Part A: [Marks 60]

1. [Marks: 10] What is an Intrusion Detection System? Is it possible to implement an Intrusion Detection System on this dataset? Explain the workflow described in the paper for implementing the Intrusion Detection System.

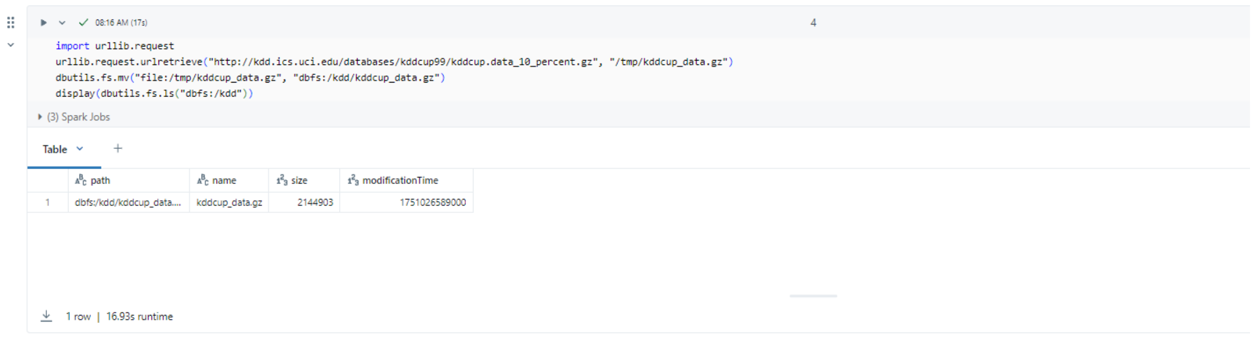
- Intrusion Detection System is a system that detect the existence of intrusion (Attack) like malicious behavior or policy violations within a system/framework with the analysis of big data

- Yes, implementation of an Intrusion Detection System in this dataset in feasible, as described in the paper that "The KDDCUP99 are tested in this study"

- Workflow described in the paper of Intrusion Detection System Implementation:

1. Data loading: use Apache Spark and load data into RDD
2. Data preprocessing: prepare data and convert categorical data in the dataset to numerical data.
3. Standardization: Apply the standardization (the standardizes feature if applicable) to numerical data.
4. Feature selection: Remove all redundant and irrelevant features in the data
5. Model classifier: Apply supervised learning method Support vector machine (SVM) into the system together Stochastic Gradient Descent (SGD) on the selected features for training.
6. Evaluate the result: Use AUROC and AUPR as metric to measure and evaluate training results

2. [Marks: 4] Use the python urllib library to extract the KDD Cup 99 data from their web repository, store it in a temporary location and then move it to the Databricks filesystem which can enable easy access to this data for analysis. {Hint: You can use the following commands in Databricks to get your data.}



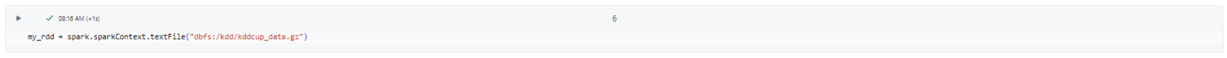
```
import urllib.request
urllib.request.urlretrieve("http://kdd.ics.uci.edu/databases/kddcup99/kddcup.data_10_percent.gz", "/tmp/kddcup_data.gz")
dbutils.fs.mv("file:/tmp/kddcup_data.gz", "dbfs:/kdd/kddcup_data.gz")
display(dbutils.fs.ls("dbfs:/kdd"))
```

	path	name	size	modificationTime
1	dbfs:/kdd/kddcup_data....	kddcup_data.gz	2144903	1751026589000

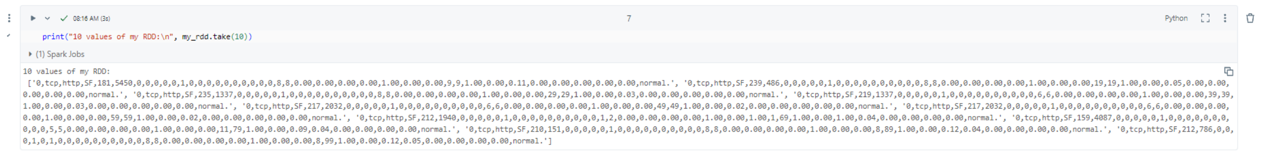
1 row | 16.93s runtime

3. [Marks: 4] After storing the data in the Databricks filesystem. Load your data from the disk into Sparks RDD. Print 10 values of your RDD and verify the type of data structure of your data (RDD).

10 values of RDD



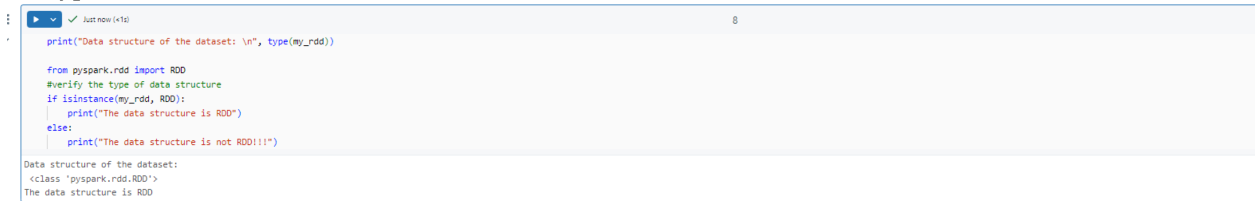
```
my_rdd = spark.sparkContext.textFile("dbfs:/kdd/kddcup_data.gz")
```



```
print("10 values of my RDD:\n", my_rdd.take(10))
```

```
10 values of my RDD:
 ['0,tcp,http,SF,181,5450,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,8,8,0.00,0.00,0.00,0.00,1.00,0.00,0.00,9,9,1.00,0.00,0.11,0.00,0.00,0.00,0.00,0.00,normal.', '0,tcp,http,SF,239,486,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,8,8,0.00,0.00,0.00,0.00,1.00,0.00,0.00,19,19,1.00,0.00,0.05,0.00,0.00,
0.00,0.00,0.00,normal.', '0,tcp,http,SF,235,1337,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,8,8,0.00,0.00,0.00,0.00,1.00,0.00,0.00,29,29,1.00,0.00,0.03,0.00,0.00,0.00,0.00,0.00,normal.', '0,tcp,http,SF,219,1337,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,6,6,0.00,0.00,0.00,0.00,1.00,0.00,0.00,39,39,
1.00,0.00,0.03,0.00,0.00,0.00,0.00,0.00,normal.', '0,tcp,http,SF,217,2032,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,6,6,0.00,0.00,0.00,0.00,1.00,0.00,0.00,49,49,1.00,0.00,0.02,0.00,0.00,0.00,0.00,0.00,normal.', '0,tcp,http,SF,217,2032,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,6,6,0.00,0.00,0.00,
0.00,1.00,0.00,0.00,59,59,1.00,0.00,0.02,0.00,0.00,0.00,0.00,0.00,normal.', '0,tcp,http,SF,212,1940,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,2,0.00,0.00,0.00,0.00,1.00,0.00,1.00,1,69,1.00,0.00,1.00,0.04,0.00,0.00,0.00,0.00,normal.', '0,tcp,http,SF,159,4087,0,0,0,0,0,1,0,0,0,0,0,0,0,
0,0,0,5,5,0.00,0.00,0.00,0.00,1.00,0.00,0.00,11,79,1.00,0.00,0.09,0.04,0.00,0.00,0.00,0.00,normal.', '0,tcp,http,SF,210,151,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,8,8,0.00,0.00,0.00,0.00,1.00,0.00,0.00,8,89,1.00,0.00,0.12,0.04,0.00,0.00,0.00,0.00,normal.', '0,tcp,http,SF,212,786,0,0,
0,1,0,1,0,0,0,0,0,0,0,0,0,0,8,8,0.00,0.00,0.00,0.00,1.00,0.00,0.00,8,99,1.00,0.00,0.12,0.05,0.00,0.00,0.00,0.00,normal.']
```

