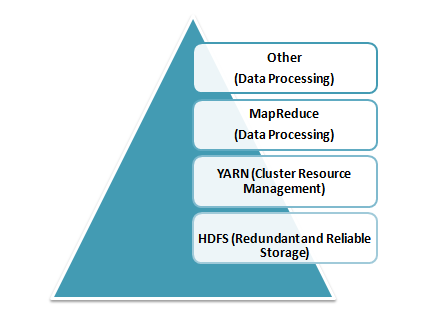
# Explain the core components of Hadoop.

**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]**

# What is partitioning in Hive?

In general partitioning in Hive is a logical division of tables into related columns such as date, city, and department based on the values of partitioned columns. Then these partitions are subdivided into buckets so that they provide extra structure to the data that may be used for more efficient querying.

# Bucketing Hive

 Partitions are sub-divided into **buckets,** to provide extra structure to the data that may be used for more efficient querying. Bucketing works based on the value of hash function of some column of a table.

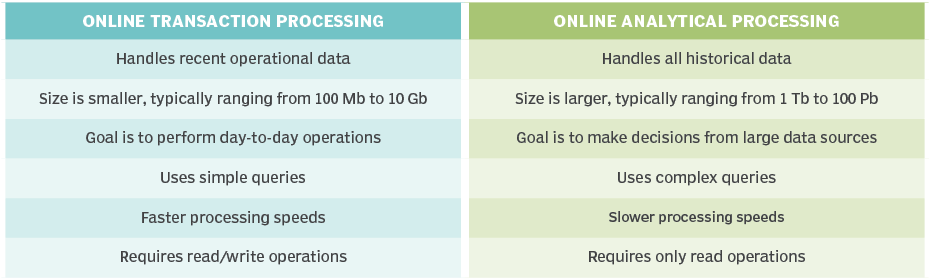
# OLTP

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.

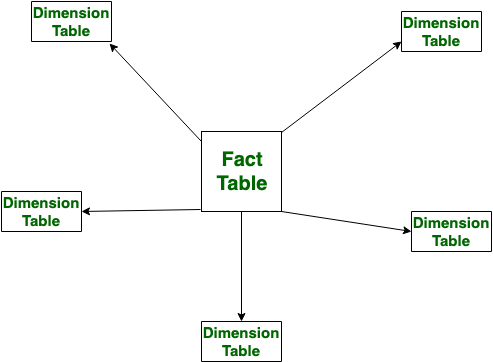
The basic difference between OLTP and OLAP is that OLTP is an online database modifying system, whereas, OLAP is an online database query system



# Fact and Dimension Table

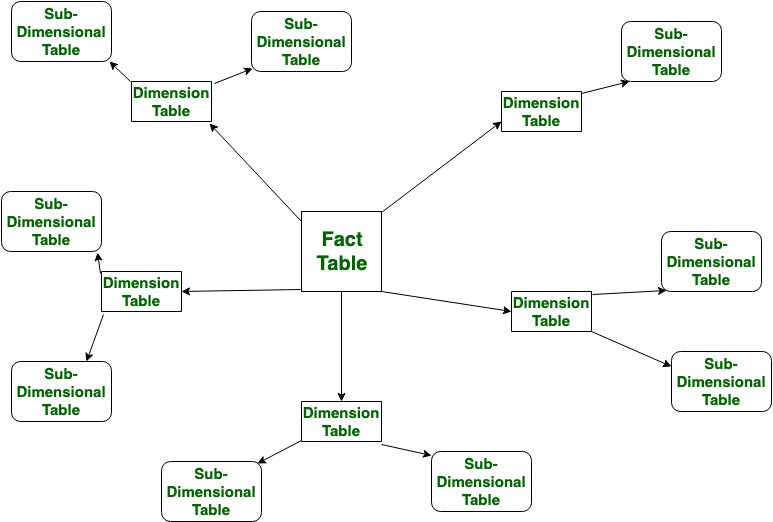
A fact table holds the data to be analyzed, and a dimension table stores data about the ways in which the data in the fact table can be analyzed.

Star Schema  
Star schema is the type of multidimensional model which is used for data warehouse. In star schema, the fact tables and the dimension tables are contained. In this schema fewer foreign-key join is used. This schema forms a star with fact table and dimension tables



# Snowflake Schema

# Snowflake Schema is also the type of multidimensional model which is used for data warehouse. In snowflake schema, the fact tables, dimension tables as well as sub dimension tables are contained. This schema forms a snowflake with fact tables, dimension tables as well as sub-dimension tables.



|  |  |  |
| --- | --- | --- |
| 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.

**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:

# Spark Architecture, an open-source, framework-based component that processes a large amount of unstructured, semi-structured, and structured data for analytics, is utilised in Apache Spark.

## Spark architecture works on master/slave node. it contains three layers

## Driver manager

## Cluster manager

## Worker node

## Inbetween the driver manager and worker node is a cluster manager

## In master node, you have the *driver program*, which drives our application. The code you are writing behaves as a driver program or if you are using the interactive shell, the shell acts as the driver program.Inside the driver program, the first thing you do is, you *create* a *Spark Context.* Assume that the Spark context is a gateway to all the Spark functionalities. It is similar to our database connection.

# How does Spark work?

**STEP 1:**The client submits spark user application code. When an application code is submitted, the driver implicitly converts user code that contains transformations and actions into a logically *directed acyclic graph* called ***DAG.***At this stage, it also performs optimizations such as pipelining transformations.

**STEP 2:** After that, it converts the logical graph called DAG into physical execution plan with many stages. After converting into a physical execution plan, it creates physical execution units called tasks under each stage. Then the tasks are bundled and sent to the cluster.

**STEP 3:** Now the driver talks to the cluster manager and negotiates the resources. Cluster manager launches executors in worker nodes on behalf of the driver. At this point, the driver will send the tasks to the executors based on data placement. When executors start, they register themselves with drivers. So, the driver will have a complete view of executors that are executing the task

**STEP 4:** During the course of execution of tasks, driver program will monitor the set of executors that runs. Driver node also schedules future tasks based on data placement.

# 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:

## Transforms the input RDD and creates new 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)

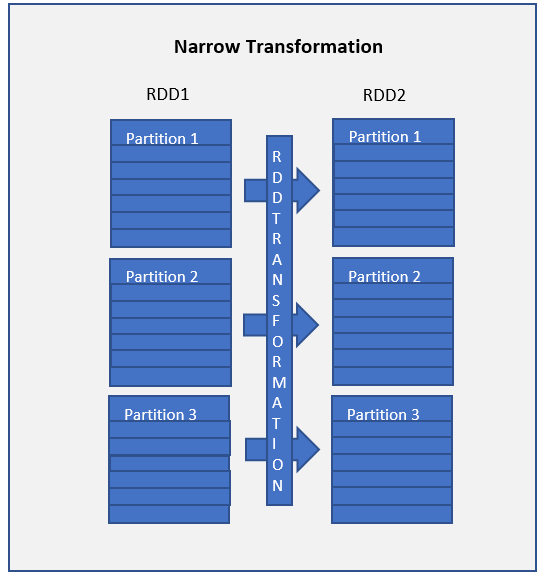
# Two types of transformations in SPARK:

* Wide Transformations
* Narrow Transformations

# Narrow Transformations:

These types of transformations convert each input partition to only one output partition. When each partition at the parent RDD is used by at most one partition of the child RDD or when each partition from child produced or dependent on single parent RDD.

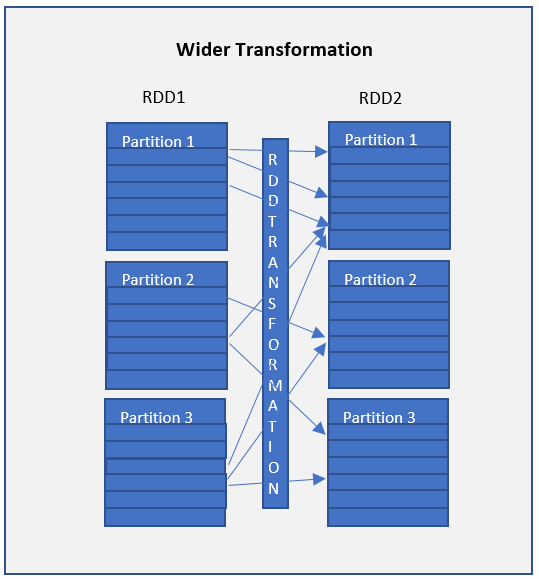
* This kind of transformation is basically fast.
* Does not require any data shuffling over the cluster network or no data movement.
* Operation of map () and filter () belongs to this transformation.



# 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 speed might be significantly affected as it might be required to shuffle data around different nodes when creating new partitions.
* 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.

## Pesris

## :

## 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

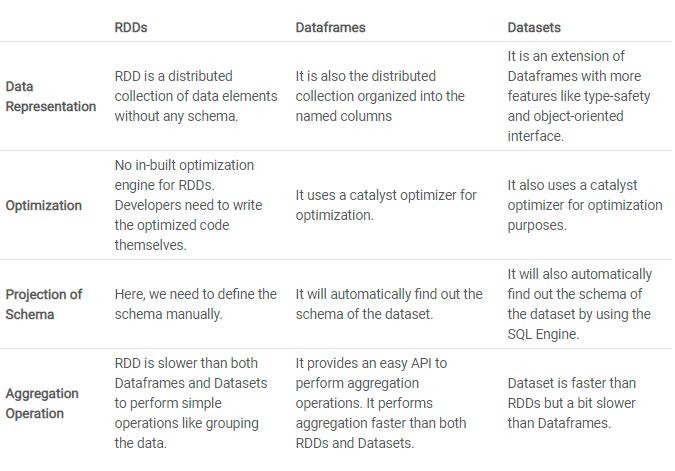
## Core:

## 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



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

DAG (Directed Acyclic Graph) in Apache Spark is an alternative to the MapReduce. It is a programming style used in distributed systems. In MapReduce, you just have two functions (map and reduce), while DAG has multiple levels that form a tree structure. Hence, DAG execution is faster than MapReduce because intermediate results do not write to disk.

# Advantage of DAG:

* The lost RDD can recover using the Directed Acyclic Graph.
* Map Reduce has just two queries the map, and reduce but in DAG you have multiple levels. So to execute SQL query, DAG is more flexible.
* DAG helps to achieve fault tolerance. Thus you can recover the lost data.
* It can do a better global optimization than a system like Hadoop Map Reduce.

# 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.
* Immutability of RDD and lineage graph helps in recreating missing RDD in case of failure making it fault tolerant

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 & Accumulator Variable

The main difference between these two is Broadcast variable is primarily used for reading some data across worker node .Accumulator is used for writing some data across worker node.

# 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. The following code block has the details of a Broadcast class for PySpark.

## 

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

## To submit broadcast variable:

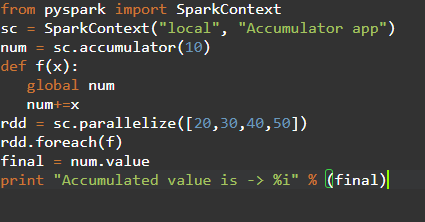
## Spark-submit broadcast.py

## OUTPUT:

## Stored data 🡪[‘raju’,’tharun’]

# Accumulator

Accumulator variables are used for aggregating the information through associative and commutative operations. For example, you can use an accumulator for a sum operation or counters (in MapReduce).



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.

# What is cluster mode and client mode in Spark?

Cluster mode puts the Spark driver in an application master process managed by YARN on the cluster. In client mode, the driver can run in the client process without an application master, which simplifies deployment and makes the driver easier to test.

# 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.

# 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 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.

In 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 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. |

# Class vs static method:

# 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 behaviour 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()

## Difference between Rank and Dense Rank?

This a sql question but I included it because we can expect this question if we go in window-partition section. Suppose, we have a dataset as given below:

Name Salary Rank Dense\_rank  
Abid 1000 1 1  
Ron 1500 2 2  
Joy 1500 2 2  
Aly 2000 4 3  
Raj 3000 5 4

## Here salary is in increasing order and we are getting rank() an dense\_rank() for the dataset. As Ron and Joy have same salary they get the same rank, but rank() will leave hole and keep “3” as empty whereas dense\_rank() will fill all the gaps even though same values are encountered

# 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 you 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
* You 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

# Sqoop

## Apache Sqoop in Hadoop is used to fetch structured data from RDBMS systems like Teradata, Oracle, MySQL, MSSQL, PostgreSQL and on the other hand

# Flume

## Apache Flume is used to fetch data that is stored on various sources as like the log files on a Web Server or an Application Server.

# 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?

select salary AS SecondHighestSalary from ( select row\_number () over ( order by salary desc ) row\_ , salary from CUSTOMERS ) as emp where emp.row\_ = 2

select salary AS ThirdHighestSalary from ( select row\_number () over ( order by salary desc ) row\_ , salary from CUSTOMERS ) as emp where emp.row\_ = 3

# **What is the ACID property in a database?**

ACID stands for Atomicity, Consistency, Isolation, and Durability. It is used to ensure that the data transactions are processed reliably in a database system.

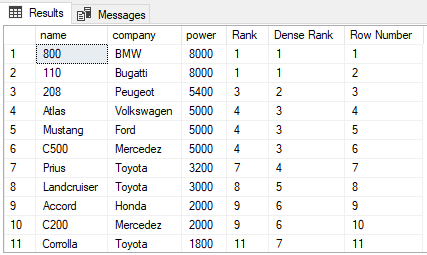
* **Atomicity** - each statement in a transaction (to read, write, update or delete data) is treated as a single unit. Either the entire statement is executed, or none of it is executed. This property prevents data loss and corruption from occurring if, for example, if you’re streaming data source fails mid-stream.
* **Consistency** - ensures that transactions only make changes to tables in predefined, predictable ways. Transactional consistency ensures that corruption or errors in your data do not create unintended consequences for the integrity of your table.
* **Isolation** - when multiple users are reading and writing from the same table all at once, isolation of their transactions ensures that the concurrent transactions don’t interfere with or affect one another. Each request can occur as though they were occurring one by one, even though they're actually occurring simultaneously.
* **Durability** - ensures that changes to your data made by successfully executed transactions will be saved, even in the event of system failure

# Rank vs Dense Rank vs Row \_Number:

RANK function skips the next N-1 ranks if there is a tie between N previous ranks.

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

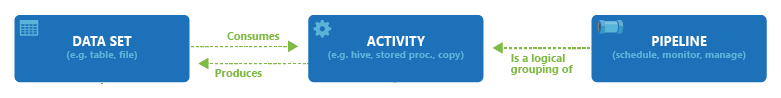
ROW\_NUMBER function has no concern with ranking. It simply returns the row number of the sorted records



# AZURE

# **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.

# **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.

# **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.

## 1.What is a Lineage Graph?

This is another frequently asked spark interview question. A Lineage Graph is a dependencies graph between the existing RDD and the new RDD. It means that all the dependencies between the RDD will be recorded in a graph, rather than the original data.

