requeteSpark_011

July 24, 2020

1 4. Requêtes Spark SQL

```
In [1]: # import des librairies
        from pyspark import SparkContext
        from pyspark.sql import SparkSession, Row
```

```
In [2]: # Initialisation de spark
        sc = SparkContext()
        spark = SparkSession.builder.config("spark.sql.broadcastTimeout", "36000").getOrCreate()
```

```
In [3]: # Définition du mot à chercher
        mot_cle = 'cinéma surréaliste'.lower()
```

1.0.1 4.1. Création des dataframes à partir des fichiers avro:

```
In [4]: # Chargement des données sur history contribition
        path_contribHistory = "hdfs://localhost:8000/data/frwiki/frwiki-20200201/master/full/frw
        df_contribHistory = spark.read.format("avro").load(path_contribHistory)
        df_contribHistory.printSchema()
        df_contribHistory.createOrReplaceTempView("pagesHistory")
```

```
root
 |-- p_title: string (nullable = true)
 |-- p_namespace: long (nullable = true)
 |-- p_id: long (nullable = true)
 |-- p_revisions: array (nullable = true)
 |    |-- element: struct (containsNull = true)
 |    |    |-- r_id: long (nullable = true)
 |    |    |-- r_parent_id: long (nullable = true)
 |    |    |-- r_timestamp: string (nullable = true)
 |    |    |-- r_contributor: struct (nullable = true)
 |    |    |    |-- r_username: string (nullable = true)
 |    |    |    |-- r_contributor_id: long (nullable = true)
 |    |    |    |-- r_contributor_ip: string (nullable = true)
 |    |    |-- r_minor: string (nullable = true)
 |    |    |-- r_comment: string (nullable = true)
 |    |    |-- r_model: string (nullable = true)
 |    |    |-- r_format: string (nullable = true)
 |    |    |-- r_text: struct (nullable = true)
 |    |    |    |-- r_text_id: long (nullable = true)
 |    |    |    |-- r_text_bytes: long (nullable = true)
 |    |    |-- r_sha1: string (nullable = true)
```

```
In [5]: # Charger les données sur les liens des pages
        path_pagelink = 'hdfs://localhost:8000/data/frwiki/frwiki-20200201/master/full/frwiki-20
        df_pl = spark.read.format("avro").load(path_pagelink)
        df_pl.printSchema()
```

```
root
 |-- pl_from: long (nullable = true)
 |-- pl_namespace: long (nullable = true)
 |-- pl_title: string (nullable = true)
 |-- pl_from_namespace: long (nullable = true)
```

```
In [6]: # Enlever les colonnes qu'on n'aura pas besoins et renommer celles à garder pour nous re
        df_pl2 = df_pl.drop("pl_namespace").drop("pl_from_namespace")\
                    .withColumnRenamed("pl_from", "src")\
                    .withColumnRenamed("pl_title", "dest_title")
        df_pl2.printSchema()
        df_pl2.createOrReplaceTempView("pageslink")
```

```
root
 |-- src: long (nullable = true)
 |-- dest_title: string (nullable = true)
```

1.0.2 4.2. Requêtes filtrages des dataframes:

4.2.1. Filtrer les titres qui correspondent à notre sujet

```
In [7]: # Recherche du mot-clé dans le titre de history
        premier_df = spark.sql("""
            SELECT p_id, p_title, p_revisions FROM pagesHistory WHERE LOWER(p_title) LIKE '%{}%'
            """.format(mot_cle))
        premier_df.persist()
        premier_df.createOrReplaceTempView("premier_result")
        premier_df.show()
```

```
+-------+--------------------+--------------------+
|   p_id|             p_title|         p_revisions|
+-------+--------------------+--------------------+
|2785024|  Cinéma surréaliste|[[26419526,, 2008...|
|4394563|Discussion:Cinéma...|[[48947910,, 2010...|
+-------+--------------------+--------------------+
```

