Scala Spark & Pyspark

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

Scala

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

val data = sc.textFile("file:///home/cloudera/revdata/file1.txt")

val gymdata = data.filter( x => x.contains("Gymnastics"))

gymdata.foreach(println)

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

Python

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

data = sc.textFile("file:///home/cloudera/revdata/file1.txt")

gymdata = data.filter( lambda x : 'Gymnastics' in x)

gymdata.foreach(print)

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

Scala

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

case class schema(txnno:String,txndate:String,custno:String,amount:String,category:String,product:String,city:String,state:String,spendby:String)

val mapsplit = gymdata.map( x => x.split(",") )

val schemardd = mapsplit.map( x => schema(x(0),x(1),x(2),x(3),x(4),x(5),x(6),x(7),x(8)))

val prodfilter = schemardd.filter( x => x.product.contains("Gymnastics"))

prodfilter.foreach(println)

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

Python

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

from collections import namedtuple

schema=namedtuple("schema",["txnno","txndate","custno","amount","category","product","city","state","spendby"])

mapsplit = gymdata.map( lambda x : x.split(","))

schemardd = mapsplit.map( lambda x : schema(x[0],x[1],x[2],x[3],x[4],x[5],x[6],x[7],x[8]))

prodfilter = schemardd.filter( lambda x : 'Gymnastics' in x.product)

prodfilter.foreach(print)

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

Scala

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

val schemadf = prodfilter.toDF()

schemadf.show()

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

Python

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

schemadf = prodfilter.toDF()

schemadf.show()

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

Scala

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

val file2 = sc.textFile("file:///home/cloudera/revdata/file2.txt")

val mapsplit1 = file2.map( x => x.split(","))

import org.apache.spark.sql.Row

val rowrdd = mapsplit1.map( x => Row(x(0),x(1),x(2),x(3),x(4),x(5),x(6),x(7),x(8)))

rowrdd.foreach(println)

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

Python

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

file2 = sc.textFile("file:///home/cloudera/revdata/file2.txt")

mapsplit1 = file2.map( lambda x : x.split(","))

from pyspark.sql import Row

rowrdd = mapsplit1.map( lambda x : Row(x[0],x[1],x[2],x[3],x[4],x[5],x[6],x[7],x[8]))

rowrdd.foreach(print)

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

Scala

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

import org.apache.spark.sql.types.\_

val schema = StructType(Array(

StructField("txnno",StringType,true),

StructField("txndate",StringType,true),

StructField("custno",StringType,true),

StructField("amount", StringType, true),

StructField("category", StringType, true),

StructField("product", StringType, true),

StructField("city", StringType, true),

StructField("state", StringType, true),

StructField("spendby", StringType, true)

))

val rowdf = spark.createDataFrame(rowrdd,schema)

rowdf.show()

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

Python

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

from pyspark.sql.types import \*

schema = StructType([ \

StructField("txnno",StringType(),True), \

StructField("txndate",StringType(),True), \

StructField("custno",StringType(),True), \

StructField("amount", StringType(), True), \

StructField("category", StringType(), True), \

StructField("product", StringType(), True), \

StructField("city", StringType(), True), \

StructField("state", StringType(), True), \

StructField("spendby", StringType(), True) \

])

rowdf = spark.createDataFrame(rowrdd,schema)

rowdf.show()

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

Scala

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

val csvdf = spark.read.format("csv").option("header","true").load("file:///home/cloudera/revdata/file3.txt")

csvdf.show()

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

Python

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

csvdf = spark.read.format("csv").option("header","true").load("file:///home/cloudera/revdata/file3.txt")

csvdf.show()

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

Scala

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

val jsondf = spark.read.format("json").load("file:///home/cloudera/revdata/file4.json")

val parqdf = spark.read.load("file:///home/cloudera/revdata/file5.parquet")

jsondf.show()

parqdf.show()

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

Python

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

jsondf = spark.read.format("json").load("file:///home/cloudera/revdata/file4.json")

parqdf = spark.read.load("file:///home/cloudera/revdata/file5.parquet")

jsondf.show()

parqdf.show()

