding_Amount_in_EUR")
                     .alias("Somma_Finanziata_EUR")) \
              .orderBy(col("Somma_Finanziata_EUR").desc(),
                 col("Numero_Progetti").desc()) \
              .limit(10)
```

**Tabella 2.54:** Numero e Somma Finanziata nei settori medici di ricerca - **Spark**

**Pig**

```
settori_medici = FOREACH dataset_unina_research GENERATE
    FLATTEN(TOKENIZE(TRIM(RCDC_Categories), ';')) AS Cat_Medica,
    Funding_Amount_in_EUR;
settori_medici_filtrati = FILTER settori_medici BY Cat_Medica IS NOT
    NULL;
settori_medici_contati = GROUP settori_medici_filtrati BY Cat_Medica;
settori_medici_contati_final = FOREACH settori_medici_contati GENERATE
    group AS Settore_Medico, COUNT(settori_medici_filtrati) AS
    Numero_Progetti, SUM(settori_medici_filtrati.Funding_Amount_in_EUR)
    AS Somma_Finanziata_EUR;
settori_medici_ordinati = ORDER settori_medici_contati_final BY
    Somma_Finanziata_EUR DESC, Numero_Progetti DESC;
settori_medici_finali = LIMIT settori_medici_ordinati 10;
```

**Tabella 2.55:** Numero e Somma Finanziata nei settori medici di ricerca - **Pig**

### 2.14.2 Risultati

In accordo con i risultati della query precedente, nella Tabella 2.56, si può notare che la Federico II ha lavorato su molti progetti di ricerca e sviluppo nei settori medici, in particolare in Genetics, Biotechnology e Clinical Research. La somma finanziata per i vari progetti varia notevolmente, ma questo potrebbe essere dovuto alla complessità e alla durata dei progetti stessi. In generale, si può dire che la Federico II si concentra sulla ricerca e lo sviluppo di nuove terapie e tecnologie per migliorare la salute umana.

| Settore_Medico | Numero_Progetti | Somma_Finanziata_EUR |
|---|---|---|
| Neurosciences | 60 | 316032667 |
| Genetics | 146 | 255874341 |
| Bioengineering | 44 | 217362848 |
| Biotechnology | 99 | 188532055 |
| Prevention | 55 | 172584138 |
| Clinical Research | 76 | 158073322 |
| Nutrition | 34 | 102271921 |
| Human Genome | 35 | 98437210 |
| Rare Diseases | 72 | 88654024 |
| Digestive Diseases | 41 | 76116470 |

**Tabella 2.56:** Risultati della query: Numero e Somma Finanziata nei settori medici di ricerca

## 2.15 Esecuzione definitiva: Hive - PySpark - Pig

Si lascia il lettore ad eseguire i seguenti notebook:

**Hive - DataBricks**

`https://shorturl.at/sKOS2`

**PySpark - Colab**

`https://colab.research.google.com/drive/1lgeCpTdMp-aDPy4cMqcJuhbkXfrqDdHY`

**Pig - Hadoop**[1]

`https://github.com/giuseppericcio/BigData/tree/main/HW1/ScriptPigLatin`

È possibile inoltre visualizzare il report realizzato con **Streamlit** al seguente link:

`https://progettiunina.streamlit.app/`

[1] **Si noti** che per eseguire gli script di Pig bisogna necessariamente configurare e installare correttamente Hadoop e Pig su un'opportuna macchina virtuale.

# 3 Conclusioni e Confronto tra le piattaforme

Contenuti



## 3.1 Il Confronto

Per confrontare le piattaforme utilizzate nella seguente trattazione si possono considerare i seguenti fattori:

- **Compatibilità e tecnologie**: bisogna valutare la compatibilità delle piattaforme con i tool e le tecnologie esistenti, come i database, i linguaggi di programmazione e i framework di analisi dei dati.
