860 263 0660 // SamirZada@gmail.com

(1) I'm a **Data Engineer** with 7 years experience, during my carrier I've been
- ingesting data from many sources
- used different processing tools
- also store my data with a large variety of databases.

(2) I'm currently working as **Data engineer** at The Hartford, hardfort Connecticut.
We Ingesting Data from the Company rest API and store them in relational database. Also we use AWS S3 as datalake, We have 52 data sources uploading data daily and weekly, then we transform and clean data with Spark Databricks and store data in Redshift.

Prior to that,

(3) I worked as **AWS Cloud Data engineer** at Progressive Corporation, Mayfield Ohio.
We were ***Migrating Data*** from a local data warehouse to ***AWS S3 using AWS DMS*** also worked on pipelines using ***AWS S3, EMR spark, Lambda functions and Redshift as Data-warehouse.***

All this years, I have worked in many other projects, with different teams and different management styles and I believe with my experience, I will be a good fit for this position and will add some value to your project if you were to give me the opportunity to be part of your team.

DATABRICKS

Databricks is a tool based on cloud for data engineering were you can create clusters in few seconds.
It’s Based on Zeplin notebook.
Its very similar as **Jupiter Notebooks**.
In **Databricks** you create **notebooks** where **spark** jobs are running.
Each notebook can be run independently.
At The Hartford, I used Databricks platform for data processing using pyspark.
My spark job essentially transformed created RDD into dataframes with specified schema across the cluster and the results are eventually saved in Redshift or S3 buckets.
Spark supports different languages such as **Scala**, **Python** and **R**, **Databricks** does as well since it actually runs Spark in a nice graphical user interface where we can easily access the DAG for spark optimization, select the language, write queries using pyspark.sql library, display the tables.
Databricks is an analytic platform for engineering, data science to process large set of data in a distributed and fault tolerant manner.

Map Reduce Stages

Input => slpit(byLine) => Map => Sort/Shuffle => Reduce

Hadoop components

Map Reduce = Data Processing

Yarn = Cluster Resource Management

HDFS = Hadoop Distributed File System (Storage)

Hadoop Ecosystem

MapReduce, Yarn, HDFS, Hbase, Hive, Pig, oozie, Sqoop, Flume, Zookeeper, Ambari.

SPARK BIG FILE

Parquet file, you can split the file by column and process each column.

JSON File, you can split by using a separator, or by keys, and process it.

Text file, you can split by lines and process it.

CSV file, you can split line by line, they are the rows or you can slit by column.

SPARK ########### SPARK

Open source distributed computing framework - based on parallel processing
Spark follows a Master / slave architecture

In your Master node, you have the driver program witch drives your application.
The code you are writing behaves as a driver program or if you are using the shell, the shell act as driver program.
Inside the driver , the first thing you do is to create a Spark Context.
The Spark Context is your gateway to all Spark functionalities, anything you do on Spark goes through Spark Context.
Now this Spark Context works with Cluster Manager to manage various jobs.
The Driver program and Spark Context takes care of the job execution in the cluster.
A job is split into multiple tasks witch are redistributed across nodes.
Worker nodes are the Slaves nodes, their jobs is to execute the Tasks.

To resume, we have a Master node with a Driver inside of it and a Spark Context inside of the Driver. The Spark Context takes the job, break it into Tasks and distribute them to the Worker Nodes.

Datasets are a collection of data of strongly typed JVM objects.
Dataset is a data structure in SparkSQL wich is strongly typed and is a map to a relational schema. It represents structured queries with encoders. Its an extension to Data-frame API.
Spark dataset provides both type safety and object-oriented programming interface.

Dataframes, data are organized in columns with column name and types info. We can say data in dataframe are the same as the table in relational database.

RDD Resisilent Distributed Datasets. Its an immutable distributed collection of objects. Each dataset in RDD is divided into logical partitions, wich may be computed on different nodes of the cluster. An RDD is read-only, partitioned collection of records.

Dataset VS DataFrame:
Dataframe is columnar, it's like a table in Relational database.
Dataset are extension of dataframe API, they are Type-safe, strongly typed.

Dataset faster than DataFrame ? Data-frame is faster.
Dataframe is more expressive and more efficient (Catalyst Optimizer), untyped can lead to runtime errors.
Datasets look like dataframe but it's typed. So you will have compile time errors instead of run-time

Narrow transformation: map(), flatMap(), filter()

map() VS flatMap() are used for transformations. The map() transformation takes in a function and applies it to each element in the RDD and the result of the function is a new value of each element in the resulting RDD. The flatMap() is used to produce multiple output elements for each input element.

Map() apply a function to each element in a collections
FlatMap() flattens a multidimensional collection into one dimension collection of element and then applies a map() to that collection.

wide transformation: Distinct(), reduceByKey(), groupByKey(), Join(), Repartition(), Coalesce()

groupByKey() can cause out of disk problems as data is sent over the network and collected on the reduce workers. Data is combined at each partition, only one output for one key at each partition to send over network. reduceByKey() required combining all your values into another value with the exact same type.

actions: collect(), count(), foreach(func), countByKey(), saveAsTextFile(path) reduce() fold()

reduce() takes the two elements as input from the RDD and then produces the output of the same type as that of the input elements.

fold() it's a wide operation, aggregate the elements of each partition, and then the results for all the partitions, using a given associative function and a neutral "zero value"

reduce() VS fold(), reduce() throw a an exception for empty collection, fold() is defined for empty collection.

lazy evaluation in spark means that the execution will not start until an action is called.

Inner join: creates a new result table by combining column values of two tables. The query compares each row of Table1 with each row of table 2 to find all pairs of rows which satisfy the join-predicate.

Left outer join: returns all the values from an inner join plus all values in the Left table that do not match to the right table, including rows with NULL values in the link column.

Right outer join: returns all the values from an inner join plus all values in the Right table that do not match to the right table, including rows with NULL values in the link column.

Full outer join: returns all records when there is a match in left table or right table records. Can return a very large result-sets! Full outer join and full join are the same.

Cross Join: it's used to generate a paired combination of each row of the first table with each row of the second table. This join type is also known as cartesian join. The Idea of Cross Join is to returns the Cartesian product of the joined tables. Say you have a 'x' row in table1 and 'y' in another, this would give (x*y) rows.

Broadcast join: spark can "broadcast" a small Data-frame by sending all the data in that small Data-frame to all nodes in the cluster. After that spark can perform a join without shuffling any of the data in the large Data-frame.