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

Scala

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

val liscol = List("txnno","txndate","custno","amount","category","product","city","state","spendby")

val schemadf1 = schemadf.select(liscol.map(col):\_\*)

val rowdf1 = rowdf.select(liscol.map(col):\_\*)

val csvdf1 = csvdf.select(liscol.map(col):\_\*)

val parqdf1 = parqdf.select(liscol.map(col):\_\*)

val jsondf1 = jsondf.select(liscol.map(col):\_\*)

val uniondf =schemadf1.union(rowdf1).union(csvdf1).union(parqdf1).union(jsondf1)

uniondf.show()

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

Python

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

liscol = ["txnno","txndate","custno","amount","category","product","city","state","spendby"]

schemadf1 = schemadf.select(\*liscol)

rowdf1 = rowdf.select(\*liscol)

csvdf1 = csvdf.select(\*liscol)

parqdf1 = parqdf.select(\*liscol)

jsondf1 = jsondf.select(\*liscol)

uniondf =schemadf1.union(rowdf1).union(csvdf1).union(parqdf1).union(jsondf1)

uniondf.show()

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

Scala

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

import org.apache.spark.sql.functions.\_

val procdf = uniondf.withColumn("txndate",expr("split(txndate,'-')[2]")).withColumnRenamed("txndate","year")

val condf = procdf.withColumn("status",expr("case when spendby='cash' then 1 else 0 end"))

val fildata = condf.filter("txnno>50000")

fildata.show()

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

Python

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

from pyspark.sql.functions import \*

procdf = uniondf.withColumn("txndate",expr("split(txndate,'-')[2]")).withColumnRenamed("txndate","year")

condf = procdf.withColumn("status",expr("case when spendby='cash' then 1 else 0 end"))

fildata = condf.filter("txnno>50000")

fildata.show()

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

Scala

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

val aggdf = fildata.groupBy("category").agg(sum("amount").alias("total"))

aggdf.show()

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

Python

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

aggdf = fildata.groupBy("category").agg(sum("amount").alias("total"))

aggdf.show()

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

Scala

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

aggdf.coalesce(1).write.format("csv").mode("overwrite").save("file:///home/cloudera/scalaout")

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

Python

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

aggdf.coalesce(1).write.format("csv").mode("overwrite").save("file:///home/cloudera/pythonout")

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

Scala

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

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

Python: <https://www.youtube.com/watch?v=bZ6NL59FMoc>

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

MISTAKE CORRECTED Updated Code

Pyspark Handson

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

EMR hostname -- 13.233.215.178

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

Put in putty or Mobaxterm

Upload key pair

username - hadoop

Open pyspark shell -- typing "pyspark"

Execute below command

DONT DONT FORGET -- CLOSE PUTTY IMMEDIATELY after the execution

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

Pyspark Code

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

data = sc.textFile("s3://zeyohbuck/revdata/file1.txt")

gymdata = data.filter( lambda x : 'Gymnastics' in x)

from collections import namedtuple

schema=namedtuple("schema",["txnno","txndate","custno","amount","category","product","city","state","spendby"])

mapsplit = gymdata.map( lambda x : x.split(","))

schemardd = mapsplit.map( lambda x : schema(x[0],x[1],x[2],x[3],x[4],x[5],x[6],x[7],x[8]))

prodfilter = schemardd.filter( lambda x : 'Gymnastics' in x.product)

schemadf = prodfilter.toDF()

schemadf.show()

file2 = sc.textFile("s3://zeyohbuck/revdata/file2.txt")

mapsplit1 = file2.map( lambda x : x.split(","))

from pyspark.sql import Row

rowrdd = mapsplit1.map( lambda x : Row(x[0],x[1],x[2],x[3],x[4],x[5],x[6],x[7],x[8]))

from pyspark.sql.types import \*

schema = StructType([ \

StructField("txnno",StringType(),True), \

StructField("txndate",StringType(),True), \

StructField("custno",StringType(),True), \

StructField("amount", StringType(), True), \

StructField("category", StringType(), True), \

StructField("product", StringType(), True), \

StructField("city", StringType(), True), \

StructField("state", StringType(), True), \

StructField("spendby", StringType(), True) \

])