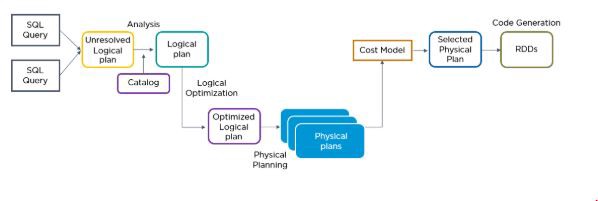
The need for an RDD lineage graph happens when we want to compute a new RDD or if we want to recover the lost data from the lost persisted RDD. Spark does not support data replication in memory. So, if any data is lost, it can be rebuilt using RDD lineage. It is also called an RDD operator graph or RDD dependency graph.

## What is the role of accumulators in Spark?

Accumulators are variables used for aggregating information across the executors. This information can be about the data or API diagnosis like how many records are corrupted or how many times a library API was called.

## What is the role of Catalyst Optimizer in Spark SQL?

Catalyst optimizer leverages advanced programming language features (such as Scala’s pattern matching and quasi quotes) in a novel way to build an extensible query optimizer.



## What is shuffling in Spark? When does it occur?

Shuffling is the process of redistributing data across partitions that may lead to data movement across the executors. The shuffle operation is implemented differently in Spark compared to Hadoop.

Shuffling has 2 important compression parameters:

spark.shuffle.compress — checks whether the engine would compress shuffle outputs or not spark.shuffle.spill.compress — decides whether to compress intermediate shuffle spill files or not

It occurs while joining two tables or while performing byKey operations such as GroupByKey or ReduceByKey

## What is a Parquet file and what are its advantages?

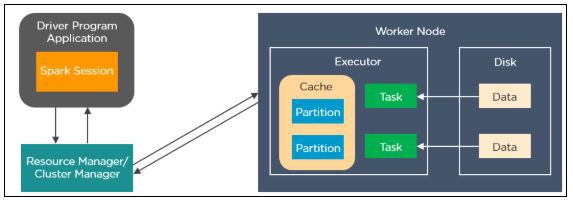
Parquet is a columnar format that is supported by several [data processing](https://www.simplilearn.com/what-is-data-processing-article) systems. With the Parquet file, Spark can perform both read and write operations.

Some of the advantages of having a Parquet file are:

* It enables you to fetch specific columns for access.
* It consumes less space
* It follows the type-specific encoding
* It supports limited I/O operations

## Explain how Spark runs applications with the help of its architecture.

This is one of the most frequently asked spark interview questions, and the interviewer will expect you to give a thorough answer to it.



Spark applications run as independent processes that are coordinated by the SparkSession object in the driver program. The resource manager or cluster manager assigns tasks to the worker nodes with one task per partition. Iterative algorithms apply operations repeatedly to the data so they can benefit from caching datasets across iterations. A task applies its unit of work to the dataset in its partition and outputs a new partition dataset. Finally, the results are sent back to the driver application or can be saved to the disk.

## What are the different cluster managers available in Apache Spark?

* Standalone Mode: By default, applications submitted to the standalone mode cluster will run in FIFO order, and each application will try to use all available nodes. You can launch a standalone cluster either manually, by starting a master and workers by hand, or use our provided launch scripts. It is also possible to run these daemons on a single machine for testing.
* Apache Mesos: Apache Mesos is an open-source project to manage computer clusters, and can also run Hadoop applications. The advantages of deploying Spark with Mesos include dynamic partitioning between Spark and other frameworks as well as scalable partitioning between multiple instances of Spark.
* Hadoop YARN: Apache [YARN](https://www.simplilearn.com/tutorials/hadoop-tutorial/yarn) is the cluster resource manager of Hadoop 2. Spark can be run on YARN as well.
* Kubernetes: [Kubernetes](https://www.simplilearn.com/tutorials/kubernetes-tutorial/what-is-kubernetes" \t "_blank) is an open-source system for automating deployment, scaling, and management of[containerized applications.](https://www.simplilearn.com/docker-alternatives-article)

## How metadata is managed in Databricks?

* Hive metastore: Azure Databricks stores all the metadata for the built-in Hive metastore as a managed service. An instance of the metastore deploys to each cluster and securely accesses metadata from a central repository for each customer workspace.
* [External metastore](https://learn.microsoft.com/en-us/azure/databricks/data/metastores/): you can also bring your own metastore to Azure Databricks.

## Broadcast Join

Broadcasting works in Spark by broadcasting the data from executors to the drivers and then have the drivers broadcast it back to the executors.

Joining two tables is one of the main transactions in Spark. It mostly requires shuffle which has a high cost due to data movement between nodes. If one of the tables is small enough, any shuffle operation may not be required. By broadcasting the small table to each node in the cluster, shuffle can be simply avoided.

Let’s assume that you are working on a force field dataset and have a data frame named df\_work\_order which contains the work orders the force field teams handle. And additionally, you have another data frame that includes the city information of field force teams. While there are more than 100M rows and lots of columns in df\_work\_order, there are approximately 100 records in the df\_city data frame. To add city information to the df\_work\_order data frame, broadcasting the small table would work fine.

df\_work\_order = df\_work\_order.join(broadcast(df\_city), on=[‘TEAM\_NO’], how=’inner’)

The maximum size for the broadcast table is **8GB**.

## What is a catalog?

A catalog is the highest abstraction (or coarsest grain) in the Databricks Lakehouse relational model. Every database will be associated with a catalog. Catalogs exist as objects within a metastore.

Before the introduction of Unity Catalog, Azure Databricks used a two-tier namespace. Catalogs are the third tier in the Unity Catalog namespacing model:

Copy

catalog\_name.database\_name.table\_name

The built-in Hive metastore only supports a single catalog, hive\_metastore.

## How to write the in single partition on spark

df.coalesce(1).write.csv('result.csv')

## How to register the UDf

spark.udf.register("convertUDF", convertCase,StringType())

## Difference between RDD and Data frames

RDD is slow as it does not have any optimization

* RDD — It can easily and efficiently process data which is structured as well as unstructured. But like Dataframe and DataSets, RDD does not infer the schema of the ingested data and requires the user to specify it.
* DataFrame — It works only on structured and semi-structured data. It organizes the data in the named column. DataFrames allow the Spark to manage schema.
* RDD — Whenever [Spark](https://data-flair.training/blogs/important-apache-spark-terminologies-and-concepts-you-must-know/) needs to distribute the data within the cluster or write the data to disk, it does so use Java serialization.
* DataFrame — Spark DataFrame Can serialize the data into off-heap storage ([in memory](https://data-flair.training/blogs/apache-spark-in-memory-computing/)) in binary format and then perform many transformations directly on this off heap memory because spark understands the schema
* RDD — There is overhead for garbage collection that results from creating and destroying individual objects.
* DataFrame — Avoids the garbage collection costs in constructing individual objects for each row in the dataset.

## Wide and narrow transformation in spark

Example 1-Let us see a simple example of map transformation on an RDD.

val listRDD = sc.parallelize(List(“cat”,”hat”,”mat”,”cat”,”mat”))

val mappedWordsRDD = listRDD.map(x => (x, 1))

If you check for dependencies in the scala shell, it shows OneToOneDependency

scala> mappedWordsRDD.dependencies

res3: Seq[org.apache.spark.Dependency[\_]] = List(org.apache.spark.OneToOneDependency@29aa5291)

So, this dependencies method on RDDs show you a sequence of Dependency objects.This shows the dependencies used by spark scheduler to know about RDDs dependency on other RDDs.This is showing one-to-one dependency because each partition of the parent RDD(listRDD) is used by at most one partition of the child RDD(mappedWordsRDD) and that is why it a Narrow Transformation.

transformations which require shuffling are Wide transformations.In wide transformations RDDs have multiple dependencies.

Transformations like groupByKey, combineByKey, join require shuffling of data because the data required to compute the records in a single partition may be available in multiple partitions of the parent RDD.In general, avoiding shuffle will make your program run faster.

## ****Spark optimization technique Performance Boosting****

1 — Join by broadcast

2 — Replace Joins & Aggregations with Windows

3 — Minimize Shuffles

4 — Cache Properly

5 — Break the Lineage — Checkpointing

6 — Avoid using UDFs

7 — Tackle with Skew Data — salting & repartition

8 — Utilize Proper File Formats — Parquet

9 — Use toPandas with pyArrow

# TAKEAWAYS

* Don’t use count() when you don’t need to return the exact number of rows. To check if data frame is empty, len(df.head(1))>0 will be more accurate considering the performance issues.
* Do not use show() in your production code.
* It is a good practice to use df.explain() to get insight into the internal representation of a data frame in Spark(the final version of the physical plan).
* Always try to minimize the data size by filtering irrelevant data(rows/columns) before joinings.
* Monitor Spark applications online/offline. It might give you [any clues](https://developer.hpe.com/blog/Ql2DXNL4rmhWB8AEDMQz/tips-and-best-practices-to-take-advantage-of-spark-2x) about unbalanced data partitions, where the jobs are stuck, and query plans. An alternative to Spark UI might be [Ganglia](https://medium.com/knoldus/ganglia-cluster-monitoring-monitoring-spark-cluster-77786b8bd38d).
* Basically, avoid using loops.
* Focus on built-in functions rather than custom solutions.
* Ensure that key columns in join operation do not include null values.
* [Put the bigger dataset on the left in joins](https://medium.com/softwaresanders/improving-your-apache-spark-application-performance-ec5d64bc3c75).
* Keep in mind that Spark runs with Lazy Evaluation logic. So, nothing is triggered until an action is called. That might result in [meaningless error](https://medium.com/enigma-engineering/things-i-wish-id-known-about-spark-when-i-started-one-year-later-edition-d767430181ed) codes.
* Unpersist the data in the cache, if you don’t need it for the rest of the code.
* Close/stop your Spark session when you are done with your application.
* In [Spark 3.0](https://medium.com/@ch.nabarun/whats-new-in-spark-3-8250a65b3144), significant improvements are achieved to tackle performance issues by [Adaptive Query Execution](https://databricks.com/blog/2020/05/29/adaptive-query-execution-speeding-up-spark-sql-at-runtime.html), take upgrading the version into consideration.
* Prefer data frames to RDDs for data manipulations.
* In general, tasks larger than about **20 KiB** are probably worth optimizing.
* In general, it is recommended **2–3 tasks per CPU core** in your cluster.
* It is always good to have a block within **128MB** per partition to achieve parallelism.
* Csv and Json data file formats give high write performance but are slower for reading, on the other hand, Parquet file format is very fast and gives the best performance in reading and slower than the other mentioned file formats concerning writing operation.
* The physical plan is read from the bottom up, whereas the DAG is read from the top down.
* The Exchange means a shuffle occurred between stages, and it is basically a performance degradation.
* An excessive number of stages might be a sign of a performance problem.
* [Garbage collection(GC)](https://spark.apache.org/docs/latest/tuning.html#garbage-collection-tuning) is another key factor that might cause performance issues. Check it out from the Executors tab of Spark UI. You may typically use Java GC options in any GC-related case.

# 1. What is SQL?

SQL is a non-procedural programming language developed by IBM in the 1970s and then later by Oracle. SQL is used by almost all relational databases to write queries, access, edit, and retrieve data.

# 2. What is a database?

A database is an organized collection of digital information, or data, stored in a computer system. Unlike spreadsheets, a database can handle massive quantities of information while allowing multiple users access to the same database to run secure and independent queries.

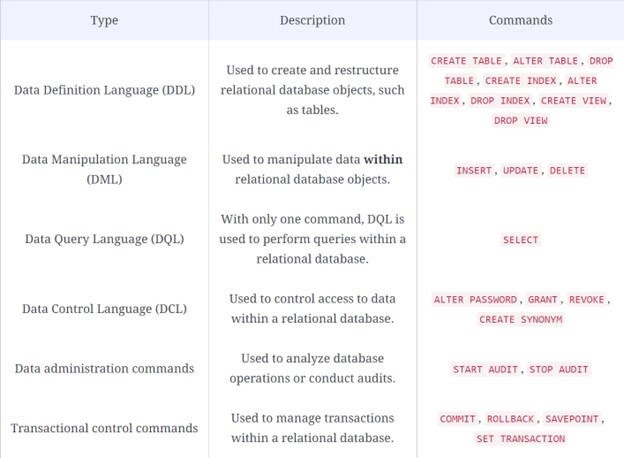
# 3. What is an RDBMS, and how is it different from a traditional DBMS?

An RDBMS or Relational Database Management System is essentially a database utilizing a tabular schema to organize multiple data elements related to each other. An RDBMS lets your define, create, and maintain relational databases in addition to providing controlled access to the data within.  
A DBMS or Database Management System functions similarly to the RDBMS described above. However, DBMS data elements are stored as files rather than in tabular form, and there is no relationship between different data elements.

**Related article**: [What is a relational database? A deep dive](https://www.educative.io/blog/relational-database-deep-dive?eid=5082902844932096)

# 4. Explain the different types of SQL commands.

There are six basic types of SQL commands.



# 5. What is the difference between SQL and MySQL?

SQL is a programming language, while MySQL is a popular, open-source RDBMS. MySQL is used to store and organize data, while SQL is used to access, edit, update and maintain data in MySQL.

**Related article**: [MySQL tutorial: The beginner’s guide to using MySQL](https://www.educative.io/blog/mysql-tutorial?eid=5082902844932096)

# 6. What is a PRIMARY KEY constraint?

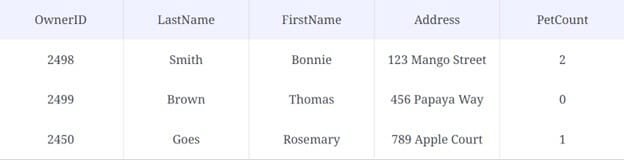
A PRIMARY KEY constraint is a column (or combination of columns) used to designate each table row with a unique identifier. You can think of a primary key as having a similar function to national government-issued identification numbers, a citizen's Social Security Number, or a vehicle identification number (VIN).

**Note**: There’s a limit of one PRIMARY KEY constraint per table. All columns defined within a PRIMARY KEY constraint must be defined as NOT NULL.

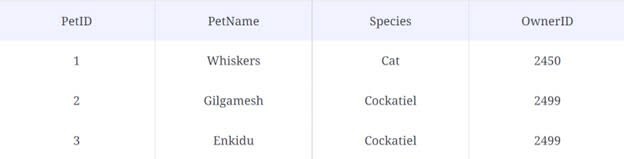
# 7. What is a FOREIGN KEY constraint?

A FOREIGN KEY is a column or collection of fields in a table referencing a PRIMARY KEY in another table. The table containing the primary key is known as the parent table, and the table containing the foreign key is called the child table.

For example, the PRIMARY KEY in the parent table below is OwnerID. The PRIMARY KEY uniquely identifies individual pet owners.



