



TED GO

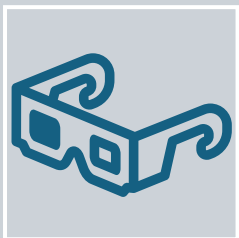
I TUOI TEDX OVUNQUE TU SIA!

Beccarelli Raissa mat. 1086785

Locatelli Giacomo mat. 1086262

Valceschini Marco mat. 1086356

JOB IMPLEMENTATI



JOB PER I WATCH-NEXT

Questo job è stato sviluppato per poter inserire nel database principale un **array di ID** corrispondenti ai video correlati al video che si sta guardando, con l'obiettivo di creare un array di **watch-next** per ciascun video. In questo modo, una volta finito di vedere un determinato video, sarà possibile consigliarne altri agli utenti in modo opportuno. Infatti l'idea è che un utente che guarda un determinato video-talk sia poi interessato a vederne altri che trattano lo stesso argomento o tematiche simili. Ovviamente, video correlati possono non appartenere allo stesso canale tematico (vedi sotto) e quindi questo job consentirà di proporre nell'applicazione finale una «griglia» di video correlati che rimandano l'utente al talk specifico sul sito di TedX.



JOB PER LA SUDDIVISIONE DEI CANALI

Questo job è stato sviluppato in linea con gli scopi del nostro progetto. La sua funzione principale è quella di restituire per ciascun **tag** l'elenco dei **video associati** sotto forma di ID. In questo modo possiamo poi effettuare una **suddivisione** rapida nei nostri canali tematici (arte e design, business, scienza, educazione, intrattenimento, politica, sport e tecnologia). Il job produce una risposta sotto forma di array, la quale permetterà nell'applicazione finale di avere a disposizione una coda di riproduzione che possa anche fare da programmazione dei canali, come accade per il palinsesto tv, in modo che gli utenti possano sapere quali sono i prossimi talk che verranno riprodotti. Inoltre, considerando il fatto che il dataset ha una dimensione fissa, la struttura ad array permetterà un accesso più rapido e semplice ai dati.

JOB PER I WATCH NEXT

```
#FROM FILES
watch_next_dataset_path= "s3://tedx-2025-data-br-20252315/related_videos.csv"

#READ THE NEXT VIDEO
watch_next_dataset=spark.read\
    .option("header","true") \
    .option("quote", "\"") \
    .option("escape", "\\") \
    .csv(watch_next_dataset_path)

watch_next_dataset=watch_next_dataset.dropDuplicates()

#CREATE A TABLE WHERE THERE IS ONLY ID AND INTERNALID
watch_next_dataset_prova=watch_next_dataset.select((col("id")).alias("IDtradotto"), (col("internalId")).alias("internalIDtradotto"))
```

Abbiamo quindi eseguito **due join**:

- il primo join tra il dataset di related-videos e la tabella intermedia, in modo da ottenere una tabella unificata con gli ID reali;
- Il secondo join tra quest'ultima tabella e il dataset tedx_dataset, per poter associare a ciascun video i suoi correlati.

Infine abbiamo applicato una **groupBy** su ogni ID e una **collect_list** per poter creare un array di video watch-next per ogni video.

Per sviluppare la **funzionalità watch-next**, siamo partiti dalla tabella fornita related-videos.

Successivamente, abbiamo creato una tabella intermedia che ci ha permesso di mappare gli ID interni alla tabella con i loro ID reali.

```
#JOIN OF THE TABLE OF RELATED VIDEOS WITH THE TABLE ABOVE AND WITH THE FINAL LIST
watch_next_dataset_tradotto=watch_next_dataset.join(watch_next_dataset_prova, watch_next_dataset.related_id==watch_next_dataset_prova.internalIDtradotto, "left").drop("internalId", "related_id", "slug", "title", "duration", "viewedCount", "presenterDisplayName")

watch_next_dataset_join=tedx_dataset_agg.join(watch_next_dataset_tradotto, tedx_dataset_agg._id==watch_next_dataset_tradotto.id, "left")

#CREATE THE FINAL TABLE
tedx_dataset_complete=watch_next_dataset_join.groupBy(col("_id")) \
    .agg(array_distinct(collect_list("IDtradotto")).alias("watch_next"), \
        first("slug").alias("slug"), first("speakers").alias("speakers"), \
        first("title").alias("title"), first("url").alias("url"), \
        first("description").alias("description"), first("duration").alias("duration"), \
        first("publishedAt").alias("publishedAt"), first("tags").alias("tags"))
```