rowdf = spark.createDataFrame(rowrdd,schema)

csvdf = spark.read.format("csv").option("header","true").load("s3://zeyohbuck/revdata/file3.txt")

csvdf.show()

jsondf = spark.read.format("json").load("s3://zeyohbuck/revdata/file4.json")

parqdf = spark.read.load("s3://zeyohbuck/revdata/file5.parquet")

jsondf.show()

parqdf.show()

liscol = ["txnno","txndate","custno","amount","category","product","city","state","spendby"]

schemadf1 = schemadf.select(\*liscol)

rowdf1 = rowdf.select(\*liscol)

csvdf1 = csvdf.select(\*liscol)

parqdf1 = parqdf.select(\*liscol)

jsondf1 = jsondf.select(\*liscol)

uniondf =schemadf1.union(rowdf1).union(csvdf1).union(parqdf1).union(jsondf1)

uniondf.show()

from pyspark.sql.functions import \*

procdf = uniondf.withColumn("txndate",expr("split(txndate,'-')[2]")).withColumnRenamed("txndate","year")

condf = procdf.withColumn("status",expr("case when spendby='cash' then 1 else 0 end"))

fildata = condf.filter("txnno>50000")

fildata.show()

aggdf = fildata.groupBy("category").agg(sum("amount").alias("total"))

aggdf.show()

===========Close immediately=====

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

Spark

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

Proficient in developing and implementing Spark RDD-based data processing workflows using Scala, Java, or Python programming languages.

Experienced in optimizing Spark RDD performance by tuning various configuration settings, such as memory allocation, caching, and serialization.

Expertise in using Spark RDD transformations and actions to process large-scale structured and unstructured data sets, including filtering, mapping, reducing, grouping, and aggregating data.

Skilled in using Spark RDD persistency and caching mechanisms to reduce data processing overhead and improve query performance.

Familiarity with Spark RDD lineage and fault tolerance mechanisms and their impact on data processing reliability and performance.

Knowledge of Spark RDD optimization techniques, such as data partitioning, shuffle tuning, and pipelining, and their impact on query performance and resource utilization.

Strong understanding of Spark RDD integration with other big data technologies, such as Hadoop, Hive, and Kafka, and their impact on data processing workflows and performance.

Ability to troubleshoot common issues with Spark RDD, such as data processing errors, performance bottlenecks, and scalability limitations.

Experience working with Spark RDD in production environments and implementing performance monitoring and alerting systems to detect and resolve performance issues proactively.

Familiarity with Spark RDD-based data processing libraries and frameworks, such as Apache Spark SQL, MLlib, and GraphX, and their features and limitations.

Knowledge of Spark RDD best practices in data engineering and data science domains, such as data preprocessing, feature engineering, model training, and inference.

Proficient in developing and implementing Spark DataFrame-based data processing workflows using Scala, Java, or Python programming languages.

Experienced in optimizing Spark DataFrame performance by tuning various configuration settings, such as memory allocation, caching, and serialization.

Expertise in using Spark DataFrame transformations and actions to process large-scale structured and semi-structured data sets, including filtering, mapping, reducing, grouping, and aggregating data.

Skilled in using Spark DataFrame persistency and caching mechanisms to reduce data processing overhead and improve query performance.

Familiarity with Spark DataFrame schema and data type operations, such as adding, renaming, and dropping columns, casting data types, and handling null values.

Knowledge of Spark DataFrame optimization techniques, such as predicate pushdown, column pruning, and vectorized execution, and their impact on query performance and resource utilization.

Ability to troubleshoot common issues with Spark DataFrame, such as data processing errors, performance bottlenecks, and scalability limitations.

Experience working with Spark DataFrame in production environments and implementing performance monitoring and alerting systems to detect and resolve performance issues proactively.

Familiarity with Spark DataFrame-based data processing libraries and frameworks, such as Apache Spark SQL, MLlib, and GraphFrames, and their features and limitations.