For this child table, the PRIMARY KEY is PetID, and the OwnerID column is a FOREIGN KEY because it references the primary key of another table.



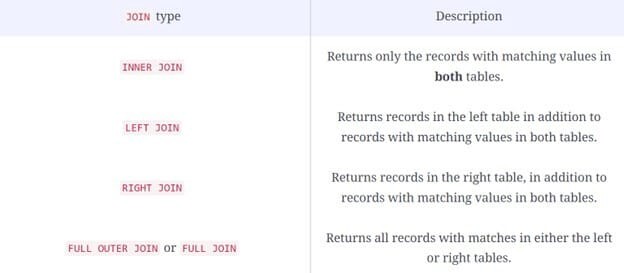
# 8. What is a UNIQUE constraint?

Like the PRIMARY KEY, the UNIQUE constraint also ensures that each value is different from the others in its column. However, tables can have multiple columns with UNIQUE constraints, unlike the PRIMARY KEY constraint, limited to just one.

# 9. What are SQL joins? What are the different types of joins?

In SQL, a JOIN clause combines rows of data in different tables with a shared column. You can SELECT and return records with matching values in both tables based on this relationship.

There are four kinds of JOIN clauses in SQL:



**Related article**: [What are SQL joins? Types of SQL joins explained](https://www.educative.io/blog/what-are-sql-joins?eid=5082902844932096)

# 10. What is a self-join?

A JOIN clause combines rows from two or more tables based on a related column between them. A self-join is a regular join, but the table is joined with itself – this is extremely useful for comparisons within a table.

Joining a table with itself means that each table row is combined with itself and with every other row of the table.

# 11. What is a cross join (Cartesian join)?

In SQL, the cross join combines each row of the **first** table with each row of the **second** table. It is also known as the Cartesian join since it returns the Cartesian product of the sets of rows from the joined tables.

# 12. What’s the difference between a WHERE clause and a HAVING clause?

The WHERE clause can be used to establish the first condition that groups and returns only the rows that meet that condition into a result set. Then, secondary conditions can be applied using the HAVING clause to return only the groups within that set that meet your new criteria.

# 13. What’s the difference between a TRUNCATE command and a DELETE command?



The TRUNCATE command is faster than DELETE, but unlike the DELETE command, data cannot be rolled back after using it to recover data that has been mistakenly deleted.

# 14. What is a query?

In the context of this article, a query is a set of instructions written in a query language like SQL that allows an individual to access information held in a database.

# 15. What is a subquery?

A subquery or **nested query** is a query within a query.

There are two types of subqueries: **Correlated** and **Non-correlated**.

* **Correlated** subqueries refer to a column in a table specified by the FROM keyword of the main query.
* **Non-correlated** subqueries are independent and their output is substituted in the main query.

# 16. What are UNION, UNION ALL, MINUS, and INTERSECT set operators?

The UNION operation combines the results of two or more SELECT statements. For example, getting the UNION of sets A and B, this operation would return all rows from both sets, excluding any duplicate rows.

The UNION ALL operation does the same thing as UNION, but includes duplicate rows in its result set.

The INTERSECT operation combines the results of two SELECT statements but only returns the rows with matching values in **both** sets.

The MINUS operation combines the results of two SELECT statements but only returns rows with values that belong to the first set of the result.

# 17. What are Normalization and Denormalization?

**Normalization** refers to the methods used to remove redundancies and inconsistencies in a database.

**Denormalization** refers to methods used to improve the performance of queries.

Normalization introduces more tables to a database, whereas Denormalization reduces the number of tables.

# 18. What are scalar functions?

Scalar functions are defined by the user and return a single value (i.e., int, char, float, etc.) based on the input value.

**Common SQL scalar functions**:

* CONCAT() concatenates two or more character strings.
* FORMAT() sets the format to display a collection of values.
* LEN() calculates the total length of a given column.
* MID() extracts substrings from a collection of string values.
* ROUND() rounds the integer value for a numeric field.
* NOW() returns the current date and time.
* RAND() calculates a random collection of numbers of a given length.

# 19. What are aggregate functions?

In SQL, aggregate functions (also known as group functions) are applied to a group of values (or all values) to calculate and return a single value.

**Common SQL aggregate functions**:

* AVG calculates the average or mean of all values in a group.
* COUNT calculates the number of rows in group, **including** rows with NULL values.
* MIN and MAX returns the smallest and largest value in a group, respectively.
* SUM returns the sum of all non-NULL values in a group.
* STDDEV calculates the standard deviation.
* VARIANCE calculates the variance.

# 20. What is a stored procedure?

Instead of writing the same SQL query multiple times, you can save it as a stored procedure and call on it whenever necessary to execute it.

Store an SQL query:

CREATE PROCEDURE procedure\_name  
AS  
sql\_statements  
GO;

Execute a stored procedure:

EXEC procedure\_name;

# 21. What is the SELECT statement?

The SELECT statement is used in SQL queries to store specific data elements or fields from a table and return them in a result set.

The SELECTsyntax:

SELECT column\_1, column\_2, ...  
FROM table\_name;

To select all data elements from a table, use

SELECT \* FROM table\_name;

# 22. What is an index?

An SQL index is a lookup table used by the database search engine to find and retrieve data quickly. An index can help makeSELECT and WHERE clauses faster but can slow down the use of UPDATE and INSERT statements.

To create an index:

CREATE INDEX index\_name ON table\_name;

# 23. What are some common clauses used with SELECT queries in SQL?

The basic SQL SELECT statement contains three clauses:

* SELECT specifies the table columns to retrieve
* FROM specifies the tables to access
* WHERE is optional and specifies which rows in the FROM tables to use

The GROUP BY clause is used with aggregate functions to group the result set according to specified columns.

The HAVING clause functions similarly to the WHERE clause but allows the use of aggregate functions.

The ORDER BY clause sorts the result set in ascending (ASC) or descending (DESC) order according to a specified column.

When writing your SELECT queries, make sure that your syntax follows this order:

SELECT column\_name(s)  
FROM table\_name  
WHERE condition  
GROUP BY column\_name(s)  
HAVING condition  
ORDER BY column\_name(s);

# 24. What are character manipulation functions?

Character manipulation functions can edit, change, or reformat character strings.

For example, you can concatenate two character strings by passing them into the CONCAT function using a SELECT query.

# 25. What is an SQL Server cursor? How do you use it?

When you want to process result sets one row at a time, you can use a database cursor, a control structure that allows you to traverse records in a database. Cursors can be used to point to individual rows in a group of rows.

You can DECLARE a cursor after any variable declaration.

DECLARE variable\_name CHAR(20)   
DECLARE cursor\_name CURSOR FOR  
SELECT column\_name   
FROM table\_name

# 26. What are the different types of indexes?

* **Clustered indexes** are **clustered** together with the main body of data. A clustered index sorts and stores rows of data in a table or view sequentially, based on key values of the table to match the order of the index. There can only be **one** clustered index per table.
* **Non-clustered indexes** are separate from, and cannot be used to store or sort data in the main table. The key values of the index, and not the table are used to define the order of a non-clustered index.
* **Column store indexes** are a standard form of index that efficiently stores data in a column-based format, rather than row-oriented.
* **Filtered indexes** are used to index a section of rows within a table.
* **Hash indexes** are arrays, and use the Hash function F(K, N), where K is critical and N is the number of slots containing a pointer and row.
* **Unique indexes** assign unique values to every row of data, so that the index key does not contain any duplicates.

# 27. What is the difference between a clustered index and a non-clustered index?

The order of rows in a **clustered index** corresponds to the order of rows in the database. A table can only have one clustered index at a time.

A **non-clustered index** functions similarly to a clustered index, but is slower and creates a separate entity within the table that references the original table. A table can have multiple non-clustered indices.

# 28. What are ACID properties?

The ACID properties refer to properties that must be followed for transactions in a database management system to remain consistent.

* **Atomicity**: The entire transaction takes place at once or not at all.
* **Consistency**: A database must be consistent before and after a transaction takes place.
* **Isolation**: Transactions occur independently and can run concurrently with others.
* **Durability**: Updates to the database must be stored in and written to disk so that transaction records can persist in the event of a system failure.

# 29. What is a schema?

An SQL schema is an abstract representation of logically structured data elements. Database schemas in SQL are defined at the logical level by a database user known as the schema owner.

# 30. What is an alias command?

The alias (AS) command makes columns or tables easier to read by giving them temporary names for the duration of a query.

# 31. How do you create empty tables with the same structure as another table?

You can use shallow cloning to create a copy of an existing table’s data structure and column attributes.

CREATE TABLE new\_table LIKE table\_1;

This command creates an empty table based on the parent table.

# 32. How do you select unique records from a table?

The SELECT DISTINCT clause will only return unique values from a table.

# 33. What is the default ordering of data using the order by clause? How could it be changed?

The default ordering of data is ascending (ASC). You can change the order by using the descending (DESC) keyword with the ORDER BY clause like so:

SELECT \* FROM table\_name ORDER BY column\_name DESC;

# 34. What are some case manipulation functions in SQL?

* LOWER or LCASE takes in a given character string and converts it to lower case.
* UPPER or UCASE takes in a given character string and converts it to upper case.

# 35. What’s the standard syntax for group functions?

The general syntax is:

SELECT column\_name, group\_function(column\_name)  
FROM table\_name  
WHERE condition  
GROUP BY column\_name  
ORDER BY column\_name

# 39. What are entities and relationships?

An **entity** can be a real-world object that can be identified by a collection of related attributes or properties. An example of an entity in a zoo database might include zookeepers, veterinarians, different public outreach initiatives, or species of animals.

**Relationships** are connections between entities that are associated with each other.

The logical relationship between entities creates a database.

# 40. What is collation? What are the different collation sensitivity?

Collation is a configuration setting that specifies how a database sorts and compares data. Different collation rules can be configured to determine the correct character sequence used to sort the character data.

Collation sensitivity can be used to specify how different characters are treated.

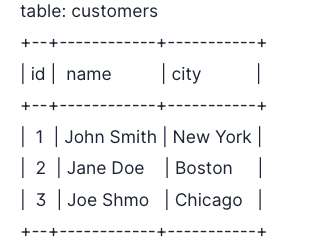
* **Accent sensitivity** differentiates between **a** and **á**.
* **Case sensitivity** differentiates between **A** and **a**.
* **Kana sensitivity** differentiates between Japanese Hiragana and Katakana.
* **Width sensitivity** treats characters of different widths (single-byte and double-byte) differently.

26

# 7. What’s a subquery and how is it used?

A subquery is a SQL query that is embedded within another SQL query. Subqueries are often used to find data that satisfies certain conditions; for example, you could use a subquery to find all customers who live in the same city as a particular customer. Subqueries can be used with various SQL commands, including SELECT, FROM, WHERE, and ORDER BY.

For example, consider the following customer table:



Suppose we want to find all customers who live in the same city as customer with id=1. We could use the following SQL query:

SELECT \* FROM customers WHERE city IN (SELECT city FROM customers WHERE id = 1)

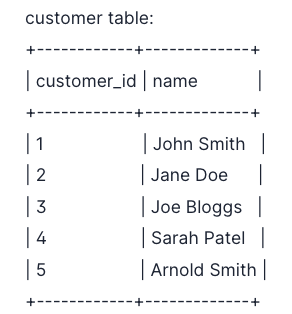
# 11. What’s the difference between a WHERE and a HAVING clause?

The WHERE clause is used to filter rows from a table based on specified conditions; for example, you could use a WHERE clause to find all customers who live in a particular city. The HAVING clause is used to filter rows from a table based on aggregated values; for example, you could use a HAVING clause to find all customers who have placed more than 10 orders.

# 19. What’s the difference between a lag and lead function in SQL?

Lag and lead functions are used to access data from a previous or future row in a table. Lag functions return data from a row that is preceding the current row, while lead functions return data from a row that is following the current row.

For example,



If the current row is customer\_id 3 (Joe Bloggs), a lag function would return customer\_id 2 (Jane Doe), while a lead function would return customer\_id 4 (Sarah Connor).

**20. Write a SQL query to select the second-highest salary in the engineering department.**

SELECT DISTINCT salary

FROM employee

WHERE department = ‘engineering’

ORDER BY salary DESC LIMIT 1 OFFSET 1;

This SQL query will select the second-highest salary from the engineering department by first selecting all distinct salaries from employees in the engineering department, then ordering them in descending order, and finally selecting the top 2 salaries.

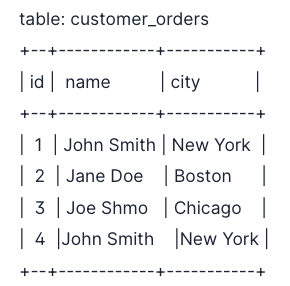
# 25. How do you optimize a SQL query?

There are a few different ways to optimize a SQL query. One way is to make sure that the columns you’re interested in are indexed, so the database can more quickly find the data you’re looking for. Another way is to use the EXPLAIN command to see how the database will execute your query, and then make changes to your query based on that information. Finally, you can use query hints to give database-specific instructions on how to execute your query.

Another way you can optimize a SQL query is to use a tool like SQL Profiler to see where the bottlenecks are in your query and then make changes accordingly.

# 26. How do you find the top 5 customers by sales?

There are a few different ways to find the top 5% of customers by sales. One way is to use the GROUP BY clause to group together rows with the same values in the columns you’re interested in. For example, suppose we have a customer orders table with the following data:



If we wanted to find the top 5 customers by sales, we could use the following SQL query:

SELECT name, city, SUM(sales) AS “Total Sales”

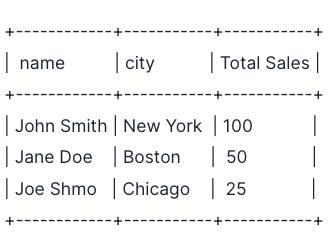
FROM customer\_orders

GROUP BY name, city

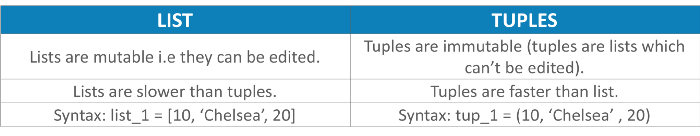
ORDER BY “Total Sales” DESC

LIMIT 5

This would return the following result:



## Q1. What is the difference between list and tuples in Python?



## Q2. What are the key features of Python?

* Python is an **interpreted** language. That means that, unlike languages like C and its variants, Python does not need to be compiled before it is run. Other interpreted languages include PHP and Ruby.
* Python is **dynamically typed**, this means that you don’t need to state the types of variables when you declare them or anything like that. You can do things like x=111 and then x="I'm a string" without error
* Python is well suited to **object orientated programming** in that it allows the definition of classes along with composition and inheritance. Python does not have access specifiers (like C++’s public, private).
* In Python, **functions** are**first-class objects**. This means that they can be assigned to variables, returned from other functions and passed into functions. Classes are also first class objects
* **Writing Python code is quick** but running it is often slower than compiled languages. Fortunately，Python allows the inclusion of C based extensions so bottlenecks can be optimized away and often are. The [numpy package](https://www.edureka.co/blog/python-numpy-tutorial?utm_source=medium&utm_medium=content-link&utm_campaign=python-interview-questions" \t "_blank) is a good example of this, it’s really quite quick because a lot of the number crunching it does isn’t actually done by Python
* Python finds **use in many spheres** — web applications, automation, scientific modeling, big data applications and many more. It’s also often used as “glue” code to get other languages and components to play nice.

