Database Project (Fall 2023)

HW3 (100pts, Due date: Nov 3)

Student ID: 2020315798

Student Name: Choi Jin Woo

Instruction: In this homework, we provide you with a jupyter notebook file (DBP_HW3.ipynb). You should follow the instructions in these documents carefully.

Submit two files as follows:

- DBP_HW3_StudentID.zip
  - DBP_HW3_StudentID.ipynb
  - DBP_HW3_StudentID.pdf

[Spark SQL]

1. [20pts] The following data is related to the manufacturing process of computer manufacturers.

Data:

- Already in the manufacturing process → used for problem (a):

[Row(phase='packing', model='book_pro', serial='book_pro1'),
Row(phase='packing', model='book_pro', serial='book_pro2'),
Row(phase='packing', model='plus', serial='plus1'),
Row(phase='packing', model='book_pro', serial='book_pro3'),
Row(phase='packing', model='plus', serial='plus2'),
Row(phase='inspection', model='book_pro', serial='book_pro4'),
Row(phase='inspection', model='plus', serial='plus3'),
Row(phase='inspection', model='book_pro', serial='book_pro5'),
Row(phase='inspection', model='book_pro', serial='book_pro6')]

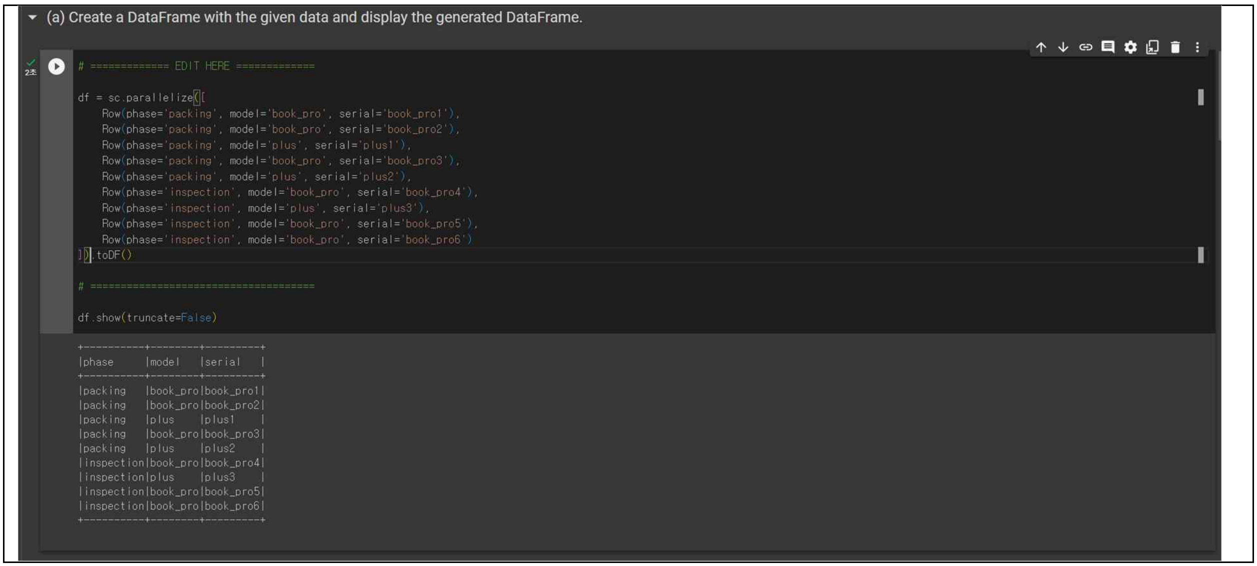
- The manufacturing process to add → used for problem (b):

[Row(phase='assembly', model='book_pro', serial='book_pro7'),
Row(phase='assembly', model='plus', serial='plus4')]

(a) Create a DataFrame with the given data and display the generated DataFrame.

[Answer]

Enter your code and result here. You must show your result (captured image).



(a) Create a DataFrame with the given data and display the generated DataFrame.

