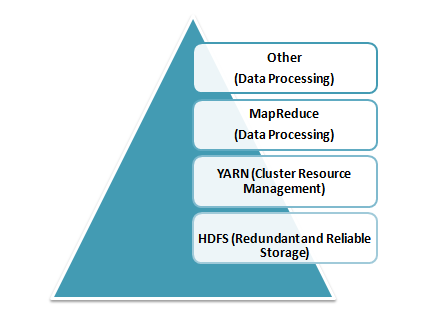
# Explain the core components of Hadoop.

**Answer:**Hadoop is an open source framework that is meant for storage and processing of big data in a distributed manner. The core components of Hadoop are –

* **HDFS (Hadoop Distributed File System) –** HDFS is the basic storage system of Hadoop. The large data files running on a cluster of commodity hardware are stored in HDFS. It can store data in a reliable manner even when hardware fails.

[](https://www.whizlabs.com/wp-content/uploads/2017/11/Core-Components-of-Hadoop.png)

* **Hadoop MapReduce –** MapReduce is the Hadoop layer that is responsible for data processing. It writes an application to process unstructured and structured data stored in HDFS. It is responsible for the parallel processing of high volume of data by dividing data into independent tasks. The processing is done in two phases Map and Reduce. The Map is the first phase of processing that specifies complex logic code and the Reduce is the second phase of processing that specifies light-weight operations.
* **YARN –** The processing framework in Hadoop is YARN. It is used for resource management and provides multiple data processing engines i.e. data science, real-time streaming, and batch processing.

# Define respective components of HDFS and YARN

The two main components of HDFS are-

* Name Node – This is the master node for processing metadata information for data blocks within the HDFS
* Data Node/Slave node – This is the node which acts as slave node to store the data, for processing and use by the Name Node

The two main components of YARN are**–**

* Resource Manager– This component receives processing requests and accordingly allocates to respective Node Managers depending on processing needs.
* Node Manager– It executes tasks on each single Data Node

# Write the command used to copy data from the local system onto HDFS?

The command used for copying data from the Local system to HDFS is:  
**hadoop fs –copyFromLocal [source][destination]**

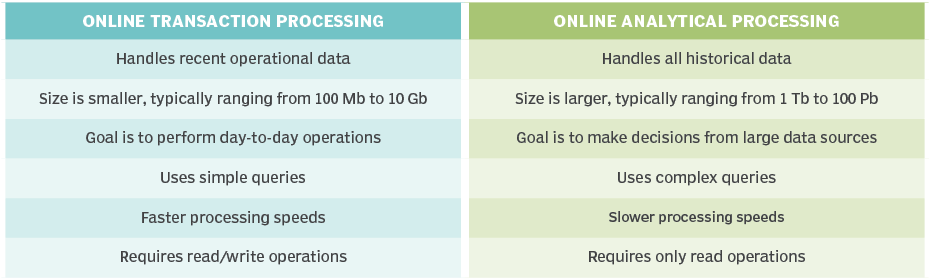
# OLTP

Online transaction processing (OLTP) captures, stores, and processes data from transactions in real time.

# OLAP

Online analytical processing (OLAP) uses complex queries to analyze aggregated historical data from OLTP systems.

The basic difference between OLTP and OLAP is that OLTP is an online database modifying system, whereas, OLAP is an online database query system

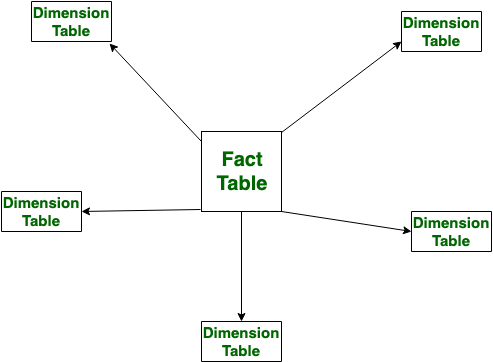


# Fact and Dimension Table

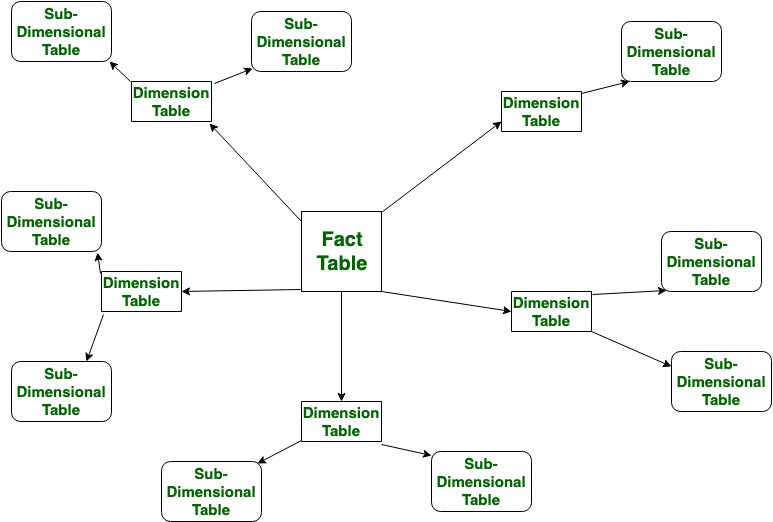
A fact table holds **the data to be analyzed,** and a

