Top 200 **Data Engineer** Interview Questions & Answers

Knowledge Powerhouse

Top 200 Data Engineer

Interview Questions & Answers

Knowledge Powerhouse

DEDICATION

To our readers!

CONTENTS

3. **What are the main components of a Hadoop Application?**

4. **What is the core concept behind Apache Hadoop framework?**

5. **What is Hadoop Streaming?**

6. **What is the difference between NameNode, Backup Node and Checkpoint NameNode in HDFS?**

7. **What is the optimum hardware configuration to run Apache Hadoop?**

8. **What do you know about Block and Block scanner in HDFS?**

9. **What are the default port numbers on which Name Node, Job Tracker and**

Task Tracker run in Hadoop?

10. How will you disable a Block Scanner on HDFS DataNode?

11. How will you get the distance between two nodes in Apache Hadoop?

12. Why do we use commodity hardware in Hadoop?

13. How does inter cluster data copying works in Hadoop?

14. How can we update a file at an arbitrary location in HDFS?

15. What is Replication factor in HDFS, and how can we set it?

16. What is the difference between NAS and DAS in Hadoop cluster?

17. What are the two messages that NameNode receives from DataNode in Hadoop?

18. How does indexing work in Hadoop?

19. What data is stored in a HDFS NameNode?

20. What would happen if NameNode crashes in a HDFS cluster?

21. What are the main functions of Secondary NameNode?

22. What happens if HDFS file is set

with replication factor of 1 and DataNode crashes?

23. What is the meaning of Rack Awareness in Hadoop?

24. If we set Replication factor 3 for a file, does it mean any computation will also take place 3 times?

25. How will you check if a file exists in HDFS?

26. Why do we use fsck command in HDFS?

27. What will happen when NameNode is down and a user submits a new job?

28. What are the core methods of a Reducer in Hadoop?

29. What are the primary phases of a Reducer in Hadoop?

30. What is the use of Context object in Hadoop?

31. How does partitioning work in Hadoop?

32. What is a Combiner in Hadoop?

33. What is the default replication factor in HDFS?

34. How much storage is allocated by HDFS for storing a file of 25 MB size?

35. Why does HDFS store data in Block structure?

36. How will you create a custom

Partitioner in a Hadoop job?

37. What are the differences between RDBMS and HBase data model?

38. What is a Checkpoint node in HDFS?

39. What is a Backup Node in HDFS?

40. What is the meaning of term Data Locality in Hadoop?

41. What is the difference between Data science, Big Data and Hadoop?

42. What is a Balancer in HDFS?

43. What are the important points a NameNode considers before selecting the DataNode for placing a data

block?

44. What is Safemode in HDFS?

45. How will you replace HDFS data volume before shutting down a DataNode?

46. What are the important configuration files in Hadoop?

47. How will you monitor memory used in a Hadoop cluster?

48. Why do we need Serialization in Hadoop map reduce methods?

49. What is the use of Distributed Cache in Hadoop?

50. How will you synchronize the

changes made to a file in Distributed Cache in Hadoop?

Apache Hive Questions

51. How will you improve the performance of a program in Hive?

52. Can we use Hive for Online Transaction Processing (OLTP) systems?

53. How will you change the data type of a column in Hive?

54. What is Metastore in Hive?

55. What is SerDe in Hive?

56. What are the components in Hive data model?

57. What are the different modes in which we can run Hive?

58. What are the main components of Hive?

59. What is the use of Hive in Hadoop eco-system?

60. What Collection/Complex data types are supported by Hive?

61. What is the use of .hiverc file in Hive?

62. How will you run Unix commands from Hive?

63. What is the purpose of USE command in Hive?

64. What is the precedence order in Hive configuration?

65. How will you display header row with the results of a Hive query?

66. Can we create multiple tables in Hive for a data file?

67. How does CONCAT function work in Hive?

68. How will you change settings of a Hive session?

69. How will you rename a table in Hive without using ALTER command?

70. What is the difference between SORT BY and ORDER BY in Hive?

71. What is the use of strict mode in Hive?

72. What is the use of IF EXISTS clause in Hive statements?

73. What is the use of PURGE in DROP statement of Hive?

74. What are the main limitations of Apache Hive?

75. What is the difference between HBase and Hive?

76. What is ObjectInspector in Hive?

77. What are the main components of Query Processor in Apache Hive?

78. How will you resolve an out of

memory error while running a JOIN query?