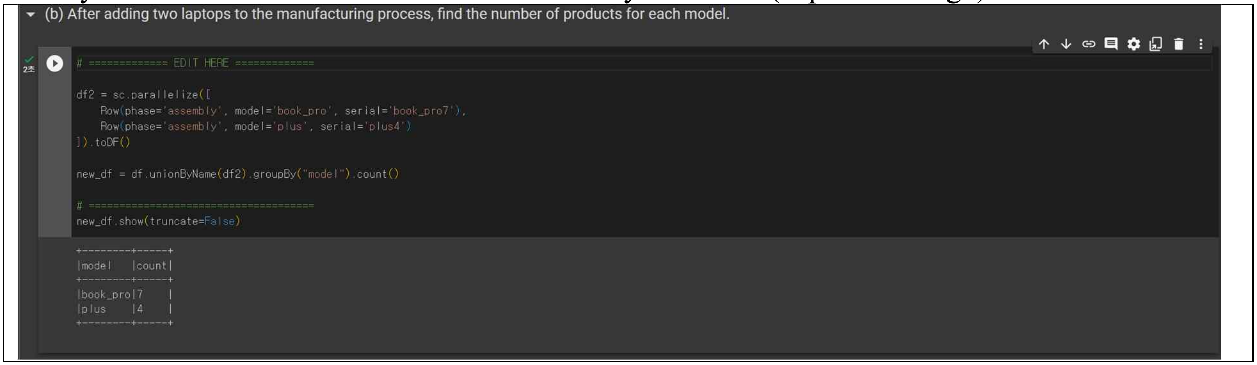
```
# ============= EDIT HERE =============

df = sc.parallelize([
    Row(phase='packing', model='book_pro', serial='book_pro1'),
    Row(phase='packing', model='book_pro', serial='book_pro2'),
    Row(phase='packing', model='plus', serial='plus1'),
    Row(phase='packing', model='book_pro', serial='book_pro3'),
    Row(phase='packing', model='plus', serial='plus2'),
    Row(phase='inspection', model='book_pro', serial='book_pro4'),
    Row(phase='inspection', model='plus', serial='plus3'),
    Row(phase='inspection', model='book_pro', serial='book_pro5'),
    Row(phase='inspection', model='book_pro', serial='book_pro6')
]).toDF()

# ====================================

df.show(truncate=False)
```

```
+----------+--------+---------+
|phase     |model   |serial   |
+----------+--------+---------+
|packing   |book_pro|book_pro1|
|packing   |book_pro|book_pro2|
|packing   |plus    |plus1    |
|packing   |book_pro|book_pro3|
|packing   |plus    |plus2    |
|inspection|book_pro|book_pro4|
|inspection|plus    |plus3    |
|inspection|book_pro|book_pro5|
|inspection|book_pro|book_pro6|
+----------+--------+---------+
```

(b) After adding ***two laptops to the manufacturing process***, find the number of products for each model.

[Answer]

Enter your code and result here. You must show your result (captured image).



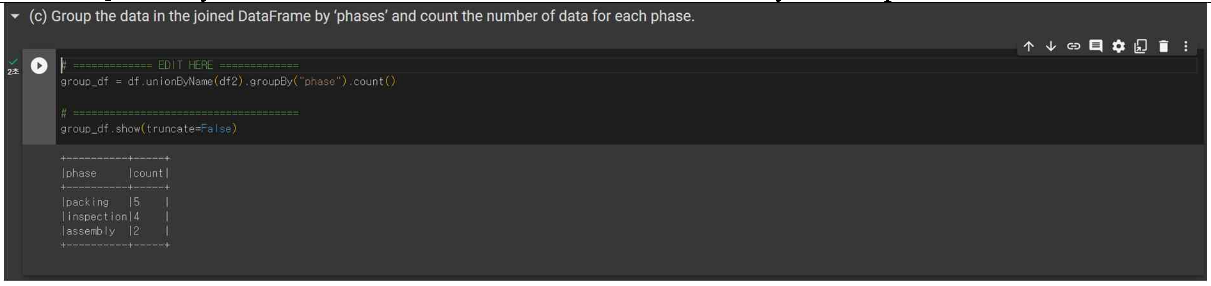
(b) After adding two laptops to the manufacturing process, find the number of products for each model.

```
# ============= EDIT HERE =============

df2 = sc.parallelize([
    Row(phase='assembly', model='book_pro', serial='book_pro7'),
    Row(phase='assembly', model='plus', serial='plus4')
]).toDF()

new_df = df.unionByName(df2).groupBy("model").count()

# ====================================
new_df.show(truncate=False)
```

```
+--------+-----+
|model   |count|
+--------+-----+
|book_pro|7    |
|plus    |4    |
+--------+-----+
```

(c) Group the data in the joined DataFrame by 'phases' and count the number of data for each phase.

[Answer] Enter your code and result here. You have to show your snapshot result.