4.2.2. Faire les liens sortants

```
In [8]: # Les liens sortants de notre page
        deuxieme_df = spark.sql("""
            SELECT p_id, p_title, p_revisions FROM pagesHistory ph INNER JOIN
            (
                SELECT pageslink.dest_title FROM premier_result INNER JOIN pageslink
                ON (premier_result.p_id = pageslink.src)
            ) pUsingOurLink ON (ph.p_title = pUsingOurLink.dest_title)
            """)
        deuxieme_df.persist()
        deuxieme_df.createOrReplaceTempView("deuxieme_result")
        deuxieme_df.show()
```

```
+-------+------------+--------------------+
|   p_id|     p_title|         p_revisions|
+-------+------------+--------------------+
| 123579|      Fluxus|[[888563,, 2004-0...|
| 876757|      Arzach|[[8209060,, 2006-...|
|   1030|  Eraserhead|[[2289,, 2002-06-...|
|4700547|      Rubber|[[53061781,, 2010...|
|4094875|  Doppelherz|[[44929883,, 2009...|
| 799539|       Freud|[[7302075,, 2006-...|
|  42242|Psychanalyse|[[206063,, 2003-1...|
|1305148|     Destino|[[13714307,, 2007...|
| 746810|   Lettriste|[[6752181,, 2006-...|
|  64290|  Taxidermie|[[344808,, 2004-0...|
|3216891|     Accueil|[[32735716,, 2008...|
| 307239|      Aaltra|[[2594075,, 2005-...|
+-------+------------+--------------------+
```

4.2.3. Faire les liens entrants

```
In [9]: # Les liens entrants dans notre page
        troisieme_df = spark.sql("""
            SELECT p_id, p_title, p_revisions FROM pagesHistory ph
            INNER JOIN
            (
                SELECT pageslink.src FROM pageslink
                WHERE LOWER(dest_title) LIKE '%{}%'
            ) linkThatOurPageUse
            ON (ph.p_id = linkThatOurPageUse.src)
            """.format(mot_cle))
        troisieme_df.persist()
        troisieme_df.createOrReplaceTempView("troisieme_result")
        troisieme_df.show()
```

```
+----+-------+-----------+
|p_id|p_title|p_revisions|
+----+-------+-----------+
+----+-------+-----------+
```

1.0.3 4.3. Union des trois dataframes obtenues:

```
In [10]: # Union verticale des trois dataframes
         union_df = spark.sql("""
                 (SELECT p_revisions.r_contributor FROM premier_result)
                 UNION ALL
                 (SELECT p_revisions.r_contributor FROM deuxieme_result)
                 UNION ALL
                 (SELECT p_revisions.r_contributor FROM troisieme_result)
             """)
         union_df.persist()
         union_df.show()
```

```
+--------------------+
|       r_contributor|
+--------------------+
|[[ARoublev68, 358...|
|[[Bub's wikibot, ...|
|[[,, 65.92.107.10...|
|[[Kostia, 39699,]...|
|[[Shaihulud, 4,],...|
|[[,, 81.67.26.83]...|
|[[Thekeuponsauvag...|
|[[16@r, 40933,], ...|
|[[Charlie brown, ...|
|[[,, 86.198.68.12...|
|[[,, 86.71.191.150]]|
|[[Somniman, 3066,...|
|[[VIGNERON, 3942,...|
|[[Gadro, 16433,],...|
+--------------------+
```

1.0.4 4.4. Extraction des Contributeurs:

```
In [12]: # La fonction explode transforme éléments d'une liste en lignes
         from pyspark.sql.functions import explode
         df_exploded = union_df.select(explode(union_df.r_contributor))
         contributor_df = df_exploded.select(df_exploded.col.r_username, df_exploded.col.r_contr
         contributor_df = contributor_df.withColumnRenamed('col.r_username', 'name')\
                                         .withColumnRenamed('col.r_contributor_id', 'id')\
                                         .withColumnRenamed('col.r_contributor_ip', 'ip')
```

1.0.5 4.5. Tri et présentation du résultat:

```
In [15]: contributor_count_df = contributor_df.groupby(["name", "id", "ip"]).count().orderBy("co
         contributor_count_df.show(3)
```

```
+-----------+-------+----+-----+
|       name|     id|  ip|count|
+-----------+-------+----+-----+
|     Léon66| 100556|null|  234|
|      Jolek| 862133|null|  205|
|Pierrette13|1220016|null|  133|
+-----------+-------+----+-----+
only showing top 3 rows
```

```
In [ ]:
```