Dimension table **stores data about the ways in which the data in the fact table can be analyzed.**

Star Schema



# Snowflake Schema



|  |  |  |
| --- | --- | --- |
| S.NO | **Star Schema** | **Snowflake Schema** |
| 1. | In star schema, The fact tables and the dimension tables are contained. | While in snowflake schema, The fact tables, dimension tables as well as sub dimension tables are contained. |
| 2. | Star schema is a top-down model. | While it is a bottom-up model. |
|  |  |  |
| 3. | In star schema, Normalization is not used. | While in this, Both normalization and demoralization are used. |
| 4. | It has less number of foreign keys. | While it has more number of foreign keys. |
| 5. | It has high data redundancy. | While it has low data redundancy. |

# What is SCD in data warehouse?

A Slowly Changing Dimension (SCD) is **a dimension that stores and manages both current and historical data over time in a data warehouse**

* Type 1 – This model involves overwriting the old current value with the new current value. Overwrite the changes
* Type 2 – The current and the historical records are kept and maintained in the same file or table. History will be added as a new row.
* Type 3 – The current data and historical data are kept in the same record. History will be added as a new column.

# File Formats:

**AVRO is a row-based storage format.** Writing operations in AVRO are better than in PARQUET

**PARQUET is a columnar-based storage format**. PARQUET is much better for analytical querying, i.e., reads and querying are much more efficient than writing. Parquet is more efficient in terms of storage and performance

**ORC** is Optimized Row Columnar, and it is a free and open-source columnar storage format designed for Hadoop workloads.

ORC supports ACID properties **ORC reduces the size of the original data up to 75%**. As a result the speed of data processing also increases and shows better performance than Text

**CSV** is a **comma-separated values file**, which allows data to be saved in a tabular format.(row and columns)

**JSON** file is **a file that stores simple data structures and objects in JavaScript Object Notation (JSON) format, which is a standard data interchange format**. It is primarily used for transmitting data between a web application and a server

# Spark:

## Open source distributed computing engine, you can store and process huge volume of data 100 time faster than hadoop.Its uses in memory and parallel processing which makes spark faster

# Spark architecture:

Its works on the principle of master slave architecture

It is the **central point and the entry point** of the Spark Shell (Scala, Python, and R).

The driver program runs the main () function of the application and is the place where the Spark Context and RDDs are created, and also where transformations and actions are performed.

Spark Driver contains various components – DAGScheduler, TaskScheduler, BackendScheduler, and BlockManager responsible for the translation of spark user code into actual spark jobs executed on the cluster.

Spark Driver performs two main tasks: Converting user programs into tasks and planning the execution of tasks by executors

# How does Spark work?

 Once we submit a spark job, a driver program is launched and this requests for resources to the cluster manager and at the same time the main program of the user function of the user processing program is initiated by the driver program.

Based on that, the execution logic is processed and parallelly Spark context is also created. Using the Spark context, the different transformations and actions are processed. So, till the time the action is not encountered, all the transformations will go into the Spark context in the form of DAG that will create RDD lineage.

Once the action is called job is created. Job is the collection of different task stages. Once these tasks are created, they are launched by the cluster manager on the worker nodes and this is done with the help of a class called task scheduler.

The conversion of RDD lineage into tasks is done by the DAG scheduler. Here DAG is created based on the different transformations in the program and once the action is called these are split into different stages of tasks and submitted to the task scheduler as tasks become ready.

Then these are launched on the different executors in the worker node through the cluster manager. The entire resource allocation and the tracking of the jobs and tasks are performed by the cluster manager.

# What is meant by Lazy evaluation in spark?

Execution will not start until action is called. That means Data is not loaded until the point where action is called which helps spark engine to have better optimization

# Terminologies of Spark

## Driver and worker Process:

## These are nothing but JVM process. Within one worker node, there could be multiple executors. Each executor runs its own JVM process.

## Application:

## It could be single command or combination of multiple notebooks with complex logic. When code is submitted to spark for execution, Application starts.

## Jobs:

## When an application is submitted to Spark, driver process converts the code into job.

## Stage:

## Jobs are divided into stages. If the application code demands shuffling the data across nodes, new stage is created. Number of stages is determined by number of shuffling operations. Join is example of shuffling operation

## Tasks:

## Stages are further divided into multiple tasks. In a stage, all the tasks would execute same logic. Each task will process 1 partition at a time. So number of partition in the distributed cluster determines the number of tasks in each stage

## Transformation:

## It creates new RDD and from the existing RDD. Until action is called, transformations are evaluated lazily. Some of the transformations are

## (map,filter,flatMap,mapPartitions,mapPartitionsWithIndex,groupBy,sortBy,union,intersection,subtract,distinct,Cartesian,zip,sample,randomSplit,keyBy,coalesce,repartition)