the type of data structure of RDD:

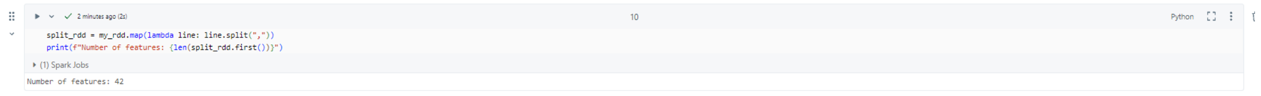


```
print("Data structure of the dataset: \n", type(my_rdd))

from pyspark.rdd import RDD
#verify the type of data structure
if isinstance(my_rdd, RDD):
    print("The data structure is RDD")
else:
    print("The data structure is not RDD!!!")
```

```
Data structure of the dataset:
 <class 'pyspark.rdd.RDD'>
The data structure is RDD
```

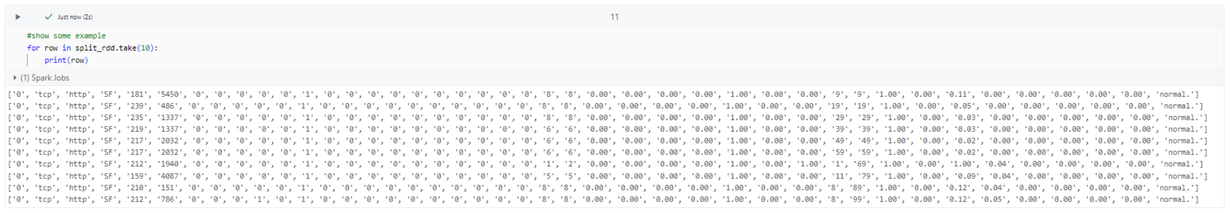
4. [Marks: 4] Split the data. (Each entry in your RDD is a comma-separated line of data, which you first need to split before you can parse and build your data frame.) Show the total number of features (columns) and print results.

Total number and results



```
split_rdd = my_rdd.map(lambda line: line.split(","))
print(f"Number of features: {len(split_rdd.first())}")
```

```
Number of features: 42
```

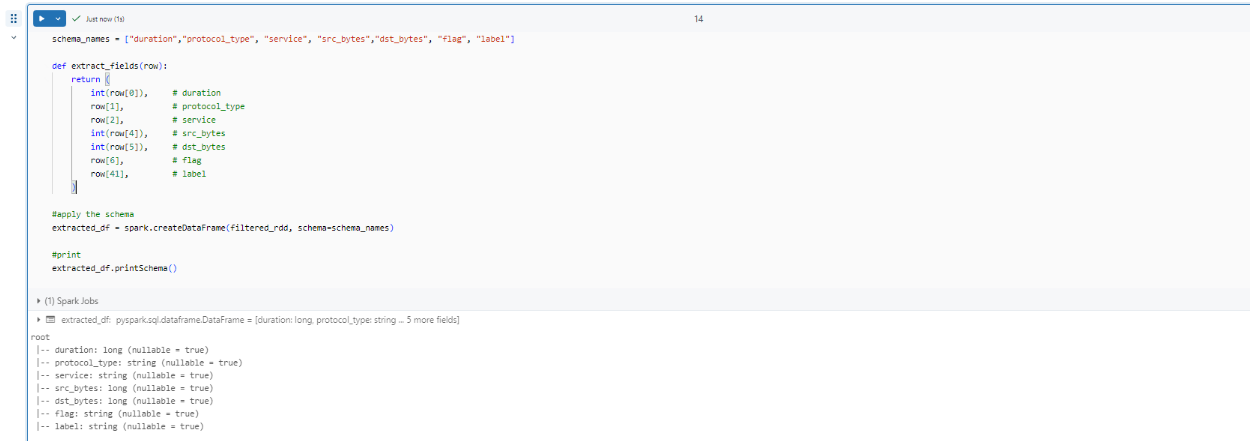


```
#show some example
for row in split_rdd.take(10):
    print(row)
```

```
['0', 'tcp', 'http', 'SF', '181', '5450', '0', '0', '0', '0', '0', '1', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '8', '8', '0.00', '0.00', '0.00', '0.00', '1.00', '0.00', '0.00', '9', '9', '1.00', '0.00', '0.11', '0.00', '0.00', '0.00', '0.00', '0.00', 'normal.']
['0', 'tcp', 'http', 'SF', '239', '486', '0', '0', '0', '0', '0', '1', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '8', '8', '0.00', '0.00', '0.00', '0.00', '1.00', '0.00', '0.00', '19', '19', '1.00', '0.00', '0.05', '0.00', '0.00', '0.00', '0.00', '0.00', 'normal.']
['0', 'tcp', 'http', 'SF', '235', '1337', '0', '0', '0', '0', '0', '1', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '8', '8', '0.00', '0.00', '0.00', '0.00', '1.00', '0.00', '0.00', '29', '29', '1.00', '0.00', '0.03', '0.00', '0.00', '0.00', '0.00', '0.00', 'normal.']
['0', 'tcp', 'http', 'SF', '219', '1337', '0', '0', '0', '0', '0', '1', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '6', '6', '0.00', '0.00', '0.00', '0.00', '1.00', '0.00', '0.00', '39', '39', '1.00', '0.00', '0.03', '0.00', '0.00', '0.00', '0.00', '0.00', 'normal.']
['0', 'tcp', 'http', 'SF', '217', '2032', '0', '0', '0', '0', '0', '1', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '6', '6', '0.00', '0.00', '0.00', '0.00', '1.00', '0.00', '0.00', '49', '49', '1.00', '0.00', '0.02', '0.00', '0.00', '0.00', '0.00', '0.00', 'normal.']
['0', 'tcp', 'http', 'SF', '217', '2032', '0', '0', '0', '0', '0', '1', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '6', '6', '0.00', '0.00', '0.00', '0.00', '1.00', '0.00', '0.00', '59', '59', '1.00', '0.00', '0.02', '0.00', '0.00', '0.00', '0.00', '0.00', 'normal.']
['0', 'tcp', 'http', 'SF', '212', '1940', '0', '0', '0', '0', '0', '1', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '1', '2', '0.00', '0.00', '0.00', '0.00', '1.00', '0.00', '1.00', '1', '69', '1.00', '0.00', '1.00', '0.04', '0.00', '0.00', '0.00', '0.00', 'normal.']
['0', 'tcp', 'http', 'SF', '159', '4087', '0', '0', '0', '0', '0', '1', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '5', '5', '0.00', '0.00', '0.00', '0.00', '1.00', '0.00', '0.00', '11', '79', '1.00', '0.00', '0.09', '0.04', '0.00', '0.00', '0.00', '0.00', 'normal.']
['0', 'tcp', 'http', 'SF', '210', '151', '0', '0', '0', '0', '0', '1', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '8', '8', '0.00', '0.00', '0.00', '0.00', '1.00', '0.00', '0.00', '8', '89', '1.00', '0.00', '0.12', '0.04', '0.00', '0.00', '0.00', '0.00', 'normal.']
['0', 'tcp', 'http', 'SF', '212', '786', '0', '0', '0', '1', '0', '1', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '8', '8', '0.00', '0.00', '0.00', '0.00', '1.00', '0.00', '0.00', '8', '99', '1.00', '0.00', '0.12', '0.05', '0.00', '0.00', '0.00', '0.00', 'normal.']
```