## Q3. What type of language is python? Programming or scripting?

Python is capable of scripting, but in general sense, it is considered as a general-purpose programming language

## Q4. How is Python an interpreted language?

An interpreted language is any programming language which is not in machine level code before runtime. Therefore, Python is an interpreted language.

## Q5. What is pep 8?

PEP stands for **Python Enhancement Proposal.**It is a set of rules that specify how to format Python code for maximum readability.

## Q6. How is memory managed in Python?

1. Memory management in python is managed by **Python private heap space**. All Python objects and data structures are located in a private heap. The programmer does not have access to this private heap. The python interpreter takes care of this instead.
2. The allocation of heap space for Python objects is done by Python’s memory manager. The core API gives access to some tools for the programmer to code.
3. Python also has an inbuilt garbage collector, which recycles all the unused memory and so that it can be made available to the heap space.

## Q7. What is namespace in Python?

A namespace is a naming system used to make sure that names are unique to avoid naming conflicts.

## Q8. What is PYTHONPATH?

It is an environment variable which is used when a module is imported. Whenever a module is imported, PYTHONPATH is also looked up to check for the presence of the imported modules in various directories. The interpreter uses it to determine which module to load.

## Q9. What are python modules? Name some commonly used built-in modules in Python?

Python modules are files containing Python code. This code can either be functions classes or variables. A Python module is a .py file containing executable code.

Some of the commonly used built-in modules are:

* os
* sys
* math
* random
* data time
* JSON

## Q10. What are local variables and global variables in Python?

**Global Variables:**

Variables declared outside a function or in global space are called global variables. These variables can be accessed by any function in the program.

**Local Variables:**

Any variable declared inside a function is known a local variable. This variable is present in the local space and not in the global space.

**Example:**

a=2 #Global Variable  
def add():   
b=3 #Local Variable  
c=a+b   
print(c)   
add()

**Output:**5

When you try to access the local variable outside the function add(), it will throw an error.

## Q11. Is python case sensitive?

Yes. Python is a case sensitive language.

## Q12. What is type conversion in Python?

Type conversion refers to the conversion of one data type into another.

1. **int()** — converts any data type into integer type
2. **float()** — converts any data type into float type
3. **ord()** — converts characters into integer
4. **hex(**) — converts integers to hexadecimal
5. **oct()** — converts integer to octal
6. **tuple() —**This function is used to convert to a tuple.
7. **set() —**This function returns the type after converting to set.
8. **list() —**This function is used to convert any data type to a list type.
9. **dict() —**This function is used to convert a tuple of order (key,value) into a dictionary.
10. **str() —**Used to convert integer into a string.
11. **complex(real,imag)**— This functionconverts real numbers to complex(real,imag) number.

## ****Q13. How to install Python on Windows and set path variable?****

**Ans:**To install Python on Windows, follow the below steps:

* Install python from this link: <https://www.python.org/downloads/>
* After this, install it on your PC. Look for the location where PYTHON has been installed on your PC using the following command on your command prompt: cmd python.
* Then go to advanced system settings and add a new variable and name it as PYTHON\_NAME and paste the copied path.
* Look for the path variable, select its value and select ‘edit’.
* Add a semicolon towards the end of the value if it’s not present and then type %PYTHON\_HOME%

## Q14. Is indentation required in python?

Indentation is necessary for Python. It specifies a block of code. All code within loops, classes, functions, etc is specified within an indented block. It is usually done using four space characters. If your code is not indented necessarily, it will not execute accurately and will throw errors as well.

## Q15. What is the difference between Python Arrays and lists?

Arrays and lists, in Python, have the same way of storing data. But, arrays can hold only a single data type elements whereas lists can hold any data type elements.

**Example:**

import array as arr  
My\_Array=arr.array('i',[1,2,3,4])  
My\_list=[1,'abc',1.20]  
print(My\_Array)  
print(My\_list)

**Output:**

array(‘i’, [1, 2, 3, 4]) [1, ‘abc’, 1.2]

## Q16. What are functions in Python?

A function is a block of code which is executed only when it is called. To define a function in Python, the **def** keyword is used.

**Example:**

def Newfunc():  
print("Hi, Welcome to Edureka")  
Newfunc(); #calling the function

**Output:**

Hi, Welcome to Edureka

## Q17. What is \_\_init\_\_?

\_\_init\_\_ is a method or constructor in Python. This method is automatically called to allocate memory when a new object/ instance of a class is created. All classes have the \_\_init\_\_ method.

Here is an example of how to use it.

class Employee:  
def \_\_init\_\_(self, name, age,salary):  
self.name = name  
self.age = age  
self.salary = 20000  
E1 = Employee("XYZ", 23, 20000)  
# E1 is the instance of class Employee.  
#\_\_init\_\_ allocates memory for E1.   
print(E1.name)  
print(E1.age)  
print(E1.salary)

**Output:**

XYZ

23

20000

## Q18. What is a lambda function?

An anonymous function is known as a lambda function. This function can have any number of parameters but, can have just one statement.

**Example:**

a = lambda x,y : x+y   
print(a(5, 6))

**Output:**

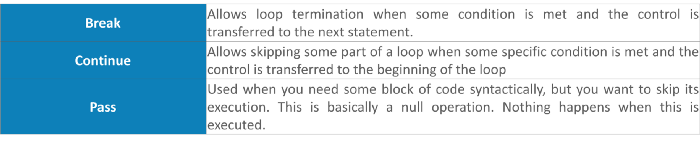
11

## Q19. What is self in Python?

self is an instance or an object of a class. In Python, this is explicitly included as the first parameter. However, this is not the case in Java where it’s optional. It helps to differentiate between the methods and attributes of a class with local variables.

The self variable in the init method refers to the newly created object while in other methods, it refers to the object whose method was called.

## Q20. How does break, continue and pass work?



## Q21. What does [::-1} do?

[::-1] is used to reverse the order of an array or a sequence.

**For example:**

import array as arr  
My\_Array=arr.array('i',[1,2,3,4,5])  
My\_Array[::-1]

**Output**:

array(‘i’, [5, 4, 3, 2, 1])

## Q22. How can you randomize the items of a list in place in Python?

Consider the example shown below:

from random import shuffle  
x = ['Keep', 'The', 'Blue', 'Flag', 'Flying', 'High']  
shuffle(x)  
print(x)

The output of the following code is as below.

['Flying', 'Keep', 'Blue', 'High', 'The', 'Flag']

## Q23. What are python iterators?

Iterators are objects which can be traversed though or iterated upon.

## Q24. How can you generate random numbers in Python?

Random module is the standard module that is used to generate a random number. The method is defined as:

import random   
random.random

The statement random.random() method return the floating point number that is in the range of [0, 1). The function generates random float numbers. The methods that are used with the random class are the bound methods of the hidden instances. The instances of the Random can be done to show the multi-threading programs that creates a different instance of individual threads. The other random generators that are used in this are:

1. randrange(a, b): it chooses an integer and define the range in-between [a, b). It returns the elements by selecting it randomly from the range that is specified. It doesn’t build a range object.
2. uniform(a, b): it chooses a floating point number that is defined in the range of [a,b).Iyt returns the floating point number
3. normalvariate(mean, sdev): it is used for the normal distribution where the mu is a mean and the sdev is a sigma that is used for standard deviation.
4. The Random class that is used and instantiated creates an independent multiple random number generators.

## Q25. What is the difference between range & xrange?

For the most part, xrange and range are the exact same in terms of functionality. They both provide a way to generate a list of integers for you to use, however you please. The only difference is that range returns a Python list object and x range returns an xrange object.

This means that xrange doesn’t actually generate a static list at run-time like range does. It creates the values as you need them with a special technique called yielding. This technique is used with a type of object known as generators. That means that if you have a really gigantic range you’d like to generate a list for, say one billion, xrange is the function to use.

This is especially true if you have a really memory sensitive system such as a cell phone that you are working with, as range will use as much memory as it can to create your array of integers, which can result in a Memory Error and crash your program. It’s a memory hungry beast.

## Q26. How do you write comments in python?

Comments in Python start with a # character. However, alternatively at times, commenting is done using doc-strings(strings enclosed within triple quotes).

**Example:**

#Comments in Python start like this   
print("Comments in Python start with a #")

**Output:**

Comments in Python start with a #

## Q27. What is pickling and unpickling?

Pickle module accepts any Python object and converts it into a string representation and dumps it into a file by using dump function, this process is called pickling. While the process of retrieving original Python objects from the stored string representation is called unpickling.

## Q28. What are the generators in python?

Functions that return an iterable set of items are called generators.

## Q29. How will you capitalize the first letter of string?

In Python, the capitalize() method capitalizes the first letter of a string. If the string already consists of a capital letter at the beginning, then, it returns the original string.

## Q30. How will you convert a string to all lowercase?

To convert a string to lowercase, lower() function can be used.

**Example:**

stg='ABCD'   
print(stg.lower())

**Output:**

abcd

## Q31. How to comment multiple lines in python?

Multi-line comments appear in more than one line. All the lines to be commented are to be prefixed by a #. You can also a very good **shortcut method to comment multiple lines**. All you need to do is hold the ctrl key and **left click** in every place wherever you want to include a # character and type a # just once. This will comment all the lines where you introduced your cursor.

## Q32. What are docstrings in Python?

Docstrings are not actually comments, but, they are **documentation strings**. These docstrings are within triple quotes. They are not assigned to any variable and therefore, at times, serve the purpose of comments as well.

**Example:**

"""  
Using docstring as a comment.  
This code divides 2 numbers  
"""  
x=8  
y=4  
z=x/y  
print(z)

**Output:**

2.0

## Q33. What is the purpose of is, not and in operators?

Operators are special functions. They take one or more values and produce a corresponding result.

is: returns true when 2 operands are true (Example: “a” is ‘a’)

not: returns the inverse of the boolean value

in: checks if some element is present in some sequence

## Q34. What is the usage of help() and dir() function in Python?

Help() and dir() both functions are accessible from the Python interpreter and used for viewing a consolidated dump of built-in functions.

1. Help() function: The help() function is used to display the documentation string and also facilitates you to see the help related to modules, keywords, attributes, etc.
2. Dir() function: The dir() function is used to display the defined symbols.

## Q35. Whenever Python exits, why isn’t all the memory de-allocated?

1. Whenever Python exits, especially those Python modules which are having circular references to other objects or the objects that are referenced from the global namespaces are not always de-allocated or freed.
2. It is impossible to de-allocate those portions of memory that are reserved by the C library.
3. On exit, because of having its own efficient clean up mechanism, Python would try to de-allocate/destroy every other object.

## Q36. What is a dictionary in Python?

The built-in datatypes in Python is called dictionary. It defines one-to-one relationship between keys and values. Dictionaries contain pair of keys and their corresponding values. Dictionaries are indexed by keys.

Let’s take an example:

The following example contains some keys. Country, Capital & PM. Their corresponding values are India, Delhi and Modi respectively.

dict={'Country':'India','Capital':'Delhi','PM':'Modi'}print dict[Country]

**Output:**

India

Similarly if you wish to print the Capital, then:

print dict[Capital]

**Output:**

Delhi

Similarly if you wish to print the PM, then:

print dict[PM]

**Output:**

Modi

## Q37. How can the ternary operators be used in python?

The Ternary operator is the operator that is used to show the conditional statements. This consists of the true or false values with a statement that has to be evaluated for it.

**Syntax**:

The Ternary operator will be given as:  
[on\_true] if [expression] else [on\_false]x, y = 25, 50big = x if x < y else y

**Example:**

The expression gets evaluated like if x<y else y, in this case if x<y is true then the value is returned as big=x and if it is incorrect then big=y will be sent as a result.

## Q38. What does this mean: \*args, \*\*kwargs? And why would we use it?

We use \*args when we aren’t sure how many arguments are going to be passed to a function, or if we want to pass a stored list or tuple of arguments to a function. \*\*kwargs is used when we don’t know how many keyword arguments will be passed to a function, or it can be used to pass the values of a dictionary as keyword arguments. The identifiers args and kwargs are a convention, you could also use \*bob and \*\*billy but that would not be wise.

## Q39. What does len() do?

**Ans:**It is used to determine the length of a string, a list, an array, etc.

**Example:**

stg='ABCD'   
len(stg)

## Q40. Explain split(), sub(), subn() methods of “re” module in Python.

**Ans:** To modify the strings, Python’s “re” module is providing 3 methods. They are:

* split() — uses a regex pattern to “split” a given string into a list.
* sub() — finds all substrings where the regex pattern matches and then replace them with a different string
* subn() — it is similar to sub() and also returns the new string along with the no. of replacements.

## Q41. What are negative indexes and why are they used?

The sequences in Python are indexed and it consists of the positive as well as negative numbers. The numbers that are positive uses ‘0’ that is uses as first index and ‘1’ as the second index and the process goes on like that.

The index for the negative number starts from ‘-1’ that represents the last index in the sequence and ‘-2’ as the penultimate index and the sequence carries forward like the positive number.

The negative index is used to remove any new-line spaces from the string and allow the string to except the last character that is given as S[:-1]. The negative index is also used to show the index to represent the string in correct order.

## Q42. What are Python packages?

Python packages are namespaces containing multiple modules.

## Q43. How can files be deleted in Python?

To delete a file in Python, you need to import the OS Module. After that, you need to use the os.remove() function.

**Example:**

import os   
os.remove("xyz.txt")

## Q44. What are the built-in types of python?

**Ans:**Built-in types in Python are as follows –

* Integers
* Floating-point
* Complex numbers
* Strings
* Boolean
* Built-in functions

## Q45. What advantages do NumPy arrays offer over (nested) Python lists?

1. Python’s lists are efficient general-purpose containers. They support (fairly) efficient insertion, deletion, appending, and concatenation, and Python’s list comprehensions make them easy to construct and manipulate.
2. They have certain limitations: they don’t support “vectorized” operations like element-wise addition and multiplication, and the fact that they can contain objects of differing types mean that Python must store type information for every element, and must execute type dispatching code when operating on each element.
3. NumPy is not just more efficient; it is also more convenient. You get a lot of vector and matrix operations for free, which sometimes allow one to avoid unnecessary work. And they are also efficiently implemented.
4. NumPy array is faster and You get a lot built in with NumPy, FFTs, convolutions, fast searching, basic statistics, linear algebra, histograms, etc.

## Q46. How to add values to a python array?

Elements can be added to an array using the **append()**, **extend()** and the **insert (i,x)** functions.