- **Prestazioni**: bisogna valutare le prestazioni di ogni piattaforma in termini di velocità di esecuzione, tempo di risposta e scalabilità. Si può utilizzare un insieme di query di benchmarking standard per eseguire test comparativi tra le piattaforme.

  Le query che si prendono in considerazione sono:

  - *Capacità di gestione delle risorse finanziarie dei progetti negli ultimi 20 anni*;
  - *Le parole chiavi più utilizzate nei titoli dei progetti*;
  - *Gli ambiti di ricerca con progetti di maggiore durata*.
- **Facilità d'uso**: bisogna considerare la facilità con cui è possibile creare e gestire workflow, configurare e monitorare le risorse e le query, e il livello di documentazione e supporto disponibile per ogni piattaforma.

### 3.1.1 Compatibilità e tecnologie

Come ampiamente scritto nelle descrizioni dei framework in precedenza, ciascuno di esse deve essere opportunamente configurato ed installato al fine di eseguire l'analisi dei dati correttamente ed efficientemente. Pertanto si discriminano le feature hardware delle macchine su cui vengono eseguite le piattaforme:

| Macchina su cui è installata la piattaforma | Features | | |
|---|---|---|---|
| | **CPU** | **RAM** | **Disco** |
| **PySpark - Google Compute Engine** | Intel Xeon, 2.3 GHz | 12,7 GB | 107,7 GB |
| **Databricks - Cluster Virtuale Remoto** [1] | Dual Core | 15,3 GB | — |
| **Pig Hadoop - Locale (WSL)** | Intel-i71165G7 | 7,8 GB | 476 GB |

**Tabella 3.1:** Caratteristiche Hardware su cui vengono installate le piattaforme

**Si noti** che non sono state trovate informazioni più dettagliate per il Cluster virtuale remoto di Databricks.

### 3.1.2 Prestazioni

Per effettuare una comparazione tra i 3 frameworks, si scelgono le query ipoteticamente più complesse e si valutano i tempi di esecuzione. La complessità della query è stata valutata in funzione del: numero di subquery, numero di funzioni ed operazioni di manipolazione e struttura della query.
I risultati:

| Query | Tempo di esecuzione in media/query | | |
|---|---|---|---|
| | **HiveQL** | **PySpark** | **Pig** |
| **Capacità di gestione delle risorse finanziarie dei progetti negli ultimi 20 anni** | 0,93 s | 0,46 s | 12,57 s |
| **Le parole chiavi più utilizzate nei titoli dei progetti** | 2,77 s | 1,29 s | 12,31 s |
| **Gli ambiti di ricerca con progetti di maggiore durata** | 1,22 s | 0,38 s | 12,49 s |

**Tabella 3.2:** Tempi di esecuzione - Hive, Pig, Spark

Dalla Tabella 3.2 dei tempi di esecuzione delle query emerge che PySpark ha prestazioni migliori rispetto ad HiveQL e Pig. In particolare, per la query sulla capacità di gestione delle risorse finanziarie dei progetti negli ultimi 20 anni, PySpark ha impiegato circa la metà del tempo di esecuzione di HiveQL. Per la query sulle parole chiavi più utilizzate nei titoli dei progetti, PySpark ha impiegato poco più della metà del tempo di esecuzione di HiveQL. Infine, per la query sugli ambiti di ricerca con progetti di maggiore durata, PySpark ha impiegato circa un terzo del tempo di esecuzione di HiveQL. Questi dati suggeriscono che, in generale, **PySpark è la piattaforma più performante delle tre per l'esecuzione di query su grandi dataset**. Questo proprio perché l'esecuzione delle query avviene su hardware più performante.

**Si noti**, infine, che i tempi di esecuzione di Pig risultano nettamente peggiori rispetto alle altre due piattaforme, ciò è dovuto in maggior parte alla fase di parserizzazione e traduzione delle query da Pig Latin in MapReduce (ovvero Java); questa caratteristica rende particolarmente svantaggioso lavorare con Pig quando tali tecniche non sono strettamente necessarie come nel caso in esame.