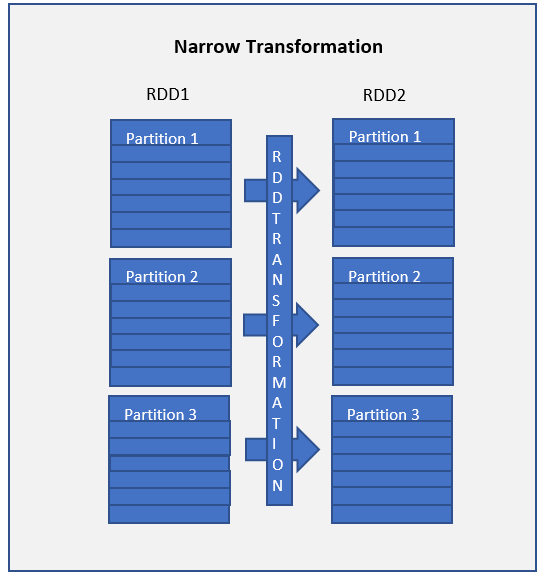
# Two types of transformations in SPARK:

* Wide Transformations
* Narrow Transformations

# Narrow Transformations:

These types of transformations convert each input partition to only one output partition.

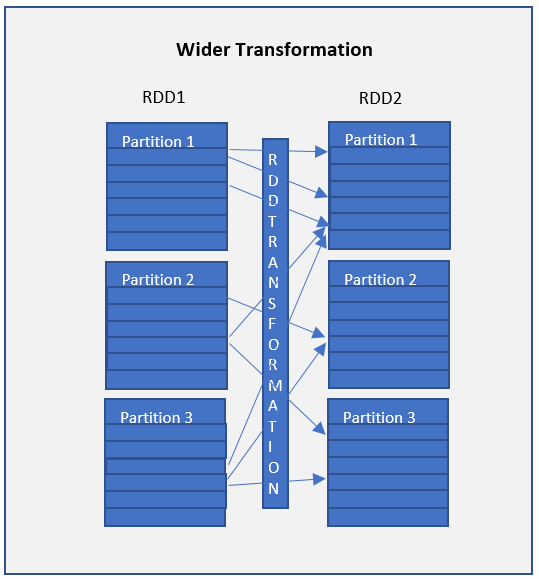
* This kind of transformation is **basically fast.**
* **Does not require any data shuffling over** the cluster network or no data movement.
* Operation of **map () and filter ()** belongs to this transformation.



# Wide Transformations:

This type of **transformation will have input partitions contributing to many output partitions.** When each partition at the parent RDD is used by multiple partitions of the child RDD or when each partition from child produced or dependent on multiple parents RDD.

* **Slow as compare to narrow dependencies**
* Might Require **data shuffling over the cluster network or no data movement.**
* Functions such as **groupByKey** (), **aggregateByKey(), aggregate(), join(), repartition()** are some examples of wider transformations.



When working with Spark, it is always good or keeps in mind all operations or transformations which might require data shuffling and hence slow down the process. Try to optimize and reduce the usage of wide dependencies as much as you can

**Repartition() and coalesce()**

The Repartition() can be used to **increase or decrease the number of partitions**, but it involves heavy data shuffling across the cluster

coalesce() can be **used only to decrease the number of partitions**. In most of the cases, coalesce() does not trigger a shuffle.

Coalesce() is best as it doesn’t trigger shuffle

## DAG:

## Directed Acyclic Graph keeps track of all transformation. For each transformation, logical plan is created and lineage graph is maintained by DAG

## Action:

## When data output is needed for developer or for storage purpose, action is called. Action would be executed based on DAG and processes the actual data.

## Some of the actions are

## (reduce,collect,aggregate,foldfirst,take,forEach,top,treeAggregate,treeReduce,Partitioncount,takeSample,max,min,sum,histogram,mean,variance,Save)

## RDD:

## Resilient Distributed Dataset is basic data structure of Spark. When spark reads or creates data, it creates RDD which is distributed across nodes in the form of partition.

* ***Resilient:*** Fault tolerant and is capable of quickly recover from failure
* ***Distributed:*** Distributed data among the multiple nodes in a cluster
* ***FFDataset:*** Collection of partitioned data with values

## Executor:

## Each worker node consist of many executors.it can be configure by spark setting

## Core:

## Each executor can consist of multiple cores. This is configurable by spark settings. Each core can process on task at a time

## Important:

A Spark application can have many jobs. A job can have many stages. A stage can have many tasks. A task executes a series of instructions.