(c) Group the data in the joined DataFrame by 'phases' and count the number of data for each phase.

```
# ============= EDIT HERE =============
group_df = df.unionByName(df2).groupBy("phase").count()

# ====================================
group_df.show(truncate=False)
```

```
+----------+-----+
|phase     |count|
+----------+-----+
|packing   |5    |
|inspection|4    |
|assembly  |2    |
+----------+-----+
```

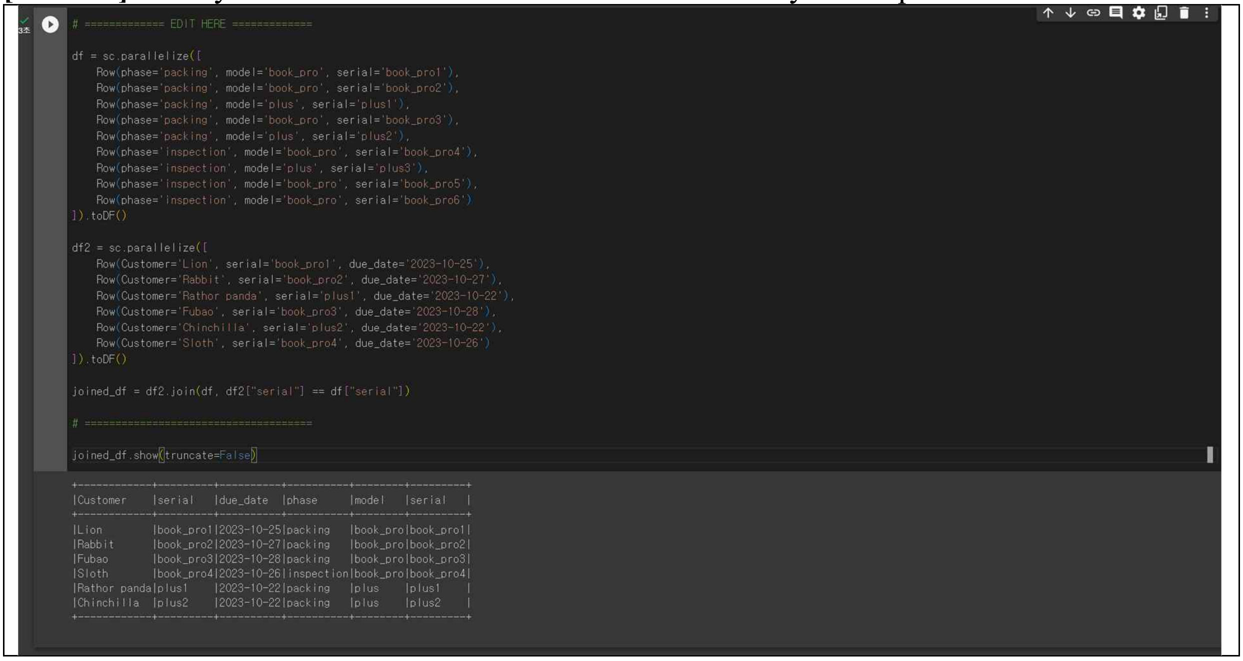
2. [20pts] The following data are *manufacturing process* and *customer information* for computer manufacturers.

- Data 1(manufacturing process):
[Row(phase='packing', model='book_pro', serial='book_pro1'),
Row(phase='packing', model='book_pro', serial='book_pro2'),
Row(phase='packing', model='plus', serial='plus1'),
Row(phase='packing', model='book_pro', serial='book_pro3'),
Row(phase='packing', model='plus', serial='plus2'),
Row(phase='inspection', model='book_pro', serial='book_pro4'),
Row(phase='inspection', model='plus', serial='plus3'),
Row(phase='inspection', model='book_pro', serial='book_pro5'),
Row(phase='inspection', model='book_pro', serial='book_pro6')]

- Data 2(customer information):
[Row(Customer='Lion', serial='book_pro1', due_date='2023-10-25'),
Row(Customer='Rabbit', serial='book_pro2', due_date='2023-10-27'),
Row(Customer='Rathor panda', serial='plus1', due_date='2023-10-22'),
Row(Customer='Fubao', serial='book_pro3', due_date='2023-10-28'),
Row(Customer='Chinchilla', serial='plus2', due_date='2023-10-22'),
Row(Customer='Sloth', serial='book_pro4', due_date='2023-10-26')]

(a) Create a DataFrame for the two given data and join Data 1 with Data 2 using an inner join based on the 'serial' column. (left side: Data 2, right side: Data 1)

[Answer] Enter your code and result here. You have to show your snapshot result.