Sort-MergeJoin is composed of 2 steps, first is to sort the datasets and second is to merge the sorted data in the partition by iterating over the elements and according to the key join the rows having the same values.

Accumulators are variables that are used for aggregating information across the executors. For example to see how many records are corrupted or how many times an API is called.

Broadcast variable are read-only Accumulators are read and write.

Checkpoint: check-pointing is actually a feature of Spark Core that allows a driver to be restarted on failure with previously computed state of distributed computation described as an RDD.

RDD Lineage: Is a graph of all the parent RDDs of an RDD. It's built as a consequence of doing transformations to the RDD and creates a logical execution plan.

Cache() VS Persist(): cache() will cache the RDD into memory, Persist(level) can cache in memory, on disk or off-heap memory according to the caching strategy specified by level. Persist() without argument is the same as cache().
persist(StorageLevel.MEMORY_ONLY) // MEMORY_ONLY_2, MEMORY_AND_DISK, MEMORY_AND_DISK_2, MEMORY_ONLY_SER

Key Salting is a technique where we will add random values to the join key of one of the tables. In the other table, we need to replicate the rows to match the random keys. The ideas is if the join condition is satisfied by key1 **==** key1, It should get satisfied by key1_<salt> **==** key1_<salt>

DAG is a scheduling layer of Spark that implements stage-oriented scheduling.
Vertices are the RDD and edges are the transformations that will be applied to this RDD's.
It's the Logical Execution Plan that will be converted into Physical plan for the the tasks to be executed in the worker nodes.
CLUSTER is a group of JVMs (nodes) connected by the network, each of which runs Spark, either in Driver or Worker roles.
EXECUTORS are JVMs that run on worker nodes.This are the JVMs that actually run tasks on data partitions.
JOB is a sequence of stages, triggered by an Action such as .count()
STAGE is a wild transformation.
TASK lowest unit of work. - - is a single operation such as .map() narrow transformation.
All jobs are sent to the executors.

Spark Hardware

- RAM per core
- core count / size
- disk size / type
- network speed / topology
- limitations of your data lake

Architecture: Master - WorkerNode: Master contains the driver (inside the driver, spark context) --> Cluster manager --> Worker Node (Executor)

Project Tungsten --> Improve Spark's efficiency in memory and CPU
Catalyst optimizer --> Improve the performance of queries in Spark SQL

Spark tuning - Performance

- Broadcast variables - immutable and read only - caching the small tables in the worker node for join purposes
- Accumulators - immutable and write only
- Check the DAG - and see if we can possibly change some stages to states
- In case of skewness : repartition and coalesce
- Dynamic resource allocation: you can that in Spark config. When an executor is down, another executor will ensure the continuity of the job

Spark uses 60% of CPU and memory resource
40% Heap memory and resource manager
You can configure that using Spark Conf.

Cache: memory only
Persist: memory only, disk only, memory and disk, serialized

Spark streaming types: - DStream (RDD), - structured streaming (Dataframes)
Spark serialization: 2 types: marshall (for more datasets), pickle (some classes)
Kryo is internally used by Spark from spark 2.0.0

## Partition by key
Table1.partitionby("ID")
Table2.partitionby("ID")
Table1.join(Table2, on={"ID"}, how= "inner")

### SPARK SUBMIT OPTIONS ####

Class = for Java and Scala applications, the fully qualified classname of the class containing the main method of the application.

Master = The location to run the application.

Conf = Spark configuration property in *key=value* format. For values that contain spaces, surround "*key=value*" with quotes (as shown). EX .....

Deploy-mode = Deployment mode: cluster and client. In cluster mode, the driver runs on worker hosts. In client mode, the driver runs locally as an external client. Use cluster mode with production jobs; client mode is more appropriate for interactive and debugging uses, where you want to see your application output immediately.

Driver-cores = Number of cores used by the driver in cluster mode. Default = 1

Driver-memory = Maximum heap size (represented as a JVM string; for example 1024m, 2g, and so on) to allocate to the driver. Alternatively, you can use the spark.driver.memory property.

Jars = Additional JARs to be loaded in the classpath of drivers and executors in cluster mode or in the executor classpath in client mode. For the client deployment mode, the path must point to a local file. For the cluster deployment mode, the path can be either a local file or a URL globally visible inside your cluster;

Packages = Comma-separated list of Maven coordinates of JARs to include on the driver and executor classpaths. The local Maven, Maven central, and remote repositories specified in repositories are searched in that order.

Client mode VS **Cluster mode**:
In Cluster mode, the Spark driver runs inside an application master process which is managed by YARN on the cluster, and the client can go away after initiating the application.
In Client mode, the driver runs in the client process, and the application master is only used for requesting resources from YARN.
To resume: client mode, the master node is your local machine and cluster you access the master node remotely. (ssh)

PYTHON GENERATOR
Generator => Create Iterator
purpose => code efficiency
to create a generator you use yield() // for x in mylist: yield(x) //
print(next(functionName))

SCALA
scala closures are functions witch uses one or more Free variables and the return value of this function is dependent of this variables.

Free variables are defined outside of the closure function and is not included as parameters of this function. They are not bound to a function with a valid value.

Traits are used to share interfaces and fields between Classes. Classes and Object can extend Traits, but a traits cannot be instantiate and can't have parameters.

Higher order functions take other functions as parameters or return a function as a result. This is possibles because functions are first-class values in Scala.

Currying transform a function that takes multiple parameter into a chain of functions, each taking a single parameter. Curried functions are defined with multiple parameter lists, as follows: **def curry(s1: String)(s2: String) = s1 + s2**

A monad is an object tat wraps another object in Scala

The **Tail recursion** is basically using the recursive functions as the last statement of the function. So when nothing is left to do after coming back from the recursive call, that is called tail recursive.

Scala Implicits allow you to omit calling methods or referencing variables directly bun instead rely on the compiler to make the connections for you.
For example: you could write a function to convert from Interger to String and rarher than call that function explicitly, you can let the compiler do it for you.
Def multiply(implicit by: Int) = value * by
implicit val multiplier = 2

Mock vs Spy object:
mock you are creating a complete fake object
spy, you have real objects and you just spying or stubbing specifics method of it.

Companion object is when you have a file and the class with the same name. The object is the companion of the class.
Ex: **samir.scala**, samir object is the companion of samir class
class Samir {} object Samir {}

Scala Testing framework: ScalaTest, Junit.