79. **What are the different SerDe implementations in Hive?**

80. **What is the use of HCatalog?**

81. **What is the Data Model of HCatalog?**

82. **What is RLIKE operator in Hive?**

83. **Can we use same name for a TABLE and VIEW in Hive?**

84. **How will you load data into a VIEW in Hive?**

85. **What is Bucketing in Hive?**

86. **What are the pros and cons of**

archiving a partition in Hive?

87. What are the table generating functions in Hive?

88. How can we specify in Hive to load an HDFS file in LOAD DATA?

89. What is a Skewed table in Hive?

90. What is the use of CLUSTERED BY clause during table creation in Hive?

91. What is a Managed table in Hive?

92. How will you prevent data to be dropped or queried from a partition in Hive?

93. What is the use of TOUCH in

ALTER statement?

94. How does OVERWRITE clause work in CREATE TABLE statement in Hive?

95. What are the options to connect an application to a Hive server?

96. How TRIM and RPAD functions work in Hive?

97. How will you recursively access sub-directories in Hive?

98. What is the optimization that can be done in SELECT * query in Hive?

99. What is the use of ORC format tables in Hive?

100. What are the main use cases for using Hive?

101. What is STREAMTABLE in Hive?

Apache Spark Questions

102. What are the main features of Apache Spark?

103. What is a Resilient Distribution Dataset in Apache Spark?

104. What is a Transformation in Apache Spark?

105. What are security options in Apache Spark?

106. How will you monitor Apache

Spark?

107. What are the main libraries of Apache Spark?

108. What are the main functions of Spark Core in Apache Spark?

109. How will you do memory tuning in Spark?

110. What are the two ways to create RDD in Spark?

111. What are the main operations that can be done on a RDD in Apache Spark?

112. What are the common Transformations in Apache Spark?

113. What are the common Actions in Apache Spark?

114. What is a Shuffle operation in Spark?

115. What are the operations that can cause a shuffle in Spark?

116. What is purpose of Spark SQL?

117. What is a DataFrame in Spark SQL?

118. What is a Parquet file in Spark?

119. What is the difference between Apache Spark and Apache Hadoop MapReduce?

120. What are the main languages

supported by Apache Spark?

121. What are the file systems supported by Spark?

122. What is a Spark Driver?

123. What is an RDD Lineage?

124. What are the two main types of Vector in Spark?

125. What are the different deployment modes of Apache Spark?

126. What is lazy evaluation in Apache Spark?

127. What are the core components of a distributed application in Apache Spark?

128. What is the difference in cache() and persist() methods in Apache Spark?

129. How will you remove data from cache in Apache Spark?

130. What is the use of SparkContext in Apache Spark?

131. Do we need HDFS for running Spark application?

132. What is Spark Streaming?

133. How does Spark Streaming work internally?

134. What is a Pipeline in Apache Spark?

135. How does Pipeline work in Apache Spark?

136. What is the difference between Transformer and Estimator in Apache Spark?

137. What are the different types of Cluster Managers in Apache Spark?

138. How will you minimize data transfer while working with Apache Spark?

139. What is the main use of MLib in Apache Spark?

140. What is the Checkpointing in Apache Spark?

141. What is an Accumulator in Apache Spark?

142. What is a Broadcast variable in Apache Spark?

143. What is Structured Streaming in Apache Spark?

144. How will you pass functions to Apache Spark?

145. What is a Property Graph?

146. What is Neighborhood Aggregation in Spark?

147. What are different Persistence levels in Apache Spark?

148. How will you select the storage

level in Apache Spark?