**Example:**

a=arr.array('d', [1.1 , 2.1 ,3.1] )  
a.append(3.4)  
print(a)  
a.extend([4.5,6.3,6.8])  
print(a)  
a.insert(2,3.8)  
print(a)

**Output:**

array(‘d’, [1.1, 2.1, 3.1, 3.4])

array(‘d’, [1.1, 2.1, 3.1, 3.4, 4.5, 6.3, 6.8])

array(‘d’, [1.1, 2.1, 3.8, 3.1, 3.4, 4.5, 6.3, 6.8])

## Q47. How to remove values to a python array?

Array elements can be removed using **pop()** or **remove()** method. The difference between these two functions is that the former returns the deleted value whereas the latter does not.

**Example:**

a=arr.array('d', [1.1, 2.2, 3.8, 3.1, 3.7, 1.2, 4.6])  
print(a.pop())  
print(a.pop(3))  
a.remove(1.1)  
print(a)

**Output:**

4.6

3.1

array(‘d’, [2.2, 3.8, 3.7, 1.2])

## Q48. Does Python have OOps concepts?

Python is an object-oriented programming language. This means that any program can be solved in python by creating an object model. However, Python can be treated as procedural as well as structural language.

## Q49. What is the difference between deep and shallow copy?

**Shallow copy** is used when a new instance type gets created and it keeps the values that are copied in the new instance. Shallow copy is used to copy the reference pointers just like it copies the values. These references point to the original objects and the changes made in any member of the class will also affect the original copy of it. Shallow copy allows faster execution of the program and it depends on the size of the data that is used.

**Deep copy** is used to store the values that are already copied. Deep copy doesn’t copy the reference pointers to the objects. It makes the reference to an object and the new object that is pointed by some other object gets stored. The changes made in the original copy won’t affect any other copy that uses the object. Deep copy makes execution of the program slower due to making certain copies for each object that is been called.

## Q50. How is Multithreading achieved in Python?

1. Python has a multi-threading package but if you want to multi-thread to speed your code up, then it’s usually not a good idea to use it.
2. Python has a construct called the Global Interpreter Lock (GIL). The GIL makes sure that only one of your ‘threads’ can execute at any one time. A thread acquires the GIL, does a little work, then passes the GIL onto the next thread.
3. This happens very quickly so to the human eye it may seem like your threads are executing in parallel, but they are really just taking turns using the same CPU core.
4. All this GIL passing adds overhead to execution. This means that if you want to make your code run faster then using the threading package often isn’t a good idea.

## Q51. What is the process of compilation and linking in python?

The compiling and linking allows the new extensions to be compiled properly without any error and the linking can be done only when it passes the compiled procedure. If the dynamic loading is used then it depends on the style that is being provided with the system. The python interpreter can be used to provide the dynamic loading of the configuration setup files and will rebuild the interpreter.

**The steps that are required in this as:**

1. Create a file with any name and in any language that is supported by the compiler of your system. For example file.c or file.cpp
2. Place this file in the Modules/ directory of the distribution which is getting used.
3. Add a line in the file Setup.local that is present in the Modules/ directory.
4. Run the file using spam file.o
5. After a successful run of this rebuild the interpreter by using the make command on the top-level directory.
6. If the file is changed then run rebuildMakefile by using the command as ‘make Makefile’.

## Q52. What are Python libraries? Name a few of them.

Python libraries are a collection of Python packages. Some of the majorly used python libraries are — Numpy, Pandas, Matplotlib, Scikit-learn and many more.

## Q53. What is split used for?

The split() method is used to separate a given string in Python.

**Example:**

a="edureka python"  
print(a.split())

**Output:**

[‘edureka’, ‘python’]

## ****Q54. How to import modules in python?****

Modules can be imported using the **import**keyword. You can import modules in three ways-

import array #importing using the original module name  
import array as arr # importing using an alias name  
from array import \* #imports everything present in the array module

# OOPS Interview Questions

## Q55. Explain Inheritance in Python with an example.

Inheritance allows One class to gain all the members(say attributes and methods) of another class. Inheritance provides code reusability, makes it easier to create and maintain an application. The class from which we are inheriting is called super-class and the class that is inherited is called a derived / child class.

They are different types of inheritance supported by Python:

1. **Single Inheritance** — where a derived class acquires the members of a single super class.
2. **Multi-level inheritance**— a derived class d1 in inherited from base class base1, and d2 are inherited from base2.
3. **Hierarchical inheritance** — from one base class you can inherit any number of child classes
4. **Multiple inheritance**— a derived class is inherited from more than one base class.

## Q56. How are classes created in Python?

Class in Python is created using the **class**keyword.

**Example:**

class Employee:  
def \_\_init\_\_(self, name):  
self.name = name  
E1=Employee("abc")  
print(E1.name)

**Output:**

abc

## Q57. What is monkey patching in Python?

**Ans:** In Python, the term monkey patch only refers to dynamic modifications of a class or module at run-time.

Consider the below example:

# m.py  
class MyClass:  
def f(self):  
print "f()"

We can then run the monkey-patch testing like this:

import m  
def monkey\_f(self):  
print "monkey\_f()"  
   
m.MyClass.f = monkey\_f  
obj = m.MyClass()  
obj.f()

The output will be as below:

monkey\_f()

As we can see, we did make some changes in the behavior of f() in MyClass using the function we defined, monkey\_f(), outside of the module m.

## Q58. Does python support multiple inheritance?

Multiple inheritance means that a class can be derived from more than one parent classes. Python does support multiple inheritance, unlike Java.

## Q59. What is Polymorphism in Python?

**Ans:**Polymorphism means the ability to take multiple forms. So, for instance, if the parent class has a method named ABC then the child class also can have a method with the same name ABC having its own parameters and variables. Python allows polymorphism.

## Q60. Define encapsulation in Python?

Encapsulation means binding the code and the data together. A Python class in an example of encapsulation.

## Q61. How do you do data abstraction in Python?

Data Abstraction is providing only the required details and hiding the implementation from the world. It c an be achieved in Python by using interfaces and abstract classes.

## Q62. Does python make use of access specifiers?

Python does not deprive access to an instance variable or function. Python lays down the concept of prefixing the name of the variable, function or method with a single or double underscore to imitate the behavior of protected and private access specifiers.

## Q63. How to create an empty class in Python?

An empty class is a class that does not have any code defined within its block. It can be created using the pass keyword. However, you can create objects of this class outside the class itself. IN PYTHON THE PASS command does nothing when its executed. it’s a null statement.

**For example-**

class a:  
 pass  
obj=a()  
obj.name="xyz"  
print("Name = ",obj.name)

**Output:**

Name = xyz

## Q64. What does an object() do?

It returns a featureless object that is a base for all classes. Also, it does not take any parameters.

# Basic Python Programs

## Q65. Write a program in Python to execute the Bubble sort algorithm.

def bs(a): # a = name of list  
 b=len(a)-1 # minus 1 because we always compare 2 adjacent values  
   
 for x in range(b):  
 for y in range(b-x):  
 if a[y]>a[y+1]:  
 a[y],a[y+1]=a[y+1],a[y]  
 return a  
a=[32,5,3,6,7,54,87]  
bs(a)

**Output:**

[3, 5, 6, 7, 32, 54, 87]

## Q66. Write a program in Python to produce Star triangle.

def pyfunc(r):  
 for x in range(r):  
 print(' '\*(r-x-1)+'\*'\*(2\*x+1))   
pyfunc(9)

**Output:**

\*  
 \*\*\*  
 \*\*\*\*\*  
 \*\*\*\*\*\*\*  
 \*\*\*\*\*\*\*\*\*  
 \*\*\*\*\*\*\*\*\*\*\*  
 \*\*\*\*\*\*\*\*\*\*\*\*\*  
 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## Q67. Write a program to produce Fibonacci series in Python.

# Enter number of terms needed #0,1,1,2,3,5....  
a=int(input("Enter the terms"))  
f=0 #first element of series  
s=1 #second element of series  
if a<=0:  
 print("The requested series is  
",f)  
else:  
 print(f,s,end=" ")  
 for x in range(2,a):  
 next=f+s   
 print(next,end=" ")  
 f=s  
 s=next</pre>

**Output:**

Enter the terms 5 0 1 1 2 3

## Q68. Write a program in Python to check if a number is prime.

a=int(input("enter number"))   
if a>1:  
 for x in range(2,a):  
 if(a%x)==0:  
 print("not prime")  
 break  
 else:  
 print("Prime")  
else:  
 print("not prime")

**Output:**

enter number 3Prime

## Q69. Write a program in Python to check if a sequence is a Palindrome.

a=input("enter sequence")  
b=a[::-1]  
if a==b:  
 print("palindrome")  
else:  
 print("Not a Palindrome")

**Output:**

enter sequence 323 palindrome

## Q70. Write a one-liner that will count the number of capital letters in a file. Your code should work even if the file is too big to fit in memory.

Let us first write a multiple line solution and then convert it to one-liner code.

with open(SOME\_LARGE\_FILE) as fh:  
count = 0  
text = fh.read()  
for character in text:  
 if character.isupper():  
count += 1

We will now try to transform this into a single line.

count sum(1 for line in fh for character in line if character.isupper())

## Q71. Write a sorting algorithm for a numerical dataset in Python.

The following code can be used to sort a list in Python:

list = ["1", "4", "0", "6", "9"]  
list = [int(i) for i in list]  
list.sort()  
print (list)

## Q72. Looking at the below code, write down the final values of A0, A1, …An.

A0 = dict(zip(('a','b','c','d','e'),(1,2,3,4,5)))  
A1 = range(10)A2 = sorted([i for i in A1 if i in A0])  
A3 = sorted([A0[s] for s in A0])  
A4 = [i for i in A1 if i in A3]  
A5 = {i:i\*i for i in A1}  
A6 = [[i,i\*i] for i in A1]  
print(A0,A1,A2,A3,A4,A5,A6)

The following will be the final outputs of A0, A1, … A6

A0 = {'a': 1, 'c': 3, 'b': 2, 'e': 5, 'd': 4} # the order may vary  
A1 = range(0, 10)   
A2 = []  
A3 = [1, 2, 3, 4, 5]  
A4 = [1, 2, 3, 4, 5]  
A5 = {0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81}  
A6 = [[0, 0], [1, 1], [2, 4], [3, 9], [4, 16], [5, 25], [6, 36], [7, 49], [8, 64], [9, 81]]

# Python Libraries Interview Questions

## Q73. Explain what Flask is and its benefits?

Flask is a web microframework for Python based on “Werkzeug, Jinja2 and good intentions” BSD license. Werkzeug and Jinja2 are two of its dependencies. This means it will have little to no dependencies on external libraries. It makes the framework light while there is a little dependency to update and fewer security bugs.

A session basically allows you to remember information from one request to another. In a flask, a session uses a signed cookie so the user can look at the session contents and modify. The user can modify the session if only it has the secret key Flask.secret\_key.

## Q74. Is Django better than Flask?

Django and Flask map the URL’s or addresses typed in the web browsers to functions in Python.

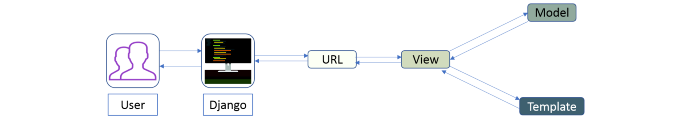
Flask is much simpler compared to Django but, Flask does not do a lot for you meaning you will need to specify the details, whereas Django does a lot for you wherein you would not need to do much work. Django consists of prewritten code, which the user will need to analyze whereas Flask gives the users to create their own code, therefore, making it simpler to understand the code. Technically both are equally good and both contain their own pros and cons.

## Q75. Mention the differences between Django, Pyramid and Flask.

* Flask is a “microframework” primarily build for a small application with simpler requirements. In flask, you have to use external libraries. Flask is ready to use.
* Pyramid is built for larger applications. It provides flexibility and lets the developer use the right tools for their project. The developer can choose the database, URL structure, templating style and more. Pyramid is heavy configurable.
* Django can also be used for larger applications just like Pyramid. It includes an ORM.

## Q76. Discuss Django architecture.

**Ans:** Django MVT Pattern:



The developer provides the Model, the view and the template then just maps it to a URL and Django does the magic to serve it to the user.

## Q77. Explain how you can set up the Database in Django.

You can use the command edit mysite/setting.py , it is a normal python module with module level representing Django settings.

Django uses SQLite by default; it is easy for Django users as such it won’t require any other type of installation. In the case your database choice is different that you have to the following keys in the DATABASE ‘default’ item to match your database connection settings.

* **Engines**: you can change database by using ‘django.db.backends.sqlite3’ , ‘django.db.backeneds.mysql’, ‘django.db.backends.postgresql\_psycopg2’, ‘django.db.backends.oracle’ and so on
* **Name**: The name of your database. In the case if you are using SQLite as your database, in that case database will be a file on your computer, Name should be a full absolute path, including file name of that file.
* If you are not choosing SQLite as your database then settings like Password, Host, User, etc. must be added.

We will add the following lines of code to the setting.py file:

DATABASES = {  
 'default': {  
 'ENGINE' : 'django.db.backends.sqlite3',  
 'NAME' : os.path.join(BASE\_DIR, 'db.sqlite3'),  
 }  
}

## Q78. Give an example how you can write a VIEW in Django?

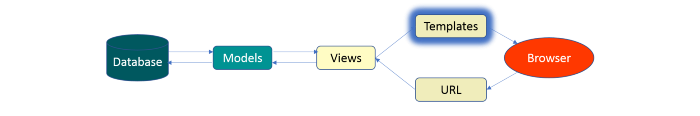
This is how we can use write a view in Django:

from django.http import HttpResponse  
import datetime  
   
def Current\_datetime(request):  
 now = datetime.datetime.now()  
 html = "<html><body>It is now %s</body></html> % now  
 return HttpResponse(html)

Returns the current date and time, as an HTML document

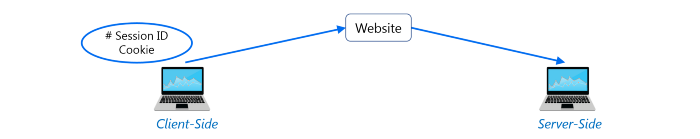
## Q79. Mention what the Django templates consist of.

The template is a simple text file. It can create any text-based format like XML, CSV, HTML, etc. A template contains variables that get replaced with values when the template is evaluated and tags (% tag %) that control the logic of the template.



## Q80. Explain the use of session in Django framework?

Django provides a session that lets you store and retrieve data on a per-site-visitor basis. Django abstracts the process of sending and receiving cookies, by placing a session ID cookie on the client side, and storing all the related data on the server side.



So the data itself is not stored client side. This is nice from a security perspective.