Compile tool: SBT, Meaven, Ant

```
\>scalac Demo.scala // compile first
\>scala Demo // then run the command

object Demo {
   def main(args: Array[String]) {
      val str1:String = "Hello, "
      val str2:String = "Scala!"

      println( "str1 + str2 = " +  strcat(str1)(str2) )
   }

   def strcat(s1: String)(s2: String) = {
      s1 + s2
   }
}
```

CDC

Change data capture is a process that captures changes made in a database, and ensures that those changes are replicated to a destination such as a data warehouse.

DATA MASKING

Data masking is the process of replacing sensitive information copied from production database to test non-production databases with realistic, but scrubbed, data based on masking rules.

PII // HIPAA/ PCI

Sensitive data

PII (pii = personally identifiable information) is defined as information that identifies a person (name, address, ssn, phone, email)

HIPAA (hipaa = Health Insurance Portability and Accountability) is a law to protect sensitive patient health information from being disclosed without the patient’s consent or knowledge.

PCI (pci = Payment Card Industry)

AES 256

Advanced Encryption Standard (AES) 128 bits or 256 bits

Docker

Docker is container manager.
Container is a light way version of a computer that isolate applications and the platform is independent of the system that is running in.
Image is a blueprint of the container.

At Progressive Corp, we used Docker to run containers, we have an image of the pipeline similar of AWS pipeline, hdfs, spark, postgressSQL (redshift).

MongoDB
In MongoDB stores data in documents and all the documents are stored in the collection.

CDC in mongoDB:
- Using Timestamp column:
The most simple way to execute a MongoDB change data capture is to have a timestamp column in collections that chnages with insert and update operatins. The object_id is representative of the time at which the row was created.
- Using Change Stream:
Change streams work by listening to the operational log, that contains all information related to writes in the database on a storage level. Change stream only work if replica are enable in MongoDB. A replica set is a second instance that helps application maintain high availability in case of failures. Replication is accomplished throught operation log and changes streams rely on the same operation log.

ALL Operator VS IN Operator
ALL operator retrieves all the documents which contains the subset of the values we pass. The subset might be in any order.
IN operator retrieves all the documents which contains the either of the values we pass.

SAVE VS INSERT
SAVE can insert or update a document. Look if it exist, if yes update, if not insert.
INSERT only perform the insertion.

3V
volume, **variety**, **velocity**

Volume refers to the amount of data, Am I doing Big data, do I need big data tools?
Variety refers to the number of type of data, what ingestion tool? Unstructured, Semi-structured, structured data. Schema, database??
Velocity refers to the speed of the data processing, real-time, batch, streaming? How many time per day or minutes my data are coming?

DELTA LAKE

it will create a delta_log folder with the log in Json format 0000000000.json // 00000001.json and crate files with the data in snappy compression and parquet format

```
spark.read.format("delta").save("/temp/folder1")
spark.read.format("delta").load("/temp/folder1")
dt = DeltaTable.forPath(spark, "/temp/folder1")
dt.toDF.show()
dt.delete("columnName == 'row name to delete' ") // dt.delete("FName == 'samir' ")
```

dt.toDF.show() will show you the updated delta table without loading again.
dt.updateExpr(“Fname == ‘samir’, map(“Age” → “Age + 5”) ”)
dt,toDF.show() delta tables are autorefreshed. You can see the new changes made.

```
dt.as(“aliasName”)
        .merge(df1.as(“inputs”), “aliasName.FName = inputs.FName” )
        .whenMatched()
        .updateExpr(
                Map(
                        “LName” → “inputs.LName”,
                        “Phone” → “inputs.Phone”,
                        “Age” → “inputs.Age”
                        )
                )
        .whenNotMatched
        .insertAll
        .execute();
dt.history.show(false)
spark.read.format(“delta”)
        .option(“versionAsOf”, 0).load(“/temp/aliasName”).show()
spark.read.format(“delta”)
        .option(“timestampAsOf”, “2020-08-22 00:57:00”)
        .load(“/temp/aliasName”)
        .show()
```

SQL Functions

rank (skip the duplicated rank)
dense rank (don’t skip the duplicated rank)
SELECT subjects, names, prices, RANK() over (partition by subjects order by prices DESC) my_rank from Products;

lag and **lead** compare rows and add a new column in your result (query).
Then you can analyze this 2 column and know the diff between them.
Used id for dates.
Ex: when the customer paid his bills and when this bill was due. 10 days for exemple.

WHERE VS HAVING
WHERE is used to filter the records from the table based on the specialized condition.
HAVING is used to filter record from the groups based on the specified condition.
We can have WHERE before a HAVING but never After.

SQL VS NoSQL
SQL are relational database (RDBMS), table-based.
NoSQL is non-relational, document-based, wide-column stores or key-value pairs.

Data lakehouse

The data lakehouse is an emerging new data repository structure that combines the benefits of both the data warehouse and the data lake. The data lakehouse will allow BI users and data scientists to work on the same sources. It will also make it easier for organizations to implement data governance policies.

KAFKA ########### KAFKA #################
kafka is a good ingestion tool, it's fault tolerant.
Kafka is a Distributed Messaging System
4 APIs: Producer / Consumer / Connector / Stream Processor

Five cores of Kafka
- Producer API: allows applications to send streams of data to topics in the Kafka cluster
- Consumer API:allows applications to read streams of data from topics in the Kafka cluster
- Streams API: allows transforming streams of data from input topics to output topics
- Connect API: allows implementing connectors that continually pull from some source system or application into Kafka or push from Kafka into some sink system or application
- AdmiClient API: allows managing and inspecting topics, brokers, and other Kafka objects

Components: Broker / Topic / Producer / Consumer / Zookeeper
The Producer, push the message to the Broker, inside the broker we have the Topic where the message is stored. The Consumer will pull the message from the Topic.
Zookeeper is used for managing and coordinating Kafka Broker also used to notify producer and consumer for failures and keep track of the offsets.

- broker offset: every time when the consumer asks for message, and the transaction is complete, it generates the broker offset
- consumer offset: help to identify where to restart when the consumer goes down.

Messaging system - ingestion tool -
Consumer group

Kafka Connect is a tool for scalably and reliably streaming data between Apache Kafka and other data systems. It makes it simple to quickly define connectors that move large data sets into and out of Kafka.