# Different betyouen RDD vs. Dataframes vs. Datasets

**DataFrame**- In dataframe data is **organized into named columns**. Basically, it is as same as a table in a relational database.

whereas, **DataSets**- As we know, it is an **extension of dataframe API, which provides the functionality of type-safe, object-oriented programming interface**

## RDDs vs Dataframes vs Datasets

|  |  |  |  |
| --- | --- | --- | --- |
|  | **RDDs** | **Dataframes** | **Datasets** |
| **Data Representation** | RDD is a distributed collection of data elements without any schema. | It is also the distributed collection organized into the named columns | It is an extension of Dataframes with more features like type-safety and object-oriented interface. |
| **Optimization** | No in-built optimization engine for RDDs. we need to write the optimized code themselves. | It uses a catalyst optimizer for optimization. | It also uses a catalyst optimizer for optimization purposes. |
| **Projection of Schema** | Here, we need to define the schema manually. | It will automatically find out the schema of the dataset. | It will also automatically find out the schema of the dataset by using the SQL Engine. |
| **Aggregation Operation** | RDD is slower than both Dataframes and Datasets to perform simple operations like grouping the data. | It provides an easy API to perform aggregation operations. It performs aggregation faster than both RDDs and Datasets. | Dataset is faster than RDDs but a bit slower than Dataframes. |

# What is DAG and how it works in Fault Tolerance?

DAG (Directed Acyclic Graph)

DAG converts logical plan to physical plan

# How spark achieves fault tolerance?

* **Spark provides fault tolerance through lineage graph. Lineage graph keeps the track of the transformations to be executed once the action has been called. It helps in recomputing any missing RDD in case of any node failure**.

from pyspark.sql import SparkSession

spark=SparkSession.builder.appName("Name").getOrCreate()

# Which one do you prefer? Either groupByKey() or ReduseByKey?

# groupByKey

The groupByKey can cause out of disk problems as data is sent over the network and collected on the reduced workers. You can see the below example.

sparkContext.textFile("hdfs://")

.flatMap(line => line.split(" ") )

.map(word => (word,1))

.groupByKey()

.map((x,y) => (x,sum(y)))

# reducebykey

Whereas in reducebykey, Data are combined at each partition, only one output for one key at each partition to send over the network. reduceByKey required combining all our values into another value with the exact same type.

sparkContext.textFile("hdfs://")

.flatMap(line => line.split(" "))

.map(word => (word,1))

.reduceByKey((x,y)=> (x+y))

# Broadcast Variable

Broadcast variables are **used to save the copy of data across all nodes. This variable is cached on all the machines and not sent on machines with tasks**.

Broadcast variable is primarily used for **reading some data across worker node**

## 

## Rdd=sc.broadcast([“raju”,”tharan”])

## To submit broadcast variable:

## Spark-submit broadcast.py

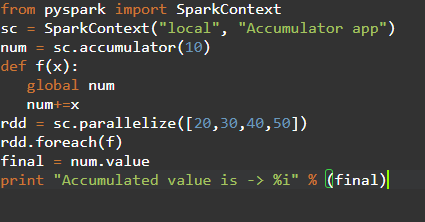
## OUTPUT:

## Stored data 🡪[‘raju’,’tharun’]

# Accumulator

Accumulator variables that **are used for aggregating information across the executors**

Accumulator is used for **writing some data across worker node**.



Num=sc.accumulator(10)

Rdd=sc.paralellize([20,30,40,50])

When you submit accumulator variable:

Spark-submit accumulator.py

**OUTPUT:**

Accumulated value is: 150

# Broadcasting Join

Broadcast Join is a type of join operation in PySpark that is used to join data frames by broadcasting it in PySpark application. This join can be used for the **data frame that is smaller in size which can be broadcasted with the PySpark application to be used further**. **The data is sent and broadcasted to all nodes in the cluster**. This is an optimal and cost-efficient join model that can be used in the PySpark application.

Clusters in spark:

Apache Spark has 3 main open source cluster managers:

**YARN,**

**Standalone,**

**Mesos,**

# What is cluster mode and client mode in Spark?

1. **Client Mode (Default Mode):** In this mode, the driver will be launched on that machine where the spark-submit command was executed.
2. **Cluster Mode:** In this mode, the driver will run inside the Standalone cluster as another procedure on one of the worker nodes, and after that, it will link back to request executors.

# What is the difference between union and union all?

Union and Union All are similar except that

Union only **selects the rows specified in the query**, while

Union All **selects all the rows including duplicates** (repeated values) from both queries

# What is Spark context in Spark?

A SparkContext **represents the connection to a Spark cluster, and can be used to create RDDs, accumulators and broadcast variables on that cluster**. Only one SparkContext should be active per JVM. You must stop() the active SparkContext before creating a new one.

.Rdd vs

1. The driver program use the SparkContext to connect and communicate with the cluster and it helps in executing and coordinating the Spark job with the resource managers like YARN or Mesos.
2. Using SparkContext you can actually get access to other contexts like  SQLContext and HiveContext.
3. Using SparkContext we can set configuration parameters to the Spark job.



# What is a Spark session?

SparkSession is the entry point to Spark SQL. It is one of the very first objects we need to create while developing a Spark SQL application. we **create a SparkSession using the SparkSession.** **builder method** (that gives you access to Builder API that you use to configure the session).

