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
g',
  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:

```
> db.embedded_movies.aggregate([
  {
    $match: {
      year: { $gte: 1975, $lte: 1980 }
    }
  },
  {
    $sort: { runtime: 1 }
```

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

The output from the MongoDB compass:

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:

```
> db.embedded_movies.aggregate([
  {
    $match: {
      year: { $gte: 1975, $lte: 1980 }
    }
  },
  {
    $setWindowFields: {
      sortBy: { runtime: 1 },
      output: {
        rankRuntime: {
          $denseRank: {}
        }
      }
    }
```

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

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

```
> db.embedded_movies.aggregate([
  {
    $match: {
      year: { $gte: 1975, $lte: 1980 }
    }
  },
  {
    $group: {
      _id: "$year",
      sumRuntime: { $sum: "$runtime" }
    }
  },
```

```
    {
      $project: {
        _id: 0,
        year: "$_id",
        sumRuntime: "$sumRuntime"
      }
    }
  ]).sort({ year: 1 })
```

The output from the MongoDB compass:

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.

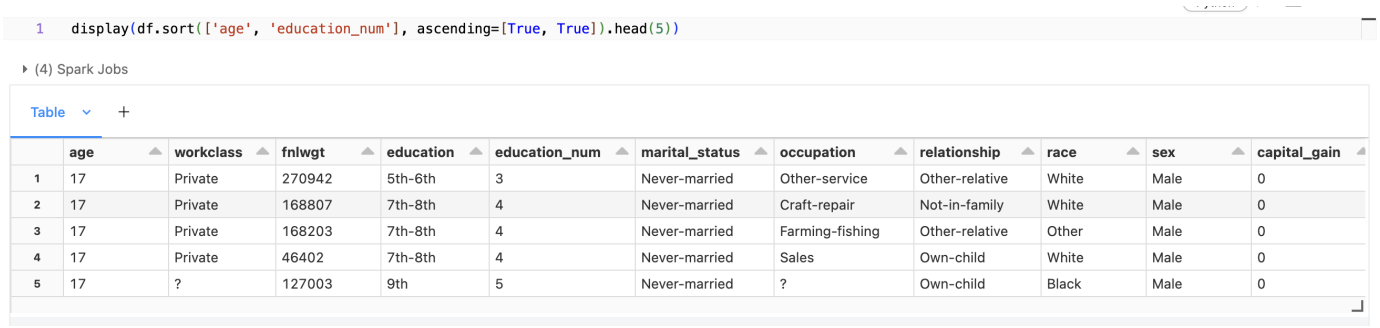
```
df = (
  spark.read
  .format("csv")
  .option("header", "false")
  .schema(adultSchema)
  .load("/databricks-datasets/adult/adult.data")
)
display(df.sort(['age', 'education_num'], ascending=[True, True]).head(5))
```

age	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	?	...

relationship	race	sex	capital_gain	capital_loss	hours_per_week	native_country	income
Other-relative	White	Male	0	0	48	Mexico	<=50K
Not-in-family	White	Male	0	0	45	United-States	<=50K

relationship	race	sex	capital_gain	capital_loss	hours_per_week	native_country	income
Other-relative	Other	Male	0	0	40	Mexico	<=50K
Own-child	White	Male	0	0	8	United-States	<=50K
Own-child	Black	Male	0	0	40	United-States	<=50K

Output from the Databricks Notebook:



```
1  display(df.sort(['age', 'education_num'], ascending=[True, True]).head(5))
```

(4) Spark Jobs

Table +

	age	workclass	fnlwgt	education	education_num	marital_status	occupation	relationship	race	sex	capital_gain
1	17	Private	270942	5th-6th	3	Never-married	Other-service	Other-relative	White	Male	0
2	17	Private	168807	7th-8th	4	Never-married	Craft-repair	Not-in-family	White	Male	0
3	17	Private	168203	7th-8th	4	Never-married	Farming-fishing	Other-relative	Other	Male	0
4	17	Private	46402	7th-8th	4	Never-married	Sales	Own-child	White	Male	0
5	17	?	127003	9th	5	Never-married	?	Own-child	Black	Male	0

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

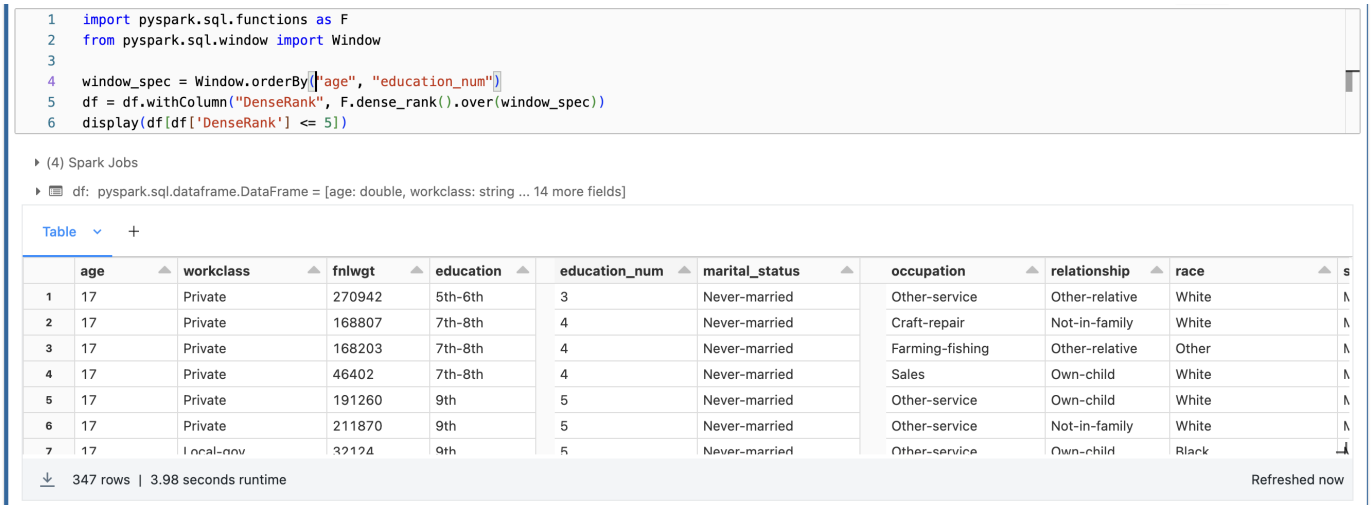
```
import pyspark.sql.functions as F
from pyspark.sql.window import Window

window_spec = Window.orderBy("age", "education_num")
df = df.withColumn("DenseRank", F.dense_rank().over(window_spec))
display(df[df['DenseRank'] <= 5])
```

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	...
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	...
...	...	...	...	...	...	...	...

...	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

...	relationship	race	sex	capital_gain	capital_loss	hours_per_week	native_country	income	rank_
...	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`.