Knowledge of Spark DataFrame best practices in data engineering and data science domains, such as data preprocessing, feature engineering, model training, and inference.

Familiarity with Spark DataFrame APIs and SQL syntax and ability to write complex SQL queries and DataFrame operations to solve business problems

Experienced in optimizing Spark SQL performance by tuning various configuration settings, such as memory allocation, caching, and serialization.

Expertise in using Spark SQL to process large-scale structured and semi-structured data sets, including querying, filtering, mapping, reducing, grouping, and aggregating data.

Skilled in using Spark SQL persistency and caching mechanisms to reduce data processing overhead and improve query performance.

Familiarity with Spark SQL schema and data type operations, such as creating, modifying, and dropping tables, views, and indexes, and handling null values.

Knowledge of Spark SQL optimization techniques, such as cost-based query optimization, column pruning, and predicate pushdown, and their impact on query performance and resource utilization.

Strong understanding of Spark SQL integration with other big data technologies, such as Hadoop, Hive, and Kafka, and their impact on data processing workflows and performance.

Ability to troubleshoot common issues with Spark SQL, such as data processing errors, performance bottlenecks, and scalability limitations.

Experience working with Spark SQL in production environments and implementing performance monitoring and alerting systems to detect and resolve performance issues proactively.

Proficient in processing serialized data in Spark using various formats, such as Avro, Parquet, ORC, and Protobuf, and their features and limitations.

Experienced in using Spark serialization libraries, such as Kryo and Java serialization, to optimize data serialization and deserialization performance.

Skilled in working with binary and textual data formats in Spark, such as CSV, JSON, and XML, and their serialization and deserialization using Spark DataFrames and RDDs.

Expertise in using Spark serialization and compression techniques, such as block-level compression, dictionary encoding, and off-heap storage, to reduce data storage and processing overhead.

Maintained and monitored Spark clusters on AWS EMR, ensuring high availability and fault tolerance.

Designed and developed batch processing data pipelines on Amazon EMR using Apache Spark, Python, and Scala to process terabytes of data in a cost-effective and scalable manner.

Designed and implemented data lake architectures on Amazon S3, leveraging partitioning and columnar formats such as Parquet to optimize query performance and minimize storage costs.

Optimized Spark jobs and data processing workflows for scalability, performance, and cost efficiency using techniques such as partitioning, compression, and caching

Designed and developed Spark applications to implement complex data transformations and aggregations for batch processing jobs, leveraging Spark SQL and DataFrames.

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

GCP POINTS

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

Proficient in managing and optimizing data storage solutions using Google Cloud Storage, ensuring efficient data organization, access, and security.

Experienced in deploying and managing data processing clusters with Google Dataproc, leveraging its scalability and automation features for large-scale data analysis.

Skilled in provisioning and configuring virtual machines on Google Compute Engine to meet specific computing requirements for various projects.

Expertise in developing and deploying serverless applications using Google Cloud Functions, enabling cost-effective and scalable solutions.

Familiarity with Google Cloud Storage buckets, object lifecycle policies, and access control mechanisms to ensure data availability and compliance.

Proficient in setting up and customizing Google Dataproc clusters, including cluster resizing and configuration tuning for optimal performance.

Hands-on experience with managing Google Compute Engine instances, including image creation, network configuration, and instance scaling.

Strong knowledge of Google Cloud Functions triggers and bindings for seamless integration with various event-driven workflows.

Implemented data replication and backup strategies using Google Cloud Storage to ensure data durability and disaster recovery.

Designed and executed data processing pipelines on Google Dataproc, incorporating tools like Spark and Hadoop for data transformation and analysis.

Utilized Google Compute Engine for high-performance computing tasks, such as machine learning model training and batch processing.

Developed serverless applications on Google Cloud Functions, leveraging event-driven architecture for real-time data processing and automation.

Proficient in Google Cloud Storage's versioning and object archiving features, ensuring data retention and compliance with data governance policies.