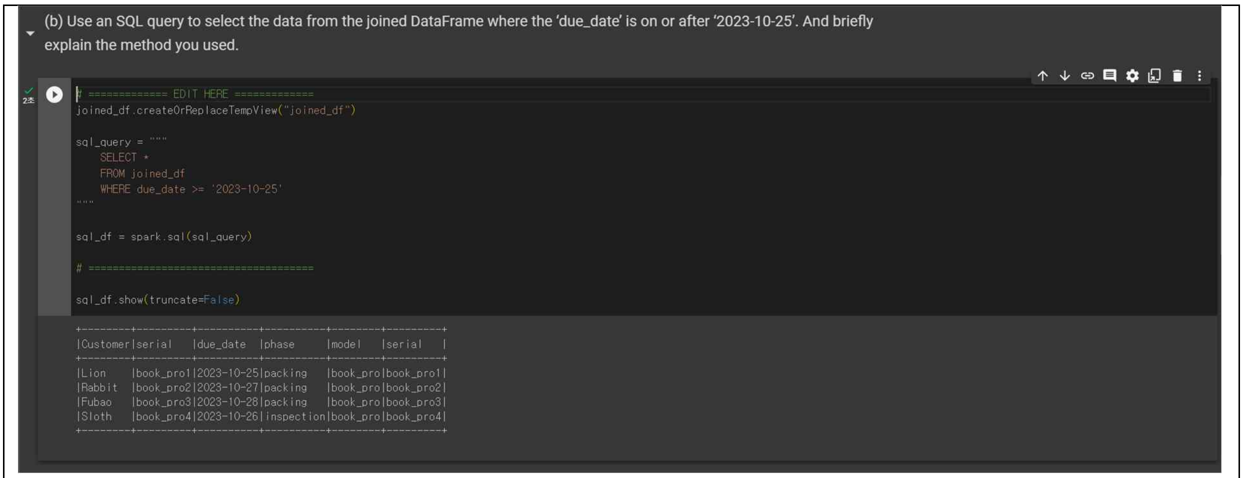


```
# ============= EDIT HERE =============

df = sc.parallelize([
    Row(phase='packing', model='book_pro', serial='book_pro1'),
    Row(phase='packing', model='book_pro', serial='book_pro2'),
    Row(phase='packing', model='plus', serial='plus1'),
    Row(phase='packing', model='book_pro', serial='book_pro3'),
    Row(phase='packing', model='plus', serial='plus2'),
    Row(phase='inspection', model='book_pro', serial='book_pro4'),
    Row(phase='inspection', model='plus', serial='plus3'),
    Row(phase='inspection', model='book_pro', serial='book_pro5'),
    Row(phase='inspection', model='book_pro', serial='book_pro6')
]).toDF()

df2 = sc.parallelize([
    Row(Customer='Lion', serial='book_pro1', due_date='2023-10-25'),
    Row(Customer='Rabbit', serial='book_pro2', due_date='2023-10-27'),
    Row(Customer='Rathor panda', serial='plus1', due_date='2023-10-22'),
    Row(Customer='Fubao', serial='book_pro3', due_date='2023-10-28'),
    Row(Customer='Chinchilla', serial='plus2', due_date='2023-10-22'),
    Row(Customer='Sloth', serial='book_pro4', due_date='2023-10-26')
]).toDF()

joined_df = df2.join(df, df2["serial"] == df["serial"])

# ====================================

joined_df.show(truncate=False)
```

```
+------------+---------+----------+----------+--------+---------+
|Customer    |serial   |due_date  |phase     |model   |serial   |
+------------+---------+----------+----------+--------+---------+
|Lion        |book_pro1|2023-10-25|packing   |book_pro|book_pro1|
|Rabbit      |book_pro2|2023-10-27|packing   |book_pro|book_pro2|
|Fubao       |book_pro3|2023-10-28|packing   |book_pro|book_pro3|
|Sloth       |book_pro4|2023-10-26|inspection|book_pro|book_pro4|
|Rathor panda|plus1    |2023-10-22|packing   |plus    |plus1    |
|Chinchilla  |plus2    |2023-10-22|packing   |plus    |plus2    |
+------------+---------+----------+----------+--------+---------+
```

(b) Use an SQL query to select the data from the joined DataFrame where the **'due_date'** is on or after **'2023-10-25'**. And briefly explain the method you used.

[Answer] Enter your code and result here. You have to show your snapshot result.