### 3.1.3 Facilità d'uso

#### 3.1.3.1 In relazione all'architettura

Nel caso in esame, sono state usate tecnologie e configurazioni delle piattaforme già facilitate attraverso l'uso di macchine virtuali e/o macchine in cloud, come già descritto nel Capitolo 1.3.
A partire dai workflows, mostrati nella Figura 3.1, è possibile dedurre la lunghezza della relativa pipeline, ciò significa che il numero di operazioni da fare prima di effettuare un Analytic è tanto maggiore quanto più lunga è la pipe. Pertanto osservando i 3 workflows in questione, si può evincere che Pig ha una complessità maggiore rispetto a PySpark e Hive, perché il suo workflow prevede più passaggi, ciò è un ulteriore fattore che condiziona anche i tempi di esecuzione visti prima.
Mentre, si può notare che Databricks-Hive e PySpark presentano dei workflows con la pipe minore e per questo motivo risultano spesso anche i più semplici e rapidi da usare per l'utente.

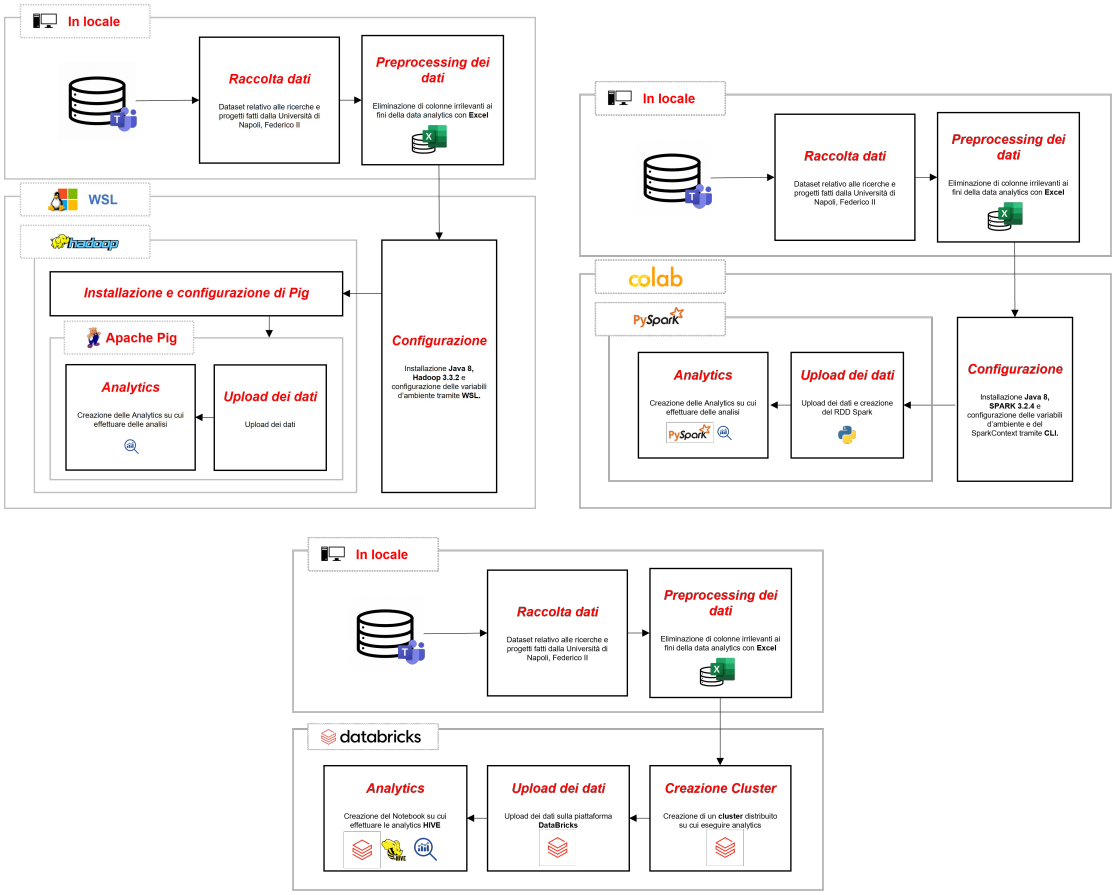


**Figura 3.1:** Workflows Hive, PySpark, Pig

#### 3.1.3.2 In relazione alla curva di apprendimento del linguaggio utilizzato

In generale, come già detto nei paragrafi precedenti, **Hive** utilizza un linguaggio di query basato su SQL, noto come HiveQL, che può risultare familiare a chi ha esperienza con il linguaggio SQL. Pertanto, per gli utenti che hanno familiarità con SQL, Hive potrebbe risultare relativamente facile da utilizzare.

