Projeto modernização

Olá caros alunos, como uma das formar de avaliar o conhecimento obtido até este momento, iremos trabalhar com um projeto de modernização de uma demanda fictícia do ministério da saúde, o enunciado a seguir fornece os detalhes.

O ministério da saúde está com uma emergência, e necessita que seja tomada as medidas adequadas em tempo hábil para combater a COVID-19. Para tal demanda, o ministério público abriu um edital emergencial para contratar um profissional que seja responsável por migrar o sistema tradicional de Big Data que está processando em lotes (batches) para processamento contínuo, ou seja, Big Data Stream.

A modernização será feita com base nos exercícios da semana 3 com pequenas mudanças no enunciado, porém foi mantido o mesmo número da questão. É necessário modernizar a coleta de sete informações, sendo que três delas já foram atualizadas para Big Data Stream por outra pessoa que também foi contratada para cumprir a demanda e as outras 4 informações estão sob sua responsabilidade. Você pode usar as informações resolvidas como exemplo para a resolução dos outros exercícios. A baixo segue a lista das informações solicitadas para a modernização.

Considerando o dataset detalhado a seguir, extraia o conjunto de informações solicitadas.

Dataset dados de COVID no Brasil

Os dados utilizados neste trabalho são os mesmos do dataset da semana 3, porém é simulado uma stream de dados. O dataset é enviado para um socket de rede na porta 4545. Cada linha do dataset é enviado para o socket a cada 100 milissegundos.

- Para alimentar a stream de dados execute no terminal:
  - cd ~/BigDataStream/apache_spark/Semana_4
  - ./alimenta_stream.sh projeto (mantenha o comando executando)
- Dados relativos a pacientes que realizaram exames de Covid19 no Brasil
- ~1k de instâncias

#	Nome do campo	Descrição
0	id	identificador
1	dataNotificacao	Data da notificação
2	dataInicioSintomas	Data do inicio dos sintomas
3	dataNascimento	Data de nascimento
4	sintomas	Sintomas do paciente
5	profissionalSaude	Relacionado a profissional de saúde
6	cbo	Ocupação
7	condicoes	Condições do paciente

#	Nome do campo	Descrição
8	estadoTeste	Estado do teste
9	dataTeste	Data do teste
10	tipoTeste	Tipo de teste realizado
11	resultadoTeste	Resultado do Teste
12	paisOrigem	Pais de Origem do paciente
13	sexo	Sexo do paciente
14	bairro	Bairro do paciente
15	estado	Estado do paciente
16	estadoIBGE	Estado do paciente IBGE
17	municipio	Municipio do paciente
18	municipioIBGE	Municipio do paciente
19	cep	CEP
20	origem	Origem do paciente
21	cnes	Código da unidade de saúde
22	estadoNotificacao	Estado da notificação
23	estadoNotificacaoIBGE	Estado da notificação IBGE
24	municipioNotificacao	Município da notificação
25	municipioNotificacaoIBGE	Município da notificação IBGE
26	numeroNotificacao	Número da notificação
27	excluido	ID de exclusão
28	validado	Local validação
29	idade	Idade do paciente
30	dataEncerramento	Data do encerramento da avaliação do paciente
31	evolucaoCaso	Evolução do caso do paciente
32	classificacaoFinal	Avaliação final do caso

#	Informações a serem extraídas:
1.	Quantidade de pacientes positivos para corona virus no último minuto e atualização a cada 30 segundos (resultadoTeste)
3.	Quantidade de pacientes de acordo com o sexo e o resultado do teste nos últimos 50 segudnos e atualização a cada 20 segundos (resultadoTeste)
4.	Sintomas mais comuns para casos positivos para corona virus no último minuto e atualização a cada 30 segundos
6.	Quantidade de casos positivos no Paraná nos últimos 40 segudnos e atualização a cada 20 segundos
15.	Idade das mulheres positivas para covid
16.	Município do Paraná com a maior quantidade de mulheres positivos para covid no último minuto e atualização a cada 20 segundos
17.	Dia da semana com a maior quantidade de testes realizados nos últimos dois minutos e atualização

In [ ]:
```
from pyspark import SparkContext #, SparkConf
from pyspark.streaming import StreamingContext


sc = SparkContext("local[2]", "NetworkWordCount")
sc.setLogLevel("ERROR")

ssc = StreamingContext(sc, 10)
ssc.checkpoint("/tmp/checkpoint")

DStream = ssc.socketTextStream("localhost", 4545)
```

In [ ]:
```
#Informação 1. Quantidade de pacientes positivos para corona virus no últi
result = DStream.filter(lambda linha: linha.split(';')[11]=='Positivo')\
            .countByWindow(60, 30)
```

In [ ]:
```
#Informação 3. Quantidade de pacientes de acordo com o sexo e o resultado 
#e atualização a cada 20 segundos (resultadoTeste)
result = DStream.map(lambda linha: [(linha.split(';')[11], linha.split(';'
            .reduceByKeyAndWindow(lambda valor1, valor2: valor1+valor2, 50
```

In [ ]:
```
#Informação 4. Sintomas mais comuns para casos positivos para corona virus
```

In [ ]:
```
result = DStream.filter(lambda linha: linha.split(';')[11]=='Positivo')\
 .map(lambda linha: [linha.split(';')[4], 1])\
 .reduceByKeyAndWindow(lambda valor1, valor2: valor1+valor2, 60, 30)
```

In [ ]:
```
#Informação 6. Quantidade de casos positivos no Paraná nos últimos 40 segu
```

In [ ]:
```
result = DStream.filter(lambda linha: linha.split(';')[15]=='PARANÁ' and
 linha.split(';')[11]=='Positivo')\
 .countByWindow(40, 20)
```

In [ ]:
```
#Informação 15. Idade das mulheres positivas para covid
```

In [ ]:
```
result = DStream.filter(lambda linha: linha.split(';')[11]=='Positivo' and
 linha.split(';')[13]=='Feminino')\
 .map(lambda linha: int(linha.split(';')[29]))\
 .window(60, 30)
```

In [ ]:
```
#Informação 16. Município do Paraná com a maior quantidade de mulheres pos
#a cada 20 segundos
result = DStream.filter(lambda linha: linha.split(';')[15]=='PARANÁ' and
                                      linha.split(';')[13]=='Feminino' and
                                      linha.split(';')[11]=='Positivo')\
    .map(lambda linha: [linha.split(';')[17], 1])\
    .reduceByKeyAndWindow(lambda valor1, valor2: valor1+valor2, 60, 20)
```

In [ ]:
```
#Informação 17. Dia da semana com a maior quantidade de testes realizados
```

In [ ]:
```
from datetime import date
dia_semana = [
 'Segunda-Feira',
 'Terça-Feira',
 'Quarta-Feira',
 'Quinta-Feira',
 'Sexta-Feira',
 'Sábado',
 'Domingo'
]
```

In [ ]:
```
result = DStream.filter(lambda linha: linha.split(';')[0] != 'id')\
 .map(lambda linha: linha.split(';')[9].split('T')[0])\
 .map(lambda t: [dia_semana[date(int(t.split('-')[0]),
                                 int(t.split('-')[1]),
                                 int(t.split('-')[2]))\
                            .weekday()
                        ], 1])\
 .reduceByKeyAndWindow(lambda v1, v2: v1+v2, 120, 40)
```

In [ ]:
```
result.pprint()

ssc.start()
ssc.awaitTermination()
```

In [ ]:
```

```