(b) Use an SQL query to select the data from the joined DataFrame where the ‘due_date’ is on or after ‘2023-10-25’. And briefly explain the method you used.

```
# ============= EDIT HERE =============
joined_df.createOrReplaceTempView("joined_df")

sql_query = """
    SELECT *
    FROM joined_df
    WHERE due_date >= '2023-10-25'
"""

sql_df = spark.sql(sql_query)

# =====================================

sql_df.show(truncate=False)
```

```
+--------+---------+----------+----------+--------+---------+
|Customer|serial   |due_date  |phase     |model   |serial   |
+--------+---------+----------+----------+--------+---------+
|Lion    |book_pro1|2023-10-25|packing   |book_pro|book_pro1|
|Rabbit  |book_pro2|2023-10-27|packing   |book_pro|book_pro2|
|Fubao   |book_pro3|2023-10-28|packing   |book_pro|book_pro3|
|Sloth   |book_pro4|2023-10-26|inspection|book_pro|book_pro4|
+--------+---------+----------+----------+--------+---------+
```

[createOrReplaceTempView]
To save Dataframe into temporary SQL table named “joined_df”

[sql_query]
SQL query that filters where “due_date” is on or after “2023-10-25”

[spark.sql()]
Returns new Dataframe with query result

[Spark ML]
3. [60pts] We provide you with a ***Fashion-MNIST*** dataset.

Dataset Description:

Training set: 60,000 examples
Test set: 10,000 examples
Each example is a 28x28 grayscale image associated with a label from 10 classes.

Label	Description
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

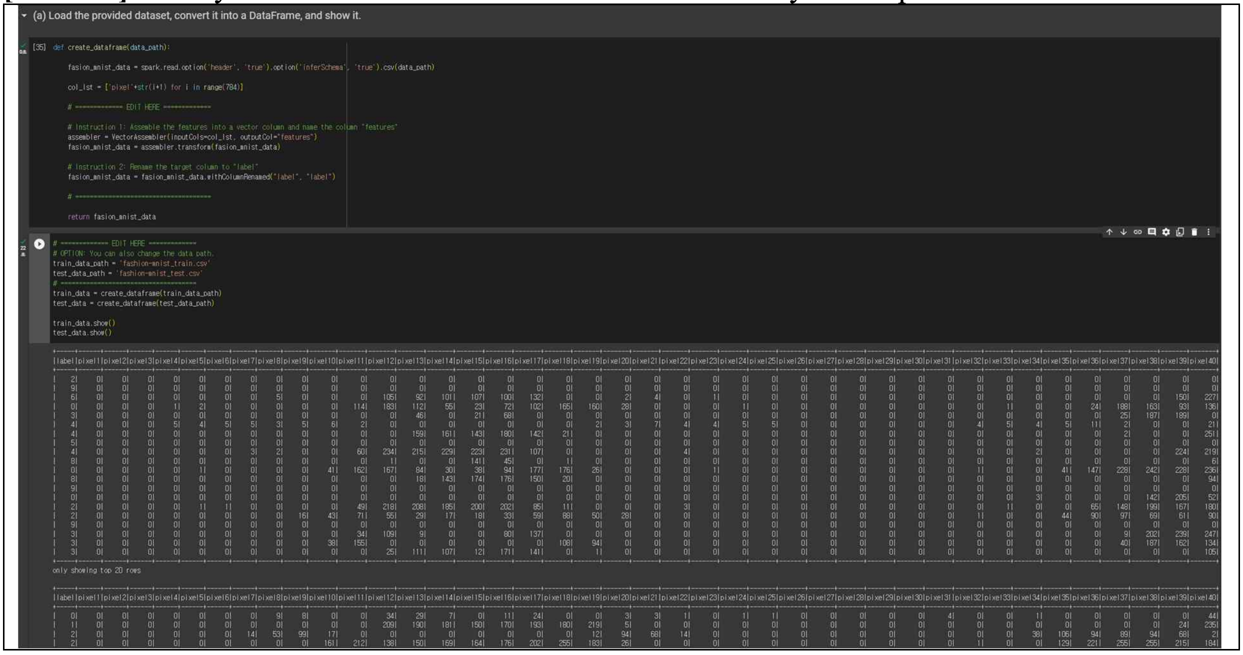
For more information, visit this website: https://github.com/zalandoresearch/fashion-mnist

(a) Load the provided dataset, convert it into a DataFrame, and show it. You should follow the following instructions.