5. [Marks: 4] Now extract these 6 columns (*duration*, *protocol_type, service, src_bytes, dst_bytes, flag and label*) from your dataset. Build a new RDD and data frame. Print schema and display 10 values.

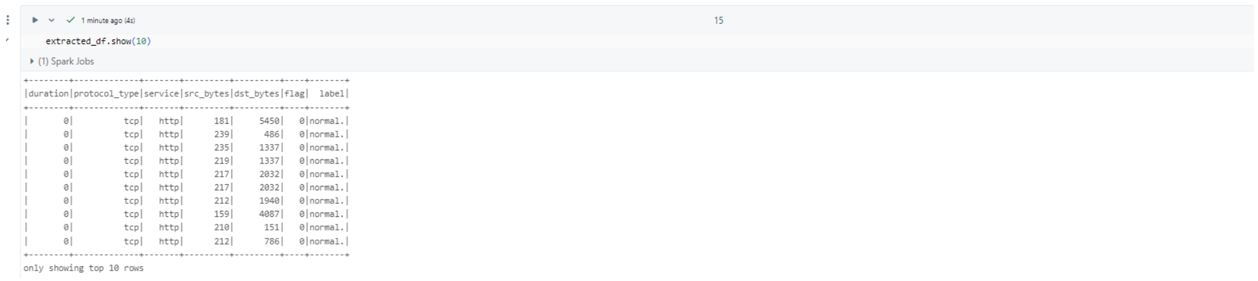
Schema:



```
schema_names = ["duration","protocol_type", "service", "src_bytes","dst_bytes", "flag", "label"]

def extract_fields(row):
    return (
        int(row[0]),     # duration
        row[1],          # protocol_type
        row[2],          # service
        int(row[4]),     # src_bytes
        int(row[5]),     # dst_bytes
        row[6],          # flag
        row[41],         # label
    )

#apply the schema
extracted_df = spark.createDataFrame(filtered_rdd, schema=schema_names)

#print
extracted_df.printSchema()
```

```
(1) Spark Jobs
extracted_df:  pyspark.sql.dataframe.DataFrame = [duration: long, protocol_type: string ... 5 more fields]
root
 |-- duration: long (nullable = true)
 |-- protocol_type: string (nullable = true)
 |-- service: string (nullable = true)
 |-- src_bytes: long (nullable = true)
 |-- dst_bytes: long (nullable = true)
 |-- flag: string (nullable = true)
 |-- label: string (nullable = true)
```

10 values:

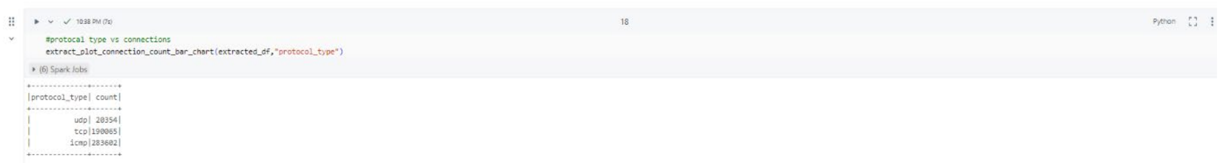


```
extracted_df.show(10)
```

```
(1) Spark Jobs
+--------+-------------+-------+---------+---------+----+-------+
|duration|protocol_type|service|src_bytes|dst_bytes|flag|  label|
+--------+-------------+-------+---------+---------+----+-------+
|       0|          tcp|   http|      181|     5450|   0|normal.|
|       0|          tcp|   http|      239|      486|   0|normal.|
|       0|          tcp|   http|      235|     1337|   0|normal.|
|       0|          tcp|   http|      219|     1337|   0|normal.|
|       0|          tcp|   http|      217|     2032|   0|normal.|
|       0|          tcp|   http|      217|     2032|   0|normal.|
|       0|          tcp|   http|      212|     1940|   0|normal.|
|       0|          tcp|   http|      159|     4087|   0|normal.|
|       0|          tcp|   http|      210|      151|   0|normal.|
|       0|          tcp|   http|      212|      786|   0|normal.|
+--------+-------------+-------+---------+---------+----+-------+
only showing top 10 rows
```

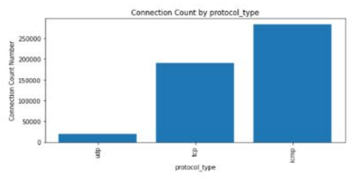
6. [Marks: 4] Get the total number of connections based on the *protocol_type* and based on the *service*. Show results in an ascending order. Plot the bar graph for both.