# cache () and persist ()?

## Both persist () and cache () are the Spark optimization technique, used to store the data, but only difference is cache () method by default stores the data in-memory (MEMORY\_ONLY) whereas in persist () method developer can define the storage level to in-memory or in-disk.

cache() - default storage level is **MEMORY\_ONLY**

**Persist()** -default storage level is **MEMORY\_AND\_DISK**.We have many option of storage levels that can be used with persist()

* **MEMORY\_ONLY,**
* **MEMORY\_AND\_DISK,**
* **MEMORY\_ONLY\_SERialized**
* **MEMORY\_AND\_DISK\_SER,**
* **DISK\_ONLY,**
* **MEMORY\_ONLY\_2,**
* **MEMORY\_AND\_DISK\_2,**
* **DISK\_ONLY\_2**
* **MEMORY\_ONLY\_SER\_2,**
* **MEMORY\_AND\_DISK\_SER\_2**

To check the storage level of the dataframe or RDD, we can use ***rdd.getStorageLevel*** or ***df.storageLevel***

# Class vs static method:

|  |  |
| --- | --- |
| **Class Method** | **Static Method** |
| The class method takes cls (class) as first argument. | The static method does not take any specific parameter. |
| Class method can access and modify the class state. | Static Method cannot access or modify the class state. |
| The class method takes the class as parameter to know about the state of that class. | Static methods do not know about class state. These methods are used to do some utility tasks by taking some parameters. |
| @classmethod decorator is used here. | @staticmethod decorator is used here. |

# Decorators:

A decorator is **a design pattern in Python that allows a user to add new functionality to an existing object without modifying its structure**. Decorators are usually called before the definition of a function

we use a decorator **when you need to change the behavior of a function without modifying the function itself**. A few good examples are when you want to add logging, test performance, perform caching, verify permissions

# iterators and generators?

Iterators are the **objects that use the next () method to get the next value of the sequence.**

A generator is **a function that produces or yields a sequence of values using a yield statement**. Classes are used to Implement the iterators. Functions are used to implement the generator.

# Lambda function:

A lambda function is **a small anonymous function**. A lambda function can take any number of arguments, but can only have one expression.

Lambda functions are used **when you need a function for a short period of time**

# Polymorphism in python defines methods in the child class that have the same name as the methods in the parent class

# Isin():

The isin() method **checks if the Dataframe contains the specified value(s)**. It returns a DataFrame similar to the original DataFrame, but the original values have been replaced with True if the value was one of the specified values, otherwise False

# Partition and Bucketing:

Both Partitioning and Bucketing in Hive are used to improve performance by eliminating table scans when dealing with a large set of data on a Hadoop file system (HDFS). The major difference between Partitioning vs Bucketing lives in the way how they split the data.

**Partition** is a way to organize large tables into smaller logical tables based on values of columns; one logical table (partition) for each distinct value.

**Bucketing** is a technique to split the data into more manageable files, (By specifying the number of buckets to create). The value of the bucketing column will be hashed by a user-defined number into buckets

Below are some of the differences between Partitioning vs bucketing

| **PARTITIONING** | **BUCKETING** |
| --- | --- |
| Directory is created on HDFS for each partition. | File is created on HDFS for each bucket. |
| You can have one or more Partition columns | You can have only one Bucketing column |
| You can’t manage the number of partitions to create | You can manage the number of buckets to create by specifying the count |
| NA | Bucketing can be created on a partitioned table |
| Uses PARTITIONED BY | Uses CLUSTERED BY |

# How to connect Hive through Spark SQL?



Solution to this is to copy your hive-site.xml and core-site.xml in spark conf folder which will give Spark job all the required metadata about Hive metastore and you have to enable Hive Support along with specifying your warehouse directory location of Hive in configuration while starting your Spark Session as given below:

spark = SparkSession \  
 .builder \  
 .appName("Python Spark SQL Hive integration example") \  
 .config("spark.sql.warehouse.dir", warehouse\_location) \  
 .enableHiveSupport() \  
 .getOrCreate()

# Databricks

# Cluster types

1. **All-purpose cluster**

* Also known as interactive cluster because
* All-purpose cluster used for **mainly used for developing purpose.** While **developing we should see the intermediate result.**
* All-purpose cluster also can be used for job.
* **Can be paused ,stop ,started and multiple user can share this cluster**

1. **Job cluster**

* **Mainly used for schedule jobs**
* While scheduling jobs you need to configure the cluster parameter based on the cluster would be created during runtime and once the job got completed it will terminate automatically
* **we couldn’t control manually. Job cluster are visible during job runtime**

1. **Pool cluster**

* When you **have multiple cluster you want to combine then you can create pool**
* Advantage of pool while **creating you can set parameters such as this many number of instances should be active always and ready to use**
* **Suitable for larger teams**
* **It will be costly**

# Cluster Modes

* Standard
* High concurrency
* Single

# What is auto scaling?

## Databricks chooses dynamically the appropriate number of workers required to run the job based on range of number of workers.