## ****Q81. List out the inheritance styles in Django.****

In Django, there are three possible inheritance styles:

1. **Abstract Base Classes:**This style is used when you only want parent’s class to hold information that you don’t want to type out for each child model.
2. **Multi-table Inheritance:**This style is used If you are sub-classing an existing model and need each model to have its own database table.
3. **Proxy models:** You can use this model, If you only want to modify the Python level behavior of the model, without changing the model’s fields.

# Web Scraping — Python Interview Questions

## Q82. How To Save An Image Locally Using Python Whose URL Address I Already Know?

We will use the following code to save an image locally from an URL address

import urllib.request   
urllib.request.urlretrieve("URL", "local-filename.jpg")

## Q83. How can you Get the Google cache age of any URL or web page?

Use the following URL format:

<http://webcache.googleusercontent.com/search?q=cache:URLGOESHERE>

Be sure to replace “URLGOESHERE” with the proper web address of the page or site whose cache you want to retrieve and see the time for. For example, to check the Google Webcache age of edureka.co you’d use the following URL:

<http://webcache.googleusercontent.com/search?q=cache:edureka.co>

## Q84. You are required to scrap data from IMDb top 250 movies page. It should only have fields movie name, year, and rating.

We will use the following lines of code:

from bs4 import BeautifulSoup  
   
import requests  
import sys  
   
url = '<http://www.imdb.com/chart/top'>  
response = requests.get(url)  
soup = BeautifulSoup(response.text)  
tr = soup.findChildren("tr")  
tr = iter(tr)  
next(tr)  
   
for movie in tr:  
title = movie.find('td', {'class': 'titleColumn'} ).find('a').contents[0]  
year = movie.find('td', {'class': 'titleColumn'} ).find('span', {'class': 'secondaryInfo'}).contents[0]  
rating = movie.find('td', {'class': 'ratingColumn imdbRating'} ).find('strong').contents[0]  
row = title + ' - ' + year + ' ' + ' ' + rating  
   
print(row)

The above code will help scrap data from IMDb’s top 250 list

# Data Analysis — Python Interview Questions

## Q85. What is map function in Python?

The map function executes the function given as the first argument on all the elements of the iterable given as the second argument. If the function given takes in more than 1 arguments, then many iterables are given. #Follow the link to know more similar functions.

## Q86. Is python numpy better than lists?

We use python numpy array instead of a list because of the below three reasons:

1. Less Memory
2. Fast
3. Convenient

## Q87. How to get indices of N maximum values in a NumPy array?

We can get the indices of N maximum values in a NumPy array using the below code:

import numpy as np  
arr = np.array([1, 3, 2, 4, 5])  
print(arr.argsort()[-3:][::-1])

**Output**

[ 4 3 1 ]

## Q88. How do you calculate percentiles with Python/ NumPy?

We can calculate percentiles with the following code

import numpy as np  
a = np.array([1,2,3,4,5])  
p = np.percentile(a, 50) #Returns 50th percentile, e.g. median  
print(p)

**Output**

3

## Q89. What is the difference between NumPy and SciPy?

1. In an ideal world, NumPy would contain nothing but the array data type and the most basic operations: indexing, sorting, reshaping, basic elementwise functions, et cetera.
2. All numerical code would reside in SciPy. However, one of NumPy’s important goals is compatibility, so NumPy tries to retain all features supported by either of its predecessors.
3. Thus NumPy contains some linear algebra functions, even though these more properly belong in SciPy. In any case, SciPy contains more fully-featured versions of the linear algebra modules, as well as many other numerical algorithms.
4. If you are doing scientific computing with python, you should probably install both NumPy and SciPy. Most new features belong in SciPy rather than NumPy.

## Q90. How do you make 3D plots/visualizations using NumPy/SciPy?

Like 2D plotting, 3D graphics is beyond the scope of NumPy and SciPy, but just as in the 2D case, packages exist that integrate with NumPy. Matplotlib provides basic 3D plotting in the mplot3d subpackage, whereas Mayavi provides a wide range of high-quality 3D visualization features, utilizing the powerful VTK engine.

# Multiple Choice Questions (MCQ)

## Q91. Which of the following statements create a dictionary? (Multiple Correct Answers Possible)

a) d = {}  
b) d = {“john”:40, “peter”:45}  
c) d = {40:”john”, 45:”peter”}  
d) d = (40:”john”, 45:”50”)

**Answer:**b, c & d.

Dictionaries are created by specifying keys and values.

## Q92. Which one of these is floor division?

a) /  
b) //  
c) %  
d) None of the mentioned

**Answer:**b) //

When both of the operands are integer then python chops out the fraction part and gives you the round off value, to get the accurate answer use floor division. For ex, 5/2 = 2.5 but both of the operands are integer so answer of this expression in python is 2. To get the 2.5 as the answer, use floor division using //. So, 5//2 = 2.5

## Q93. What is the maximum possible length of an identifier?

a) 31 characters  
b) 63 characters  
c) 79 characters  
d) None of the above

**Answer:**d) None of the above

Identifiers can be of any length.

## Q94. Why are local variable names beginning with an underscore discouraged?

a) they are used to indicate a private variables of a class  
b) they confuse the interpreter  
c) they are used to indicate global variables  
d) they slow down execution

**Answer:**a) they are used to indicate a private variable of a class

As Python has no concept of private variables, leading underscores are used to indicate variables that must not be accessed from outside the class.

## Q95. Which of the following is an invalid statement?

a) abc = 1,000,000  
b) a b c = 1000 2000 3000  
c) a,b,c = 1000, 2000, 3000  
d) a\_b\_c = 1,000,000

**Answer:**b) a b c = 1000 2000 3000

Spaces are not allowed in variable names.

## Q96. What is the output of the following?

try:  
 if '1' != 1:  
 raise "someError"  
 else:  
 print("someError has not occured")  
except "someError":  
 print ("someError has occured")

a) someError has occured  
b) someError has not occured  
c) invalid code  
d) none of the above

**Answer:**c) invalid code

A new exception class must inherit from a BaseException. There is no such inheritance here.

## Q97. Suppose list1 is [2, 33, 222, 14, 25], What is list1[-1] ?

The index -1 corresponds to the last index in the list.

a) Error  
b) None  
c) 25  
d) 2

**Answer:**c) 25

The index -1 corresponds to the last index in the list.

## Q98. To open a file c:scores.txt for writing, we use

a) outfile = open(“c:scores.txt”, “r”)  
b) outfile = open(“c:scores.txt”, “w”)  
c) outfile = open(file = “c:scores.txt”, “r”)  
d) outfile = open(file = “c:scores.txt”, “o”)

**Answer:**b) The location contains double slashes ( ) and w is used to indicate that file is being written to.

## Q99. What is the output of the following?

f = None  
   
for i in range (5):  
 with open("data.txt", "w") as f:  
 if i > 2:  
 break  
   
print f.closed

a) True  
b) False  
c) None  
d) Error

**Answer:**a) True

The WITH statement when used with open file guarantees that the file object is closed when the with block exits.

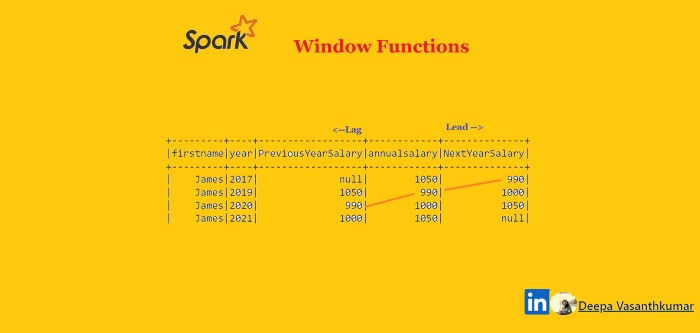
## Q100. When will the else part of try-except-else be executed?

a) always  
b) when an exception occurs  
c) when no exception occurs  
d) when an exception occurs into except block

**Answer:**c) when no exception occurs

# Apache Spark Window Functions

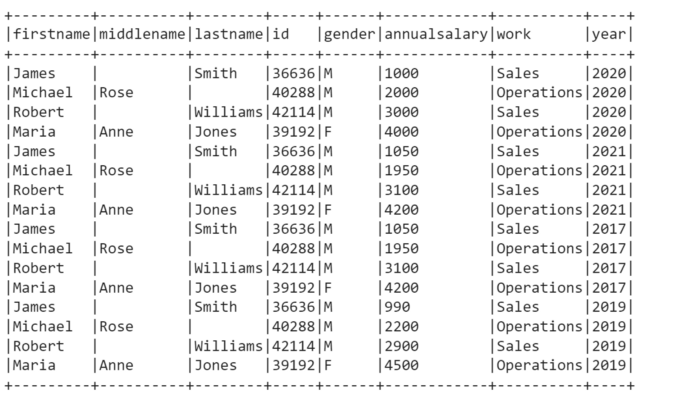
**Window functions** perform calculations on a set of rows that are related together. But, unlike the aggregate functions, windowing functions do not collapse the result of the rows into a single value. Instead, all the rows maintain their original identity and the calculated result is returned for every row.



To demo the purpose, let us create some data and dummy dataframe to demo the same



The dataframe looks like this



Now, to implement Rank Function, we can define window speculation and the ordering

from pyspark.sql.window import Window

from pyspark.sql.functions import row\_number

windowSpec = Window.partitionBy(“work”).orderBy(“annualsalary”)

When ordering is not defined, an unbounded window frame (rowFrame, unboundedPreceding, unboundedFollowing) is used by default. When ordering is defined, a growing window frame (rangeFrame, unboundedPreceding, currentRow) is used by default.

# Rank Function

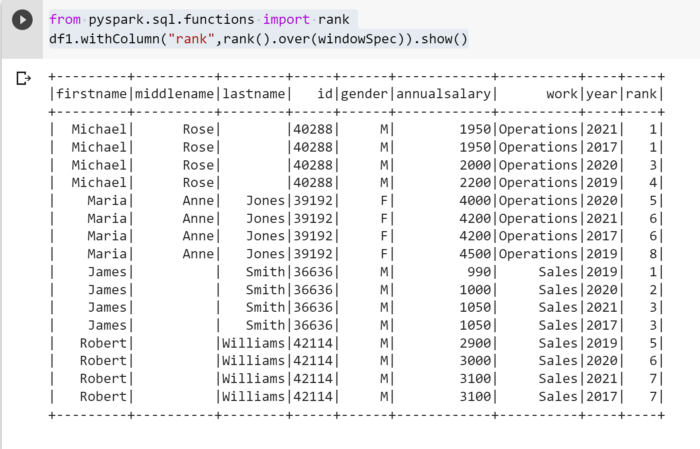
rank() window function is used to provide a rank to the result within a window partition. This function leaves gaps in rank when there are ties.

The difference between **rank** and **dense\_rank** is that denseRank leaves no gaps in ranking sequence when there are ties. That is, if you were ranking a competition using dense\_rank and had three people tie for second place, you would say that all three were in second place and that the next person came in third. Rank would give me sequential numbers, making the person that came in third place (after the ties) would register as coming in fifth.

**Rank**

from pyspark.sql.functions import rank

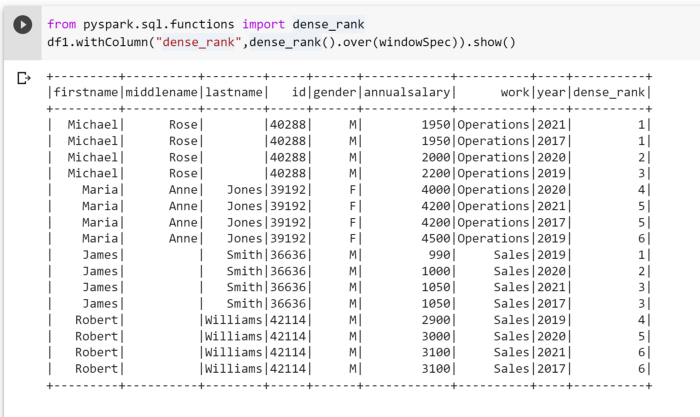
df1.withColumn(“rank”,rank().over(windowSpec)).show()



**Dense-Rank**

from pyspark.sql.functions import dense\_rank

df1.withColumn(“dense\_rank”,dense\_rank().over(windowSpec)).show()

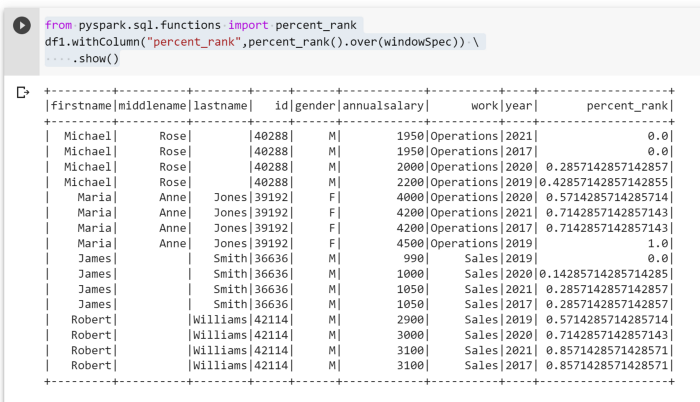


**Percent Rank**

from pyspark.sql.functions import percent\_rank

df1.withColumn(“percent\_rank”,percent\_rank().over(windowSpec)) \

.show()

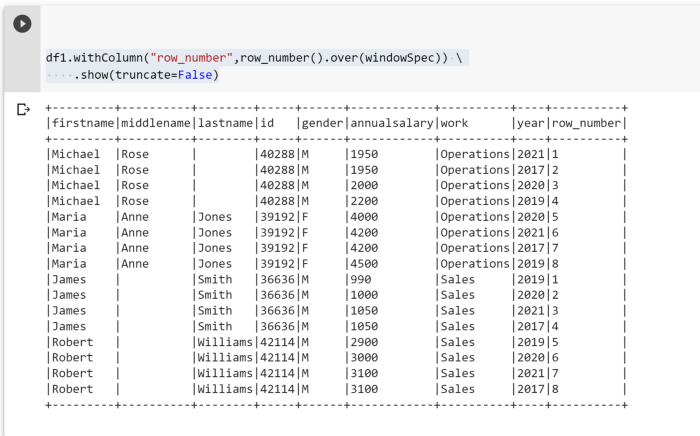


**Row Number Window Function**

returns a sequential number starting at 1 within a window partition.

df1.withColumn(“row\_number”,row\_number().over(windowSpec)) \

.show(truncate=False)



# Analytical Window Functions — Lag & Lead

When we need to compare the the result with the one previous or ahead for the same windowing group, we can use lag or lead functions.

Consider a scenario, where we need to find the what salary he is drawing now vs what was his previous salary, or

what was the previous sales when compared to current or

Given a week time, what is the difference of spend,

In these kind of scenarios , where we need to compare the immediate vs next or previous for same set of conditions — lag or lead functions come handy !