the total number of connections based on the *protocol_type*

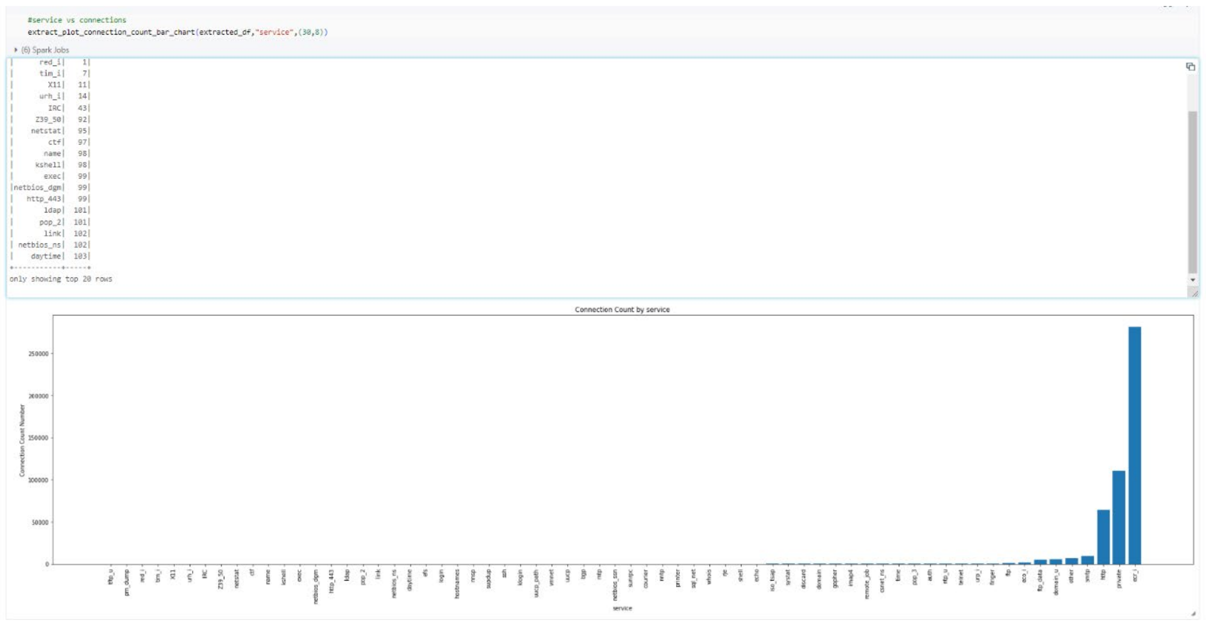


```
#protocal type vs connections
extract_plot_connection_count_bar_chart(extracted_df,"protocol_type")
```

```
(6) Spark Jobs
+-------------+------+
|protocol_type| count|
+-------------+------+
|          udp| 20354|
|          tcp|190065|
|         icmp|283602|
+-------------+------+
```



the total number of connections based on the *service*



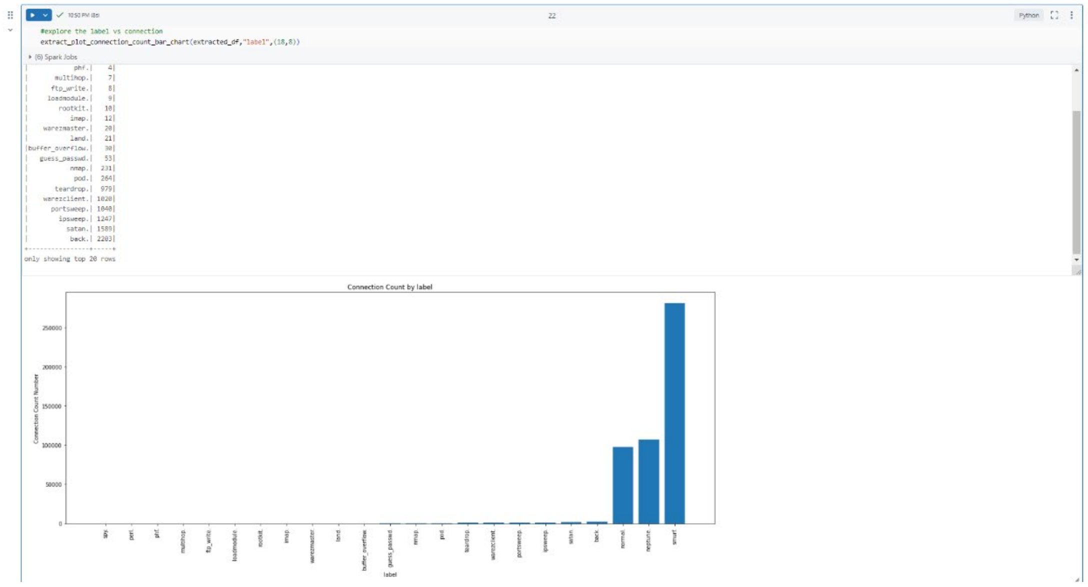
```
#service vs connections
extract_plot_connection_count_bar_chart(extracted_df,"service",(30,8))
```

▸ (6) Spark Jobs

```
|      red_i|     1|
|      tim_i|     7|
|        X11|    11|
|      urh_i|    14|
|        IRC|    43|
|     Z39_50|    92|
|    netstat|    95|
|        ctf|    97|
|       name|    98|
|     kshell|    98|
|       exec|    99|
|netbios_dgm|    99|
|   http_443|    99|
|       ldap|   101|
|      pop_2|   101|
|       link|   102|
| netbios_ns|   102|
|    daytime|   103|
+-----------+------+
only showing top 20 rows
```

7. [Marks: 10] Do a further exploratory data analysis, including other columns of this dataset and plot graphs. Plot at least 3 different charts and explain them.

the total number of connections based on the *flag*

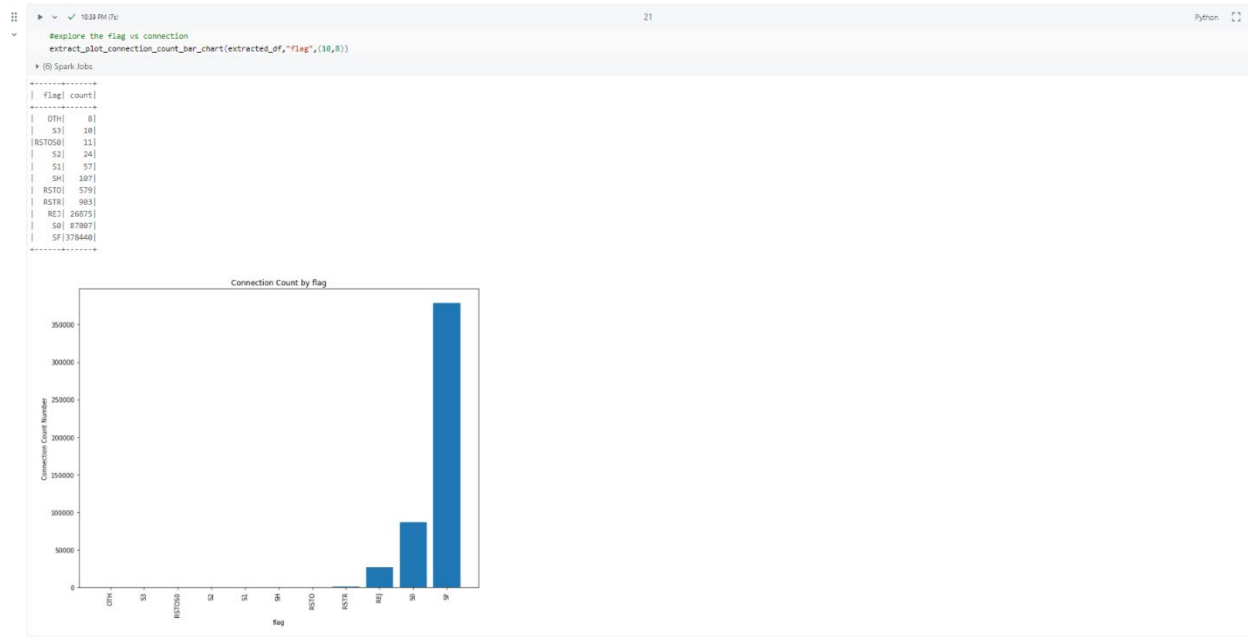


```
#explore the label vs connection
extract_plot_connection_count_bar_chart(extracted_df,"label",(18,8))
```