149. What are the options in Spark to create a Graph?

150. What are the basic Graph operators in Spark?

151. What is the partitioning approach used in GraphX of Apache Spark?

SQL Tricky Questions

152. Write SQL query to get the second highest salary among all Employees?

153. How can we retrieve alternate records from a table in Oracle?

154. Write a SQL Query to find Max salary and Department name from each department.

155. Write a SQL query to find records in Table A that are not in Table B without using NOT IN operator.

156. What is the result of following query?

157. Write SQL Query to find employees that have same name and email.

158. Write a SQL Query to find Max salary from each department.

159. Write SQL query to get the nth

highest salary among all Employees.

160. How can you find 10 employees with Odd number as Employee ID?

161. Write a SQL Query to get the names of employees whose date of birth is between 01/01/1990 to 31/12/2000.

162. Write a SQL Query to get the Quarter from date.

163. Write Query to find employees with duplicate email.

165. Is it safe to use ROWID to locate a record in Oracle SQL queries?

166. What is a Pseudocolumn?

167. What are the reasons for de-normalizing the data?

168. What is the feature in SQL for writing If/Else statements?

169. What is the difference between DELETE and TRUNCATE in SQL?

170. What is the difference between DDL and DML commands in SQL?

171. Why do we use Escape characters in SQL queries?

172. What is the difference between Primary key and Unique key in SQL?

173. What is the difference between INNER join and OUTER join in SQL?

174. What is the difference between Left OUTER Join and Right OUTER Join?

175. What is the datatype of ROWID?

176. What is the difference between where clause and having clause?

MySQL Questions

177. How will you calculate the number of days between two dates in MySQL?

178. What are the different types of Triggers in MySQL?

179. What are the differences between Heap table and temporary

table in MySQL?

180. What is a Heap table in MySQL?

181. What is the difference between BLOB and TEXT data type in MySQL?

182. What will happen when AUTO_INCREMENT on an INTEGER column reaches MAX_VALUE in MySQL?

183. What are the advantages of MySQL as compared with Oracle DB?

184. What are the disadvantages of MySQL?

185. What is the difference between

CHAR and VARCHAR datatype in MySQL?

186. What is the use of 'i_am_a_dummy flag' in MySQL?

187. How can we get current date and time in MySQL?

188. What is the difference between timestamp in Unix and MySQL?

189. How will you limit a MySQL query to display only top 10 rows?

190. What is automatic initialization and updating for TIMESTAMP in a MySQL table?

191. How can we get the list of all the indexes on a table?

192. What is SAVEPOINT in MySQL?

193. 17. What is the difference between ROLLBACK TO SAVEPOINT and RELEASE SAVEPOINT?

194. How will you search for a String in MySQL column?

195. How can we find the version of the MySQL server and the name of the current database by SELECT query?

196. What is the use of IFNULL() operator in MySQL?

197. How will you check if a table

exists in MySQL?

198. How will you see the structure of a table in MySQL?

199. What are the objects that can be created by CREATE statement in MySQL?

200. 24. How will you see the current user logged into MySQL connection?

201. How can you copy the structure of a table into another table without copying the data?

202. What is the difference between Batch and Interactive modes of MySQL?

203. How can we get a random

number between 1 and 100 in MySQL?

ACKNOWLEDGMENTS

We thank our readers who constantly send feedback and reviews to motivate us in creating these useful books with the latest information!

INTRODUCTION

Big Data and Data Science are the most popular technology trends. There is a growing demand for Data Engineer job in technology companies.

This book contains popular technical interview questions that an interviewer asks for Data Engineer position. The questions cover Hadoop, Hive, Spark, SQL and MySql areas.

Each question is accompanied with an answer so that you can prepare for job interview in short time.

