```
#instalando o módulo wget
%%capture
!pip install -q wget
!mkdir data

#baixando os dados das tabelas de dimensão e das tabelas de fatos
import wget

url = "https://raw.githubusercontent.com/cdaciferri/DataMartBISolutions/main/data.csv"
wget.download(url, "data/data.csv")

url = "https://raw.githubusercontent.com/cdaciferri/DataMartBISolutions/main/funcionario.c
wget.download(url, "data/funcionario.csv")

url = "https://raw.githubusercontent.com/cdaciferri/DataMartBISolutions/main/equipe.csv"
wget.download(url, "data/equipe.csv")

url = "https://raw.githubusercontent.com/cdaciferri/DataMartBISolutions/main/cargo.csv"
wget.download(url, "data/cargo.csv")

url = "https://raw.githubusercontent.com/cdaciferri/DataMartBISolutions/main/cliente.csv"
wget.download(url, "data/cliente.csv")

url = "https://raw.githubusercontent.com/cdaciferri/DataMartBISolutions/main/pagamento.csv
wget.download(url, "data/pagamento.csv")

url = "https://raw.githubusercontent.com/cdaciferri/DataMartBISolutions/main/negociacao.cs
wget.download(url, "data/negociacao.csv")
```

```
#instalando Java Runtime Environment (JRE) versão 8
%%capture
!apt-get remove openjdk*
!apt-get update --fix-missing
!apt-get install openjdk-8-jdk-headless -qq > /dev/null
```

```
#baixando Apache Spark versão 3.0.0
%%capture
!wget -q https://archive.apache.org/dist/spark/spark-3.0.0/spark-3.0.0-bin-hadoop2.7.tgz
!tar xf spark-3.0.0-bin-hadoop2.7.tgz && rm spark-3.0.0-bin-hadoop2.7.tgz
```

```
import os
#configurando a variável de ambiente JAVA_HOME
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
#configurando a variável de ambiente SPARK_HOME
os.environ["SPARK_HOME"] = "/content/spark-3.0.0-bin-hadoop2.7"
```

```
%%capture
#instalando o pacote findspark
!pip install -q findspark==1.4.2
#instalando o pacote pyspark
!pip install -q pyspark==3.0.0
```

```
import findspark
findspark.init()
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName("pyspark-notebook").master("local[*]").getOrCreate()

from pyspark.sql.types import IntegerType
from pyspark.sql.types import FloatType
from pyspark.sql.functions import round, desc

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from numpy.random import seed
from tensorflow.random import set_seed
from tensorflow import keras
from tensorflow.keras import layers


#criando os DataFrames em Spark
cargo = spark.read.csv(path="data/cargo.csv", header=True, sep=",")
cliente = spark.read.csv(path="data/cliente.csv", header=True, sep=",")
data = spark.read.csv(path="data/data.csv", header=True, sep=",")
equipe = spark.read.csv(path="data/equipe.csv", header=True, sep=",")
funcionario = spark.read.csv(path="data/funcionario.csv", header=True, sep=",")
negociacao = spark.read.csv(path="data/negociacao.csv", header=True, sep=",")
pagamento = spark.read.csv(path="data/pagamento.csv", header=True, sep=",")


#convertendo os dados necessários para o tipo de dado inteiro
colunas_cargo = ["cargoPK"]
colunas_cliente = ["clientePK"]
colunas_data = ["dataPk", "dataDia", "dataMes", "dataBimestre", "dataTrimestre", "dataSeme
colunas_equipe = ["equipePK"]
colunas_funcionario = ["funcPK", "funcDiaNascimento", "funcMesNascimento", "funcAnoNascime
colunas_negociacao = ["equipePK", "clientePK", "dataPK", "quantidadeNegociacoes"]
colunas_pagamento = ["funcPK", "equipePK", "dataPK", "cargoPK", "quantidadeLancamentos"]

for coluna in colunas_cargo:
  cargo = cargo.withColumn(coluna, cargo[coluna].cast(IntegerType()))

for coluna in colunas_cliente:
  cliente = cliente.withColumn(coluna, cliente[coluna].cast(IntegerType()))

for coluna in colunas_data:
  data = data.withColumn(coluna, data[coluna].cast(IntegerType()))

for coluna in colunas_equipe:
  equipe = equipe.withColumn(coluna, equipe[coluna].cast(IntegerType()))

for coluna in colunas_funcionario:
  funcionario = funcionario.withColumn(coluna, funcionario[coluna].cast(IntegerType()))

for coluna in colunas_negociacao:
  negociacao = negociacao.withColumn(coluna, negociacao[coluna].cast(IntegerType()))

for coluna in colunas_pagamento:
  pagamento = pagamento.withColumn(coluna, pagamento[coluna].cast(IntegerType()))


#convertendo os dados necessários para o tipo de dado float
colunas_negociacao = ["receita"]
colunas_pagamento = ["salario"]

for coluna in colunas_negociacao:
  negociacao = negociacao.withColumn(coluna, negociacao[coluna].cast(FloatType()))

for coluna in colunas_pagamento:
  pagamento = pagamento.withColumn(coluna, pagamento[coluna].cast(FloatType()))


#criando as visões temporárias
cargo.createOrReplaceTempView("cargo")
cliente.createOrReplaceTempView("cliente")
data.createOrReplaceTempView("data")
equipe.createOrReplaceTempView("equipe")
funcionario.createOrReplaceTempView("funcionario")
negociacao.createOrReplaceTempView("negociacao")
pagamento.createOrReplaceTempView("pagamento")


#especificando a consulta que retorna os valores dos atributos
#funcPK, equipePK, dataPK, cargoPK e salario
#para pagamentos realizados na dataAno de 2016
query2016 = query = """
SELECT funcPK, equipePK, pagamento.dataPK, cargoPK, salario
FROM pagamento, data
WHERE pagamento.dataPK = data.dataPK
      AND dataAno = 2016
ORDER BY funcPK, equipePK, pagamento.dataPK, cargoPK, salario
"""

#transformando o resultado em um dataFrame em Pandas
respostaPandas16 = spark.sql(query2016).toPandas()

#exibindo algumas linhas do dataFrame respostaPandas16
respostaPandas16
```

	funcPK	equipePK	dataPK	cargoPK	salario
0	5	2	5	112	2226.659912
1	5	2	36	112	2226.659912
2	5	2	65	112	2226.659912
3	5	2	96	112	2226.659912
4	5	2	126	112	2226.659912

```
#especificando a consulta que retorna os valores dos atributos
#funcPK, equipePK, dataPK, cargoPK e salario
#para pagamentos realizados na dataAno de 2017
query2017 = query = """
SELECT funcPK, equipePK, pagamento.dataPK, cargoPK, salario
FROM pagamento, data
WHERE pagamento.dataPK = data.dataPK
      AND dataAno = 2017
ORDER BY funcPK desc, equipePK, pagamento.dataPK, cargoPK, salario
"""

#transformando o resultado em um dataFrame em Pandas
respostaPandas17 = spark.sql(query2017).toPandas()

#exibindo algumas linhas do dataFrame respostaPandas17
respostaPandas17
```

	funcPK	equipePK	dataPK	cargoPK	salario
0	200	5	371	318	10631.740234
1	200	5	402	318	10631.740234
2	200	5	430	318	10631.740234
3	200	5	461	318	10631.740234
4	200	5	491	318	10631.740234
...	...	...	...	...	...
1315	2	3	583	463	1668.939941
1316	2	3	614	463	1668.939941
1317	2	3	644	463	1668.939941
1318	2	3	675	463	1668.939941
1319	2	3	705	463	1668.939941

1320 rows × 5 columns

```
#Convertendo os dataFrames para numpy array
x_train = respostaPandas16.to_numpy(copy=False)
x_test = respostaPandas17.to_numpy(copy=False)
```

```
#inicializando as sementes
seed(1)
set_seed(1)


#Normalizando min-max para 0-1 (usando max e min do treinamento)
max = x_train.max(axis=0)
min = x_train.min(axis=0)
x_train = (x_train-min)/(max-min)
print(x_train.shape)

x_test = (x_test-min)/(max-min)
print(x_test.shape)
```

```
(600, 5)
(1320, 5)
```

```
#Escreva aqui a sua resposta para o Item 1
#denoising overcomplete autoencoder para pré-treinamento baseado em auto-supervisão
def auto_encoder(input_shape):
    input = keras.layers.Input(shape=(input_shape))
    #encoder:
    a = keras.layers.Dense(4, activation='relu')(input)
    b = keras.layers.Dense(3, activation='relu')(a)
    c = keras.layers.Dropout(0.33)(b)
    d = keras.layers.Dense(4, activation='relu')(c)
    e = keras.layers.Dense(5, activation='relu')(d)
    #decoder
    #decoder = Dense(32,activation='relu',name='input_decoder')(encoder)
    #decoder = Dense(32,activation='relu')(decoder)
    #decoder = Dense(28,activation='tanh')(decoder)
    #autoencoder
    autoencoder = keras.models.Model(input, e)
    autoencoder.summary()
    return autoencoder

modelo_A = auto_encoder(5)
```

```
Model: "functional_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_5 (InputLayer)         [(None, 5)]               0
_________________________________________________________________
dense_9 (Dense)              (None, 4)                 24
_________________________________________________________________
dense_10 (Dense)             (None, 3)                 15
_________________________________________________________________
dropout_2 (Dropout)          (None, 3)                 0
_________________________________________________________________
dense_11 (Dense)             (None, 4)                 16
_________________________________________________________________
dense_12 (Dense)             (None, 5)                 25
=================================================================
Total params: 80
Trainable params: 80
```

```
Non-trainable params: 0
_________________________________________________________________
```

```
# Função que define decaimento para a taxa de aprendizado
def scheduler(epoch, lr):