▸ (6) Spark Jobs

```
|            phf.|     4|
|       multihop.|     7|
|      ftp_write.|     8|
|     loadmodule.|     9|
|        rootkit.|    10|
|           imap.|    12|
|    warezmaster.|    20|
|           land.|    21|
|buffer_overflow.|    30|
|   guess_passwd.|    53|
|           nmap.|   231|
|            pod.|   264|
|       teardrop.|   979|
|    warezclient.|  1020|
|      portsweep.|  1040|
|        ipsweep.|  1247|
|          satan.|  1589|
|           back.|  2203|
+----------------+------+
only showing top 20 rows
```

the total number of connections based on the *label*



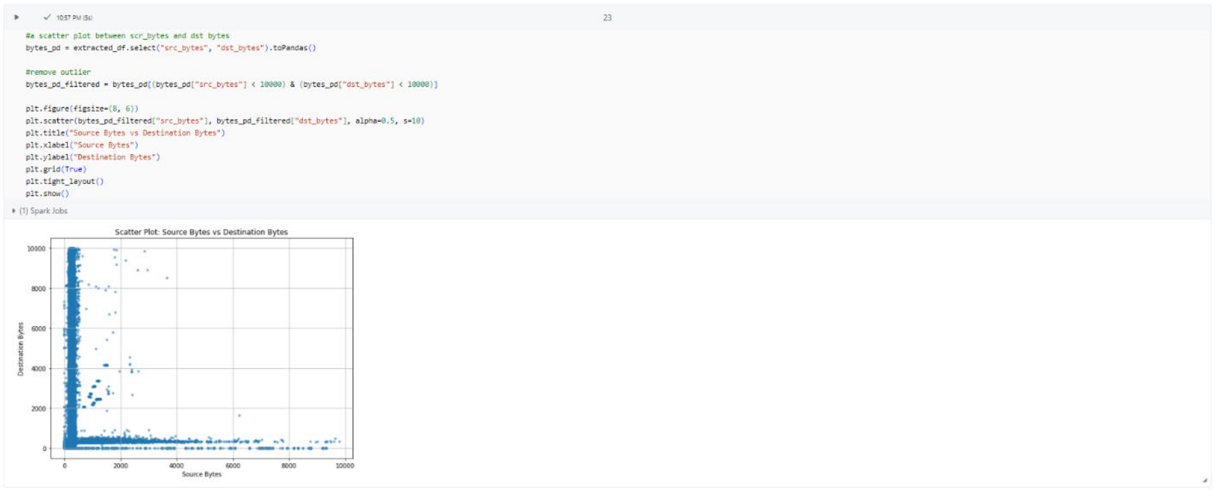
10:39 PM (7s)　21　Python

```
#explore the flag vs connection
extract_plot_connection_count_bar_chart(extracted_df,"flag",(10,8))
```

(6) Spark Jobs

```
+------+------+
|  flag| count|
+------+------+
|   OTH|     8|
|    S3|    10|
|RSTOS0|    11|
|    S2|    24|
|    S1|    57|
|    SH|   107|
|  RSTO|   579|
|  RSTR|   903|
|   REJ| 26875|
|    S0| 87007|
|    SF|378440|
+------+------+
```

a scatter plot between scr_bytes and dst_bytes（outlier removed）



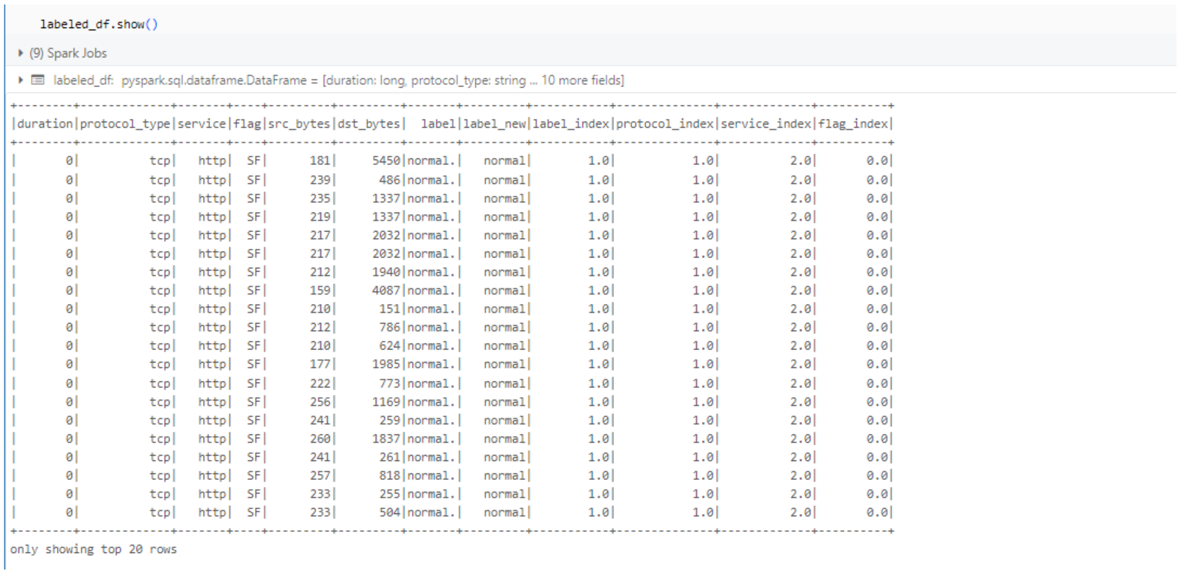
10:57 PM (5s)　23

```
#a scatter plot between scr_bytes and dst bytes
bytes_pd = extracted_df.select("src_bytes", "dst_bytes").toPandas()

#remove outlier
bytes_pd_filtered = bytes_pd[(bytes_pd["src_bytes"] < 10000) & (bytes_pd["dst_bytes"] < 10000)]

plt.figure(figsize=(8, 6))
plt.scatter(bytes_pd_filtered["src_bytes"], bytes_pd_filtered["dst_bytes"], alpha=0.5, s=10)
plt.title("Source Bytes vs Destination Bytes")
plt.xlabel("Source Bytes")
plt.ylabel("Destination Bytes")
plt.grid(True)
plt.tight_layout()
plt.show()
```

(1) Spark Jobs

8. [Marks: 20] Look at the label column where label == ‘normal’. Now create a new label column where you have a label == ‘normal’ and everything else is considered as an ‘attack’. Split your data (train/test) and based on your new label column now build a simple machine learning model for intrusion detection (you can use few selected columns for your model out of all). Explain which algorithm you have selected and why. Show the results with some success metrics.

