1- Uma empresa de transporte público de uma grande capital brasileira está enfrentando um desafio para analisar seus dados. A empresa possui sensores em cada um de seus ônibus, os quais enviam sua localização em tempo real para um banco de dados relacional localizado na sede da empresa. Além disso, a empresa também gerencia dados cadastrais relacionados aos ônibus, estações, motoristas e pontos disponíveis, os quais são armazenados em diferentes planilhas Excel, cada uma com um formato de padronização distinto, e são atualizados uma vez por semana. Os gestores precisam fazer diferentes tipos de análises, incluindo traçar rotas em tempo real para os ônibus e realizar investigações históricas dos dados cadastrais. Considerando este contexto, os gestores da empresa solicitaram a implementação de uma aplicação de big data warehousing englobando todas as fontes de dados supracitadas com o intuito de executar consultas analíticas.
Assinale a alternativa que corresponde à melhor estratégia de implementação.

Deve-se utilizar tanto um data lake quanto um data warehouse para se carregar os dados. O data lake pode receber os dados dos sensores em tempo real via streaming de dados devido à sua frequência de atualização, enquanto recebe o restante dos dados em cargas diárias. Uma vez carregados no data lake, os dados passam por transformações e são carregados no data warehouse a fim de otimizar as consultas analíticas. Desta forma, tem-se consultas em tempo real para os dados dos sensores dos ônibus no data lake e consultas otimizadas para todos os dados históricos no data warehouse.

2 - Considere a etapa de integração de instâncias do processo de ETL relativa ao código em Pandas ilustrado a seguir. Considere que a função processar_conflitos recebe como parâmetro três fontes de dados, os índices e a coluna a ser processada. Considere também que a função retorna uma tabela em Pandas com o resultado final, processando apenas a coluna especificada no parâmetro. Adicionalmente, considere alguns exemplos de dados armazenados nas fontes “fonte1”, “fonte2” e “fonte3”.

```
import pandas as pd


def processar_conflitos(indices, coluna, fonte_A, fonte_B, fonte_C):
    """Função de processamento de conflitos das fontes."""
    for indice in indices:
        fonte_A.at[indice, coluna] = (
            fonte_B.at[indice, coluna] if not pd.isna(fonte_B.at[indice, coluna])
            else fonte_A.at[indice, coluna] if not pd.isna(fonte_C.at[indice, coluna])
            else fonte_C.at[indice, coluna] if not pd.isna(fonte_A.at[indice, coluna])
            else fonte_B.at[indice, coluna]
        )
```

```
    return fonte_A


# Exemplos de dados armazenados nas fontes
fonte1 = pd.DataFrame(columns=["codigo", "nome", "email"], data=[["01", None, "maria@email.com"],
                                                                 ["02", "João", None],
                                                                 ["03", None, None]]).set_index("codigo")
fonte2 = pd.DataFrame(columns=["codigo", "nome", "email"], data=[["01", "Maria", "mariaaa@email.com"],
                                                                 ["02", "João", "joao@email.com"],
                                                                 ["03", "José", None]]).set_index("codigo")
fonte3 = pd.DataFrame(columns=["codigo", "nome", "email"], data=[["01", "Mariah", "mariah@email.com"],
                                                                 ["02", None, "joao@email.com"],
                                                                 ["03", "José", None]]).set_index("codigo")


# Execução do processamento de conflitos
indices = ["01", "02", "03"]
df = processar_conflitos(indices, "nome", fonte1, fonte3, fonte2)
df = processar_conflitos(indices, "email", df, fonte2, fonte3)
```

Q2Process.txt

De acordo com a lógica da função processar_conflitos, pode haver registro com o valor vazio. No resultado final da variável “df”, o registro de código “01” contém o nome “Mariah” e o e-mail “mariaaa@email.com”.

3 - Considere o diagrama conceitual modelado na Figura 1. Assinale a alternativa que corresponde ao que está sendo modelado no diagrama.

Os atributos “PK”, “Destino”, “Data” e “Preço” são extraídos da fonte “passagens aereas” e passam por uma função de tratamento que adiciona o atributo “Região”, que usa como base o atributo “Destino” previamente extraído. Na sequência, os dados são utilizados para a execução de tarefas paralelas de agregação e depois filtrados

usando o novo atributo “Região”, para então serem armazenados em seus respectivos data marts regionais.

4 - Considere o código especificado a seguir, o qual utiliza o método merge() sobre três DataFrames criados a partir dos dicionários “alunos”, “turma1” e “turma2”.

```
'import pandas as pd

alunos = {
   "cpf": ["204.927.060-77", "116.948.760-20", "327.639.610-61", "904.716.030-40",
"750.286.140-83"],
   "nome": ["Joao Paulo", "Jose Carlos", "Maria Eduarda", "Ana Julia", "Carlos
Alberto"],
   "idade": [21, 23, 20, 21, 22]
}

df_alunos = pd.DataFrame(alunos)

turma1 = {
   "cpf": ["204.927.060-77", "116.948.760-20", "327.639.610-61", "904.716.030-40"],
   "nota_progamacao": [9.0, 5.0, 10.0, 8.0]
}

df1 = pd.DataFrame(turma1).query("nota_progamacao > 5")

turma2 = {
   "cpf": ["750.286.140-83", "116.948.760-20", "327.639.610-61", "904.716.030-40"],
   "nota_calculo": [3.0, 4.0, 8.0, 9.0]
}

df2 = pd.DataFrame(turma2).query("nota_calculo > 5")

dfR = df1.merge(df2, how="inner", on="cpf").merge(df_alunos, how="left", on="cpf")
print(dfR)
```

dfR:

| | cpf | nota_progamacao | nota_calculo | nome | idade |
|---|---|---|---|---|---|
| 0 | 327.639.610-61 | 10.0 | 8.0 | Maria Eduarda | 20 |
| 1 | 904.716.030-40 | 8.0 | 9.0 | Ana Julia | 21 |

