catalyst optimizer

Spark driver

Spark context

spark session

spark.driver.memory

spark.sql.shuffle.partition

spark.default.parallelism

spark context

DAG

Transformation and actions

Lazily evaluated

modes in dataframe reader , dataframe writer and spark structured streaming

explain ()

Delta table properties and usage

Repartition vs coalesce

narrow vs wide transfromation

scala spark vs pyspark

map vs mappartition

case class vs struct type

parquet vs orc file

manged table vs External table

static vs Dynamic partition

### Catalyst Optimizer

The Catalyst Optimizer is a query optimization framework in Apache Spark that optimizes query plans by applying rules and optimizations. It leverages an extensible framework to manipulate query plans, including predicate pushdown, constant folding, and more.

scala

Copy code

import org.apache.spark.sql.SparkSession

val spark = SparkSession.builder()

.appName("CatalystOptimizerExample")

.master("local[\*]")

.getOrCreate()

import spark.implicits.\_

val df = spark.read.csv("path/to/data.csv")

df.select($"col1" + $"col2").show()

df.explain()

### Spark Driver

The Spark Driver is the process that coordinates the execution of a Spark application. It converts user code into tasks and submits them to the Spark Cluster Manager for execution.

### Spark Context

Spark Context (sc) is the entry point for interacting with Spark functionality from a Spark application. It represents the connection to a Spark cluster and is used to create RDDs and broadcast variables.

### Spark Session

Spark Session is a unified entry point for accessing Spark functionality and represents the connection to a Spark cluster. It is introduced in Spark 2.0 to combine SparkContext, SQLContext, HiveContext, and StreamingContext into a single entry point.

### spark.driver.memory

spark.driver.memory is a Spark configuration property that determines the amount of memory to allocate to the driver process.

### spark.sql.shuffle.partition

spark.sql.shuffle.partition is a configuration property that determines the number of partitions to use when shuffling data for joins or aggregations in Spark SQL.

### spark.default.parallelism

spark.default.parallelism is a configuration property that sets the default number of partitions to use for RDDs and dataframes in Spark.

### DAG (Directed Acyclic Graph)

DAG represents the logical execution plan of a Spark job. It consists of a sequence of stages, where each stage represents a set of tasks that can be executed in parallel.

### Transformations and Actions

Transformations in Spark are operations that transform RDDs/DataFrames into another RDD/DataFrame. Actions, on the other hand, trigger computation and return results to the driver.

### Lazily Evaluated

In Spark, transformations are lazily evaluated, meaning they are not executed immediately. They are evaluated only when an action is called, which triggers the execution of the entire DAG.

### Modes in DataFrame Reader/Writer and Spark Structured Streaming

Modes in DataFrame Reader/Writer and Spark Structured Streaming specify how to handle existing data when writing to a file/data store. Modes include "append," "overwrite," "ignore," and "errorIfExists."

### explain()

explain() is a method in Spark DataFrame API used to display the execution plan (physical and logical) of a DataFrame.

### Delta Table Properties and Usage

Delta Table is a table format in Delta Lake, an open-source storage layer for data lakes. Delta tables support ACID transactions, schema evolution, and time travel. Properties include data retention, partitioning, and more.

### Repartition vs Coalesce

Repartition and Coalesce are methods to control the partitioning of data in Spark. Repartition shuffles data and allows specifying the number of partitions, while Coalesce reduces the number of partitions without shuffling.

### Narrow vs Wide Transformations

Narrow transformations are transformations where each input partition contributes to only one output partition, while wide transformations are transformations that require data from multiple partitions to compute the output partition.

### Scala Spark vs PySpark

Scala Spark and PySpark are APIs provided by Apache Spark for Scala and Python programming languages, respectively. Both APIs offer similar functionality, but they differ in syntax and performance characteristics.

### map vs mappartition

map is a transformation in Spark that applies a function to each element of an RDD/DataFrame, while mappartition applies a function to each partition of an RDD/DataFrame.

### Case Class vs StructType

Case class is a Scala feature used to define a lightweight data structure, while StructType is a data type in Spark SQL used to define the schema of a DataFrame.

### Parquet vs ORC File

Parquet and ORC are columnar storage file formats used in Apache Hadoop ecosystem. They offer similar performance characteristics and are optimized for analytics workloads.

### Managed Table vs External Table

Managed tables are tables whose metadata and data are managed by the Spark catalog, while External tables are tables whose metadata is managed by the Spark catalog but data resides outside of Spark's control.

### Static vs Dynamic Partition

Static partitioning involves explicitly specifying partition values when writing data, while dynamic partitioning automatically determines partition values based on data characteristics.

Day 15

Azure Databricks ⇒

Types of Storage :

Azure Blob

Azure Data Lake Storage (ADLS)

File share

Table

Queue

Azure Databricks → AWS /GCP

Blob --access key

SAS token --Shared Access signature Token

dbutils.fs.mount(

source = "wasbs://datasets@akdatabricksblob.blob.core.windows.net",

mount\_point = "/mnt/anand1",

extra\_configs = {"fs.azure.account.key.akdatabricksblob.blob.core.windows.net":dbutils.secrets.get(scope = "akscope", key = "adbblobkey")})

https://learn.microsoft.com/en-us/azure/databricks/security/secrets/secret-scopes

storage\_account\_name = "akdatabricksblob"

storage\_account\_access\_key = "o3Eq8XooO0OD2aO367lPNoILB2PwdPHZrVn0+/CpN0VV9quPZ2e5MQJDsXptrkpFi7Sfwqnc8n0q+AStVgTJqw=="

file\_location = "wasbs://datasets@akdatabricksblob.blob.core.windows.net/superstore.csv"

file\_type = "csv"

spark.conf.set(

"fs.azure.account.key."+storage\_account\_name+".blob.core.windows.net",

storage\_account\_access\_key)