6 steps for data quality:
- completeness (number of records sent = number of records received)
- uniqueness (distinct data - no redundancy)
- timeliness (precision in time for logs for instance)
- validation (a string coming as a string, order of the schema, i.e make sure that the metadata is correct)
- accuracy (how well the info reflect reality) - make sure the source is producing the correct data, ex: outside the possible range)
- consistency (info stored in one place = info stored in a different place)
(no null values, no special character)
Streaming and batch data -

Messages order;
Look for partitions, check topic logs, by default FIFO
To solve change in order of the topic generated by the producer —> Each consumer has its own broker

Boto 3 module is used in Python to connect to AWS
Kafka Consumer - S3 - Lambda (Spark)- Redshift

Topic (boto3) —> S3 (Name, Region, Versioning) —> Lambda (Name, pass permission, pass path) —> Databases
SNS: use SNS in this above pipeline and it will send the info

FLUME ########### FLUME

Apache Flume is a data ingestion tool for collecting, aggregating and transporting large amounts of streaming data such as log data, events from various sources to a centralized data store.

Components of Agent: source / channel / sink

```
TwitterAgent.sources = Twitter
TwitterAgent.channels = MemChannel
TwitterAgent.sinks = AVRO

# Describing/Configuring the source
TwitterAgent.sources.Twitter.type=org.apache.flume.source.twitter.TwitterSource
TwitterAgent.sources.Twitter.consumerKey=qGYCt6WQypxgfGZn5f2yvo9ZE
TwitterAgent.sources.Twitter.consumerSecret=IsxCfnLjvST8WRhsSsFZU
TwitterAgent.sources.Twitter.accessToken=12812503522019573
TwitterAgent.sources.Twitter.accessTokenSecret=XVcL102M4L4knXtkJOelh
TwitterAgent.sources.Twitter.keywords= miami

# Describing/Configuring the Channel
TwitterAgent.channels.MemChannel.type=memory
TwitterAgent.channels.MemChannel.capacity=10000
TwitterAgent.channels.MemChannel.transactionCapacity=1000

# Describing/Configuring the sink
TwitterAgent.sinks.AVRO.type = avro
TwitterAgent.sinks.AVRO.hostname = localhost
TwitterAgent.sinks.AVRO.port = 6669

# Bind the source and sink to the channel
TwitterAgent.sources.Twitter.channels = MemChannel
TwitterAgent.sinks.AVRO.channel = MemChannel
```

FLUME VS KAFKA

Kafka can support data streams for multiple applications, whereas FLUME is specific for hadoop and big data analysis. Kafka can process and monitor data in distributed system whereas FLUME gathers data from distributed systems to land data on a centralized data store.

########################## INTERV PREP ##########################
Big data is a problem and Spark and Hadoop are solution
Pipeline is a set of data platform s
Bunch of technology and main contribution: consuming from rest API different files, from relational databases
SFTP (Secure File Transfer Protocol)

Data layers

Raw layer --> Datalake (S3, HDFS)
DSL (Data Service layer) --> csv to Hive
ASL (Application Service layer) --> Partitioning
Consumption layer - Tableau,
Datamart subset of data warehouse
Data lake is a collection of data from various sources

Agile

Agile core is SCRUM
2-3 weeks sprint
Scrum master coordinate with all teams
Backlog is the list of tasks in a project.
4 Scrum ceremonies =
- Sprint Backlog Refinement. (Product Backlog refinement is the act of adding detail, estimates, and order to items in the Product Backlog)
- Sprint planning (beginning of the sprint)
- Daily stand up (15 min every day)
- Spring review (end of sprint 30-60 min)
- Spring Retrospective (End of Iteration 60 min, give feedback, debriefing, what went well, what went bad, when the next sprint will start...)

Jira - user story (different tasks) --> assigned to each developer as a task --> Testing team
to-do / In progress / code review / done
task or ticket can have point depending of the complexity of the task. 1-8
8 points usually will take you a week.

SDLC

SDLC: System Deployment Life Cycle
Planning - Analysis - Design - Implementation - Maintenance

SOAP: Simple Object Access Protocol

It is a communication protocol design to communicate via internet - it is the xml way of defining what information is sent and how
SOAP API - is a service API like Ambari API -

HIVE

Hive is an ETL data warehouse tool on top of Hadoop Ecosystem and used for processing structured and semi-structured data.

UI → Driver → Compiler → Metastore → Execution Engine

- Change the **execution engine** for better execution plan. In traditional query, it does the load before filtering, in tez, filter is first.
SET hive.execution.engine=mr; // **SET hive.execution.engine=tez**;
Once the query is parsed, a logical query is generated, for use by the query execution engine. Hive performs both logical and physical optimization, including partition pruning, projection pruning and predicate pushdown.

- Partitioning (Static (Load data in tables) and dynamic(insert data into tables) hive.partition)
Ex Partitioning on date column
- Bucketing - Create ranges for id for instance
- Use joins (map side(only map, no reduce), sort merge bucketing())
- Change file format to ORC ()
- Sort by (uses multiple reducers) instead of order by (uses only one reducer)
- Cluster by is combination of distributed by and sort by

ORC is better than parquet if the data is not nested

Update hive table is not possible in version before 3.0,
In Hive 3.0 ACID transaction are enabled so you can do update using UPSERS
You can use Staging table then track by date diff or key diff

Beeline is CLI for Hive

Avro, Parquet, ORC

Avro (row-based, schema evolution; has header (metadata) and body (binary format)) --> for raw records, it has a good compression. It is divided by stripe and each stripe is 250MB

Parquet - column based, it is the default format for Spark --> Perfect for nested data, and has better compression that Avro. Ex: Multiple phones in phone number field

ORC (ACID transaction) --> fast query; has by default the statics available in footer (min, max, avg, count) — better compression than parquet to my opinion.

ACID Transaction:
In database systems, **ACID** (Atomicity, Consistency, Isolation, Durability) refers to a standard set of properties that guarantee database transactions are processed reliably.

Atomicity - All or nothing
Consistency -
Isolation - Independent and not visible to the receiver
Durability - Committed data

########### OLTP / OLAP ##############
On**L**ine **T**ransaction **P**rocessing captures, stores, and processes data from transactions in real-time. (MySQL, PostgressSQL, Oracle)

On**L**ine **A**nalytical **P**rocessing uses complex queries to analyze aggregated historical data from OLTP systems. (IBM Cognos, Apache Kylin, Oracle OBIEE) save in data-warehouse.

########## NORMALIZATION / DENORMALIZATION #############

Normalization is the technique of dividing the data into multiple tables to reduce data redundancy and inconsistency and to achieve data integrity.