Index the features of column after mapping all “normal.” to “normal” and others for “attack”



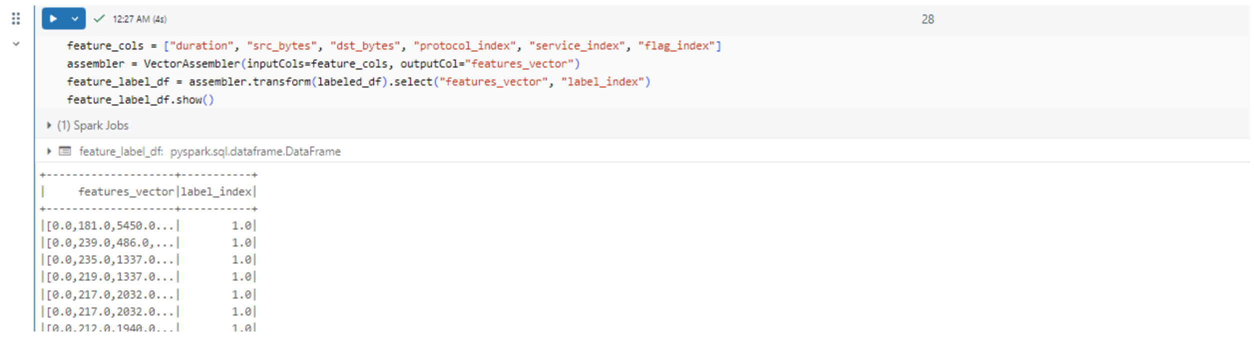
```
labeled_df.show()
```

▸ (9) Spark Jobs

▸ labeled_df: pyspark.sql.dataframe.DataFrame = [duration: long, protocol_type: string ... 10 more fields]

```
+--------+-------------+-------+----+---------+---------+-------+---------+-----------+--------------+-------------+----------+
|duration|protocol_type|service|flag|src_bytes|dst_bytes|  label|label_new|label_index|protocol_index|service_index|flag_index|
+--------+-------------+-------+----+---------+---------+-------+---------+-----------+--------------+-------------+----------+
|       0|          tcp|   http|  SF|      181|     5450|normal.|   normal|        1.0|           1.0|          2.0|       0.0|
|       0|          tcp|   http|  SF|      239|      486|normal.|   normal|        1.0|           1.0|          2.0|       0.0|
|       0|          tcp|   http|  SF|      235|     1337|normal.|   normal|        1.0|           1.0|          2.0|       0.0|
|       0|          tcp|   http|  SF|      219|     1337|normal.|   normal|        1.0|           1.0|          2.0|       0.0|
|       0|          tcp|   http|  SF|      217|     2032|normal.|   normal|        1.0|           1.0|          2.0|       0.0|
|       0|          tcp|   http|  SF|      217|     2032|normal.|   normal|        1.0|           1.0|          2.0|       0.0|
|       0|          tcp|   http|  SF|      212|     1940|normal.|   normal|        1.0|           1.0|          2.0|       0.0|
|       0|          tcp|   http|  SF|      159|     4087|normal.|   normal|        1.0|           1.0|          2.0|       0.0|
|       0|          tcp|   http|  SF|      210|      151|normal.|   normal|        1.0|           1.0|          2.0|       0.0|
|       0|          tcp|   http|  SF|      212|      786|normal.|   normal|        1.0|           1.0|          2.0|       0.0|
|       0|          tcp|   http|  SF|      210|      624|normal.|   normal|        1.0|           1.0|          2.0|       0.0|
|       0|          tcp|   http|  SF|      177|     1985|normal.|   normal|        1.0|           1.0|          2.0|       0.0|
|       0|          tcp|   http|  SF|      222|      773|normal.|   normal|        1.0|           1.0|          2.0|       0.0|
|       0|          tcp|   http|  SF|      256|     1169|normal.|   normal|        1.0|           1.0|          2.0|       0.0|
|       0|          tcp|   http|  SF|      241|      259|normal.|   normal|        1.0|           1.0|          2.0|       0.0|
|       0|          tcp|   http|  SF|      260|     1837|normal.|   normal|        1.0|           1.0|          2.0|       0.0|
|       0|          tcp|   http|  SF|      241|      261|normal.|   normal|        1.0|           1.0|          2.0|       0.0|
|       0|          tcp|   http|  SF|      257|      818|normal.|   normal|        1.0|           1.0|          2.0|       0.0|
|       0|          tcp|   http|  SF|      233|      255|normal.|   normal|        1.0|           1.0|          2.0|       0.0|
|       0|          tcp|   http|  SF|      233|      504|normal.|   normal|        1.0|           1.0|          2.0|       0.0|
+--------+-------------+-------+----+---------+---------+-------+---------+-----------+--------------+-------------+----------+
only showing top 20 rows
```

Input feature vectors and their label



▶ ✓ 12:27 AM (4s) 28

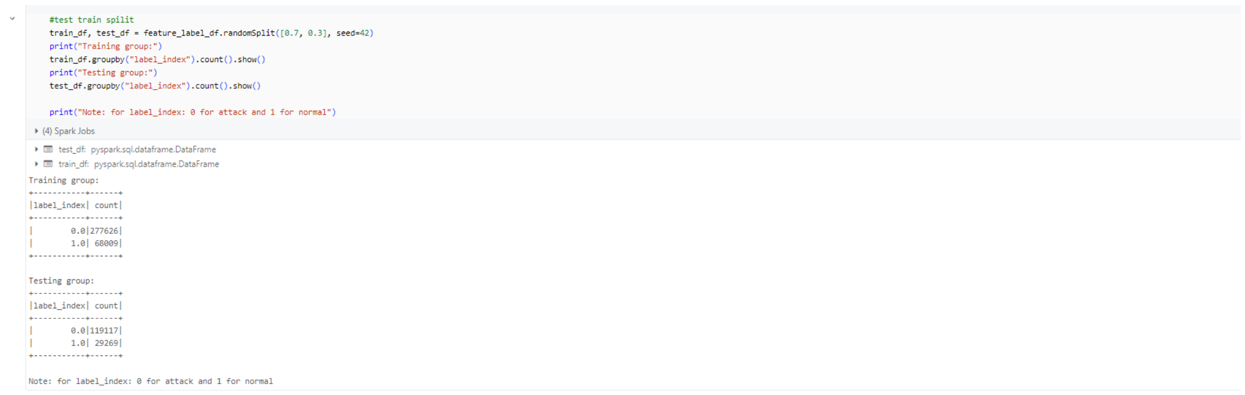
```
feature_cols = ["duration", "src_bytes", "dst_bytes", "protocol_index", "service_index", "flag_index"]
assembler = VectorAssembler(inputCols=feature_cols, outputCol="features_vector")
feature_label_df = assembler.transform(labeled_df).select("features_vector", "label_index")
feature_label_df.show()
```

