Homework 3: Data Pipelines

MongoDB

1. Figure out what the schema of the collection is.

```
> db.embedded_movies.find().limit(1)
< {
  _id: ObjectId("573a1390f29313caabcd5293"),
  plot: "Young Pauline is left a lot of money when her wealthy uncle dies. However, her
uncle's secretary has been named as her guardian until she marries, at which time she will
officially take ...",
  genres: [
    'Action'
  ],
  runtime: 199,
  cast: [
    'Pearl White',
    'Crane Wilbur',
    'Paul Panzer',
    'Edward Josè'
 ],
  num_mflix_comments: 0,
  poster: 'https://m.media-
amazon.com/images/M/MV5BMzgxODk1Mzk2Ml5BMl5BanBnXkFtZTgwMDg0NzkwMjE@._V1_SY1000_SX677_AL_.jp
  title: 'The Perils of Pauline',
  fullplot: `Young Pauline is left a lot of money when her wealthy uncle dies. However, her
uncle's secretary has been named as her guardian until she marries, at which time she will
officially take possession of her inheritance. Meanwhile, her "guardian" and his
confederates constantly come up with schemes to get rid of Pauline so that he can get his
hands on the money himself.,
  languages: [
    'English'
 ],
  released: 1914-03-23T00:00:00.000Z,
  directors: [
    'Louis J. Gasnier',
    'Donald MacKenzie'
  ],
```

- 2. Query that captures the following requirements:
 - 1. Movies with year between 1975 and 1980.
 - 2. Display only 3 columns title, year and runtime.
 - 3. Order by runtime (asc or dsc).
 - 4. Display the top 5 results.

The following displays the first 5 results in ascending order of runtime:

```
},
  {
    $limit: 5
  },
  {
    $project: {
      title: 1,
      year: 1,
      runtime: 1,
      _id: 0
  },
    $project: {
      title: "$title",
      year: "$year",
      runtime: "$runtime"
    }
  }
])
```

The output from the MongoDB compass:

```
title: 'Tyll the Giant',
    year: 1980,
    runtime: 14
}
{
    title: 'Pinocchio',
    year: 1976,
    runtime: 73
}
{
    title: 'Forbidden Zone',
    year: 1980,
    runtime: 74
}
{
    title: 'Allegro non troppo',
    year: 1976,
    runtime: 75
}
{
    title: 'Wizards',
    year: 1977,
    runtime: 80
}
```

There's a script called retrieve_movies.js that can be run using node. Ensure that you create a file called db_creds.json before you run the file. The output is given in movies_query_results.json, this includes all the movies without any limit.

The following displays the top 5 as ranked outputs in ascending order of runtime:

```
},
  {
    $match: {
      rankRuntime: { $lte: 5 }
  }
  },
  {
    $sort: {
      rankRuntime: 1
    }
  },
  {
    $project: {
      title: 1,
      year: 1,
      runtime: 1,
      _id: 0
    }
 },
  {
    $project: {
     title: "$title",
      year: "$year",
      runtime: "$runtime"
  }
])
```

```
title: 'Tyll the Giant',
    year: 1980,
    runtime: 14
}
{
    title: 'Pinocchio',
    year: 1976,
    runtime: 73
}
{
    title: 'Forbidden Zone',
    year: 1980,
    runtime: 74
}
{
    title: 'Allegro non troppo',
    year: 1976,
    runtime: 75
}
{
    title: 'Wizards',
    year: 1977,
    runtime: 80
}
```

3. Write an aggregation aggregating year which calculates sum of all runtime for movies where year is between 1975 and 1980 (inclusive).

The output from the MongoDB compass:

```
{
    year: 1975,
    sumRuntime: 2314
}

{
    year: 1976,
    sumRuntime: 1892
}

{
    year: 1977,
    sumRuntime: 2097
}

{
    year: 1978,
    sumRuntime: 3471
}

{
    year: 1979,
    sumRuntime: 1440
}

{
    year: 1980,
    sumRuntime: 3303
}
```