Step 2: Read the data

Now that we have specified our file metadata, we can create a DataFrame. Notice that we use an option to specify that we want to infer the schema from the file. We can also explicitly set this to a particular schema if we have one already.

df = spark.read.format(file\_type).option("inferSchema", "true").load("wasbs://datasets@akdatabricksblob.blob.core.windows.net/emp.csv",header=True)

Mounting Azure blob into azure databricks 👍

Azure key vault to store the blob/ADLS access key or SAS token

Secret scope in Databricks to integrate Key vault with DAtabricks

patientdf=spark.read.csv("dbfs:/mnt/anand/patient.txt")

df.write.mode("overwrite").parquet("/mnt/anand/adbmntblob")

spark.sql.dynamicPartition.enabled=true

https://<databricks-instance>#secrets/createScope

https://adb-6021290970286233.13.azuredatabricks.net/

dbutils.fs.mount(

source = "wasbs://<container-name>@<storage-account-name>.blob.core.windows.nSet",

mount\_point = "/mnt/iotdata",

extra\_configs = {"fs.azure.account.key.<storage-account-name>.blob.core.windows.net":dbutils.secrets.get(scope = "<scope-name>", key = "<key-name>")})

============================================================

Important terms and configurations :

Spark session

Role of : spark driver , executor, HDFS , partition , YARN ,

RDD

Dataframe

Partition -> dynamic and static

memory ->

spark.executor.memory

spark.default.parallelism= number of cores in the cluster (RDD) ==> customize to 100 ,200,50

spark.sql.shuffle.partitions= 200 (DF)

spark.serializer

spark.driver.cores=1 (defaullt)

spark.driver.memory=1G (default)

pycharm installation and virtualenv setup

===================================================

Day 13

https://drive.google.com/file/d/0B6LDnz6V2Lv9SnBsYnZ0NHB6c0E/view?usp=sharing&resourcekey=0-L98f-Cnsu6Sr-RBbfezq8A

Databricks :

==================apply ====================

salesdf.write.format("parquet").partitionBy("country","state").bucketBy(5,"city").saveAsTable("sales\_bucket")

==========================================================

userdf=spark.read.parquet("dbfs:/FileStore/users.parquet")

from pyspark.sql.functions import \*

userdf.select("name",explode\_outer("favorite\_numbers")).show()

================================================================

%fs ls /FileStore

salesdf=spark.read.csv("dbfs:/FileStore/superstore.csv",header=True,inferSchema=True)

salesdf.write.saveAsTable("sales")

%sql

update sales set Discount=1 where ID=32298;

%sql

delete from sales where Country='India'

===================time travel

%sql

DESCRIBE HISTORY sales;

%sql

SELECT count(\*) FROM sales TIMESTAMP AS OF '2024-03-07T10:23:59.000+0000'

%sql

SELECT \* FROM sales VERSION AS OF 1

=============================================================

kakfadf = spark.readStream.format("kafka").option("kafka.bootstrap.servers", "localhost:9092").option("subscribe", "womensday-celeb").load()

kakfadf.selectExpr("CAST(key AS STRING)", "CAST(value AS STRING)")

query = kafkadf.writeStream.outputMode('append').format('console').option("truncate","false").start().awaitTermination()

pyspark --packages org.apache.spark:spark-sql-kafka-0-10\_2.11:2.2.0

cd /usr/local/kafka/bin

sudo ./zookeeper-server-start.sh ../config/zookeeper.properties

Kafka Topics

Go to kafka bin directory to execute the following commands :open different terminal tabs to execute

cd kafka-2.xxx/bin

1.#### To Start zookeeper daemons (no need to start)

--------------------------------------------

sudo ./zookeeper-server-start.sh ../config/zookeeper.properties

2.###To start the kafka server (in new terminal tab, run this)

-------------------------------------------------

sudo ./kafka-server-start.sh ../config/server.properties

3.###To create the kafka topic(new tab)

------------------------------------------------

./kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 2 - -topic securityalerts

4.###To list the topics

----------------------------------------------------------

./kafka-topics.sh --list --zookeeper localhost:2181

5### To Run the producer(new tab)

-----------------------------------------------------------

./kafka-console-producer.sh --broker-list localhost:9092 --topic securityalerts

6### To Run consumer (new tab)

-------------------------------------------------------------

./kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic securityalerts --from-beginning

List existing topics

kafka-topics.sh --zookeeper localhost:2181 --list

Describe a topic

kafka-topics.sh --zookeeper localhost:2181 --describe --topic securityalerts

Purge a topic

kafka-topics.sh --zookeeper localhost:2181 --alter --topic securityalerts --config retention.ms=1000

... wait a minute ...

kafka-topics.sh --zookeeper localhost:2181 --alter --topic securityalerts --delete-config retention.ms

Delete a topic

kafka-topics.sh --zookeeper localhost:2181 --delete --topic securityalerts

Get number of messages in a topic ???

kafka-run-class.sh kafka.tools.GetOffsetShell --broker-list localhost:9092 --topic securityalerts --time -1 --offsets 1 | awk -F ":" '{sum += $3} END {print sum}'

Get the earliest offset still in a topic

kafka-run-class.sh kafka.tools.GetOffsetShell --broker-list localhost:9092 --topic securityalerts --time -2

Get the latest offset still in a topic

kafka-run-class.sh kafka.tools.GetOffsetShell --broker-list localhost:9092 --topic securityalerts --time -1

1-A 2-B 3-D 4-C

Driver's role in task scheduling

spark-submit --packages option

Resolving dependency conflicts with spark.jars.packages

YARN's NodeManager in Spark job execution

Options:

A) The component that is responsible for negotiating resources with the Cluster Manager and plays a crucial role in task distribution to Executors.