▸ (1) Spark Jobs

▸ feature_label_df: pyspark.sql.dataframe.DataFrame

```
+--------------------+-----------+
|     features_vector|label_index|
+--------------------+-----------+
|[0.0,181.0,5450.0...|        1.0|
|[0.0,239.0,486.0,...|        1.0|
|[0.0,235.0,1337.0...|        1.0|
|[0.0,219.0,1337.0...|        1.0|
|[0.0,217.0,2032.0...|        1.0|
|[0.0,217.0,2032.0...|        1.0|
|[0.0,212.0,1940.0...|        1.0|
```

Training and testing group separation using 7:3 ratio



```
#test train spilit
train_df, test_df = feature_label_df.randomSplit([0.7, 0.3], seed=42)
print("Training group:")
train_df.groupby("label_index").count().show()
print("Testing group:")
test_df.groupby("label_index").count().show()

print("Note: for label_index: 0 for attack and 1 for normal")
```

▸ (4) Spark Jobs

▸ test_df: pyspark.sql.dataframe.DataFrame

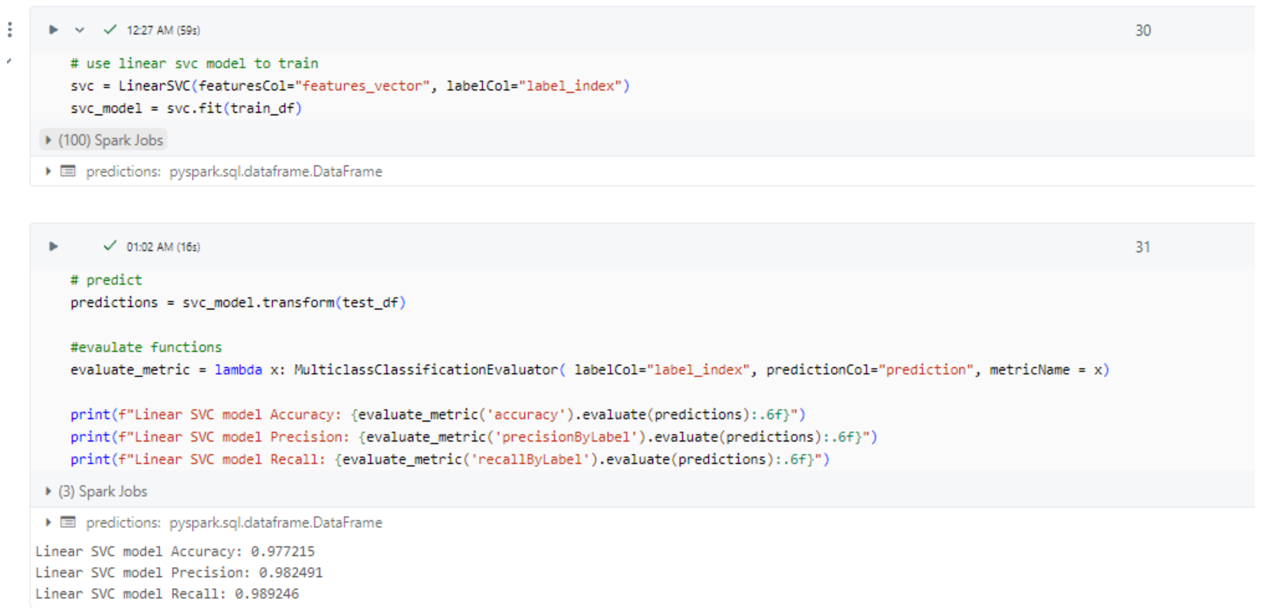
▸ train_df: pyspark.sql.dataframe.DataFrame

```
Training group:
+-----------+------+
|label_index| count|
+-----------+------+
|        0.0|277626|
|        1.0| 68009|
+-----------+------+

Testing group:
+-----------+------+
|label_index| count|
+-----------+------+
|        0.0|119117|
|        1.0| 29269|
+-----------+------+

Note: for label_index: 0 for attack and 1 for normal
```

The algorithm we deployed is LinearSVC (Support Vector Machine with a linear kernel)

Reason of selecting LinearSVC:

LinearSVC is a supervised model good at binary classification. It focuses on finding the boundary that maximizes separation between the two classes. In our model, we need to distinguish between the label "normal" and "attack" with multiple features provided. This is a typical binary classification problem and LinearSVC is a great choice for it.


12:27 AM (59s) 30

```
# use linear svc model to train
svc = LinearSVC(featuresCol="features_vector", labelCol="label_index")
svc_model = svc.fit(train_df)
```

(100) Spark Jobs

predictions: pyspark.sql.dataframe.DataFrame

01:02 AM (16s) 31

```
# predict
predictions = svc_model.transform(test_df)

#evaulate functions
evaluate_metric = lambda x: MulticlassClassificationEvaluator( labelCol="label_index", predictionCol="prediction", metricName = x)

print(f"Linear SVC model Accuracy: {evaluate_metric('accuracy').evaluate(predictions):.6f}")
print(f"Linear SVC model Precision: {evaluate_metric('precisionByLabel').evaluate(predictions):.6f}")
print(f"Linear SVC model Recall: {evaluate_metric('recallByLabel').evaluate(predictions):.6f}")
```

(3) Spark Jobs

predictions: pyspark.sql.dataframe.DataFrame

```
Linear SVC model Accuracy: 0.977215
Linear SVC model Precision: 0.982491
Linear SVC model Recall: 0.989246
```

Based on the metrics of accuracy, recall and precision using MulticlassClassificationEvaluator, results are as shown above

Part B: [Marks: 20]

1. [Marks: 4] Read the below statements, choose the correct answer, and provide explanations.

Statements
1. A platform as a service (PaaS) solution that hosts web apps in Azure provide professional development services to continuously add features to custom applications.

Yes
PaaS provides a framework that allow developers to create and manage their own cloud-based applications without worrying about provisioning, configuring or understanding OS, hardware etc.