Better for databases with Primary keys and Foreign ID
2 tables ex:
T1 table with Employee information. Col Department Name or ID (Foreign ID)
T2 table about Department info ID or name in Table 1

Denormalization is the technique of combining the data into a single table to make data retrieval faster.

Better for data ware house because tables have no relations.
1 table with all information, more column. And redundancy information.

############## DATA VALIDATION ########################
Data validation means checking the accuracy and quality of source data before using, importing or processing.
Different types of validations can be performed depending on the destination constrains or objectives.
Data validation is a form of data cleansing.

Cross-system consistency checks:
Compares data in different systems to ensure it is consistent. Systems may represent the same data differently, in which case comparison requires transformation
ex: one system may store customer's name like 'Doe, john' || First_Name: 'Doe', Last_Name: 'Doe'

File existence check:
Checks that a file with a specified name exists. This check is essential for programs that use file handling.

Format Check:
Check that the data is in a specified format
ex: dates have to be in the format YYYY-MM-DD. Regular expressions may be used for this kind of validation.

Presence Check:
Checks that the data is present
ex: customers may be required to have an email address.

SCD

slow changing dimension

when you add new record
type 1 create new row with new record
type 2 create a column with a flag ex: Old / New in front of each row or the Date updated
type 3 Create c column with new record ex: old salary // new salary
type 2 preferred.

Difference Database, Data warehouse, Data lake
- database - OLTP
- data warehouse - OLAP (aggregation of OLTP)
- Data lake - Central repository

AIRFLOW ######### AIRFLOW

Airflow is a platform to schedule and monitor workflow.
airflow need Web Server / Scheduler / Executor
Web server run the UI
Sheduler run the Dags
Executors run The task of this same dag.

Airflow can be a cluster, it will use many Executor called worked nodes.

to do parallel task you need to change the database, by default is using SQLlite and its allow only one request at the time. Mysql or PostgreSQL will allow many request at the same time so the executor can ask the metadata in this database and do task in parallel.

Sequential Executor for single node (default)
Celery for cluster
Local executor for testing purposes

Custom Operator: create a file **hello_operator.py**

```
from airflow.models.baseoperator import BaseOperator
from airflow.utils.decorators import apply_defaults
class HelloOperator(BaseOperator):
    @apply_defaults
    def __init__(
            self,
            name: str,
            *args, **kwargs) -> None:
        super().__init__(*args, **kwargs)
        self.name = name
    def execute(self, context):
        message = "Hello {}".format(self.name)
        print(message)
        return message
```

then…

```
from custom_operator.hello_operator import HelloOperator
with dag:
    hello_task = HelloOperator(task_id='sample-task', name='foo_bar')
```

Hooks act as an interface to communicate with the external shared resources in a DAG. For example, multiple tasks in a DAG can require access to a MySQL database. Instead of creating a connection per task, you can retrieve a connection from the hook and utilize it. Hook also helps to avoid storing connection auth parameters in a DAG.

CRONJOB ###################### CRONTAB

Linux Crontab is used for running specific tasks on a regular interval.
Crontab is very useful for routine tasks like scheduling system scanning, daily backups.
Crontab executes jobs automatically in the back-end at a specified time and interval.

[Minute] [Hour] [Day_of_the_Month] [Month_of_the_Year] [Day_of_the_Week] [command]

Minute = value between 0-59
Hour = value between 0-23
Day_of_the_Month = value between 1-31 (month with less days, it will ignore remaining part)
Month_of_the_Year = value between 1-12 (can also be define with 3 letters like jan, feb, mar, apr)
Day_of_the_week = value between 0-7 (0 and 7 for Sunday, 1 Monday, also 3 letters like sun, mon, tue, wed)
***** matches anything
Multiple values are separated by (**,**)
Define range by using Hyphen like 1-10 (**-**)
Define multiple range like this **jan-mar, jul-sep**
Multiple Tasks with single cronjob, separe script by (**;**)

To add or update jobs in crontab, use the below command. It will open a crontab file in the edito where a job can be added/updated.

```
$ crontab -e
```

By default, it will edit crontab entries of current logged in user. To edit other user crontab use command as below.

```
$ crontab -u username -e
```

To view crontab entries of a current users use :

```
$ crontab -l
```

To view crontab of a specified user.

```
$ crontab -u username -l
```

Example: job running **every week days at 5pm**

```
$ 0 17 * * mon,tue,wed,thu,fri /script/script.sh
```

### Compression techniques

LZO, Gzip, bzip2 (uses parquet format), Snappy

SDLC

Software Development Life Cycle

1. Analysis, **2**. Design, **3**. Implementation, **4**. Testing, **5**. Deployment, **6**. Maintenance

OOP 4 Pillars

1. Inheritance (Sharing of information)
2. Encapsulation (Grouping of information)
3. Abstraction (Hiding information)
4. Polymorphism (Redefining of information)

Hadoop

Hadoop components

- Processing: Map Reduce and Yarn
- Storage: HDFS

Yarn is the resource manager

For high availability feature, Zookeeper manage active namenode and standby namenode through QJM

Hadoop modes: 1. Standalone, 2. Pseudo Distributed, 3. Cluster node
Hadoop Distributions: 1. HortonWorks (Ambari), 2. Cloudera (Cloudera Manager), 3. MapR

Hadoop tools: Hive, HBase, Flume, Sqoop (RDBS), Pig
Hot spotting in HBase - to avoid it that you change the key …

Checkpoint merging fsimage-memory and edit log files - file log sys
High availability feature in Hadoop came with the concept of standby namenode

QJM (Quora Journal Manager - 3 nodes minimum) manages the name nodes
Failover control
Namenode 2 is updated with metadata by QJM in order for the standby namenode to ensure the backup when the active namenode goes down.

Creating Hadoop cluster:
Yarn architecture: Ressource namager

Hadoop VS AWS

AWS

- No need to maintain the infrastructure
- No configuration needed
- Scalability is no brainer
- Cost efficient

Hadoop

- Security purposes (sensitive data not in cloud)

HBASE

Hbase is a distributed, column oriented NoSQL database
follow the Master / Slave architecture.
3 majors components:
- HMaster Server (Master Node)
- Hbase Region Server (slave node)
- Zookeeper

Hmaster will have the metadata and is the one that distrubute data to Hbase region server.
Hase Region Server will have Region, Regions are tables with all the row between the start key and end key
Zookeeper acts like a coordinator inside Hbase, he listen to the heart beat of each region server.