B) A spark-submit option that allows the user to specify Maven coordinates of jars to include on the driver and executor classpaths.

C) The YARN component that manages the user job's executors on a node, monitoring their resource usage (CPU, memory, disk, network) and reporting the same to the ResourceManager.

D) A Spark property that can help resolve jar dependency conflicts by specifying the coordinates of the Maven artifacts to be included in the runtime."

1-B 2- A 3-D 4-C

spark.executor.memoryOverhead

spark.memory.fraction

spark.memory.storageFraction

spark.driver.maxResultSize

Options:

A) Adjusts the proportion of JVM memory dedicated for Spark to use, leaving the rest for user data structures and overhead.

B) Specifies the amount of non-heap memory to be allocated per executor, beyond the memory used for caching and storing tasks' data.

C) Controls the size of the maximum result that can be fetched by the driver from the executors at one time, affecting the driver's ability to handle large collected datasets.

D) Dictates the fraction of Spark memory to be reserved for storage of Spark internal data structures, with the remainder available for execution and tasks.

"

13.When setting up Spark on a standalone machine, what is one of the first steps you should take?

14. To cache a DataFrame in SparkSQL for faster access in subsequent actions, the method \_\_\_\_cache()\_\_\_\_\_\_\_ is used.

15. In SparkSQL, to read and write data in JSON format, one would use the format option \_\_\_"json"

\_\_\_\_\_\_\_\_ in the DataFrameReader and DataFrameWriter APIs.

1-A 2-D 3 -C 4-B

Stateless Transformation

Stateful Transformation

Micro-batch Processing

Garbage Collection Tuning

Options:

A) These transformations use the information from the current batch only and don't retain any state from previous batches.

B) This refers to optimizing the JVM's garbage collection to handle the frequent allocation and deallocation of short-lived objects typically seen in streaming applications.

C) Involves processing the data in small time chunks, where each batch contains the data received during the configured batch interval.

D) These transformations consider data across multiple batches and involve maintaining and updating state over time."

17. Using Bucketing what gets created in Hive Warehouse? Directories Folders Files List

18. Using Partitioning what gets created in Hive Warehouse? Directories Folders Files List

19. What config is used to read Hive Data in PySpark? Spark Context SQLContext SparkConf HiveContext

20. In a Spark application, which component is responsible for managing the execution of tasks across the cluster and keeps track of the status of executors?

21.If you read and load data with following records in a PySpark dataframe, what will be data type "topping" and "batters" ?

{'id': '0002',

'type': 'donut',

'name': 'Raised',

'ppu': 0.55,

'batters': {'batter': [{'id': '1001', 'type': 'Regular'}]},

'topping': [{'id': '5001', 'type': 'None'},

{'id': '5002', 'type': 'Glazed'},

{'id': '5005', 'type': 'Sugar'},

{'id': '5003', 'type': 'Chocolate'},

{'id': '5004', 'type': 'Maple'}]}

{"id": "0002","type": "donut","name": "Raised","ppu": "0.55","batters": {"batter": [{"id": "1001", "type": "Regular"}]},"topping": [{"id": "5001", "type": "None"},{"id": "5002", "type": "Glazed"},{"id": "5005", "type": "Sugar"},{"id": "5003", "type": "Chocolate"},{"id": "5004", "type": "Maple"}]}

22.On a spark cluster of n nodes with host1 being the master, when a spark application is run using the following command:

spark-submit --class --master spark://host1:7077

1-D 2 -A 3 - B 4 - C

File Streams

Flume

Pull-based Approach

Checkpointing

Options:

A) This input source is for data that is pushed into Spark Streaming from a distributed aggregation service.

B) A method of data retrieval in Spark Streaming where the system polls for data at regular intervals.

C) Refers to the process of writing the state of the streaming computation to reliable storage for fault tolerance.

D) An input source that monitors a directory for new files and processes them as they are detected."

1-D 2-A 3 -C 4-B

Real-time Analytics

Collaborative Data Science

Databricks MLflow

Delta Lake Management

Options:

A) A feature that allows data teams to collaboratively work on complex data science and machine learning projects with shared notebooks and code repositories.

B) The process of managing and optimizing Delta Lake, which provides ACID transactions and scalable metadata handling for big data workloads.

C) A tool within the Databricks platform that streamlines the machine learning lifecycle, including experimentation, reproducibility, and deployment.

D) The capability to process and analyze data as it arrives in real-time, enabling immediate insights and actions based on the most current data."

26. What Happens if you drop External Tables in Hive?

1-D 2- B 3 -C 4 -A

spark.executor.memory

spark.default.parallelism

spark.serializer

spark.driver.cores

Options:

A) Determines the number of cores to use for the Driver process, affecting its ability to schedule tasks.