-Instructions 1: *Assemble the features into a vector column and name the column “features.”*

-Instructions 2: *Rename the target column to “label.”*

[Answer] Enter your code and result here. You have to show your snapshot result.



(a) Load the provided dataset, convert it into a DataFrame, and show it.

```
[35] def create_dataframe(data_path):

         fasion_mnist_data = spark.read.option('header', 'true').option('inferSchema', 'true').csv(data_path)

         col_lst = ['pixel'+str(i+1) for i in range(784)]

         # ============= EDIT HERE =============

         # Instruction 1: Assemble the features into a vector column and name the column "features"
         assembler = VectorAssembler(inputCols=col_lst, outputCol="features")
         fasion_mnist_data = assembler.transform(fasion_mnist_data)

         # Instruction 2: Rename the target column to "label"
         fasion_mnist_data = fasion_mnist_data.withColumnRenamed("label", "label")

         # ====================================

         return fasion_mnist_data
```

```
# ============= EDIT HERE =============
# OPTION: You can also change the data path.
train_data_path = 'fashion-mnist_train.csv'
test_data_path = 'fashion-mnist_test.csv'
# ====================================
train_data = create_dataframe(train_data_path)
test_data = create_dataframe(test_data_path)

train_data.show()
test_data.show()
```

```
+-----+------+------+------+------+------+------+------+------+------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+
|label|pixel1|pixel2|pixel3|pixel4|pixel5|pixel6|pixel7|pixel8|pixel9|pixel10|pixel11|pixel12|pixel13|pixel14|pixel15|pixel16|pixel17|pixel18|pixel19|pixel20|pixel21|pixel22|pixel23|pixel24|pixel25|pixel26|pixel27|pixel28|pixel29|pixel30|pixel31|pixel32|pixel33|pixel34|pixel35|pixel36|pixel37|pixel38|pixel39|pixel40|
+-----+------+------+------+------+------+------+------+------+------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+
|    2|     0|     0|     0|     0|     0|     0|     0|     0|     0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|
|    9|     0|     0|     0|     0|     0|     0|     0|     0|     0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|
|    6|     0|     0|     0|     0|     0|     0|     0|     5|     0|      0|      0|    105|     92|    101|    107|    100|    132|      0|      0|      2|      4|      0|      1|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|    150|    227|
|    0|     0|     0|     0|     1|     2|     0|     0|     0|     0|      0|    114|    183|    112|     55|     23|     72|    102|    165|    160|     28|      0|      0|      0|      1|      0|      0|      0|      0|      0|      0|      0|      0|      1|      0|      0|     24|    188|    163|     93|    136|
|    3|     0|     0|     0|     0|     0|     0|     0|     0|     0|      0|      0|      0|     46|      0|     21|     68|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|     25|    187|    189|      0|
|    4|     0|     0|     0|     5|     4|     5|     5|     3|     5|      6|      2|      0|      0|      0|      0|      0|      0|      0|      2|      3|      7|      4|      4|      5|      5|      0|      0|      0|      0|      0|      0|      4|      5|      4|      5|     11|      2|      0|      0|     21|
|    4|     0|     0|     0|     0|     0|     0|     0|     0|     0|      0|      0|      0|    159|    161|    143|    180|    142|     21|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      2|      0|      0|    251|
|    5|     0|     0|     0|     0|     0|     0|     0|     0|     0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|
|    4|     0|     0|     0|     0|     0|     0|     3|     2|     0|      0|     60|    234|    215|    229|    223|    231|    107|      0|      0|      0|      0|      4|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      2|      0|      0|      0|      0|    224|    219|
|    8|     0|     0|     0|     0|     0|     0|     0|     0|     0|      0|      0|      1|      0|      0|    141|     45|      0|      1|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      6|