Spark Databriks

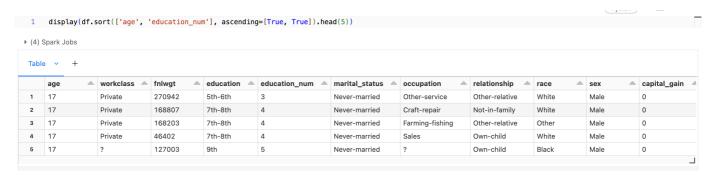
- Notebook that was used can be found in Databricks_Exercise.ipynb.
- You can also find the notebook published on databricks-prod-cloudfront
- Output for "Display top 5 rows ordered in ascending order by age and ascending order by education_num" is given below.

```
from pyspark.sql.types import StructType
from pyspark.sql.types import StructField , IntegerType, StringType, DoubleType
adultSchema = StructType([
    StructField("age", DoubleType(), True),
    StructField("workclass", StringType(), True),
    StructField("fnlwgt", DoubleType(), True),
    StructField("education", StringType(), True),
    StructField("education_num", DoubleType(), True),
    StructField("marital_status", StringType(), True),
    StructField("occupation", StringType(), True),
    StructField("relationship", StringType(), True),
    StructField("race", StringType(), True),
StructField("sex", StringType(), True),
    StructField("capital_gain", DoubleType(), True),
    StructField("capital_loss", DoubleType(), True),
    StructField("hours_per_week", DoubleType(), True),
    StructField("native_country", StringType(), True),
    StructField("income", StringType(), True),
])
df = (
  spark.read
  .format("csv")
  .option("header", "false")
  .schema(adultSchema)
```

```
.load("/databricks-datasets/adult/adult.data")
)
display(df.sort(['age', 'education_num'], ascending=[True, True]).head(5))
```

age	workclas	ss fnlv	wgt	education	ed	ucation_num	marital_status	occupation	
17	Private	270	942	5th-6th	3		Never-married	Other-service	
17	Private	168	807	7th-8th	4		Never-married	Craft-repair	
17	Private	168	3203	7th-8th	4		Never-married	Farming-fishing	
17	Private	464	102	7th-8th	4		Never-married	Sales	
17	?	127	003	9th	5		Never-married	?	
relati	ionship	race	sex	capital_g	ain	capital_loss	ha		•
			00%	capital_9	alli	capital_ioss	hours_per_weel	k native_country	y income
Other	r-relative	White	Male		alli	0	48	Mexico	<=50K
	r-relative			0					
Not-ii		White	Male	0	dili	0	48	Mexico	<=50K
Not-ii	n-family r-relative	White White	Male Male	0 0 0	alli	0 0	48	Mexico United-States	<=50K <=50K

Output from the Databricks Notebook:



If we're considering top 5 by rank, then we get:

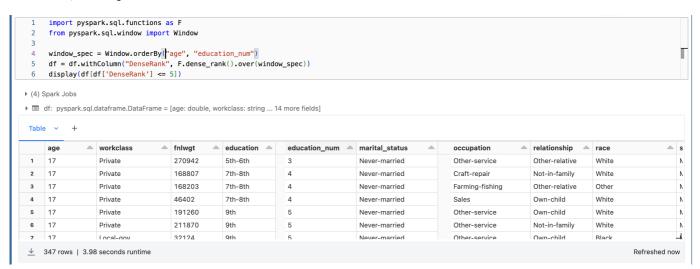
```
import pyspark.sql.functions as F
from pyspark.sql.window import Window
window_spec = Window.orderBy("age", "education_num")
df = df.withColumn("DenseRank", F.dense_rank().over(window_spec))
display(df[df['DenseRank'] <= 5])</pre>
```

age	workclass	fnlwgt	education	education_num	marital_status	occupation	•••
17	Private	270942	5th-6th	3	Never-married	Other-service	
17	Private	168807	7th-8th	4	Never-married	Craft-repair	
17	Private	168203	7th-8th	4	Never-married	Farming-fishing	
17	Private	46402	7th-8th	4	Never-married	Sales	
17	?	127003	9th	5	Never-married	?	
17	Private	221129	9th	5	Married-civ-spouse	Other-service	

age	workclass	fnlwgt	education	education_num	marital_status	occupation	•••
17	?	275778	9th	5	Never-married	?	
17	Private	166290	9th	5	Never-married	Other-service	
17	Private	73145	9th	5	Never-married	Craft-repair	
17	Self-emp-inc	413557	9th	5	Never-married	Sales	
		•••					

<u></u>	relationship	race	sex	capital_gain	capital_loss	hours_per_week	native_country	income	rank_
	Other- relative	White	Male	0	0	48	Mexico	<=50K	1
	Not-in- family	White	Male	0	0	45	United-States	<=50K	2
	Other- relative	Other	Male	0	0	40	Mexico	<=50K	2
	Own-child	White	Male	0	0	8	United-States	<=50K	2
	Own-child	Black	Male	0	0	40	United-States	<=50K	3
	Husband	White	Male	0	0	40	United-States	<=50K	3
	Own-child	White	Female	0	0	25	Mexico	<=50K	3
	Own-child	White	Female	0	0	20	United-States	<=50K	3
	Own-child	White	Female	0	0	16	United-States	<=50K	3
	Own-child	White	Female	0	0	40	United-States	<=50K	3
		•••		•••	•••				

... and so on, resulting in a total of 347 rows.



The complete output is stored in $adult_query_results.csv$.