B) Influences the level of parallelism by setting the default number of partitions in RDDs, DataFrames, and Datasets when no partitioning information is available.

C) Specifies the serializer class for serializing objects that will be sent over the network or need to be cached in serialized form, impacting speed and memory footprint.

D) Sets the amount of memory to use per executor process, impacting the amount of data that can be processed and the need for spill to disk."

1-A 2 -B 3 -D 4-C

Driver's role in task scheduling

spark-submit --packages option

Resolving dependency conflicts with spark.jars.packages

YARN's NodeManager in Spark job execution

Options:

A) The component that is responsible for negotiating resources with the Cluster Manager and plays a crucial role in task distribution to Executors.

B) A spark-submit option that allows the user to specify Maven coordinates of jars to include on the driver and executor classpaths.

C) The YARN component that manages the user job's executors on a node, monitoring their resource usage (CPU, memory, disk, network) and reporting the same to the ResourceManager.

D) A Spark property that can help resolve jar dependency conflicts by specifying the coordinates of the Maven artifacts to be included in the runtime."

====================================================

Day 12

https://github.com/akgeoinsys/retail

from pyspark import SparkContext

from pyspark.streaming import StreamingContext

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

ssc = StreamingContext(sc, 10)

lines = ssc.socketTextStream("localhost", 9999)

words = lines.flatMap(lambda line: line.split(" "))

# Count each word in each batch

pairs = words.map(lambda word: (word, 1))

wordCounts = pairs.reduceByKey(lambda x, y: x + y)

# Print the first ten elements of each RDD generated in this DStream to the console

wordCounts.pprint()

# Print the first ten elements of each RDD generated in this DStream to the console

wordCounts.pprint()

==============Spark structured streaming

lines = spark.readStream.format("socket").option("host", "localhost").option("port", 9999).load()

words = lines.select(explode(split(lines.value, " ") ).alias("word"))

wordCounts = words.groupBy("word").count()

query = wordCounts.writeStream.outputMode("complete").format("console").start()

==========================Modes =================

Streaming Mode : Append , update , complete

1st entry a a a b b

2nd a a c

3rd b b c d

Complete

1st a,3 b,2

2nd a,5 b,2 c, 1

3rd a,5 b, 4 c,2 d, 1

Append

1st a,3 b, 2

2nd a,2 c ,1

3rd b,2 c,1 d,1

Update

1st a3 , b2

2nd a,5 c, 1

3rd b,4 ,c,2 d,1

5

def add\_timestamp():

ts = time.time()

timestamp = datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d %H:%M:%S')

return timestamp

trimmedDF = fileStreamWithTS.select("borough","major\_category", 'value',"timestamp")

query = trimmedDF.writeStream\

.outputMode('append')\

.format('console')\

.option("truncate","false")\

.start()\

.awaitTermination()

=================================

Day 11

userdf=spark.read.format("parquet").load("file:///usr/local/spark/examples/src/main/resources/users.parquet")

>>> userdf=spark.read.load("file:///usr/local/spark/examples/src/main/resources/users.parquet")

>

>> userdf=spark.read.parquet("file:///usr/local/spark/examples/src/main/resources/users.parquet")

>>> userdf.printSchema()

root

|-- name: string (nullable = true)

|-- favorite\_color: string (nullable = true)

|-- favorite\_numbers: array (nullable = true)

| |-- element: integer (containsNull = true)

>>> userdf.show()

+------+--------------+----------------+

| name|favorite\_color|favorite\_numbers|

+------+--------------+----------------+

|Alyssa| null| [3, 9, 15, 20]|

| Ben| red| []|

+------+--------------+----------------+

salesdf.write.option("compression","gzip").format("csv").save("file:///home/hduser/target/sales\_gz")

>>>

>>> salesdf=spark.read.format("csv").load("file:///home/hduser/target/sales\_gz/part-00000-e818eaf5-08ef-4873-a60e-219c514fe113-c000.csv.gz")

--------------------------broadcast variable ---------------------------

# Broadcast variables are read-only shared variables that can be cached on each machine.

# They are used to efficiently distribute large read-only data structures to all tasks.

from pyspark.sql import SparkSession

def broadcast\_example(spark):

data\_to\_broadcast = [1, 2, 3, 4, 5]

broadcast\_data = spark.sparkContext.broadcast(data\_to\_broadcast)

def process\_data(x):

# Access the broadcast variable within the transformation

return x \* broadcast\_data.value[0]

rdd = spark.sparkContext.parallelize([1, 2, 3, 4, 5])

result = rdd.map(process\_data).collect()

print("Result:", result)

if \_\_name\_\_ == "\_\_main\_\_":

spark\_session = SparkSession.builder.appName("BroadcastExample").getOrCreate()

broadcast\_example(spark\_session)

spark\_session.stop()

-------------------------Accumulator--------------------

# Accumulators are variables that can be used to aggregate values across multiple tasks in a parallel manner.

# They are typically used for counters and sums.

# Code:

from pyspark.sql import SparkSession

def accumulator\_example(spark):

accumulator\_var = spark.sparkContext.accumulator(0)

def process\_data(x):

# Increment the accumulator within the transformation

accumulator\_var.add(x)

return x

rdd = spark.sparkContext.parallelize([1, 2, 3, 4, 5])

result = rdd.map(process\_data).collect()

print("Result:", result)

print("Accumulator Value:", accumulator\_var.value)

if \_\_name\_\_ == "\_\_main\_\_":

spark\_session = SparkSession.builder.appName("AccumulatorExample").getOrCreate()

accumulator\_example(spark\_session)

spark\_session.stop()