TTL(Time To Live) – Attribute
In Hbase, Column families can be set to time values in secons using TTL. Hbase will automatically delete rows once the expiration time is reached. This attribute apllies to all version of a row – even the current verion too.

CICD

Continuous Integration / Continuous Deployment
A CICD pipeline automates your softwares delivery process. The pipeline builds code, run tests, and safely deploys a new version of the application. Automated pipelines remove manual errors, provides standardized feedback loops to developers, and enable fast product iterations.

Example: Github => Jenkins => S3 => Lambda => EMR(spark) => Redshift

######## Jenkins ########
CICD pipeline
Code.py --> Github --> Jenkins
Code.py --> Github --> Build (add dependencies) —> Test --> Deploy --> Production
Unit testing using modules: pytest, unit-test, Cucumber, selenium (for web applications)

Github —> Jenkins (Build/Test/Deploy) —> S3 —> Lambda(triggered by arrival of new jar file in S3) —> EMR (Spark)

Git is version control
Github repository
Ansible is infrastructure tool (no server needed) - create infrastructure when you deploy the code and turn off at the end of the job.

### Scalability ########
Horizontal scaling - more computers for best efficiency
Vertical scaling - more memory on the same machine

Streamsets Data collector -> Nifi like

######## Service type in cloud #########
IAAS Infrastructure AAS - Example EC2
PAAS Platform AAS - Example EMR
SAAS Software AAS -

Architecture:
  Shared disk database architecture --> data stored on disk
  Shared nothing architecture --> MPP (Leader (master) - compute ())

Salesforce to check out - for data science

################
Apache
Apache Kerberos: security tool, encrypts user and password - has 2 servers: authenticator, token generator

########### AWS #############
########## Kinesis ###########
Kinesis Data Streams - Ingestion, can collect and process large streams of data records in real time as same as - Not scalable and not managed - Can integrate other tools thru EMR
Kinesis Firehose - Streaming and processing _ Scalable and managed - Can
Kinesis Video
Kinesis Data Analytics

########## Redshift ###########
Redshift is a data warehouse in AWS and uses Massive Parallel Processing (MPP)

Architecture leader: responsible of the execution plan and distributes jobs to compute nodes.
It is based on PostgreSQL
##### Redshift Performance #############
- Change HDD to SSD
- Redshift Sort Key: determines the order in which rows in a table are stored
Compound sort keys
Interleaved sort keys
- Distributions styles:
Key: quick access key based on the node (like partitions in Hive) -> each partition is a different node
Even: assigned hash values (like bucketing in Hive) - partitioned data is stored in a node according the hash values
All: use more space, because using caches for tables in each node - like broadcasting in Spark

Architecture: Client (jdbc, odbc) --> Leader --> Computer Nodes (1,2,3, ...)

S3 --> SNS --> EMR(Spark) --> Redshift

####### Kibana #########
Open source data visualization and exploration tool - can run on EC2 or Elasticsearch Service
- EC2 responsible for installing Kibana software and managing the cluster
- Elasticsearch Service: Kibana is deployed and managed automatically

Elasticsearch - real time search engine - document database like MongoDB, based on key value

ELK (Elastic (query tool) Logstash (data ingestion tool) Kibana (visualization))

############# AWS Continue #######3
AWS EC2 (Elastic Compute Cloud) - virtual machine in cloud
AWS Lambda Serverless service - Charged based on # of requests
AWS Cloudwatch jk
AWS Glue is a ETL tool to build pipelines like Apache Nifi- main components: Crawler (select sources) and Classifier (select type of files), you create or import IAM to give the access to the pipeline
Glue use for data migration

AWS EMR is a cluster to run data and has Hive, HDFS, Spark, etc.

AWS Athena like SQL Workbench is a query engine in AWS, but serverless and you pay for each query you run. It can be optimized multiple ways in the back end data but not in Athena itself:
- Partittion - supports Hive partitioning
- Bucketing
- Compression
- Optimized files
Athena performance is better by improving file compression for example

Athena query engine is Presto, it compatible with Hive metastore; it uses SSL(Secure Socket Layer at port 443)

AWS DynamoDB is like Apache HBase, not distributed and NoSQL database

RDBMS: Mysql, Oracle, Sql sever, Postgre SQL, AWS RDS, Sql database for Azure
AWS Aurora supports 2 kinds of replica, has high availability,
AWS RDS chance of data lost

AWS Glacier - data warehouse for cold data - Accessing is slow and expensive in cost while storing is almost free
AWS S3 - data warehouse for hot data - Accessing is fast
AWS Cloudwatch (Monitoring tools for Redshift) - Create materialized views

AWS Strep Functions - an orchestration tool like airflow - a serverless workflow service that allows to automate multiple services in a single execution. A PUT event can trigger a Glue job for instance.
AWS Fargate - Serverless compute engine for containers
AWS Copilot - CLI to maintain your container apps.

KMS for security on AWS

########## AWS Notification ###########
SNS (Simple Notification Service) - for multiple messages
SQS (Simple Queuing Service) - Queuing service in AWS - for 1 message at the time.

Example of Pipeline: File --> S3 --> Lambda --> SQS (Batches) --> Lambda --> Redshift

AWS IOT - 2 main components: GreenGrass, Core (IOT topics, IOT rule, IOT action --> AWS DynamoDB)

Private Cloud (Internal cloud for companies), Public Cloud (AWS, Azure, IBM, etc.), Hybrid Cloud

###### AWS Machine Learning #######
- Amazon Lex
- Amazon Polly
- Amazon Rekognition

Snowflake

Snowflake (IAAS: Infrastructure As A Service) is an analytic data warehouse system available on cloud; it's a SQL database. It is a columnar-stored relational database and works well with analytical tool like Tableau

It is distributed - data are split into micro partitions that are internally optimized and compressed.