## It is one of the performance optimization technique

## It is also one of cost saving technique

## Auto scaling has two types

## Standard

## optimized

# Different Types of SQL JOINs

Here are the different types of the JOINs in SQL:

* (INNER) JOIN: Returns records that have matching values in both tables
* LEFT (OUTER) JOIN: Returns all records from the left table, and the matched records from the right table
* RIGHT (OUTER) JOIN: Returns all records from the right table, and the matched records from the left table
* FULL (OUTER) JOIN: Returns all records when there is a match in either left or right table

# Set Operators:

Set operators are used to combine results from two or more SELECT statements. They combine the same type of data from two or more tables. This looks similar to SQL joins although there is a difference. SQL joins are used to combine columns whereas Set operators are used to join rows from multiple SELECT queries. They return only one result set.  
  
These operators work on complete rows of the queries, so the results of the queries must have the same column name, same column order and the types of columns must be compatible.

There are the following 4 set operators in SQL Server: union, unionall, intersect and except

# UNION

The UNION operator combines two or more result sets into a single result set, without duplications

# UNION ALL

Like the UNION operator the UNION ALL operator also combines two or more result sets into a single result set. The only difference between a UNION and UNION ALL is that the UNION ALL allows duplicate rows.

# INTERSECT

INTERSECT operator returns only the rows present in all the result sets. The intersection of two queries gives the rows that are present in both result sets

# EXCEPT

EXCEPT operator returns all distinct the rows that are present in the result set of the first query, but not in the result set of the second query. It means it returns the difference between the two result sets.

# Execution order of SQL

* FROM
* WHERE
* GROUP BY
* HAVING
* SELECT
* ORDER BY
* LIMIT

# Primary Key vs Foreign Key

# 

# JOINS IN SQL

* **INNER JOIN:**  return all the rows from multiple tables where the join condition is satisfied.
* **LEFT JOIN:**  return all the rows from the left table but only the matching rows from the right table where the join condition is fulfilled.
* **RIGHT JOIN:** return all the rows from the right table but only the matching rows from the left table where the join condition is fulfilled.
* **FULL JOIN:** returns all the records when there is a match in any of the tables. Therefore, it returns all the rows from the left-hand side table and all the rows from the right-hand side table.
* [**SELF JOIN**](https://www.tutorialspoint.com/sql/sql-self-joins.htm) − is used to join a table to itself as if the table were two tables, temporarily renaming at least one table in the SQL statement.
* [**CARTESIAN JOIN**](https://www.tutorialspoint.com/sql/sql-cartesian-joins.htm) − returns the Cartesian product of the sets of records from the two or more joined tables.

# Common clauses used with SELECT query in SQL?

The following are some frequent SQL clauses used in conjunction with a SELECT query:

**WHERE** clause: In SQL, the WHERE clause is used to filter records that are required depending on certain criteria.

Example: SELECT ID, NAME, SALARY

FROM CUSTOMERS

WHERE SALARY > 2000;

**ORDER BY** clause: The ORDER BY clause in SQL is used to sort data in ascending (ASC) or descending (DESC) order depending on specified field(s) (DESC).

SELECT \* FROM CUSTOMERS

ORDER BY NAME DESC;

**GROUP BY** clause: GROUP BY clause in SQL is used to group entries with identical data and may be used with aggregation methods to obtain summarized database results.

SELECT DEPT, SUM(SALARY) FROM CUSTOMERS GROUP BY DEPT;

SELECT DEPT, min(SALARY) FROM CUSTOMERS GROUP BY DEPT ;

SELECT DEPT, MAX(SALARY) FROM CUSTOMERS GROUP BY DEPT ;

SELECT DEPT, AVG(SALARY) FROM CUSTOMERS GROUP BY DEPT ;

**HAVING** clause in SQL is used to filter records in combination with the GROUP BY clause. It is different from WHERE, since the WHERE clause cannot filter aggregated records Syntax:

SELECT FROM WHEREGROUP BY

HAVING

ORDER BY

Example

SELECT ID, NAME, AGE, ADDRESS, SALARY

FROM CUSTOMERS

GROUP BY age

HAVING COUNT(age) >= 2;

# How to remove duplicate rows in SQL?

1. **DISTINCT**

SELECT DISTINCT SALARY FROM CUSTOMERS

ORDER BY SALARY;

1. **DELETE BY ROW**

* If the SQL table has duplicate rows, the duplicate rows must be removed.
* Let’s assume the following table as our dataset:

|  |  |  |
| --- | --- | --- |
| ID | Name | Age |
| 1 | A | 21 |
| 2 | B | 23 |
| 2 | B | 23 |
| 4 | D | 22 |
| 5 | E | 25 |
| 6 | G | 26 |
| 5 | E | 25 |

DELETE FROM table WHERE ID IN (  
SELECT   
ID, COUNT (ID)   
FROM   table  
GROUP BY ID  
HAVING   
COUNT (ID) > 1);

# How to find the nth highest salary in SQL?

**Second highest salary in sql**

**Max salary:**

Select max(salary)from customers;

**Second highest**

SELECT \* FROM customers ORDER BY salary DESC LIMIT 1 OFFSET 1;

**Third highest**

SELECT \* FROM customers ORDER BY salary DESC LIMIT 1 OFFSET 2;

**Fourth highest**

SELECT \* FROM customers ORDER BY salary DESC LIMIT 1 OFFSET 3;