===============Reading table from RDB -Mysql ========================

products\_mysqldf = spark.read.format("jdbc").option("url", "jdbc:mysql://localhost:3306/salesdb?allowPublicKeyRetrieval=true&useSSL=false").option("driver", "com.mysql.jdbc.Driver").option("dbtable", "products").option("user", "root").option("password", "root").load()

order\_items\_mysqldf = spark.read.format("jdbc").option("url", "jdbc:mysql://localhost:3306/salesdb?allowPublicKeyRetrieval=true&useSSL=false").option("driver", "com.mysql.jdbc.Driver").option("dbtable", "order\_items").option("lowerBound", "1").option("upperBound", "172168").option("numPartitions", "10").option("partitionColumn","order\_item\_id").option("user", "root").option("password", "root").load()

q= "(select \* from order\_items where order\_item\_order\_id>100) ord"

order\_items\_mysqldf = spark.read.format("jdbc").option("url", "jdbc:mysql://localhost:3306/salesdb?allowPublicKeyRetrieval=true&useSSL=false").option("driver", "com.mysql.jdbc.Driver").option("dbtable",q).option("user", "root").option("password", "root").load()

=======================complex data types ============================

true|10|100|1000|10000|4.0|20.0|2.2222|1969-12-31 15:59:58.174|1970-01-01 00:00:00|hello|hello|k1:v1,k2:v2|100,200|{10, "foo"}

true|20|200|2000|20000|8.0|40.0|4.2222|1970-12-31 15:59:58.174|1971-01-01 00:00:00|||k3:v3,k4:v4|200,300|{20, "bar"}

spark.sql("""

CREATE TABLE IF NOT EXISTS alltypes (

bo1 BOOLEAN,

ti1 TINYINT,

si1 SMALLINT,

i1 INT,

bi1 BIGINT,

f1 FLOAT,

d1 DOUBLE,

de1 DECIMAL,

ts1 TIMESTAMP,

da1 DATE,

s1 STRING,

vc1 VARCHAR(5),

m1 MAP<STRING, STRING>,

l1 ARRAY<INT>,

st1 STRUCT<c1:INT, c2:STRING>

)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY '|'

COLLECTION ITEMS TERMINATED BY ','

MAP KEYS TERMINATED BY ':'

STORED AS TEXTFILE

""")

spark.sql("load data local inpath 'file:///usr/local/hive/examples/files/alltypes.txt' into table alltypes")

=================================Configuration=========================

spark.conf.get("spark.sql.autoBroadcastJoinThreshold")

spark.conf.set("spark.sql.autoBroadcastJoinThreshold","100")

order\_itemdf.join(broadcast(productdf),order\_itemdf.order\_item\_product\_id == productdf.product\_id,"inner").show()

spark.conf.set("spark.sql.join.preferSortMergeJoin", "false")

# Creating a SparkSession with custom resource allocation settings

spark\_session = SparkSession.builder \

.appName("ResourceAllocationExample") \

.config("spark.executor.memory", "2g") # Set executor memory to 2 gigabytes

.config("spark.executor.cores", 2) # Allocate 2 cores per executor

.config("spark.driver.memory", "1g") # Set driver memory to 1 gigabyte

.config("spark.driver.cores", 1) # Allocate 1 core for the driver

.getOrCreate()

spark.conf.get("spark.sql.parquet.compression.codec")

-spark.conf.set("spark.sql.parquet.compression.codec","gzip")

====================Mock ===========================

1. Data structure/collections in python

2. apply(), map(), reduce (), filter()

3. use of lambda function

4. how to call exceptions

5. What is the output of the following Python code snippet?

numbers = [1, 2, 3]

for i, number in enumerate(numbers):

numbers[i] \*= 2

print(numbers)

In how many ways you can create RDDs?

6. Repartition() is a method of pyspark.sql.DataFrame class that is used to increase or decrease the number of partitions of the DataFrame.

7.What is Default Write mode in PySpark

8.How many write modes are in PySpark DataFrames?

9.What happens when you perform a map transformation on an RDD in PySpark?

10. What is the purpose of the "registerTempTable" method in Spark SQL?

Match the following Spark features with their appropriate descriptions:

1-

1)Broadcast Variables -D

2)Accumulators -B

3)MapPartitions -A

4)Piping Operations -C

Options:

A) Used to perform a function on each partition of the RDD independently.(3)

B) Variables that are used to sum up values across the cluster in a parallel and fault-tolerant manner.(2)

C) Allow executing shell commands on each partition of an RDD.(4)

D) Enable efficient distrib(1)

=============================================

Project task 1:

Problem definition /Statement :

KYD --Know Your Data

No of columns , description , primary key , foreign key

No of rows , missing , null , mismatch , delimiter

Cleansing and Scrubbing

Airlines :

Data source : csv (airport) === spark --file

Hdfs (flights) → Hive External ⇒ spark

RDB (airlines) ⇒ Mysql ==>spark

Loan : (using pandas split the data into three dataframe and write it to separate file

1947--2000 Csv ⇒ spark ⇒ df⇒ filter ---> 2001

2001-2020 Hdfs → hive external table ⇒spark ⇒ 2002

Latest Mysql → JDBC ==spark ⇒2003

Retail :