5 - Considere o trecho de código especificado a seguir.
Considere as seguintes afirmações.
I A transformação flatMap() é usada para criar um novo RDD a partir do retorno da função split() sobre cada elemento do RDD.
II O código realiza a contagem de cada palavra, exceto a palavra Spark.

III As palavras mais frequentes retornadas pelo comando especificado na linha 17 são [('É', 2), ('Olá', 2)].
IV A transformação filter() é usada para filtrar apenas a palavra Spark e adicioná-la ao RDD.
V A ação collect() é usada para retornar uma lista com elementos do RDD e suas respectivas contagens.

Assinale a alternativa que contém apenas as afirmações corretas:

I, II, V.

6 - Considere a constelação de fatos da BI Solutions e a seguinte solicitação de consulta:

“Liste a média dos salários recebidos por escolaridade mínima e por sexo em cada ano”. Arredonde a soma dos salários para até duas casas decimais. Devem ser exibidas as colunas na ordem e com os nomes especificados a seguir: “ANO”, “ESCOLARIDADE”, “SEXO”, “Média dos Salários (R$)”. Ordene as linhas exibidas primeiro por ano em ordem ascendente, depois por escolaridade em ordem ascendente e depois por sexo em ordem ascendente.

Assinale a alternativa que corresponde à consulta em SQL.

```
SELECT dataAno AS ANO,
cargoEscolaridadeMinima AS ESCOLARIDADE,
funcSexo AS SEXO,
ROUND(AVG(salario),2) AS `Média dos Salários (R$)`
FROM pagamento JOIN data ON data.dataPK = pagamento.dataPK
JOIN cargo ON cargo.cargoPK = pagamento.cargoPK
JOIN funcionario ON funcionario.funcPK = pagamento.funcPK
GROUP BY ANO, ESCOLARIDADE, SEXO
ORDER BY ANO, ESCOLARIDADE, SEXO
```

7 –

```
negociacaon
.join(data, on="dataPK")n
.join(equipe, on="equipePK")n
.where("dataAno = 2018")n
.select("equipeNome", "filialNome", "receita")n
.groupBy("equipeNome", "filialNome")n
.sum("receita")n
.orderBy(desc("sum(receita)"))
```

8 –

```
# Resposta da Questão 8
# Não se esqueça de comentar detalhadamente a sua solução

query = """
select dataAno as ANO,
filialCidade as CIDADE,
count(quantidadeNegociacoes) as TOTALNEGOCIACOES
FROM negociacao as a
INNER JOIN data as b
ON a.dataPK = b.dataPK
INNER JOIN equipe as c
ON a.equipePK = c.equipePK
INNER JOIN cliente as d
ON a.clientePK = d.clientePK
where d.clienteCidade = c.filialCidade
group by ANO, CIDADE
order by ANO asc, CIDADE asc, TOTALNEGOCIACOES
"""

# Executa a query (consulta)
result_df = spark.sql(query)

# Mostra os resultados da consulta
print("Resultado utilizando o método spark.sql():")
result_df.show()
```

c. Com exceção do ano de 2020, as negociações realizadas pelas equipes localizadas na cidade de SAO PAULO aumentaram paulatinamente.

9 –

```
pagamento.join(funcionario, on="funcPK").join(data, on="dataPK")\
  .where("funcRegiaoNome = 'SUDESTE'")\
  .select("funcSexo", "dataAno", "salario")\
  .cube("funcSexo", "dataAno").avg("salario")\
  .withColumn("avg(salario)", round("avg(salario)", 2))\
  .orderBy("funcSexo", "dataAno")\
  .withColumnRenamed("funcSexo", "SEXO")\
  .withColumnRenamed("dataAno", "ANO")\
  .withColumnRenamed("avg(salario)", "MEDIASALARIO")\
  .show()



Ou

# Resposta da Questão 9
```

```
# Não se esqueça de comentar detalhadamente a sua solução

query= '''SELECT
      func.funcSexo as SEXO,
      data.dataAno as ANO,
      round(avg(pag.salario),2) as MEDIASALARIO
      from pagamento pag
      INNER JOIN data ON pag.dataPK = data.dataPK
      inner join funcionario func on func.funcPK = pag.funcPK
      where func.funcRegiaoNome ='SUDESTE'
      group by CUBE (SEXO, ANO)
      order by SEXO asc, ANO asc, MEDIASALARIO asc
         '''

result_df = spark.sql(query)

result_df.show()
```

São retornadas 18 linhas, das quais 6 linhas são referentes ao sexo feminino e 6 linhas são referentes ao sexo masculino.

10 –
e.
Os lucros gerados pelas equipes que possuem BI & ANALYTICS em seu nome, independentemente da filial e da cidade na qual estão localizadas, são maiores do que a soma dos lucros gerados pelas demais equipes.