|    0|     0|     0|     0|     0|     1|     0|     0|     0|     0|     41|    162|    167|     84|     30|     38|     94|    177|    176|     26|      0|      0|      0|      1|      0|      0|      0|      0|      0|      0|      0|      0|      1|      0|      0|     41|    147|    228|    242|    228|    236|
|    8|     0|     0|     0|     0|     0|     0|     0|     0|     0|      0|      0|      0|     18|    143|    174|    176|    150|     20|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|     94|
|    9|     0|     0|     0|     0|     0|     0|     0|     0|     0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|
|    0|     0|     0|     0|     0|     0|     0|     0|     0|     0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      3|      0|      0|      0|    142|    205|     52|
|    2|     0|     0|     0|     0|     1|     1|     0|     0|     0|      0|     49|    218|    208|    185|    200|    202|     85|     11|      0|      0|      0|      3|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      1|      0|      0|     65|    148|    199|    167|    180|
|    2|     0|     0|     0|     0|     0|     0|     0|     0|    16|     43|     71|     55|     29|     17|     18|     33|     59|     88|     50|     28|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      1|      0|      0|     44|     90|     97|     69|     61|     90|
|    9|     0|     0|     0|     0|     0|     0|     0|     0|     0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|
|    3|     0|     0|     0|     0|     0|     0|     0|     0|     0|      0|     34|    109|      9|      0|      0|     80|    137|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      9|    202|    239|    247|
|    3|     0|     0|     0|     0|     0|     0|     0|     0|     0|     38|    155|      0|      0|      0|      0|      0|      0|    108|     94|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|     40|    187|    162|    134|
|    3|     0|     0|     0|     0|     0|     0|     0|     0|     0|      0|      0|     25|    111|    107|     12|    171|    141|      0|      1|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|    105|
+-----+------+------+------+------+------+------+------+------+------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+
only showing top 20 rows

+-----+------+------+------+------+------+------+------+------+------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+
|label|pixel1|pixel2|pixel3|pixel4|pixel5|pixel6|pixel7|pixel8|pixel9|pixel10|pixel11|pixel12|pixel13|pixel14|pixel15|pixel16|pixel17|pixel18|pixel19|pixel20|pixel21|pixel22|pixel23|pixel24|pixel25|pixel26|pixel27|pixel28|pixel29|pixel30|pixel31|pixel32|pixel33|pixel34|pixel35|pixel36|pixel37|pixel38|pixel39|pixel40|
+-----+------+------+------+------+------+------+------+------+------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+
|    0|     0|     0|     0|     0|     0|     0|     0|     9|     8|      0|      0|     34|     29|      7|      0|     11|     24|      0|      0|      3|      3|      1|      0|      1|      1|      0|      0|      0|      0|      0|      4|      0|      0|      1|      0|      0|      0|      0|      0|     44|
|    1|     0|     0|     0|     0|     0|     0|     0|     0|     0|      0|      0|    209|    190|    181|    150|    170|    193|    180|    219|      5|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|     24|    235|
|    2|     0|     0|     0|     0|     0|     0|    14|    53|    99|     17|      0|      0|      0|      0|      0|      0|      0|      0|     12|     94|     68|     14|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|     38|    106|     94|     89|     94|     68|      2|
|    2|     0|     0|     0|     0|     0|     0|     0|     0|     0|    161|    212|    138|    150|    169|    164|    176|    202|    255|    183|     26|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      0|      1|      0|      0|    129|    221|    255|    255|    215|    184|
```