We have compiled this list after attending dozens of technical interviews in top-notch companies like- Airbnb, Netflix, Amazon etc.

Often, these questions and concepts are used in our daily work. But these are most helpful when an Interviewer is trying to test your deep knowledge of Data Science and Big Data.

Once you go through them in the first pass, mark the questions that you could not answer by yourself. Then, in second pass go through only the difficult questions.

After going through this book 2-3 times, you will be well prepared to face a technical interview for a Data Engineer position.

Data Engineer Interview Questions

Apache Hadoop Questions

1. What are the four Vs of Big Data?

Data scientists at IBM have come up with an explanation for four Vs of Big Data. These are as follows:

1. **Volume**: Scale of data. Scale can be defined in the term of number of users, number of tables, size of data, number of records etc.
2. **Velocity**: Analysis of streaming data. In general Big Data keeps getting generated over time. Velocity

is the rate at which data keeps getting generated.

3. **Variety**: This point is about the different forms of Big Data. It can be in log files, media files, voice records, images etc.
4. **Veracity**: This is about certainty or uncertainty of data. How sure we are about the accuracy of the data.

2. What is the difference between Structured and Unstructured Big Data?

Structured Data can be stored in traditional database systems like-Oracle, SQL server, MS Access etc. It is stored in rows and columns. Most of the online application transactions are Structured Data. Structured data can be easily defined in a data model.

Unstructured data cannot be stored in terms of rows and columns. So it cannot be stored in traditional database systems. Generally, it has varying size and content. Some of the examples of unstructured data are log files, tweets, Facebook likes, Google search items.

Most of the Internet of Things (IOT) data is unstructured data. It is difficult to define unstructured data in a defined data model. Some of the software supporting unstructured data is: MongoDB, Hadoop etc.

3. What are the main components of a Hadoop Application?

Over the time, there are various forms in which a Hadoop application is defined. But in most of the cases there are following four core components of Hadoop application:

1. **HDFS**: This is the file system in which Hadoop data is stored. It is a distributed

file system with very high bandwidth.

2. **Hadoop Common**: This is common set of libraries and utilities used by Hadoop.

3. **Hadoop MapReduce**: This is based on MapReduce algorithm for providing large-scale data processing.

4. **Hadoop YARN**: This is used for resource management in a Hadoop cluster. It can also schedule tasks for users.

4. What is the core concept behind Apache Hadoop framework?

Apache Hadoop is based on the concept of MapReduce algorithm. In MapReduce algorithm, Map and Reduce operations are used to process very large data set.

In this concept, Map method does the filtering and sorting of data. Reduce method performs the summarizing of data.

This is a concept from functional programming.

The key points in this concept are scalability and fault tolerance. In Apache Hadoop these features are achieved by multi-threading and efficient implementation of MapReduce.

5. What is Hadoop Streaming?

Hadoop distribution provides a Java utility called Hadoop Streaming. It is packaged in a jar file. With Hadoop Streaming, we can create and run Map Reduce jobs with an executable script.

We can create executable scripts for Mapper and Reducer functions. These executable scripts are passed to Hadoop Streaming in a command.

Hadoop Streaming utility creates Map and Reduce jobs and submits these to a cluster. We can also monitor these jobs with this utility.

6. What is the difference between NameNode, Backup Node and Checkpoint NameNode in HDFS?

The differences between NameNode, BackupNode and Checkpoint NameNode are as follows:

NameNode: NameNode is at the

heart of the HDFS file system that manages the metadata i.e. the data of the files is not stored on the NameNode but rather it has the directory tree of all the files present in the HDFS file system on a Hadoop cluster. NameNode uses two files for the namespace:

- **fsimage file**: This file keeps track of the latest checkpoint of the namespace.

- **edits file**: This is a log of changes made to the namespace since checkpoint.

Checkpoint Node: Checkpoint

Node keeps track of the latest checkpoint in a directory that has same structure as that of NameNode’s directory.