Demonstrated expertise in optimizing Google Dataproc jobs through cluster optimization, task scheduling, and efficient resource utilization.

Managed auto-scaling configurations on Google Compute Engine instances to adapt to fluctuating workloads and reduce operational costs.

Designed and deployed serverless APIs using Google Cloud Functions, enabling seamless integration with other cloud services and applications.

Implemented data encryption and access controls in Google Cloud Storage to ensure data security and privacy compliance.

Leveraged Google Dataproc's preemptible VMs and cost optimization strategies to reduce the overall data processing expenses.

Automated infrastructure provisioning and management on Google Compute Engine using Infrastructure as Code (IAC) tools like Terraform.

Developed serverless event-driven workflows on Google Cloud Functions, streamlining data processing and reducing infrastructure complexity.

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

Hive resume

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

Proficient in handling hive partitions and buckets with respect to the business requirement.

Experience in handling hive schema evolution with avro file format

Skilled in handling semi structured/serialised data processing using hive (AVRO,PAQUET,ORC)

Experienced in efficiently using Hive managed and external table with respect to the business requirement.

Proficient in creating and managing Hive tables, including managed, external, and partitioned tables.

Experienced in designing efficient data models using Hive tables and partitions to optimize query performance.

Expertise in querying Hive tables using SQL-like syntax and performing data analysis using tools like Apache Spark.

Skilled in integrating Hive tables with other big data technologies, such as Hadoop, HBase, and Impala.

Familiarity with Hive metastore and its role in managing table metadata and schema evolution.

Knowledge of Hive table formats, including ORC, Parquet, and Avro, and their advantages and disadvantages for different use cases.

Strong understanding of Hive table partitioning strategies, such as range, hash, and list partitioning, and their trade-offs in terms of query performance and data distribution.

Ability to troubleshoot common issues with Hive tables, such as data skew, table corruption, and query optimization.

Proficient in optimizing Hive query performance by tuning various configuration settings, such as memory allocation, parallelism, and compression.

Experienced in identifying and resolving performance bottlenecks in Hive, such as data skew, inefficient joins, and excessive shuffling.

Expertise in using Hive explain plans, query profiling, and metrics monitoring to diagnose query performance issues and optimize query execution.

Skilled in leveraging Hive partitioning, bucketing, indexing, and caching features to improve query performance and reduce data processing overhead.

Familiarity with Hive query optimization techniques, such as subquery unnesting, predicate pushdown, and vectorization, and their impact on query performance and resource utilization.

Knowledge of Hive query tuning best practices, such as minimizing data transfers, avoiding unnecessary data conversions, and using appropriate data formats and compression codecs.

Strong understanding of Hive integration with other big data technologies, such as Hadoop, Spark, and Impala, and their impact on query performance and resource utilization.

Ability to troubleshoot common issues with Hive performance, such as out-of-memory errors, query hangs, and slow query execution times.

Experience working with Hive in production environments and implementing performance monitoring and alerting systems to detect and resolve performance issues proactively.

Familiarity with Hive performance tuning tools, such as Hive Query Profiler, Hive Query Plan Visualization, and Hive Load Testing Tools, and their features and limitations.

Proficient in optimizing Hive query performance by tuning various configuration settings, such as memory allocation, parallelism, and compression.

Experienced in identifying and resolving performance bottlenecks in Hive, such as data skew, inefficient joins, and excessive shuffling.

Expertise in using Hive explain plans, query profiling, and metrics monitoring to diagnose query performance issues and optimize query execution.

Skilled in leveraging Hive partitioning, bucketing, indexing, and caching features to improve query performance and reduce data processing overhead.

Familiarity with Hive query optimization techniques, such as subquery unnesting, predicate pushdown, and vectorization, and their impact on query performance and resource utilization.

Knowledge of Hive query tuning best practices, such as minimizing data transfers, avoiding unnecessary data conversions, and using appropriate data formats and compression codecs.

Strong understanding of Hive integration with other big data technologies, such as Hadoop, Spark, and Impala, and their impact on query performance and resource utilization.