JOB PER I WATCH NEXT

```
_id: "567505"
▶ watch_next: Array (3)
  slug: "ben_proudfoot_the_true_story_of_the_iconic_tagline_because_i_m_worth_i..."
  speakers: "Ben Proudfoot"
  title: "The true story of the iconic tagline "Because I'm worth it." | The Fin..."
  url: "https://www.ted.com/talks/ben_proudfoot_the_true_story_of_the_iconic_t..."
  description: "From two-time Oscar winner Ben Proudfoot comes THE FINAL COPY OF ILON ..."
  duration: "1059"
  publishedAt: "2025-03-07T13:49:56Z"
▶ tags: Array (8)
```

```
_id: "567505"
▼ watch_next: Array (3)
  0: "417942"
  1: "422805"
  2: "511250"
  slug: "ben_proudfoot_the_true_story_of_the_iconic_tagline_because_i_m_worth_i..."
  speakers: "Ben Proudfoot"
  title: "The true story of the iconic tagline "Because I'm worth it." | The Fin..."
  url: "https://www.ted.com/talks/ben_proudfoot_the_true_story_of_the_iconic_t..."
  description: "From two-time Oscar winner Ben Proudfoot comes THE FINAL COPY OF ILON ..."
  duration: "1059"
  publishedAt: "2025-03-07T13:49:56Z"
▶ tags: Array (8)
```

Il nostro dataset si compone quindi di **dieci attributi** totali per ogni video, tra cui anche l'array di watch-next che si trova subito dopo l'id del video e che, nel caso di questo video, contiene solo tre ID di video correlati.

JOB PER I CANALI

```
#ELIMINA TUTTE LE TUPLE CHE NON HANNO L'URL, IL TITOLO E LA DESCRIZIONE
tedx_dataset_agg=tedx_dataset_agg.filter((col("url").rlike(r"^https?://")) | (col("title").isNotNull()) | (col("description").isNotNull()))

#CANCELLA LE COLONNE CHE NON SONO DI NOSTRO INTERESSE
tedx_dataset_agg=tedx_dataset_agg.drop("slug").drop("speakers").drop("title").drop("url").drop("description").drop("publishedAt")

#CREO LA TABELLA CON SOLO ID, DURATA E TAG
tedx_dataset_explode=tedx_dataset_agg.select(col("_id"), col("duration"), explode(col("tags")).alias("tag"))

#ELIMINO LE TUPLE DUPLICATE
tedx_dataset_explode= tedx_dataset_explode.dropDuplicates()

#ELIMINIAMO LE TUPLE CHE HANNO DURATA NULLA E MAGGIORE DI 900 SECONDI
tedx_dataset_explode=tedx_dataset_explode.filter((col("duration")<900) | (col("duration").isNotNull()))
```

Abbiamo poi filtrato nuovamente i dati, per avere nella tabella solo i video con una durata minore di **15 minuti** (900 secondi) e con una durata che non fosse nulla.

Abbiamo poi creato il dataset che **aggregasse** per ciascun tag gli id dei video relativi e abbiamo poi tolto la colonna duration che non era più di nostro interesse.

Successivamente abbiamo modificato nuovamente il dataset prendendo solo i tag che sono di nostro interesse per i «canali», ovvero scienze, arte, design, sport, salute, politica, tecnologia, economia, business, intrattenimento ed educazione

Partendo dal dataset che avevamo inizialmente, abbiamo **filtrato** i dati per avere un controllo sulla forma dell'url, sul titolo e sulla descrizione del video che non devono essere nulli.

Abbiamo poi **eliminato** delle colonne che non erano di nostro interesse (slug, speakers, titolo, url, descrizione e giorno di pubblicazione). Successivamente abbiamo eseguito un **explode** sui tag, che erano stati precedentemente aggregati, per avere su ogni riga un id, la durata del video e un solo tag a cui il video si riferisce.

```
#RAGGRUPPIAMO PER TAG TUTTI GLI ID ED ELIMINIAMO LA COLONNA DURATION
tedx_dataset_canali = tedx_dataset_explode.groupBy("tag") \
    .agg(collect_list("_id").alias("id_associati")) \
    .withColumnRenamed("tag", "_id").drop(col("duration"))

#PRENDIAMO SOLO I TAG CHE SONO DI NOSTRO INTERESSE
tedx_dataset_canali = tedx_dataset_canali.filter(
    (col("_id")=="science") |
    (col("_id")=="art") | (col("_id")=="design") |
    (col("_id")=="sports") | (col("_id")=="health") |
    (col("_id")=="politics") |
    (col("_id")=="technology") |
    (col("_id")=="economics") | (col("_id")=="business") |
    (col("_id")=="entertainment") |
    (col("_id")=="education") )
```