Checkpoint node creates checkpoints for the namespace at regular intervals by downloading the edits and fsimage file from the NameNode and merging it locally. The new image is then again updated back to the active NameNode.

BackupNode: This node also provides check pointing functionality like that of the Checkpoint node but it also maintains its up-to-date in-memory copy of the file system namespace that is in sync with the active NameNode.

7. What is the optimum hardware configuration to run Apache Hadoop?

To run Apache Hadoop jobs, it is recommended to use dual core machines or dual processors. There should be 4GB or 8GB RAM with the processor with Error-correcting code (ECC) memory.

Without ECC memory, there is high

chance of getting checksum errors.

For storage high capacity SATA drives (around 7200 rpm) should be used in Hadoop cluster.

Around 10GB bandwidth Ethernet networks are good for Hadoop.

8. What do you know about Block and Block scanner in HDFS?

A large file in HDFS is broken into multiple parts and each part is stored on a different Block. By default a Block is of 64 MB capacity in HDFS.

Block Scanner is a program that every Data node in HDFS runs periodically to verify the checksum of every block stored on the data node.

The purpose of a Block Scanner is to detect any data corruption errors on Data node.

9. What are the default port numbers on which Name Node, Job Tracker and Task Tracker run in Hadoop?

Default port numbers of Name Node, Job Tracker and Task Tracker are as follows:

- NameNode runs on port 50070

- Task Tracker runs on port 50060
- Job Tracker runs on port 50030

10. How will you disable a Block Scanner on HDFS DataNode?

In HDFS, there is a configuration **dfs.datanode.scan.period.hours** in hdfs-site.xml to set the number of hours interval at which Block Scanner should run.

We can set **dfs.datanode.scan.period.hours**=0 to disable the Block Scanner. It means it will not run on HDFS DataNode.

11. How will you get the distance between two nodes in Apache Hadoop?

In Apache Hadoop we can use **NetworkTopology.getDistance()** method to get the distance between two nodes.

Distance from a node to its parent is considered as 1.

12. Why do we use commodity hardware in Hadoop?

Hadoop does not require a very high-end server with large memory and processing power. Due to this we can use any inexpensive system with average RAM and processor. Such kind of system is called commodity hardware.

Since there is parallel processing in Hadoop MapReduce, it is convenient to distribute a task among multiple servers and then do the execution. It saves cost as well as it is much faster compared to other options.

Another benefit of using commodity hardware in Hadoop is scalability. Commodity hardware is readily available in market. Whenever we need to scale up our operations in Hadoop cluster we can obtain more commodity hardware. In case of high-end machines, we have to raise purchase orders and get them built on demand.

13. How does inter cluster data copying works in Hadoop?

In Hadoop, there is a utility called **DistCP** (Distributed Copy) to perform large inter/intra-cluster copying of data. This utility is also based on MapReduce. It creates Map tasks for files given as input.

After every copy using DistCP, it is recommended to run crosschecks to confirm that there is no data corruption and copy is complete.

14. How can we update a file at an arbitrary location in HDFS?

This is a trick question. In HDFS, it is not allowed to update a file at an arbitrary location. All the files are written in append only mode. It means all writes are done at the end of a file.

So there is no possibility of updating the files at any random location.

15. What is Replication factor in HDFS, and how can we set it?

Replication factor in HDFS is the number of copies of a file in file system. A Hadoop application can specify the number of replicas of a file it wants HDFS to maintain.

This information is stored in NameNode.

We can set the replication factor in following ways:

- We can use Hadoop fs shell, to specify the replication factor for a file. Command as follows:

  ```
  $hadoop fs –setrep –w 5
  /file_name
  ```

  In above command, replication factor of file_name file is set as 5.

- We can also use Hadoop fs shell, to specify the replication factor of all the files in a directory.

  ```
  $hadoop fs –setrep –w 2
  /dir_name
  ```

In above command, replication factor of all the files under directory dir_name is set as 2.