Ability to troubleshoot common issues with Hive performance, such as out-of-memory errors, query hangs, and slow query execution times.

Experience working with Hive in production environments and implementing performance monitoring and alerting systems to detect and resolve performance issues proactively.

Familiarity with Hive performance tuning tools, such as Hive Query Profiler, Hive Query Plan Visualization, and Hive Load Testing Tools, and their features and limitations.

Proficient in designing Avro schema for Hive tables and managing schema evolution to accommodate changes in data structure and format.

Experienced in handling schema compatibility issues in Hive, such as adding or removing fields, changing field types or names, and handling default values and nullability.

Expertise in using Avro tools and libraries, such as the Avro command-line interface, Avro IDL, and Avro schema resolution rules, to manage schema evolution in Hive.

Skilled in configuring Hive Avro serialization and deserialization settings, such as the schema registry URL, the schema file path, and the schema versioning strategy.

Familiarity with Hive Avro schema evolution limitations, such as the impact on data compatibility, performance, and storage, and the need for careful schema management in distributed environments.

Knowledge of Hive Avro schema evolution best practices, such as versioning schema files, using schema registries for centralized schema management, and testing schema changes in a staging environment before deployment.

Knowledge of Hive serialized data processing best practices, such as choosing appropriate serialization formats and codecs, optimizing data compression and encoding, and avoiding serialization overhead in data processing.

Strong understanding of Hive serialized data processing performance optimization techniques, such as using columnar storage, data partitioning, and indexing, and their trade-offs in terms of query performance and resource utilization.

Ability to troubleshoot common issues with Hive serialized data processing, such as data format errors, serialization and deserialization failures, and performance bottlenecks.

Experience working with other big data technologies, such as Hadoop, Spark, and Impala, and integrating serialized data processing workflows with other data processing and analytics tools.

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

Sqoop Points

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

Experienced in importing and exporting large datasets between Hadoop and relational databases using Sqoop.

Proficient in writing Sqoop commands to transfer data between Hadoop and various databases such as MySQL, and SQL Server.

Skilled in configuring Sqoop jobs for incremental data transfers using Sqoop's incremental import feature.

Proficient in performing data validation and cleansing during data transfer using Sqoop's validation and cleansing options.

Adept in scheduling and automating Sqoop jobs for incremental runs.

Strong experience in troubleshooting and resolving Sqoop job failures and performance issues.

Experienced in integrating Sqoop with other Hadoop ecosystem components such as Hive, HBase, and Spark.

Skilled in setting up secure connections between Hadoop and databases using Sqoop's security features.

Proficient in optimizing Sqoop imports and exports for performance and scalability.

Experienced in designing and implementing complex data integration solutions using Sqoop.

Designed and implemented end-to-end data integration solutions using Sqoop for large-scale data migrations from on-premise databases to Hadoop clusters.

Developed custom Sqoop connectors to support data transfers between Hadoop and proprietary data sources.

Implemented Sqoop-based data synchronization solutions to keep Hadoop data in sync with external databases.

Proficient in using Sqoop to import and export data in various file formats such as CSV, Avro, and Parquet.

Experienced in using Sqoop to import and export data from and to cloud-based data storage services such as Amazon S3

Skilled in optimizing Sqoop jobs for high throughput and low latency using tuning parameters such as batch size and number of mappers.

Proficient in using Sqoop to import and export data with complex schemas and data types.

Adept in setting up and configuring Sqoop on Hadoop clusters, including managing dependencies and configurations.

Strong experience in integrating Sqoop with data quality tools such as Trifacta and Talend.

Implemented automated data pipelines using Sqoop, Oozie, and other Hadoop ecosystem components for large-scale data processing.

Designed and developed custom Sqoop scripts to perform complex data transformations and manipulations during data transfer.

Implemented Sqoop-based solutions to load data from external databases into Hadoop clusters in real-time.

Skilled in configuring Sqoop to work with distributed databases and data warehouses such as Teradata and Vertica.

Experienced in troubleshootingb and resolving issues related to data format conversions, data type mappings, and data consistency during data transfer.