2 files each ,load it to the following sources

Data source : csv === spark --file

Hdfs → Hive External ⇒ spark

RDB ⇒ Mysql ==>spark

Covid :

https://github.com/CSSEGISandData/COVID-19/tree/master/csse\_covid\_19\_data/csse\_covid\_19\_daily\_reports

Data source : csv (2020) === spark --file

Hdfs (2021) → Hive External ⇒ spark

RDB (2022) ⇒ Mysql ==>spark

Altenate fuel :

https://afdc.energy.gov/data\_download

Data source : Alternate fuelstations : csv === spark --file

Light duty vehicles : Hdfs → Hive External ⇒ spark

Medium and Heavy => RDB ⇒ Mysql ==>spark

========================================

Day 10

Emp = [(1,"Smith",1,"2018","10","M",3000.00),

(2,"Rose",1,"2010","20","M",4000.00),

(3,"Williams",1,"2010","10","M",1000.00),

(4,"Jones",2,"2005","10","F",2000.00),

(5,"Brown",2,"2010","40","",300.00),

(6,"Brown",2,"2010","50","",2000.00)

]

EmpSchema = StructType([

StructField('Emp\_id', IntegerType(), True),

StructField('Empname', StringType(), True),

StructField('MGR', IntegerType(), True),

StructField('YOJ', StringType(), True),

StructField('dept\_id', StringType(), True),

StructField('gender', StringType(), True),

StructField('salary', DoubleType(), True)

])

from pyspark.sql.types import StructType,StructField, StringType,IntegerType

EmpDF = spark.createDataFrame(data=Emp, schema = EmpSchema)

dept = [("Finance",10), \

("Marketing",20), \

("Sales",30), \

("IT",40) \

]

deptColumns = ["dept\_name","dept\_id"]

deptDF = spark.createDataFrame(data=dept, schema = deptColumns)

EmpDF.join(deptDF,empDF.dept\_id == deptDF.dept\_id,"inner").show()E

from pyspark.sql.functions import \*

Emp = [

(1,"Smith",1,"2018","10","M",3000.00),

(2,"Rose",1,"2010","20","M",4000.00),

(3,"Williams",1,"2010","10","M",1000.00),

(4,"Jones",2,"2005","10","F",2000.00),

(5,"Brown",2,"2010","40","",300.00),

(6,"Brown",2,"2010","50","",2000.00)

]

EmpDF.join(broadcast(deptDF),EmpDF.dept\_id == deptDF.dept\_id,"inner").explain()

============================================================

3rd question:

hive> select book\_id,book\_title,book\_type,book\_cost from books where book\_cost in (select max(book\_cost) from books);

Day 8

spark.sql("create database dataengg")

spark.sql("show databases").show()

spark.sql("use dataengg")

salesdf.write.saveAsTable("dataengg.sales")

create table patient(pid INT, pname STRING, drug STRING,gender string, tot\_amt INT) row format delimited fields terminated by ',' stored as textfile;

load data local inpath 'file:///home/hduser/datasets/patient.csv' into table patient;

select count(\*) from patient;

CREATE EXTERNAL TABLE wordcount(word string,count int) ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' LINES TERMINATED BY '\n' STORED AS TEXTFILE LOCATION '/mrout/wcout';

=====================================================

Day 7

Team 1 --Retail - Vishnu ,Sahil, Angela, Riya

https://github.com/akgeoinsys/Shell2023/tree/main/Sales360

team 2 -- Carbon emission - Pragati, Baskar,Arpit,

https://github.com/akgeoinsys/Shell2023/tree/main/EVCarbonEmission

team 3 --covid 19- > Nikhil, Nirmeetha, Rohit, Sandeep

Batch Processing

https://github.com/akgeoinsys/ltimindtree2023/tree/main/casestudies/datasets/healthcare/worldometer\_data.csv

Spark Streaming Datasets: Latest

https://github.com/akgeoinsys/ltimindtree2023/tree/main/casestudies/datasets/healthcare/countr

y\_wise\_latest.csv

team 4 -- Worldbank ->loan => Gaurav , Shrasti ,Thirumalesh,Mansi

Batch Processing 👍

https://finances.worldbank.org/Loans-and-Credits/IBRD-Statement-Of-Loans-Historical-Data/zucq-nrc3/about\_data

Spark Streaming Datasets: Latest

https://finances.worldbank.org/Loans-and-Credits/IBRD-Statement-of-Loans-Latest-Available-Snapshot/sfv5-tf7p/about\_data

team 5 --> Airlines : Manideep ,Likitha, Dinesh

https://github.com/akgeoinsys/ltimindtree2023/blob/main/casestudies/datasets/airlines.csv

Airports:

https://github.com/akgeoinsys/ltimindtree2023/blob/main/casestudies/datasets/airports.csv

Streaming datasets :

Flights: (historical)

https://github.com/akgeoinsys/ltimindtree2023/blob/main/casestudies/datasets/flights\_sample.cs

v

Flights: (latest)

https://github.com/akgeoinsys/ltimindtree2023/blob/main/casestudies/datasets/flights\_latest.csv

salesdf.select("state","city").filter("country='India'").orderBy(["state","city"],ascending=[0,1]).distinct().show()

salesdf.select("country","state").filter("country='India'").sort("state").distinct().show()

from pyspark.sql.functions import col

salesdf.select("country","state").filter("country='India'").sort(col("state").desc()).distinct().show()

from pyspark.sql.types import \*

from pyspark.sql.functions import \*

empdf.select((col("bonus")+col("SALARY")).alias("GrossSal")).show()

empdf.select("\*",(col("bonus")+col("SALARY")).alias("GrossSal")).show()

empdf.withColumn("GrossSalary",col("salary")+col("Bonus")).show()

empdf\_new=empdf.withColumnRenamed("sal","Salary").withColumn("Bonus",lit(1000)).withColumn("Loc",lit("Hyd"))