**Lag** Window function: returns the value that is offset rows before the current row, and 'nullif there is less thanoffset` rows before the current row. For example,

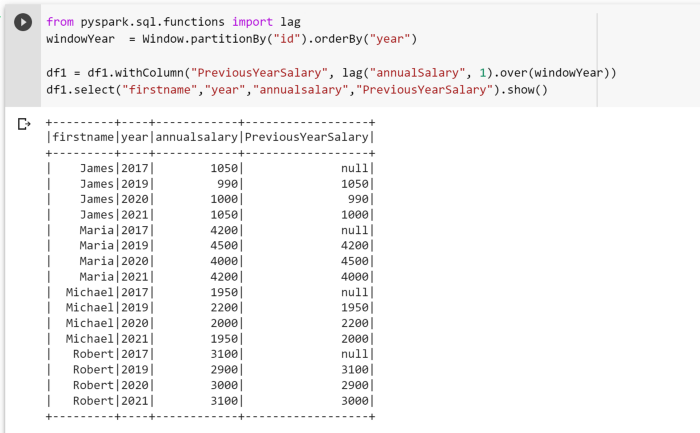
* an offset of one will return the previous row at any given point in the window partition.

from pyspark.sql.functions import lag

windowYear = Window.partitionBy(“id”).orderBy(“year”)

df1 = df1.withColumn(“PreviousYearSalary”, lag(“annualSalary”, 1).over(windowYear))

df1.select(“firstname”,”year”,”annualsalary”,”PreviousYearSalary”).show()



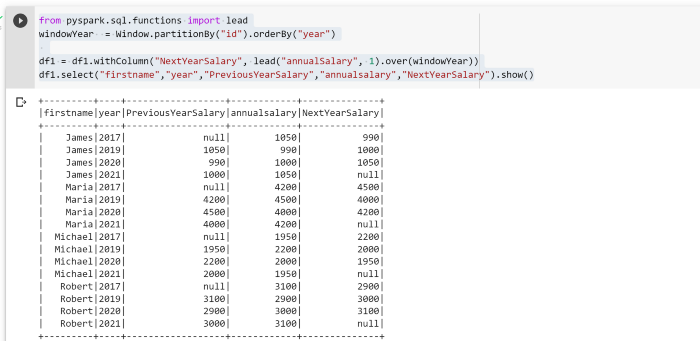
Lead Window function: returns the value that is offset rows after the current row, and null if there is less than offset rows after the current row. For example, an offset of one will return the next row at any given point in the window partition.

from pyspark.sql.functions import lead

windowYear = Window.partitionBy(“id”).orderBy(“year”)

df1 = df1.withColumn(“NextYearSalary”, lead(“annualSalary”, 1).over(windowYear))

df1.select(“firstname”,”year”,”PreviousYearSalary”,”annualsalary”,”NextYearSalary”).show()



**Lag & Lead — with negative offsets**

Tricky Part, what happens when we supply negative offsets to these functions…?

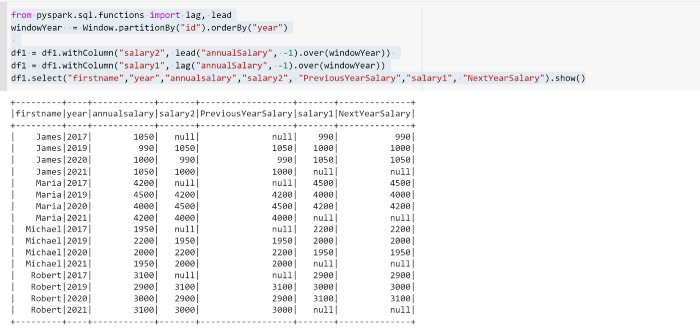
from pyspark.sql.functions import lag, lead

windowYear = Window.partitionBy(“id”).orderBy(“year”)

df1 = df1.withColumn(“salary2”, lead(“annualSalary”, -1).over(windowYear))

df1 = df1.withColumn(“salary1”, lag(“annualSalary”, -1).over(windowYear))

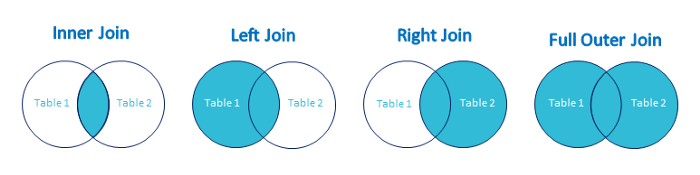
df1.select(“firstname”,”year”,”annualsalary”,”salary2", “PreviousYearSalary”,”salary1", “NextYearSalary”).show()



# Different Types of JOIN in Spark SQL

Join in Spark SQL is the functionality to join two or more datasets that are similar to the table join in SQL based databases. Spark works as the tabular form of datasets and data frames. The Spark SQL supports several types of joins such as inner join, cross join, left outer join, right outer join, full outer join, left semi-join, left anti join. Joins scenarios are implemented in Spark SQL based upon the business use case. Some of the joins require high resource and computation efficiency. For managing such scenarios spark support SQL optimizer and cross join enabler flags features.

## Types of Join in Spark SQL



Following are the different types of Joins:

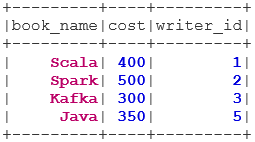
* INNER JOIN
* LEFT OUTER JOIN
* RIGHT OUTER JOIN
* FULL OUTER JOIN
* LEFT SEMI JOIN
* LEFT ANTI JOIN

## Example of Data Creation

We will use the following data to demonstrate the different types of joins:

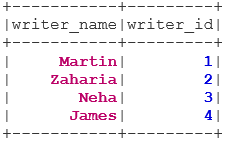
## Book Dataset:

case class Book(book\_name: String, cost: Int, writer\_id:Int)  
val bookDS = Seq(  
Book("Scala", 400, 1),  
Book("Spark", 500, 2),  
Book("Kafka", 300, 3),  
Book("Java", 350, 5)  
).toDS()  
bookDS.show()



## Writer Dataset:

case class Writer(writer\_name: String, writer\_id:Int)  
val writerDS = Seq(  
Writer("Martin",1),  
Writer("Zaharia " 2),  
Writer("Neha", 3),  
Writer("James", 4)  
).toDS()  
writerDS.show()



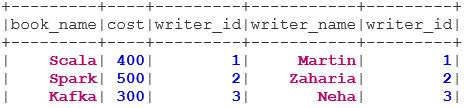
## Types of Joins

Below are mentioned 7 different types of Joins:

## 1. INNER JOIN

The INNER JOIN returns the dataset which has the rows that have matching values in both the datasets i.e. value of the common field will be the same.

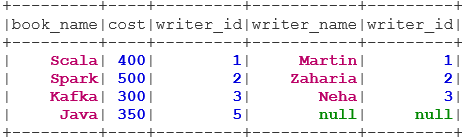
val BookWriterInner = bookDS.join(writerDS, bookDS("writer\_id") === writerDS("writer\_id"), "inner")  
BookWriterInner.show()



## 2. LEFT OUTER JOIN

The LEFT OUTER JOIN returns the dataset that has all rows from the left dataset, and the matched rows from the right dataset.

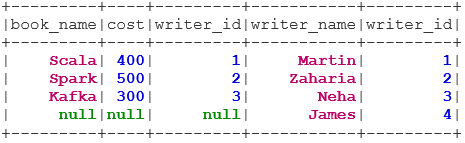
val BookWriterLeft = bookDS.join(writerDS, bookDS("writer\_id") === writerDS("writer\_id"), "leftouter")  
BookWriterLeft.show()



## 3. RIGHT OUTER JOIN

The RIGHT OUTER JOIN returns the dataset that has all rows from the right dataset, and the matched rows from the left dataset.

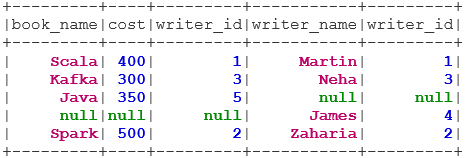
val BookWriterRight = bookDS.join(writerDS, bookDS("writer\_id") === writerDS("writer\_id"), "rightouter")  
BookWriterRight.show()



## 4. FULL OUTER JOIN

The FULL OUTER JOIN returns the dataset that has all rows when there is a match in either the left or right dataset.

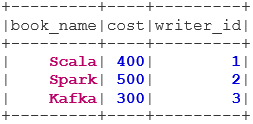
val BookWriterFull = bookDS.join(writerDS, bookDS("writer\_id") === writerDS("writer\_id"), "fullouter")  
BookWriterFull.show()



## 5. LEFT SEMI JOIN

The LEFT SEMI JOIN returns the dataset which has all rows from the left dataset having their correspondence in the right dataset. Unlike the LEFT OUTER JOIN, the returned dataset in LEFT SEMI JOIN contains only the columns from the left dataset.

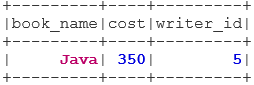
val BookWriterLeftSemi = bookDS.join(writerDS, bookDS("writer\_id") === writerDS("writer\_id"), "leftsemi")  
BookWriterLeftSemi.show()



1. LEFT ANTI JOIN

The ANTI SEMI JOIN returns the dataset which has all the rows from the left dataset that don’t have their matching in the right dataset. It also contains only the columns from the left dataset.

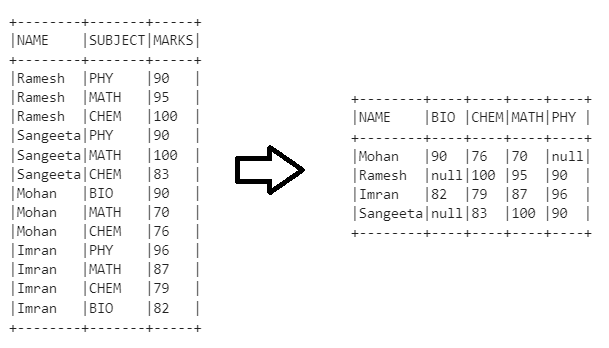
val BookWriterLeftAnti = bookDS.join(writerDS, bookDS("writer\_id") === writerDS("writer\_id"), "leftanti")  
BookWriterLeftAnti.show()



# PySpark — Optimize Pivot Data Frames like a PRO

Pivoting data is a very common scenario in every data engineering pipeline. Spark provides out-of-the-box pivot() method to do the job right. **But, do you know we have a performance trade off in Spark Data Frame using pivot(), if it is not used properly.**

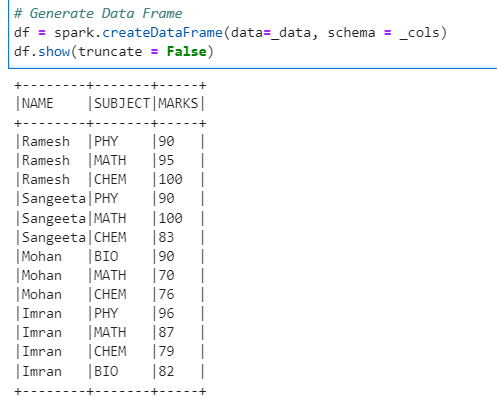
Lets check that out in action.



Pivot Data Frames

First, we create our example Data Frame

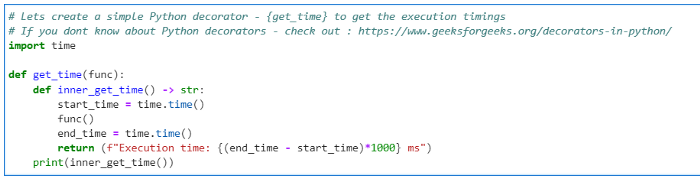
**# Example Data Set**\_data = [  
 ["Ramesh", "PHY", 90],  
 ["Ramesh", "MATH", 95],  
 ["Ramesh", "CHEM", 100],  
 ["Sangeeta", "PHY", 90],  
 ["Sangeeta", "MATH", 100],  
 ["Sangeeta", "CHEM", 83],  
 ["Mohan", "BIO", 90],  
 ["Mohan", "MATH", 70],  
 ["Mohan", "CHEM", 76],  
 ["Imran", "PHY", 96],  
 ["Imran", "MATH", 87],  
 ["Imran", "CHEM", 79],  
 ["Imran", "BIO", 82]  
]\_cols = ["NAME", "SUBJECT", "MARKS"]**# Generate Data Frame**  
df = spark.createDataFrame(data=\_data, schema = \_cols)  
df.show(truncate = False)



Example Data Frame

To measure the performance, we will create a simple Python decorator.

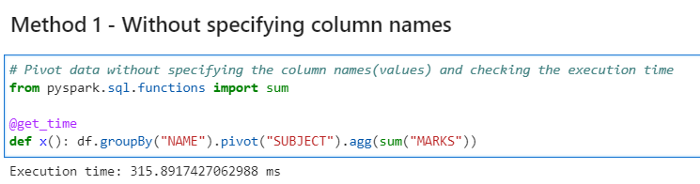
**# Lets create a simple Python decorator - {get\_time} to get the execution timings  
# If you dont know about Python decorators - check out :** [**https://www.geeksforgeeks.org/decorators-in-python/**](https://www.geeksforgeeks.org/decorators-in-python/)  
import timedef get\_time(func):  
 def inner\_get\_time() -> str:  
 start\_time = time.time()  
 func()  
 end\_time = time.time()  
 return (f"Execution time: {(end\_time - start\_time)\*1000} ms")  
 print(inner\_get\_time())



Python decorator to measure performance

**Method 1 — Pivoting the data without specifying the Pivot column names**

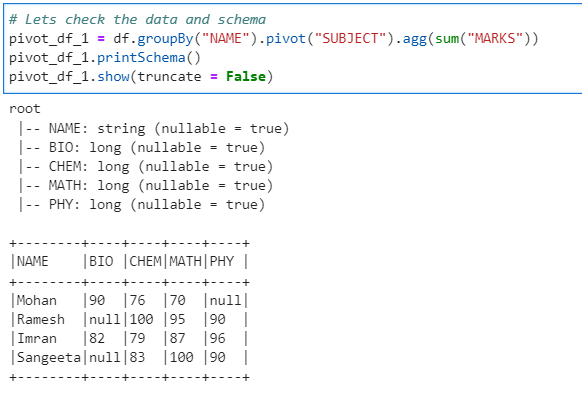
**# Pivot data without specifying the column names(values) and checking the execution time**  
from pyspark.sql.functions import sum[@get\_time](http://twitter.com/get_time)  
def x(): df.groupBy("NAME").pivot("SUBJECT").agg(sum("MARKS"))



Performance measure without specifying column names

Checking the data

**# Lets check the data and schema**  
pivot\_df\_1 = df.groupBy("NAME").pivot("SUBJECT").agg(sum("MARKS"))  
pivot\_df\_1.printSchema()  
pivot\_df\_1.show(truncate = False)

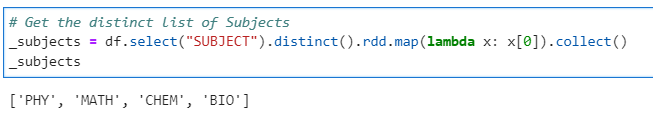


Pivot Data Frame 1

**Method 2 — Specifying the column names**

First, we have to get the distinct column names from the SUBJECT column

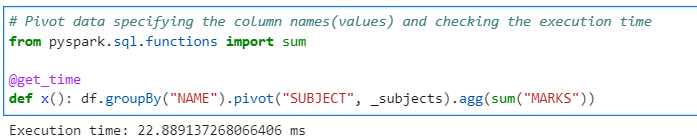
**# Get the distinct list of Subjects**  
\_subjects = df.select("SUBJECT").distinct().rdd.map(lambda x: x[0]).collect()  
\_subjects



Distinct column names

Now, if we use the distinct column name for PIVOT

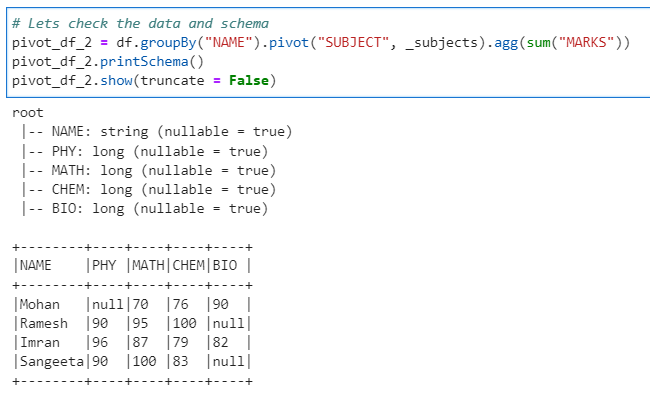
**# Pivot data specifying the column names(values) and checking the execution time**  
from pyspark.sql.functions import sum[@get\_time](http://twitter.com/get_time)  
def x(): df.groupBy("NAME").pivot("SUBJECT", \_subjects).agg(sum("MARKS"))



Performance measure with column names

Check the data

**# Lets check the data and schema**  
pivot\_df\_2 = df.groupBy("NAME").pivot("SUBJECT", \_subjects).agg(sum("MARKS"))  
pivot\_df\_2.printSchema()  
pivot\_df\_2.show(truncate = False)



Pivot Data Frame 2

As we can see the second time with column names specified the pivot() method ran much quicker.