2. A platform as a service (PaaS) database offering in Azure provides built-in high availability.

Yes, Azure has a built-in high availability and security as it can ensure that the service remain available with minimal downtime, even in the situation of failure event

2. [Marks: 4] Read the below statement, choose the correct answer, and provide explanations.
A relational database must be used when:
a. A dynamic schema is required
b. Data will be stored as key/value pairs
c. Storing large images and videos
d. Strong consistency guarantees are required

d
Relational Database is adhered to a schema thus a is wrong.
Key/value pair is for semi-structured data thus b is wrong
Image and videos are unstructured data thus c is wrong since relational data base is static.
d is correct

3. [Marks: 4] Read the below statement, choose the correct answer, and provide explanations.
When you are implementing a Software as a Service solution, you are responsible for:
a. Configuring high availability
b. Defining scalability rules
c. Installing the SaaS solution
d. Configuring the SaaS solution

d
In SaaS, a and b are provided by service providers, thus a and b are wrong,
User can directly access the SaaS solution, and they don't need to install it, where c is wrong.
In SaaS, the customer is only responsible for configuring and using the software based to their needs thus d is correct.

4. [Marks: 4] Read the below statements, choose the correct answer, and provide explanations.
Statements
1. To achieve a hybrid cloud model, a company must always migrate from a private cloud model

False
For a hybrid cloud model, company can start with public cloud model and combine with private cloud model and does not always migrate with private cloud model.

2. A company can extend the capacity of its internal network by using a public cloud

True
Public cloud model capacity can be extended due to its high scalability

3. In a public cloud model, only guest users at your company can access the resources in the cloud

False
In public cloud model, both internal and external users can access the resource controlled by the organization as long as they are granted access and authentication.

5. [Marks: 4] Read the below statements, choose the correct answer, and provide explanations.
a. A cloud service that remains available after a failure occurs <u>Fault Tolerance</u>

Fault Tolerance is how much failures system is able to handle

b. A cloud service that can be recovered after a failure occurs <u>Disaster recovery</u>

Disaster recovery is how the system can be recovered from failure

c. A cloud service that performs quickly when demand increases <u>Dynamic Scalability</u>

Dynamic Scalability is how system could deal with different requests and increase capacity based on demand of memory/traffic/computing power.

d. A cloud service that can be accessed quickly from the internet <u>Low Latency</u>

Low Latency is how people can access the system in a swift manner without communication overhead.

Disaster recovery, Fault Tolerance, Low Latency, Dynamic Scalability

Part C: [Marks: 20]

1. [Marks: 10] Hybrid Cloud Strategy: A large financial institution is considering migrating its legacy systems to the cloud. They have stringent regulatory requirements for data security and latency. They also need to maintain control over sensitive data and ensure compliance.
a. Design a hybrid cloud strategy outlining which components should reside on-premises versus in the cloud, considering security, regulatory compliance, latency, and cost effectiveness. Justify your decisions.

Components on-premises:
- The core of banking system (transaction, mortgage, stock etc.): it has high security performance requirements and establish on-premises could ensure the security and low access latency.
- The data of sensitive clients: the data that involved in private and sensitive data should store on-premises to reduce the risk of data breach
- Real-time detection and analysis system of either stock market, risk management, fraud detection etc: Usually these detection systems require immediate response for data stream and using on-premises set-up could ensure the performance and low-latency

Components in the cloud:
- Data analysis system for non-sensitive data: Using cloud model could efficiently process large-volume data and deploy the advanced analytic algorithms.
- Customer applications: set up on cloud could ensure the global accessibility and its availability.
- Non-sensitive Data Storage: set up on cloud storage could be cost-friendly and save the CapEx of setup large hard-drive and server etc.

b. Discuss the challenges in implementing and managing this hybrid cloud environment. What specific technical and operational considerations must be addressed?

Operational considerations:

One challenge is the data Security and compliance: for sensitive data like civil private information, we need to ensure that these data are handled collaboratively on-premises infrastructure and on the cloud for data residency and comply with all legal requirements and regulations.

Technical considerations:

The technical considerations could be encryption and decryption implementation for the data transferring between two models and the connection of different model use VPN or other network security rules need to be implemented.

c. Propose a plan for monitoring and managing the hybrid cloud infrastructure to ensure security, performance, and compliance.

- Security: Organizations could use some identity and access monitor by actively monitor the access history of user and log of event within the system to identify all different unauthorized access. For cloud model, we could also deploy the cloud-based threat detection tools and conduct several security assessments regularly.

- Performance: Organizations could use performance monitoring tools to track the CPU, RAM memory, GPU usage of each individual user. Also, several network monitoring tools could be used to track the network flow, latency and other statistics when data of two models are interacting.

- Compliance: Organizations could use some native policy enforcement tools to enforce encryption, and access control rules for the cloud model and the on-premises should grant access to people who are authorized.

2. [Marks: 10] Database Selection: A rapidly growing e-commerce company needs a database solution to handle increasing volumes of transactional data and customer information. They require high availability, scalability, and strong consistency guarantees. The data model involves relational and non-relational elements.
a. Recommend a database solution (or a combination of solutions) suitable for this scenario, explaining your choice based on the specific needs. Consider both relational and NoSQL options.

The recommendation would be using hybrid database solution as implement a database that combine with both relational and NoSQL database.

For the relational database like PostgreSQL or MySQL, it could support complicated relational queries and it has a high storge scale up to terabyte level, which is perfect for the e-commerce company for their transactions history or logs.

For NoSQL, it could have a high performance within all different action like read, write and its perfect for non-relational, data with no schema (semi-structure data) like session data, online access data, etc that are more event-driven.

Thus, using hybrid database model that deploy model collaboratively is better for the system as relational DB could use to handle most of transactional data while NoSQL could be in charge of those eventual data. Combine both models could ensure a high flexibility and more cost-effective.

b. Outline a data migration strategy for moving their existing data to the chosen solution, including considerations for downtime, data integrity, and testing.

- Access the current database as they need to identify their current relational and non-relational database and classify their data.
- Design the new schema for relational database and design the key for the NoSQL and conduct sanity test.
- Migrate historical relational data to relational DB and others to NoSQL and validate the data for data integrity.
- Conduct Application, network, performance, failure-recovery(downtime) as different type of testing, and perform data consistency test between old and new database
- New system online.

c. Discuss potential scalability challenges and how your chosen solution(s) can address them.

- One challenge is when the business grows, the volume of transactions and data storge demand would grow significantly.
  They could use the Relational database that ensure the support of auto-scaling storage and queried and for NoSQL, they could partition and distribute data on more data node
- One other challenge could be how to maintain the performance when user grows, and more queries are more intensive and complex. For relational DB, they could optimize the queries. Implementation of the cache system for both databases' queries is also feasible.
- Another challenge could be handling of different data based on the data model as some structural data and semi-structural data require different handling models and accessing these databases would use different method for collaboration. They could link both model with some common identifier like product ID or customer ID. To ensure the data integrity, data schemas, formats, and validation rules could be defined and reduce the data missing and data collision issue.