===========Aggregation ==================

display maxsal,min sal and avg sal of employees

empdf.select(max("salary"),min('salary'),avg("salary")).show()

empdf.agg(max("salary").alias("MaxSAL"),min('salary'),avg("salary")).show()

user defined schema creation :

##user defined schema

from pyspark.sql.types import StructType,StructField, StringType,IntegerType

EmpSchema = StructType([ StructField('Empno', IntegerType(), True),

StructField('Empname', StringType(), True),

StructField('Salary', IntegerType(), True)

])

employeedf=spark.read.option("mode","dropmalformed").schema(EmpSchema).csv("file:///home/hduser/datasets/employee.csv",header=True)

employeedf=spark.read.option("mode","failfast").schema(EmpSchema).csv("file:///home/hduser/datasets/employee.csv",header=True)

employeedf=spark.read.option("mode","permissive").schema(EmpSchema).csv("file:///home/hduser/datasets/employee.csv",header=True)

spark.read.schema(EmpSchema).csv("file:///home/hduser/datasets/employee.csv",header=True).dropna().show()

## na.drop fill or replace

employeedf\_permissive=employeedf

employeedf\_permissive.na.drop().show()

employeedf\_permissive.na.drop("any").show()

employeedf\_permissive.na.drop(how="all").show()

employeedf\_permissive.na.drop(subset="empno").show()

=====================================================

day 6

emprdd=sc.textFile("file:///home/hduser/datasets/emp.csv")

>>> emprdd.collect()

['Empno,ename,sal', '111,zzz,8000', '111,aaa,8888', '121,bbb,8000', '444,eee,9999', '555,ak,7878']

>>> empdf=spark.read.csv("file:///home/hduser/datasets/emp.csv")

>>> empdf.show()

empdf=spark.read.csv("file:///home/hduser/datasets/emp.csv",header=True)

>>> empdf.show()

+-----+-----+----+

|Empno|ename| sal|

+-----+-----+----+

| 111| zzz|8000|

| 111| aaa|8888|

| 121| bbb|8000|

| 444| eee|9999|

| 555| ak|7878|

+-----+-----+----+

empdf=spark.read.csv("file:///home/hduser/datasets/emp.csv",header=True,inferSchema=True)

>>> empdf.printSchema()

root

|-- Empno: integer (nullable = true)

|-- ename: string (nullable = true)

|-- sal: integer (nullable = true)

empdf.show(n)

empdf.limit(n).show()

Empdf.head(n)

Empdf.first()

empdf.rdd.getNumPartitions()

lines=sc.textFile("file:///home/hduser/datasets/people.txt")

>>> from pyspark.sql.types import Row

>>> parts=lines.map(lambda l:l.split(",")

... )

>>> people=parts.map(lambda p:Row(name=p[0],age=int(p[1]))

... )

>>> peopledf=spark.createDataFrame(people)

>>> peopledf.show()

+---+-------+

|age| name|

+---+-------+

| 29|Michael|

| 30| Andy|

| 19| Justin|

+---+-------+

salesdf=spark.read.option("header","true").option("inferSchema","true").csv("file:///home/hduser/datasets/superstore.csv"

salesdf.select("country","state").show()

salesdf.select("country","state").filter("country='India'").show()

salesdf.select("country","state").where("country='India'").show()

salesdf.select("country","state").where("country='India'" and "state='Gujarat'").distinct().show(50)

===========================================================================

Day 5

https://github.com/akgeoinsys/retail

##start all the daemons

start-dfs.sh

start-yarn.sh

start-master.sh

start-slave.sh localhost:7077

##to start scala sparl

spark-shell

##to open pyspark

pyspark

##wordcount in spark

##scala

val srcfile=sc.textFile("file:///home/hduser/datasets/sample.txt")

##pyspark

srcfile=sc.textFile("file:///home/hduser/datasets/sample.txt")

lines= srcfile.flatMap(lambda line: line.split(" "))

words=lines.map(lambda word: (word, 1))

counts=words.reduceByKey(lambda a, b: a + b)

counts.collect()

=================================================

Day 4

https://github.com/akgeoinsys/deloitte\_dta --

wordcount

https://github.com/akgeoinsys/retail

Quizzzzzzzz

Which of the following rightly defines a distributed system?

What does the below comprehensive list do:

print([for elem in num if len(str(elem)) == 3] elem/2)

What's wrong with this loop?

i = 0;

while (i < 10) :

…

I = i+1

==========================

Create and display a DataFrame from the following dictionary data

a. 'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],

b. 'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],

c. ‘attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],

=========================================

Day 3

###to start the daemons

start-dfs.sh

Start-yarn,sh

Open the web browser for hdfs web console and type

localhost:50070

hadoop fs -mkdir day1

###to download the big file

spark.apache.org

/usr/local/hadoop-2.9.1/etc/hadoop/

==================================to download the datasets from github ==================

https://github.com/akgeoinsys/retail

=======================================

day 2

cd

cd /

cd ..

cd ../..

cd -

ls

mkdir dta

mkdir -p 2024/feb/20

cd

1745 pwd

1746 mkdir dta

1747 mkdir /2024/feb/20

1748 mkdir

1749 mkdir --help

1750 mkdir 2024/feb/20

1751 mkdir -p 2024/feb/20

1752 cd 2024/feb/20

1753 cd ..