Proficient in using Sqoop to export Hadoop data to external databases for reporting and analytics purposes.

Adept in using Sqoop to perform data backup and disaster recovery operations on Hadoop clusters.

Strong experience in setting up and configuring Sqoop to work with Kerberos-based authentication and encryption.

Developed and implemented data governance policies and procedures using Sqoop to ensure data privacy, security, and compliance.

Proficient in using Sqoop to automate data migrations between Hadoop clusters in different geographical regions.

Skilled in setting up and managing Sqoop-based data replication and synchronization solutions for data backup and disaster recovery.

Developed Sqoop scripts to perform data transformations and data cleansing during data import from external databases into Hadoop clusters.

Proficient in configuring Sqoop to import and export data using custom SQL queries and stored procedures.

Experienced in troubleshooting and resolving performance issues related to network connectivity and resource allocation during Sqoop data transfer operations.

Adept in using Sqoop to load data into Hive tables for further processing and analysis.

Skilled in implementing Sqoop-based solutions for migrating data between different Hadoop distributions and versions.

Strong experience in configuring Sqoop to handle complex data structures such as nested and hierarchical data.

Proficient in using Sqoop to perform data validation and reconciliation checks to ensure data accuracy and completeness.

Experienced in setting up and configuring Sqoop-based data import and export solutions for large-scale data warehousing projects.

Developed and implemented Sqoop-based solutions to integrate external databases with Hadoop-based applications and workflows.

Skilled in using Sqoop to import and export data between Hadoop clusters and data lakes such as Amazon S3.

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

ETL Testing Points

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

Experienced in ETL (Extract, Transform, Load) testing methodologies and processes.

Proficient in testing data extraction processes from various sources, including databases, files, and APIs.

Skilled in validating and verifying data transformation rules and business logic applied during ETL processes.

Strong understanding of data warehouse concepts and testing data loading into data warehouse systems.

Knowledgeable about data integration and consolidation processes in ETL pipelines.

Familiarity with data quality and data cleansing techniques in ETL testing.

Expertise in designing and executing test cases for ETL processes to ensure data accuracy and completeness.

Proficient in SQL queries and scripting for data validation and verification during ETL testing.

Experienced in identifying and reporting data quality issues and data anomalies during ETL testing.

Skilled in performing regression testing on ETL processes to ensure backward compatibility and system stability.

Strong understanding of data mapping and data transformation rules documentation.

Knowledgeable about data profiling techniques to identify data patterns and anomalies in ETL processes.

Familiarity with ETL toolsets such as Informatica, Talend, DataStage, or SSIS.

Expertise in testing ETL workflows and job scheduling mechanisms.

Proficient in using data validation tools and ETL testing frameworks.

Experienced in testing data integration and synchronization between different systems using ETL processes.

Skilled in performance testing of ETL processes to ensure scalability and efficiency.

Strong understanding of data security and privacy considerations in ETL testing.

Knowledgeable about metadata management and testing metadata-driven ETL processes.

Familiarity with data lineage and impact analysis in ETL testing.

Expertise in testing error handling and exception handling mechanisms in ETL processes.

Proficient in conducting data reconciliation and data consistency checks in ETL testing.

Experienced in testing data extraction and loading from cloud-based platforms, such as Amazon S3 or Azure Data Lake.

Skilled in testing incremental data loading and change data capture mechanisms in ETL processes.

Strong understanding of data archiving and data retention strategies in ETL testing.

Knowledgeable about data encryption and data masking techniques in ETL testing.

Familiarity with testing real-time data integration and streaming ETL processes.

Expertise in testing ETL transformations, such as data aggregation, filtering, sorting, and joining.

Proficient in identifying and resolving performance bottlenecks in ETL processes.

Experienced in collaborating with development teams to analyze and troubleshoot ETL issues.

Skilled in test data management and test environment setup for ETL testing.

Strong understanding of data governance and regulatory compliance requirements in ETL testing.

Knowledgeable about testing ETL processes in Big Data platforms, such as Hadoop or Spark.