**TOP 3 HIGHEST SALARY**

SELECT \* FROM CUSTOMERS order by salary desc

LIMIT 3;

**Windows function:**

SELECT \* FROM(

SELECT emp\_name, salary, DENSE\_RANK()

over(ORDER BY salary DESC) AS ranking FROM employee) AS k

WHERE ranking=3;

SELECT \* FROM(

SELECT emp\_name, salary, ROW\_NUMBER()

over(ORDER BY salary DESC) AS ranking FROM employee) AS k

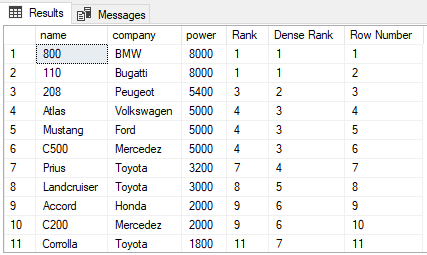
WHERE ranking=3;

# Rank vs Dense Rank vs Row \_Number:

RANK function **skips ranks** if there is a tie between the previous ranks.

DENSE\_RANK function **does not skip ranks** if there is a tie between ranks

ROW\_NUMBER function has no concern with ranking. It simply returns the **row number of the sorted records**



# AZURE

**ADF**

Azure Data Factory is **Azure's cloud ETL service for scale-out serverless data integration and data transformation**

**Azure Data Factory is composed of below key components.**

* Pipelines.
* Activities.
* Datasets.
* Linked services.
* Data Flows.
* Integration Runtimes.

### Pipeline

A pipeline is **a logical grouping of activities that performs a unit of work**. Together, the activities in a pipeline perform a task

### Activity

Activities represent a **processing step in a pipeline**. For example, we might use a copy activity to copy data from one data store to another data store

### Datasets

Datasets represent **data structures within the data stores**, which simply point to or reference the data you want to use in your activities as inputs or outputs.

### Linked services

Linked services are much like **connection strings**, which define the connection information that's needed for Data Factory to connect to external resources

### Integration Runtime

In Data Factory, An **integration runtime provides the bridge between the activity and linked Services**. It's referenced by the linked service or activity, and provides the **compute environment** where the activity either runs on or gets dispatched from

### Triggers

Triggers represent the **unit of processing that determines when a pipeline execution needs to be kicked off.**

There are different types of triggers for different types of events.

1. Schedule trigger –**we can schedule at particular time ,date ,hours**
2. Tumbling window trigger-**periodic interval(example every 1 hrs in 24)**
3. Event based trigger-**If certain changes occurs in storage layer it will run**

**Azure Key Vault**

**It is** **used to securely store and tightly control access** to

Tokens,

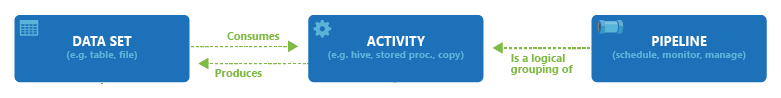
Passwords,

Certificates,

API keys and other secrets

# **What is Activity in Azure Data Factory?**

The activity is the task we performed on our data. We use activity inside the Azure Data Factory pipelines. ADF pipelines are a group of one or more activities. For ex: When you create an ADF pipeline to perform ETL you can use multiple activities to extract data, transform data and load data to your data warehouse. Activity uses Input and output datasets. Dataset represents your data if it is tables, files, folders etc. Below diagram shows the relationship between Activity, dataset and pipeline:



An Input dataset simply tells you about the input data and it’s schema. And an Output dataset will tell you about the output data and it’s schema. You can attach zero or more Input datasets and one or more Output datasets. Activities in Azure Data Factory can be broadly categorized as:

1- Data Movement Activities

2- Data Transformation Activities

3- Control Activities

Activity I have used

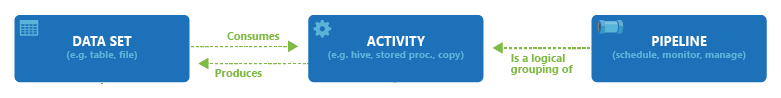
**Copy Activity:** It simply **copies the data from Source location to destination location**. Azure supports multiple data store locations such as Azure Storage, Azure DBs, NoSQL, Files, etc.

**Data Flow:**In data flow**, First, you need to design data transformation workflow to transform or move data**. **Then you can call Data Flow activity inside the ADF pipeline**. It runs on Scaled out Apache Spark Clusters. There are **two types of DataFlows: Mapping and Wrangling DataFlows**

**Databricks Notebook: It runs our databricks notebook on Azure databricks workspace**. It runs on Apache spark.

**What is Activity in Azure Data Factory?**

The activity is the task we performed on our data. We use activity inside the Azure Data Factory pipelines. ADF pipelines are a group of one or more activities. For ex: When you create an ADF pipeline to perform ETL you can use multiple activities to extract data, transform data and load data to your data warehouse. Activity uses Input and output datasets. Dataset represents your data if it is tables, files, folders etc. Below diagram shows the relationship between Activity, dataset and pipeline:



An Input dataset simply tells you about the input data and it’s schema. And an Output dataset will tell you about the output data and it’s schema. You can attach zero or more Input datasets and one or more Output datasets. Activities in Azure Data Factory can be broadly categorized as:

1- Data Movement Activities

2- Data Transformation Activities

3- Control Activities

**DATA MOVEMENT ACTIVITIES :**

**1- Copy Activity:** It simply copies the data from Source location to destination location. Azure supports multiple data store locations such as Azure Storage, Azure DBs, NoSQL, Files, etc.