Query performance is fast
Distributed, Storage, Networks, OS

############### Cassandra #################

Cassandra which columnar based (real time streaming - columnar, AP, NoSQL, can store json, xml, …), great for huge amount of data on daily basis. Consistency is flexible, you can choose strong or low consistency. Cassandra is decentralized and every node in cluster is identical; it is fault tolerant, elastically scalable with robust architecture (peer to peer) – no single point of failure.
There is a flexibility in Cassandra to create columns within the rows. That is, Cassandra is known as the schema-optional data model. Since each row may not have the same set of columns, there is no need to show all the columns needed by the application at the surface. Therefore, Schema-less/Schema

############### SQL RDBMS Engines ################
MySQL Workbench
Postgre SQL
MS SQL Server

########### Docker ###############
VM - Hypervisor
Docker - Docker engine - you can spin up your machine in 10-15s, no high availability reason of the use of Kubernetes (Orchestration tool)
Docker swarm - orchestration tool
Elastic container service () in AWS
AWS ACR - (AWS Docker like)

Spark resource manager: - yarn, - mesos, standalone, - kubernetes

##### Fact Table #######
Star schema: tables are not normalized and only fact table and one more dimension table
Snowflake schema: tables are normalized, you can have many dimensions

UDFs

File-->>(FTP)-->S3-->Event listener-->{Key:Value}-->Lambda-->Spark Engine-->Redshift (or S3)-->Quick Sight (1/10 cost of Tableau)

#### Pig ####
Pig is a Procedural Data Flow Language. Allows to do complex transformations without knowing Java, Pig scripts are converted into a series of Map Reduce jobs that runs in the cluster. Pig latin is its language, operates on file in HDFS and provides operations like join, filter, sort, group.
Pig uses metadata (schema) when available, don't have to use it like in Hive. It supports complex data like tuples.

Runs in 2 modes:
    local(no Hadoop use) for testing

MapReduce(read data from HDFS, transform it and store back in HDFS file system))
## Hive has a Declarative SQLish Language HiveQL.

############ Impala ##############
It is an open source MPP query engine for huge volume data in Hadoop cluster
It is not fault tolerant, it is required for real time query
You can access Hive table in Impala - read the Hive metadata
Impala talks to the Namenode for fast query

###########
Graph databases: Neo4j - Transitivity like Facebook friends request
Document databases (NoSQL): MongoDB - horizontally called

S3 --> KPL --> Kinesis --> Stream --> KCL --> S3 --> Lambda --> Redshift

#### Swap space ####
Swap space doesn't replace RAM, it helps the machine with a small amount of RAM when the physical memory is full. So inactive pages can be moved to swap place to free memory.
When checking the swap space, a large utilization percentage means there is a need of adding more physical memory RAM to the system.
If the system RAM is less than 1GB, it is useful to use swap space to improve the performance of running application. However, when a system needs constantly swap, I consider it bad because it is the sign that the RAM is too small for what you are using it for; physical memory increase might be necessary.

Redis - open source - NOSql database, fast, simple, scalable, support transactions
Memory, more than just key-value
- because it most of the databases are in memory, you could lose your data in case of failure

##### MAVEN # ####
Apache Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM). It is a build management tool

DevOps tools
- Jenkins
- Maven (build automation tool) like Ant, Gradle
- Docker
- Kubernetes(can be used as a resource manager in Spark; it is orchestration tool for Jenkins and Docker)
- Ansible (Infrastructure building tool)
- Terrofom (configuration managed tool)

Jenkins -> Test -> Production

#### Linux command

Grep, sed,

Snowflake is a data warehouse system available on cloud
Query performance is fast

Scala: Any is the signature

Offset internal and
Check save offset (which message was received and sent) , consumer fooset
Save offset in an external database
Consumer group read from the same topic

Panda is not parallelized and cannot be used with Spark
StructType is what we use to create a schema in Python (we use case class in Scala)

GOOGLE ############ GOOGLE CLOUD #################
DataWarehouse = BigQuery
RDBMS = Cloud SQL
NoSQL real-time database = Firebase
Spark or Hadoop cluster = Dataproc
Stream data = Dataflow
Storage = Cloud Storage

AZURE ############ AZURE #################

STORAGES == S3 => Azure Blob (unstructured data) // Azure gen2 storage (any data)
ETL === AWS GLUE => Azure Data Factory ADF
DATA WAREHOUSE === AWS Redshit => Azure Synapse
DATA ANALIST === AWS Kinana => Power BI // Azure Analysis Services
AUTH === Azure Key vault.
DATALAKE === Azure Data Lake Storage Gen 2 (ADLS Gen2)

Source => Azure gen2 => ADF(azure databricks // azure Synapse) => Power Bi

INFORMATICA ######### INFORMATICA ########################

Oracle
MySql => ETL(Power center) => IDW(Infornmatica Data Warehouse)
Posgress

MICROSOFT ########### MICROSOFT ###############

SSIS == Sql Server Integreation System (ETL tool)

Source => SSIS => SqlServer

################# WEB SCRAPING / WEB-CRAWLING ###########

Beautiful Soup 4 // BS4, scrapy, selenium.

Model Data Science

It’s a mathematical function that help’s predict or classifies data.
Framework, SparkML, Tensorflow, Keras, Theranos

EXPERIENCE

P1 - THE HARTFORD, Hartford, connecticut
- **Data engineer**
- **January 2019 - present.**
- Ingesting Data from the Company rest API and store them in relational database On Prem HortonWork // Ambari for monitoring Hadoop cluster.

---- 1 -----------------

When I start working at The Hartford, they were using Flume to ingest data from the company RestAPI in JSON format to HDFS then Spark was streaming over HDFS to do some transformation, cleaning data, make sure the schema was identical to be then stored in data-warehouse, the Data were store in Hive and MySql. The data analytics team was doing visualisation with Tableau. The data where customer information such as personal info, insurance they have, how much they pay etc... they wanted to have all pieces of information possible for the customer representatives. When the customer representative enters the client's name, all the pieces of information is shown on the computer with a nice and simple view. When the customer representative have the customer on the phone they can see his information and provide the best customer services and propose new services to the customer. After few months working at The Hartford, the Architect asked me advice on new pipelines with maybe using the Cloud, the Official Architect had another background then Big Data so I was the one who designed the new pipelines. I decided to have a Hybrid pipeline. The data from the different location overall USA was uploaded to AWS S3 in different buckets, from there I decided to use Spark over Databricks to do data cleaning and transformations. Then the data are stored in AWS Redshift (data warehouse). For each new pipeline, I decided to use Airflow as orchestration tool. Today we have 52 pipelines running that way. I don't have the Architect Title but like I said I'm able to design all sort of pipeline, as well on Cloud using AWS, GCP or Azure services and on-premises with Cloudera or Hortonworks and also Hybrid like I did at The Hartford.