16. What is the difference between NAS and DAS in Hadoop cluster?

NAS stands for Network Attached Storage and DAS stands for Direct Attached Storage.

In NAS, compute and storage layers are separated. Storage is distributed over different servers on a network.

In DAS, storage is attached to the node where computation takes place.

Apache Hadoop is based on the principle of moving processing near the location of data. So it needs storage disk to be local to computation.

With DAS, we get very good performance on a Hadoop cluster. Also DAS can be implemented on commodity hardware. So it is more cost effective.

Only when we have very high bandwidth (around 10 GbE) it is preferable to use NAS storage.

17. What are the two messages that NameNode receives from DataNode in Hadoop?

NameNode receives following two messages from every DataNode:

1. **Heartbeat**: This message signals that DataNode is still alive. Periodic receipt of Heartbeat is vey important for NameNode to decide whether to use a DataNode

or not.

2. **Block Report**: This is a list of all the data blocks hosted on a DataNode. This report is also very useful for functioning of NameNode. With this report, NameNode gets information about what data is stored on a specific DataNode.

18. How does indexing work in Hadoop?

Indexing in Hadoop has two different levels.

1. **Index based on File URI**: In this case data is indexed based on different files. When we search for data, index will return the files that contain the data.

2. **Index based on InputSplit**: In this case, data is indexed based on

locations where input split is located.

19. What data is stored in a HDFS NameNode?

NameNode is the central node of an HDFS system. It does not store any actual data on which MapReduce operations have to be done. But it has all the metadata about the data stored in HDFS DataNodes.

NameNode has the directory tree of all the files in HDFS filesystem. Using this meta data it manages all the data stored in different DataNodes.

20. What would happen if NameNode crashes in a HDFS cluster?

There is only one NameNode in a HDFS cluster. This node maintains metadata about DataNodes. Since there is only one NameNode, it is the single point of failure in a HDFS cluster. When NameNode crashes, system may become unavailable.

We can specify a secondary NameNode in HDFS cluster. The secondary

NameNode takes the periodic checkpoints of the file system in HDFS. But it is not the backup of NameNode. We can use it to recreate the NameNode and restart it in case of a crash.

21. What are the main functions of Secondary NameNode?

Main functions of Secondary NameNode are as follows:

1. **FsImage**: It stores a copy of FsImage file and EditLog.

2. **NameNode crash**: In case NameNode crashes, we can use Secondary NameNode's FsImage to recreate the NameNode.

3. **Checkpoint**: Secondary NameNode runs Checkpoint to confirm that data is not corrupt in HDFS.

4. **Update**: It periodically applies the updates from EditLog to FsImage file. In this way FsImage file on Secondary NameNode is kept up to date. This helps in saving time during NameNode restart.

22. What happens if HDFS file is set with replication factor of 1 and DataNode crashes?

Replication factor is same as the number of copies of a file on HDFS. If we set the replication factor of 1, it means there is only 1 copy of the file.

In case, DataNode that has this copy of file crashes, the data is lost. There is no way to recover it. It is essential to keep replication factor of more than 1 for any business critical data.

23. What is the meaning of Rack Awareness in Hadoop?

In Hadoop, most of the components like NameNode, DataNode etc are rack-aware. It means they have the information about the rack on which they exist. The main use of rack awareness is in implementing fault-tolerance.

Any communication between nodes on same rack is much faster than the communication between nodes on two different racks.

In Hadoop, NameNode maintains information about rack of each DataNode. While reading/writing data, NameNode tries to choose the DataNodes that are closer to each other. Due to performance reasons, it is recommended to use close data nodes for any operation.

So Rack Awareness is an important concept for high performance and fault-tolerance in Hadoop.

24. If we set Replication factor 3 for a file, does it mean any computation will also take place 3 times?

No. Replication factor of 3 means that there are 3 copies of a file. But computation takes place only one the one copy of file. If the node on which first copy exists, does not respond then computation will be done on second copy.