**Conclusion:**We can now easily conclude that if the column names are specified the execution is much quicker. But, don’t forget the execution time required to get the distinct columns as well.

So, If the column name are already known/pre-specified for a larger dataset, we should always try to specify them.

# Databricks Delta lake

Let’s first understand what Data Lake is. A data lake is a central storage repository that holds big data from many sources in a raw, granular format.  
It can store structured, semi-structured, or unstructured data,  
which means data can be kept in a more flexible format for future use. Data Lake provides a very cost effective & scalable way to store any kind of data in form of raw files. Example of Data Lake is amazon s3, azure datalake gen2, HDFS etc. The challenge with Data Lakes is that they do not provide ACID guarantees, which means Insert, Update, Delete and Merge operations.

This is where **Delta lake** comes into picture, it enables ACID/Curd operations , with the help of parquet files and \_delta\_log(Subfolder). \_delta\_log maintain all the transaction logs which are applied on data using two files .crc and .json. We will see all this files and folders with examples going forward.

Our example dataset:

emp=[([1,'EMP1','2222255555','emp1@gmail.com](mailto:1,'EMP1','2222255555','emp1@gmail.com)'),  
([2,'EMP2','3333344444','emp2@gmail.com](mailto:2,'EMP2','3333344444','emp2@gmail.com)'),  
([3,'EMP3','5555566666','emp3@gmail.com](mailto:3,'EMP3','5555566666','emp3@gmail.com)'),  
([4,'EMP4','','emp4@gmail.com](mailto:4,'EMP4','','emp4@gmail.com)'),  
([5,'EMP5','9247518345','emp5@gmail.com](mailto:5,'EMP5','9247518345','emp5@gmail.com)')]df=spark.createDataFrame(emp,['id','name','mobile','email'])  
df=df.coalesce(1)  
df.show(truncate=False)



Example Dataset

Writing the data to delta lake.

df \  
.write \  
.format('delta') \  
.save('/FileStore/tables/employees')

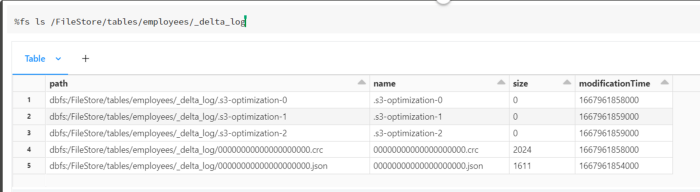
Lets see how the data is stored in employees directory.

%fs ls /FileStore/tables/employees



Delta Table

To check the transaction log, we can list the **\_delta\_log**folders where all transaction-related data get captured.



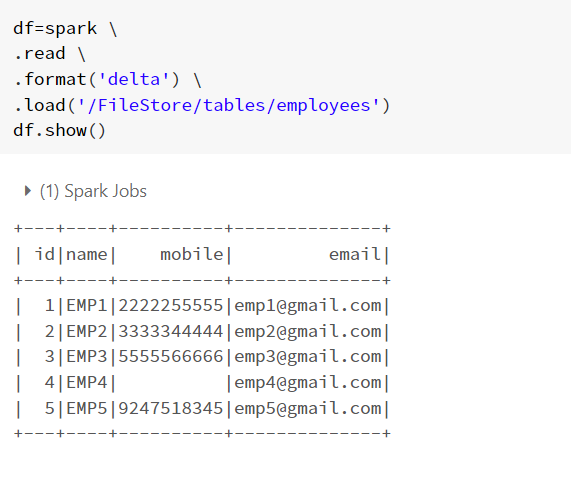
\_delta\_log Folder

Now, if we keep an eye on above pictures, when we store data into delta lake it internally stores data in the form of parquet ,\_delta\_log. \_delta\_log folder contains .crc(consistency redundancy check) and .json files.

Whenever we perform DML operation like, update or delete a new parquet ,.crc and .json files get created.

**Update Operation on Delta table:**

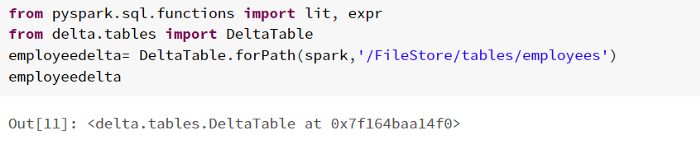
df=spark \  
.read \  
.format('delta') \  
.load('/FileStore/tables/employees')  
df.show()



Data From Delta Table

Now we will try to update the mobile for id=4 using deltatable object.

**#Detlta table object**from pyspark.sql.functions import lit, expr  
from delta.tables import DeltaTable  
employeedelta= DeltaTable.forPath(spark,'/FileStore/tables/employees')  
employeedelta



Delta Table Object

**Updating Data:**

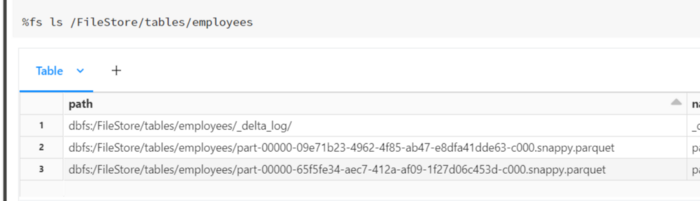
employeedelta.update( \  
condition= expr('id==4'), \  
 set = {"mobile" : lit("9456734577")}  
)



Output after the update

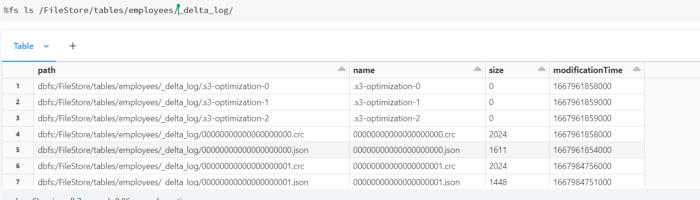
We can see for id=4 the mobile is updated to ‘9456734577’.

Lets check the number of parquet ,.crc and .json files in the given path.



Data Files

We have performed one update a new file got created ,whenever we read this folder based on this files and \_delta\_logs it will show the latest updated data.



\_delta\_log folder

**Delete Operation:**

Here we will delete the id=4 using delta object, creation of delta object is same as above which we created for update.

employeedelta.delete( \  
condition= expr('id==4') \  
 )



Result of employees data after delete.

**Conclusion :** Delta Lake allows us to perform ACID and CURD operations with the help of DeltaTable object.

### 1) Explain what is Hive?

Hive is an ETL and Data warehousing tool developed on top of Hadoop Distributed File System (HDFS). It is a data warehouse framework for querying and analysis of data that is stored in HDFS. Hive is an open-source-software that lets programmers analyze large data sets on [Hadoop](https://career.guru99.com/top-25-hadoop-admin-interview-questions-and-answers/" \o "Hadoop).

### 2) When to use Hive?

* Hive is useful when making [data warehouse](https://career.guru99.com/top-50-datawarehousing-questions-answers/) applications
* When you are dealing with static data instead of dynamic data
* When application is on high latency (high response time)
* When a large data set is maintained
* When we are using queries instead of scripting

### 3) Mention what are the different modes of Hive?

Depending on the size of data nodes in Hadoop, Hive can operate in two modes. **These modes are,**

* Local mode
* Map reduce mode

### 4) Mention when to use Map reduce mode?

Map reduce mode is used when,

* It will perform on large amount of data sets and query going to execute in a parallel way
* Hadoop has multiple data nodes, and data is distributed across different node we use Hive in this mode
* Processing large data sets with better performance needs to be achieved

### 5) Mention key components of Hive Architecture?

Key components of Hive Architecture includes,

* User Interface
* Compiler
* Metastore
* Driver
* Execute engine

### 6) Mention what are the different types of tables available in Hive?

There are two types of tables available in Hive.

* **Managed table**: In managed table, both the data and schema are under control of Hive
* **External table**: In the external table, only the schema is under the control of Hive.

7) Explain what is Metastore in Hive?

### Metastore is a central repository in Hive.  It is used for storing schema information or metadata in the external database.

### 8) Mention what Hive is composed of ?

Hive consists of 3 main parts,

1. Hive Clients
2. Hive Services
3. Hive Storage and Computing

### 9) Mention what are the type of database does Hive support ?

For single user metadata storage, Hive uses derby database and for multiple user Metadata or shared Metadata case Hive uses [MYSQL](https://career.guru99.com/top-50-mysql-interview-questions-answers/).

### 10) Mention Hive default read and write classes?

Hive default read and write classes are

1. TextInputFormat/HiveIgnoreKeyTextOutputFormat
2. SequenceFileInputFormat/SequenceFileOutputFormat

### 11) What is indexing in Hive?

Hive indexing is a query optimization technique to improve the speed of query lookup on certain columns of a table.

### 12) Why is Hive not suitable for OLTP systems?

Hive is not suitable for OLTP systems because it does not provide insert and update function at the row level.

### 13) Mention what is the difference between Hbase and Hive?

Difference between Hbase and Hive is,

* Hive enables most of the [SQL](https://www.guru99.com/sql-server-questions.html) queries, but HBase does not allow SQL queries
* Hive does not support record level insert, update, and delete operations on table
* Hive is a data warehouse framework whereas HBase is NoSQL database
* Hive run on the top of MapReduce, HBase runs on the top of HDFS

### 14) Explain what is a Hive variable? What for we use it?

Hive variable is created in the Hive environment that can be referenced by Hive scripts. It is used to pass some values to the hive queries when the query starts executing.

### 15) Mention what is ObjectInspector functionality in Hive?

ObjectInspector functionality in Hive is used to analyze the internal structure of the columns, rows, and complex objects.  It allows to access the internal fields inside the objects.

### 16) Mention what is (HS2) HiveServer2?

It is a server interface that performs following functions.

* It allows remote clients to execute queries against Hive
* Retrieve the results of mentioned queries

Some advanced features Based on Thrift RPC in its latest version include

* Multi-client concurrency
* Authentication

### 17) Mention what Hive query processor does?

Hive query processor convert graph of MapReduce jobs with the execution time framework.  So that the jobs can be executed in the order of dependencies.

### 18) Mention what are the components of a Hive query processor?

The components of a Hive query processor include,

* Logical Plan Generation
* Physical Plan Generation
* Execution Engine
* Operators
* UDF’s and UDAF’s
* Optimizer
* Parser
* Semantic Analyzer
* Type Checking

### 19) Mention what is Partitions in Hive?

Hive organizes tables into partitions.

* It is one of the ways of dividing tables into different parts based on partition keys.
* Partition is helpful when the table has one or more Partition keys.
* Partition keys are basic elements for determining how the data is stored in the table.

### 20) Mention when to choose “Internal Table” and “External Table” in Hive?

In Hive you can choose internal table,

* If the processing data available in local file system
* If we want Hive to manage the complete lifecycle of data including the deletion

You can choose External table,

* If processing data available in HDFS
* Useful when the files are being used outside of Hive

### 21) Mention if we can name view same as the name of a Hive table?

No. The name of a view must be unique compared to all other tables and as views present in the same database.

### 22) Mention what are views in Hive?

In Hive, Views are Similar to tables. They are generated based on the requirements.

* We can save any result set data as a view in Hive
* Usage is similar to as views used in SQL
* All type of DML operations can be performed on a view

### 23) Explain how Hive Deserialize and serialize the data?

Usually, while read/write the data, the user first communicate with inputformat. Then it connects with Record reader to read/write record.  To serialize the data, the data goes to row. Here deserialized custom serde use object inspector to deserialize the data in fields.

### 24) What is Buckets in Hive?

* The data present in the partitions can be divided further into Buckets
* The division is performed based on Hash of particular columns that is selected in the table.

### 25) In Hive, how can you enable buckets?

In Hive, you can enable buckets by using the following command,

set.hive.enforce.bucketing=true;

### 26) In Hive, can you overwrite Hadoop MapReduce configuration in Hive?

Yes, you can overwrite Hadoop MapReduce configuration in Hive.

### 27) Explain how can you change a column data type in Hive?

You can change a column data type in Hive by using command,

ALTER TABLE table\_name CHANGE column\_name column\_name new\_datatype;

### 28) Mention what is the difference between order by and sort by in Hive?

* SORT BY will sort the data within each reducer. You can use any number of reducers for SORT BY operation.
* ORDER BY will sort all of the data together, which has to pass through one reducer. Thus, ORDER BY in hive uses a single

### 29) Explain when to use explode in Hive?

Hadoop developers sometimes take an [array](https://career.guru99.com/top-50-array-interview-questions-answers/) as input and convert into a separate table row. To convert complex data types into desired table formats, Hive use explode.

### 30) Mention how can you stop a partition form being queried?

You can stop a partition form being queried by using the ENABLE OFFLINE clause with ALTER TABLE statement.