Familiarity with testing data replication and synchronization in ETL processes.

Expertise in testing ETL metadata repositories and data catalogs.

Proficient in documenting test plans, test cases, and test results for ETL testing.

Experienced in conducting data validation and reconciliation between source and target systems in ETL testing.

Skilled in conducting data migration testing and data conversion validation in ETL processes.

Strong understanding of ETL performance tuning techniques for optimization.

Knowledgeable about testing data extraction and loading from different database systems, such as Oracle, SQL Server, or MySQL.

Familiarity with testing ETL processes in data virtualization environments.

Expertise in testing data warehousing concepts, such as dimensional modeling and star schemas.

Proficient in testing ETL data lineage and data traceability.

Experienced in testing ETL metadata management tools and data dictionaries.

Skilled in testing data synchronization and replication across distributed systems.

Strong understanding of data deduplication and data consolidation techniques in ETL testing.

Knowledgeable about testing data migration from legacy systems to modern platforms using ETL processes.

Familiarity with testing ETL processes in real-time analytics and reporting systems.

Expertise in testing data transformation rules and business logic applied during ETL processes.

Proficient in using ETL testing tools and frameworks, such as QuerySurge, Talend Data Quality, or Informatica Data Validation Option.

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

Product Company List

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

Zoho Corporation

Freshworks

InMobi

Practo

Razorpay

BrowserStack

Druva

Helpshift

Paytm Money

Zenoti

Urban Ladder

PharmEasy

Chargebee

Postman

CleverTap

Khatabook

Unacademy

Udaan

Toppr

BlackBuck

UrbanClap

Bharti Airtel

Reliance Jio

Vodafone Idea

Tata Communications

BSNL (Bharat Sanchar Nigam Limited)

Sterlite Technologies

Tejas Networks

Subex

HFCL (Himachal Futuristic Communications Limited)

ITI Limited (Indian Telephone Industries Limited)

Matrix Comsec

Radisys India Pvt. Ltd.

Vihaan Networks Limited (VNL)

Lavelle Networks

Ceragon Networks

Mahindra Comviva

Nexge Technologies

Sterlite Power

Infinera India Pvt. Ltd.

Elitecore Technologies

Indus Towers

Viptela (now part of Cisco)

Nokia Networks India

Ericsson India Pvt. Ltd.

Huawei Technologies India Pvt. Ltd.

ZTE Corporation India

Motorola Solutions India Pvt. Ltd.

Qualcomm India Pvt. Ltd.

Intel India

Samsung Electronics India

MindTickle

Innovaccer

Icertis

Lenskart

Nykaa

CarDekho

Grofers

Licious

Hike

Zeta

PhonePe

Byju's

Lendingkart

PolicyBazaar

Dream11

Swiggy

Meesho

NoBroker

Zomato

InCred

Zerodha

Cred

Ola Electric

Quick Heal Technologies

Rupeek

Innov8

Ixigo

BigBasket

Rapido

Drivezy

WhiteHat Jr

Oyo Rooms

Rebel Foods

HealthifyMe

Delhivery

BYJU'S

Dailyhunt

Acko General Insurance

Treebo Hotels

Cars24

Milkbasket

Inshorts

Hector Beverages

Cashify

Bounce

Shuttl

Testbook

BoAt Lifestyle

Snapdeal

Jumbotail

Moglix

MagicPin

The Man Company

Little Black Book (LBB)

Droom

KredX

Juspay

Belong.co

Porter

Yulu

Spinny

Flipkart

Ola

Paytm

MakeMyTrip

OYO Rooms

Myntra

BookMyShow

Quikr

Rebel Foods (Faasos)

Dunzo

Cure.fit

Rivigo

Portea Medical

Simpl

Hector Beverages (Paper Boat)

Zilingo

Pine Labs

CaratLane

Lensico

CRED

Bira 91

Livspace

Blackbuck

ShareChat

BigRock

Testpress

Xoxoday

Haptik

Coverfox

Zoomcar

Paper Boat

OYO Workspaces

CarWale

Rebel Foods (Behrouz Biryani)