JOB PER I CANALI

Abbiamo poi creato delle «sotto-tabelle» che unissero i tag che abbiamo intenzione di mostrare nello **stesso canale** (come arte e design), facendo in modo di avere un'unica colonna che li rappresentasse. Abbiamo fatto un **distinct** sull'unione dei due tag, per non avere ripetizioni dello stesso id. Infine abbiamo preso i tag rimanenti e li abbiamo uniti, per formare un **unico dataset** con i tag utili per il nostro progetto.

```
#UNIAMO I TAG CHE SARANNO ALL'INTERNO DELLO STESSO "CANALE"
```

```
arte_design = tedx_dataset_canali.filter(col("_id").isin("arte", "design"))  
sports_health = tedx_dataset_canali.filter(col("_id").isin("sports", "health"))  
economics_business = tedx_dataset_canali.filter(col("_id").isin("economics", "business"))
```

```
#UNIAMO GLI ID TRA I TAG DELLO STESSO CANALE E FACCIAMO IL DISTINCT TRA GLI ID
```

```
a_d = arte_design.agg(array_distinct(flatten(collect_list("id_associati"))).alias("id_associati")) \  
    .withColumn("_id", lit("arte_design"))  
s_h = sports_health.agg(array_distinct(flatten(collect_list("id_associati"))).alias("id_associati")) \  
    .withColumn("_id", lit("sports_health"))  
e_b = economics_business.agg(array_distinct(flatten(collect_list("id_associati"))).alias("id_associati")) \  
    .withColumn("_id", lit("economics_business"))
```

```
#ANDIAMO A COSTRUIRE IL RISULTATO FINALE
```

```
altri_id = tedx_dataset_canali.filter(col("_id").isin("science", "politics", "technology", "entertainment", "education"))  
risultato_finale = altri_id.unionByName(a_d.select("_id", "id_associati"))  
risultato_finale = risultato_finale.unionByName(s_h.select("_id", "id_associati"))  
risultato_finale = risultato_finale.unionByName(e_b.select("_id", "id_associati"))
```


JOB PER I CANALI

In questo modo ci ritroviamo ad avere tutti i tag che ci saranno utili per il nostro progetto e per la suddivisione in «canali tematici».

```
_id: "entertainment"
▸ id_associati : Array (428)

_id: "education"
▸ id_associati : Array (1203)

_id: "science"
▸ id_associati : Array (1613)

_id: "politics"
▸ id_associati : Array (352)

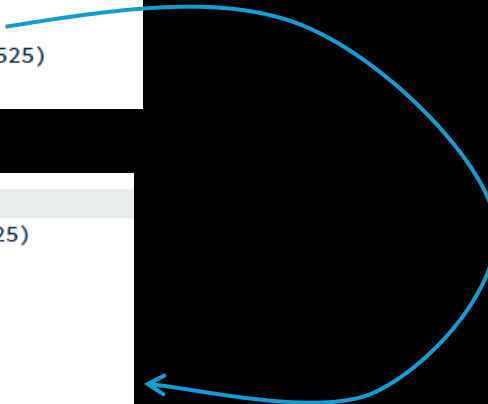
_id: "economics_business"
▸ id_associati : Array (1009)

_id: "arte_design"
▸ id_associati : Array (669)
```

```
_id: "sports_health"
▸ id_associati : Array (879)

_id: "technology"
▸ id_associati : Array (1525)
```

```
_id: "technology"
▾ id_associati : Array (1525)
  0: "544918"
  1: "516510"
  2: "498820"
  3: "446598"
  4: "429297"
  5: "403664"
  6: "402679"
  7: "394122"
  8: "340209"
  9: "323659"
 10: "306100"
```



CRITICITA'

Uno dei primi problemi avuti è quello relativo **all'aggiornamento del database MongoDB**, in quanto il rieseguire lo stesso job più volte non riusciva a sovrascrivere il database precedente.

Un altro problema è nato per la **uplicazione**. Infatti, per i **watch-next** produceva all'interno dell'array più volte lo stesso id, mentre per il database relativo al secondo job ci venivano generati più volte gli stessi **tag** con gli stessi array di id.

Relativamente al secondo job, abbiamo riscontrato difficoltà con il database MongoDB, in quanto questo necessita di avere per ciascuna riga **l'attributo _id**. La difficoltà riscontrata è stata nel fatto che non ci permetteva di assegnare all'attributo **_id** i nomi dei tag.

Un ultimo problema avuto è relativo alla cancellazione delle righe con all'interno dei **valori nulli**. Infatti, abbiamo dovuto fare più prove per riuscire a capire come cancellarli.

I NOSTRI LINK