RestAPI => Flume => HDFS => SPARK => MySQL / HIVE => Tableau
AWS S3 => Spark(Databricks) => AWS Redshift == Airflow Orchestration, GitLab CICD

---- 2 -----------
we did some incremental imports to hive using Sqoop. Country tables, services name tables.
Mysql => Sqoop => HDFS // HDFS => Sqoop => HIVE
---- 3 ---------------
Installed Airflow engine to run multiple spark jobs.
The job was running every day to perform spark job in parallel.
To do parallel task you needed to change the database, by default it's using SQLlite and its allow only one request at the time. Mysql or PostgreSQL will allow many request at the same time so the executor can ask the metadata in this database and do task in parallel.
--- 4 --------
Did some ingestion of real-time data logs to HDFS.
API => FLUME => HDFS => SPARK => MySQL => Tableau.
I lived in Bristol, duncan street, behind Walmart. 25-30 minutes from the plaza.
Financial and insurance company, Fortune 500 in 2018
Christopher Swift CEO, Doug Elliot President, Beth Costello CFO
insurance company for individual and companies
Noth of New-York and South of Boston
Italian Rest Salute 10 blocks away / Kent pizza 1 block away.

P2
- PROGRESSIVE CORP, Mayfield, Ohio
- **AWS Cloud Data Engineer**
- **May 2017 - January 2019**
- migrating data from local to AWS S3 and pipeline from S3 to EMR to Redshift
- ***AWS, S3, Redshift, Spark, Hive, Kafka***

------ 1 ---------
Data were mostly customer experiences and claims, data anilist team were doing some visualization to find out what service need more attention to and witch service needed more money invested in.

```
MySql                 Data lake                 Data warehouse
Oracle => DMS => AWS S3 => EMR(spark) => Redshift
```

-------- 2 ---------
Data click from websites in real time, analyze where the customer is clicking and what service is most view by the costumers.

RestAPI
websites
mobileApp => AWS Kinesis Data stream(real time) => EMR(spark) => Redshift

--------- 3 ----------
Query some tables with sparkSQL from Hive to do some transformations and store the data into AWS Redshift so the data analist team could do some join with other tables with costumer info. This tables were data about services, service name, prices, availability, country available ect…

Hive on Prem => Spark-SQL => Redshift.

I was living in a small city called BeachWood with family. It's 15 minutes from Mayfield.
Insurance company, cleveland campus 1, Mayfield Village, Cuyahoga County Ohio. Less than 30 minutes away from cleveland, close to Lake Erie. Noth between new-york and chicago, close to detroit and the canadian border.(Toronto closer canadian city)

P3
- MCKESSON CORP, Irving, Texas
- **Big Data Engineer**
- **December 2015 - May 2017**
- Retail data, healthcare product. Real-time click-streams.
- ***Spark, Kafka, Hive, MySQL, AWS Kibana***

--------- 1 --------------------------
We were ingesting real-time click-streams with Kafka, consume and transform the data with spark and store in hive.

We have people sending quality check data for their products to the internal API, we ingest data (1000 to 5000 message / minute) in JSON format, spark streaming to clean and remove duplicate and make sure of the same schema. Store the data

in Hive. The purpose was to have an Internal recommendation engine for the company clients. To provide the best product and services depending of the client’s choice.
Ex: client want to open a new facility, based on the client requirement, we can provide the best product to buy.

Internal - API (bacth 1k to 5k minutes) => Kafka => Spark => Hive
API (real-time) => Kafka => Spark => Hive

--------- 2 ------------------------------
Query hive tables with Spark-SQL, join some tables, some complex queries and store in MySQL so the data analist team can do some visualization with AWS Kibana.

Hive => spark => MySQL=>AWS Kibana

-------- 3 --------------------------------
Did documentation in sharepoint 2013 for the developer team.

Currently ranked 7th on the FORTUNE 500, we are a global leader in healthcare supply chain management solutions, retail pharmacy, healthcare technology, community oncology and specialty care.

5 minutes away from Dallas/Fort Worth International Airport.
Dallas/Fort Worth Airport is 20 minutes from Dallas Texas.

P4
- EPIC SYSTEM, Verona, Wisconsin
- **Big Data Engineer**
- **October 2013 - December 2015**
- Ingesting data from Oracle, manipulate and analize record for software

integration.
- ***Hortonworks, Flume, Kafka, HDFS, Spark, Oracle, MySQL, Hbase***

Epic build software to retrieve electronic record from patient.
Hortonworks and Ambari on prem
Hbase is a distributed, column oriented NoSQL database.

------- 1 ------------------------------------

Ingesting real-time data from API with Kafka, do some transformation with Spark and store into Hbase. Data about costumers info, like pulse rate, blood pressure ect.. so the data analist team could analyze the data for software integretion.

API => Kafka=> Spark=>Hbase

------- 2 -------------------------------------

Ingesting Records from API with Flume and store them into HDFS for then stream data with Spark and do some transformations and aggregation and store into Hbase.

API => Flume => Spark => Hbase

-------- 3 -------------------------------------
Transferred some tables from MySQL / Oracle to HDFS using Sqoop.
Data were names and infos of Software ect...

Mysql => Sqoop => HDFS

Health company, software & services,
Founded in 1979, Epic build software to retrieve electronic record from patient.
Wisconsin up north Chicago, verona 10 miles from madison

P5
- CATERPILLAR, Deerfield, ILinois
- **Data Engineer**
- **November 2011 - October 2013**

- She
- ***H***

Created shell script to automate process such as starting HDFS, Kafka or even file transfer.

MySQL => Sqoop=> HDFS

---- 2 ----------

From HDFS importing into Sharepoint server with cmd

HDFS=>Shell script=>SharePoint.

---- 3 ------------
creted test cases during sprints.

see how to do that

--- 4 ----------
Did some data visualization using Power BI.
Learn how to use it and write some description of the tool and optimization.

----- 5 ---------------
monitored multiple SQL servers to improve query performance.

--- 6 ----------------
did some POC on Installing, configuring and managing Hadoop binaries

https://docs.microsoft.com/en-us/sharepoint/administration/import-a-list-or-document-library

- Created UNIX shell scripts to automate the build process, and to perform regular jobs like file transfers.
- Wrote shell scripts to automate workflows to pull data from various databases into FS.
- Loaded data into a shared file system using terminal.
- Performed upgrades, patches and bug fixes in Hadoop in a cluster environment.
- Wrote shell scripts for automating the process of data loading
- Created test cases during two-week sprints using agile methodology
- Designed data visualization to present current impact and growth using Power BI
- Developed a Java program to clean up data streams from server logs
- Worked with a Java Messaging system to deliver logs
- Monitored multiple SQL servers to improve query performance
- Experience in configuring, installing and managing Hadoop binaries (PoC).