    return lr * tf.math.exp(-0.005)


#Escreva aqui a sua resposta para o Item 2
batch_size = 16
epochs = 250

modelo_A.compile(loss='mse',
                 optimizer=keras.optimizers.Adam(lr=0.002))

callbacklr = tf.keras.callbacks.LearningRateScheduler(scheduler)

historyDenoising = modelo_A.fit(x=x_train, y=x_train,
                                epochs=epochs,
                                batch_size = batch_size,
                                callbacks=[callbacklr],
                                verbose=1)
```

```
38/38 [==============================] - 0s 917us/step - loss: 0.1209
Epoch 114/250
38/38 [==============================] - 0s 914us/step - loss: 0.1189
Epoch 115/250
38/38 [==============================] - 0s 882us/step - loss: 0.1221
Epoch 116/250
38/38 [==============================] - 0s 957us/step - loss: 0.1194
Epoch 117/250
38/38 [==============================] - 0s 1ms/step - loss: 0.1198
Epoch 118/250
38/38 [==============================] - 0s 987us/step - loss: 0.1200
Epoch 119/250
38/38 [==============================] - 0s 1ms/step - loss: 0.1172
Epoch 120/250
38/38 [==============================] - 0s 1ms/step - loss: 0.1197
Epoch 121/250
38/38 [==============================] - 0s 930us/step - loss: 0.1202
Epoch 122/250
38/38 [==============================] - 0s 948us/step - loss: 0.1210
Epoch 123/250
38/38 [==============================] - 0s 948us/step - loss: 0.1190
Epoch 124/250
38/38 [==============================] - 0s 907us/step - loss: 0.1221
Epoch 125/250
38/38 [==============================] - 0s 919us/step - loss: 0.1199
Epoch 126/250
38/38 [==============================] - 0s 949us/step - loss: 0.1215
Epoch 127/250
38/38 [==============================] - 0s 942us/step - loss: 0.1195
Epoch 128/250
38/38 [==============================] - 0s 1ms/step - loss: 0.1188
Epoch 129/250
38/38 [==============================] - 0s 958us/step - loss: 0.1208
Epoch 130/250
38/38 [==============================] - 0s 1ms/step - loss: 0.1207
Epoch 131/250
Epoch 131/250
38/38 [==============================] - 0s 1ms/step - loss: 0.1194
Epoch 132/250
38/38 [==============================] - 0s 1ms/step - loss: 0.1213
Epoch 133/250
38/38 [==============================] - 0s 916us/step - loss: 0.1177
Epoch 134/250
38/38 [==============================] - 0s 932us/step - loss: 0.1188
Epoch 135/250
38/38 [==============================] - 0s 961us/step - loss: 0.1232
Epoch 136/250
38/38 [==============================] - 0s 960us/step - loss: 0.1235
Epoch 137/250
38/38 [==============================] - 0s 999us/step - loss: 0.1197
Epoch 138/250
38/38 [==============================] - 0s 1ms/step - loss: 0.1194
Epoch 139/250
38/38 [==============================] - 0s 1ms/step - loss: 0.1196

Epoch 140/250
38/38 [==============================] - 0s 1ms/step - loss: 0.1178
Epoch 141/250
38/38 [==============================] - 0s 1ms/step - loss: 0.1201
Epoch 142/250
38/38 [==============================] - 0s 1ms/step - loss: 0.1194
Epoch 143/250
```

```
#Escreva aqui a sua resposta para o Item 3
predicao = modelo_A.predict(x_test)
error = np.power( np.abs(x_test - predicao),2)
error = np.sum(error, axis=1)


# Obtendo o valor de funcPK desejado

k_maiores = 1 # obtem os k valores com maior erro
for i in np.argpartition(error,-k_maiores)[-k_maiores:]:
    print(respostaPandas17.iloc[i,0])
```

```
147
```

```
#Escreva aqui a sua resposta para o Item 4

consultaSQL = """
        SELECT funcNome as NOMEFUNCIONARIO, dataMes as MES, round(salario,2) as SALARIO
        FROM funcionario
        JOIN pagamento ON funcionario.funcPK = pagamento.funcPK
        JOIN data ON data.dataPK = pagamento.DataPK
        WHERE data.dataAno = 2017 and funcionario.funcPK = 147
        SORT BY SALARIO DESC, MES ASC
"""
spark.sql(consultaSQL).show(20,truncate=False)
```

```
+---------------+---+--------+
|NOMEFUNCIONARIO|MES|SALARIO |
+---------------+---+--------+
|ABIMAELE BORGES|7  |250000.0|
|ABIMAELE BORGES|1  |1559.94 |
|ABIMAELE BORGES|2  |1559.94 |
|ABIMAELE BORGES|3  |1559.94 |
|ABIMAELE BORGES|4  |1559.94 |
|ABIMAELE BORGES|5  |1559.94 |
|ABIMAELE BORGES|6  |1559.94 |
|ABIMAELE BORGES|8  |1559.94 |
|ABIMAELE BORGES|9  |1559.94 |
|ABIMAELE BORGES|10 |1559.94 |
|ABIMAELE BORGES|11 |1559.94 |
|ABIMAELE BORGES|12 |1559.94 |
+---------------+---+--------+
```