To know more about Data Movement activities, please use below link:

[Pipelines and activities in Azure Data Factory - Azure Data Factory | Microsoft Docs](https://docs.microsoft.com/en-us/azure/data-factory/concepts-pipelines-activities#data-movement-activities)

**DATA TRANSFORMATION ACTIVITIES:**

**1- Data Flow:**In data flow, First, you need to design data transformation workflow to transform or move data. Then you can call Data Flow activity inside the ADF pipeline. It runs on Scaled out Apache Spark Clusters. There are two types of DataFlows: Mapping and Wrangling DataFlows

**MAPPING DATA FLOW:** It provides a platform to graphically design data transformation logic. You don’t need to write code. Once your data flow is complete, you can use it as an Activity in ADF pipelines.

**WRANGLING DATA FLOW:**It provides a platform to use power query in Azure Data Factory which is available on Ms excel. You can use power query M functions also on the cloud.

**2- Hive Activity:** This is a HD insight activity that executes Hive queries on windows/linux based HDInsight cluster. It is used to process and analyze structured data.

**3- Pig activity:** This is a HD insight activity that executes Pig queries on windows/linux based HDInsight cluster. It is used to analyze large datasets.

**4- MapReduce:**This is a HD insight activity that executes MapReduce programs on windows/linux based HDInsight cluster. It is used for processing and generating large datasets with a parallel distributed algorithm on a cluster.

**5- Hadoop Streaming:** This is a HD Insight activity that executes Hadoop streaming program on windows/linux based HDInsight cluster. It is used to write mappers and reducers with any executable script in any language like Python, C++ etc.

**6- Spark:**This is a HD Insight activity that executes Spark program on windows/linux based HDInsight cluster. It is used for large scale data processing.

**7- Stored Procedure:**In Data Factory pipeline, you can use execute Stored procedure activity to invoke a SQL Server Stored procedure. You can use the following data stores: Azure SQL Database, Azure Synapse Analytics, SQL Server Database, etc.

**8- U-SQL:** It executes U-SQL script on Azure Data Lake Analytics cluster. It is a big data query language that provides benefits of SQL.

**9- Custom Activity:** In custom activity, you can create your own data processing logic that is not provided by Azure. You can configure .Net activity or R activity that will run on Azure Batch service or an Azure HDInsight cluster.

**10- Databricks Notebook:**It runs your databricks notebook on Azure databricks workspace. It runs on Apache spark.

**11- Databricks Python Activity:** This activity will run your python files on Azure Databricks cluster.

**12- Azure Functions:** It is Azure Compute service that allows us to write code logic and use it based on events without installing any infrastructure. It stores your code into Storage and keep the logs in application Insights.Key points of Azure Functions are :

1- It is a Serverless service.

2- It has Multiple languages available : C#, Java, Javascript, Python and PowerShell

3- It is a Pay as you go Model.

To know more about Data Transformation activity, use below link:

[Pipelines and activities in Azure Data Factory - Azure Data Factory | Microsoft Docs](https://docs.microsoft.com/en-us/azure/data-factory/concepts-pipelines-activities#data-transformation-activities)

**3- Control Flow Activities:**

**1- Append Variable Activity:**It assigns a value to the array variable.

**2- Execute Pipeline Activity:** It allows you to call Azure Data Factory pipelines.

**3- Filter Activity:** It allows you to apply different filters on your input dataset.

**4- For Each Activity:** It provides the functionality of a for each loop that executes for multiple iterations.

**5- Get Metadata Activity:**It is used to get metadata of files/folders. You need to provide the type of metadata you require: childItems, columnCount, contentMDS, exists, itemName, itemType, lastModified, size, structure, created etc.

**6- If condition Activity:** It provides the same functionality as If statement, it executes the set of expressions based on if the condition evaluates to true or false.

**7- Lookup Activity:** It reads and returns the content of multiple data sources such as files or tables or databases. It could also return the result set of a query or stored procedures.

**8- Set Variable Activity:** It is used to set the value to a variable of type String, Array, etc.

**9- Switch Activity:**It is a Switch statement that executes the set of activities based on matching cases.

**10- Until Activity:** It is same as do until loop. It executes a set of activities until the condition is set to true.

**11- Validation Activity:** It is used to validate the input dataset.

**12- Wait Activity:**It just waits for the given interval of time before moving ahead to the next activity. You can specify the number of seconds.

**13- Web Activity:** It is used to make a call to REST APIs. You can use it for different use cases such as ADF pipeline execution.

**14- Webhook Activity:** It is used to to call the endpoint URLs to start/stop the execution of the pipelines. You can call external URLs also.