(b) Train models to classify the classes of the Fashion MNIST dataset and **report the results for test data**. The models used are **Logistic Regression, Decision Tree, and Random Forest.**

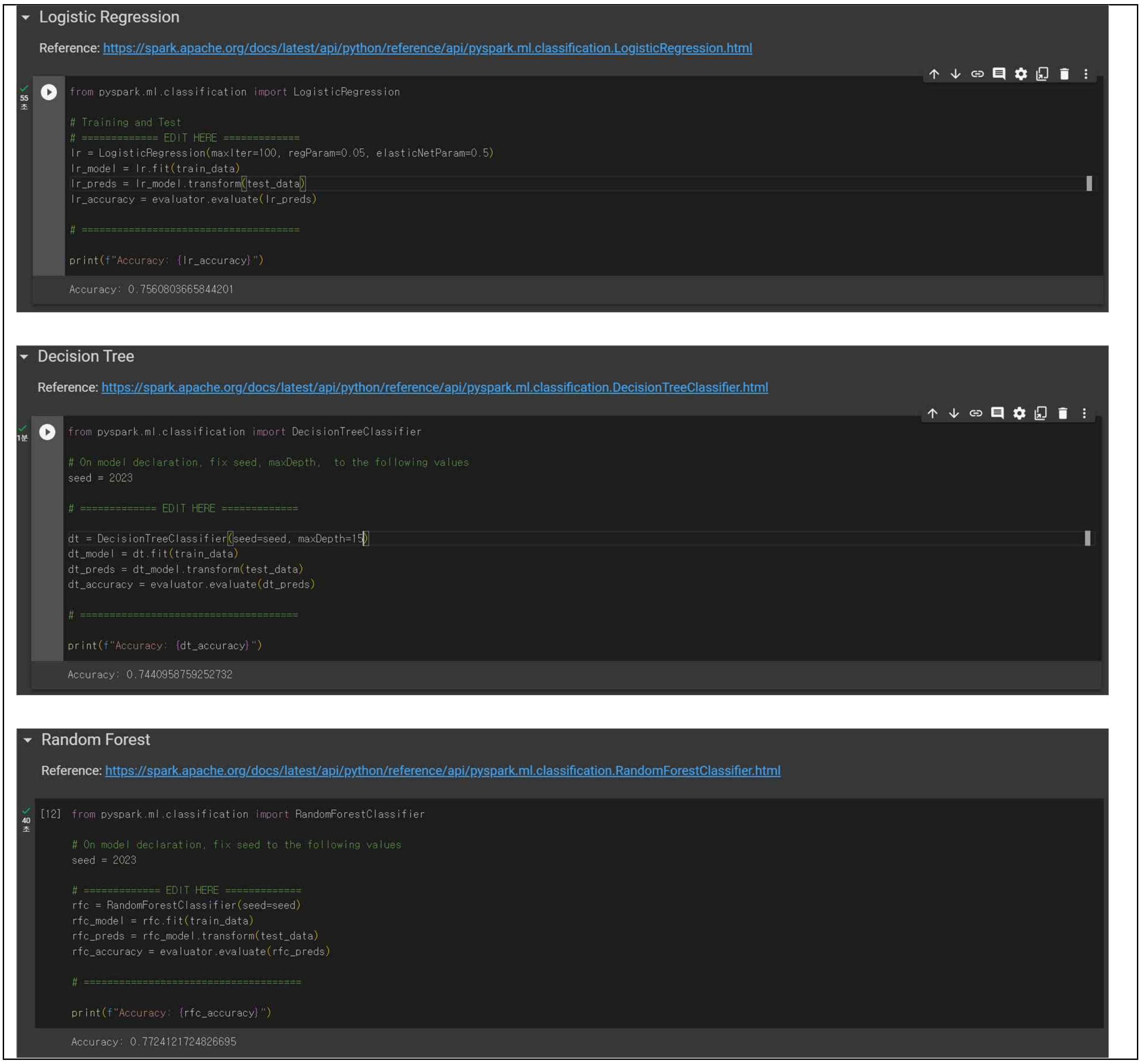
For detailed explanations of the usage of each model, please refer to the official documentation below.

- Logistic Regression: [Link]
- Decision Tree: [Link]
- Random Forest: [Link]

[Answer] Fill in the table below.

	Logistic Regression	**Decision Tree**	**Random Forest**
Test accuracy	0.7561	0.7441	0.7724

[Answer] Enter your code and result here. You have to show your snapshot result.



Logistic Regression

Reference: https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.ml.classification.LogisticRegression.html

```
from pyspark.ml.classification import LogisticRegression

# Training and Test
# ============ EDIT HERE ============
lr = LogisticRegression(maxIter=100, regParam=0.05, elasticNetParam=0.5)
lr_model = lr.fit(train_data)
lr_preds = lr_model.transform(test_data)
lr_accuracy = evaluator.evaluate(lr_preds)

# ===================================

print(f"Accuracy: {lr_accuracy}")
```

```
Accuracy: 0.7560803665844201
```

Decision Tree

Reference: https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.ml.classification.DecisionTreeClassifier.html

```
from pyspark.ml.classification import DecisionTreeClassifier

# On model declaration, fix seed, maxDepth,  to the following values
seed = 2023

# ============ EDIT HERE ============

dt = DecisionTreeClassifier(seed=seed, maxDepth=15)
dt_model = dt.fit(train_data)
dt_preds = dt_model.transform(test_data)
dt_accuracy = evaluator.evaluate(dt_preds)

# ===================================

print(f"Accuracy: {dt_accuracy}")
```

```
Accuracy: 0.7440958759252732
```

Random Forest

Reference: https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.ml.classification.RandomForestClassifier.html

```
from pyspark.ml.classification import RandomForestClassifier

# On model declaration, fix seed to the following values
seed = 2023

# ============ EDIT HERE ============
rfc = RandomForestClassifier(seed=seed)
rfc_model = rfc.fit(train_data)
rfc_preds = rfc_model.transform(test_data)
rfc_accuracy = evaluator.evaluate(rfc_preds)

# ===================================

print(f"Accuracy: {rfc_accuracy}")
```

```
Accuracy: 0.7724121724826695
```