1754 cd ../..

1755 cd -

1756 cd 20

1757 cat >sample.txt

1758 cat sample.txt

1759 cat >>sample.txt

1760 cat sample.txt

1761 ls

1762 ls -lrt

1763 ls --help

1764 cd

1765 ls -lrtt

1766 ls -lrt

1767 cd 2024

1768 ls -lrt

1769 cd

1770 cd dta

1771 ls -l

1772 cd

1773 cd datasets

1774 ls -l

1775 cd

1776 ls

1777 cd 2024/feb/20/

1778 cp sample.txt demo.txt

1779 mv demo.txt 2024/feb

1780 mv demo.txt /home/hduser/2024/feb

1781 ls

1782 cp sample.txt demo.txt

1783 ls

1784 rm sample.txt

1785 cd

1786 rm 2024

1787 rmdir 2024

1788 rm -rf 2024

1789 history

Step1: Download Vmware workstation player using link

below:

https://customerconnect.vmware.com/en/downloads/detail

s?downloadGroup=WKST-PLAYER-

1700&amp;productId=1377&amp;rPId=97014

Step2: Download Jigsaw vm using link below:

https://drive.google.com/file/d/1wCa43KSpPlZSi22KVS3jcnTy

3iLEx\_6C/view

https://unextlearning-my.sharepoint.com/:u:/g/personal/krishna\_shetty\_u-next\_com/EafqCbLFx29Et8MPqKqNdlcBb6Q43MFslSkwDsMGbRLgZg?e=X82vhr

=========================================

Day 1

employees = [

{'empno': 1, 'name': 'Alice', 'department': 'HR', 'salary': 50000},

{'empno': 2, 'name': 'Bob', 'department': 'IT', 'salary': 60000},

{'empno': 3, 'name': 'Charlie', 'department': 'Finance', 'salary': 70000},

{'empno': 4, 'name': 'David', 'department': 'IT', 'salary': 5500},

{'empno': 5, 'name': 'Eva', 'department': 'Finance', 'salary': 75000}

]

From the employees dict , extract the name as a list

Print names using for loop

Add a new employee to the dict

Update the salary of Alice

Delete the employee whose salary is less than 60000

Get the jobwise count of employees → write a function

(select job, count(employees) from employee group by job)

7 . sort the employee names in descending order

8. Extract the employee who is/are earning high(max) salary

9. Create a new department dictionary and merge it employee (join) - explore

10. Find out any null or missing values in the records

#map

def multiply\_by\_two(x):

return x \* 2

numbers = [1, 2, 3]

doubled\_numbers = list(map(multiply\_by\_two, numbers))

#filter

numbers = [1, 2, 3, 4, 5]

evens = list(filter(lambda x: x % 2 == 0, numbers))

==========================================================================

Modules and file handing

import pandas as pd

# Sample employee dataset

data = {

'empno': [1, 2, 3, 4, 5],

'name': ['Alice', 'Bob', 'Charlie', 'David', 'Eva'],

'department': ['HR', 'IT', 'Finance', 'IT', 'Finance'],

'salary': [50000, 60000, 70000, 55000, 75000],

}

emp\_df = pd.DataFrame(data)

## save it as a employee.csv file

empno,name,department,salary

1,Alice,HR,50000

2,Bob,IT,60000

3,Charlie,Finance,70000

4,David,IT,55000

5,Eva,Finance,75000

employees\_df = pd.read\_csv(file\_path)

sorted\_employees\_df = employees\_df.sort\_values(by='salary', ascending=False)

employees\_df['bonus'] = employees\_df['salary'] \* 0.05

average\_salary\_by\_department = employees\_df.groupby('department')['salary'].mean()

import matplotlib.pyplot as plt

average\_salary\_by\_department.plot(kind='bar', color='skyblue', edgecolor='black')

plt.title('Average Salary by Department')

plt.xlabel('Department')

plt.ylabel('Average Salary')

plt.show()

##reduce

from functools import reduce

numbers = [1, 2, 3, 4, 5]

product = reduce(lambda x, y: x \* y, numbers)

##exceptions

try:

result = 10 / 0

except ZeroDivisionError:

print("Cannot divide by zero")

Python :

1. split the line into words

2. print only the even length words and odd length words

3. sort the list contains positive and negative integers

4. create function for palindrome

5. create recursive function for power of n

6. list append functions and arguments

7. check the password contains one string, one number and one special character

8. collections -->indexes [-3,-1]

9. binary sort

10. string functions

sql :

1. maxsal,min,avg sal of employee

2. employees who joined in 2005 --> doj

3. departmentwise , sum of salary

4. empname , managername

5. departmentwise , sum of salary less than 50000 using having

6. departmentwise , sum of salary less than 50000 using where

7. extract day, month and year from date

8. find the experience of the employee in months

9. print only top(n) salaried employees

10. print only 30% or 50 or 75% records of employee table

11. sales table: find out most purchased product by the customer

12. find the customer who purchased on multiple dates

13. get the distinct employee records without using distinct function ?

14. how to copy only the structure of the table to the new table;

15. filter out only the duplicate records ( table minus distinct)

16 floor(35.45) , round(15.35,-1) , ceil(56)

17. how to implement for loop and if then else statement in sql

18. substr( instr )

19. string concate

20. type conversion/cast