Per quanto riguarda **Spark**, si è avvalsi dell'utilizzo delle API PySpark che forniscono alcune funzionalità per l'elaborazione dei dati basate su SQL, noto come SparkSQL, che utilizza sintassi SQL-like. Spark offre una documentazione molto dettagliata e una vasta comunità di supporto che può facilitare l'apprendimento della piattaforma rispetto ad Hive e a Pig. Inoltre, grazie all'interfacciamento con Python è possibile estendere tutte le librerie dello stesso rendendo il framework più flessibile.

Mentre **Pig** utilizza un proprio linguaggio di scripting, noto come Pig Latin. Tuttavia, il linguaggio Pig Latin è relativamente intuitivo e basato su un insieme di costrutti di base per l'elaborazione dei dati. Pertanto, per gli utenti che non hanno familiarità con altri linguaggi di programmazione, Pig potrebbe risultare relativamente facile da imparare.

#### 3.1.3.3 In relazione al numero di linee di codice

Un ulteriore aspetto da non sottovalutare nella discriminazione tra le varie piattaforme, è quello relativo al numero di linee di codice che servono per definire una query nei diversi linguaggi. Nella seguente Tabella, sono riportate le linee di codice necessarie ad eseguire le tre query considerate finora in ciascuno dei 3 linguaggi utilizzati:

| Query | Linee di Codice (LOC) | | |
|---|---|---|---|
| | **HiveQL** | **PySpark** | **Pig** |
| **Capacità di gestione delle risorse finanziarie dei progetti negli ultimi 20 anni** | 8 | 8 | 7 |
| **Le parole chiavi più utilizzate nei titoli dei progetti** | 7 | 8 | 7 |
| **Gli ambiti di ricerca con progetti di maggiore durata** | 6 | 5 | 5 |

**Tabella 3.3:** Linee di Codice - Hive, Pig, Spark

Pertanto considerando tutte le query realizzate la media delle linee di codice utilizzate è:

| Piattaforma | Numero di linee di codice medio |
|---|---|
| **Pig** | 6 |
| **HiveQL** | 7 |
| **PySpark** | 7 |

**Tabella 3.4:** Numero di linee di codice medio per la costruzione di una query

**Si noti**: il numero di linee di codice medio delle query delle rispettive piattaforme non tengono conto delle linee di codice con cui avviene il load del dataset.

## 3.2 Conclusioni

Dati i confronti effettuati nel paragrafo precedente, ai fini dello svolgimento dell'homework, è risultato più semplice usare la piattaforma e il framework **Databricks-Hive**. Questa affermazione è data sia dalla facilità di installazione, ma soprattutto dall'uso intuitivo e veloce della piattaforma stessa.
Tuttavia, non è da sottovalutare la flessibilità data da **PySpark**, essendo tale piattaforma integrata tramite API in Python, il che le permette di sfruttare tutti quei vantaggi propri di Python stesso, come l'abbondante quantità di librerie già implementate per la data analysis e le varie visualizzazioni oltre, in caso di necessità, alla presenza di numerose librerie per il Machine Learning.
Infine, **Pig** risulta essere la piattaforma con un linguaggio di interrogazione (Pig Latin) di maggiore leggibilità.

# Elenco delle figure

# Elenco delle tabelle

# Bibliografia

[1] Autori del seguente homework. *Slide del corso*. 2023.