25. How will you check if a file exists in HDFS?

In Hadoop, we can run hadoop fs command with option e to check the existence of a file in HDFS. This is generally used for testing purpose. Command will be as follows:

%>hadoop fs -test -ezd file_uri

- e is for checking the existence of file
- z is for checking non-zero size of

file

- d is for checking if the path is directory

26. Why do we use fsck command in HDFS?

fsck command is used for getting the details of files and directories in HDFS.

Main uses of fsck command in HDFS are as follows:

1. **delete**: We use this option to delete files in HDFS.
2. **move**: This option is for moving corrupt files to lost/found.

3. **locations**: This option prints all the locations of a block in HDFS.

4. **racks**: This option gives the network topology of data-node locations.

5. **blocks**: This option gives the report of blocks in HDFS.

27. What will happen when NameNode is down and a user submits a new job?

Since NameNode is the single point of failure in Hadoop, user job cannot execute. The job will fail when the NameNode is down. User will have to wait for NameNode to restart and come up, before running a job.

28. What are the core methods of a Reducer in Hadoop?

The main task of Reducer is to reduce a larger set of data that shares a key to a smaller set of data. In Hadoop, Reducer has following three core methods:

1. **setup**(): At the start of a task, setup() method is called to configure various parameters for Reducer.

2. **reduce**(): This is the main

operation of Reducer. In reduce() method we define the task that has to be done for a set of values that share a key.

3. **cleanup**(): Once reduce() task is done, we can use cleanup() to clean any intermediate data or temporary files.

29. What are the primary phases of a Reducer in Hadoop?

In Hadoop, there are three primary phases of a Reducer:

1. **Shuffle**: In this phase, Reducer copies the sorted output from each Mapper.

2. **Sort**: In this phase, Hadoop framework sorts the input to Reducer by same key. It uses

merge sort in this phase.
Sometimes, shuffle and sort phases occur at the same time.

3. **Reduce**: This is the phase in which output values associated with a key are reduced to give output result. Output from Reducer is not re-sorted.

30. What is the use of Context object in Hadoop?

Hadoop uses Context object with Mapper to interact with rest of the system. Context object gets the configuration of the system and job in its constructor.

We use Context object to pass the information in setup(), cleanup() and map() methods. This is an important object that makes the important information available during the map operations.

31. How does partitioning work in Hadoop?

Partitioning is the phase between Map phase and Reduce phase in Hadoop workflow. Since partitioner gives output to Reducer, the number of partitions is same as the number of Reducers.

Partitioner will partition the output from Map phase into distinct partitions by using a user-defined condition. Partitions can be like Hash based buckets.

E.g. If we have to find the student with the maximum marks in each gender in each subject. We can first use Map function to map the keys with each gender. Once mapping is done, the result is passed to Partitioner. Partitioner will partition each row with gender on the basis of subject. For each subject there will be a different Reducer. Reducer will take input from each partition and find the student with the highest marks.

32. What is a Combiner in Hadoop?

Combiner is an optional step between Map and Reduce. Combiner is also called Semi-Reducer. Combiner takes output from Map, creates Key-value pairs and passes these to Reducer. Combiner's task is to summarize the outputs from Map into summary records with same key.

By using Combiner, we can reduce the data transfer between Mapper and

Reducer. Combiner does the task similar to reduce but it is done on the Map machine itself.

33. What is the default replication factor in HDFS?

Default replication factor in HDFS is 3. It means there will be 3 copies of each data.

We can configure it with **dfs.replication in hdfs-site.xml** file.

We can even set it from command line in Hadoop fs command.

34. How much storage is allocated by HDFS for storing a file of 25 MB size?

This is a question to test the basic concepts of HDFS. In HDFS, all the data is stored in blocks. The size of block can be configured in HDFS.

In Apache Hadoop, the default block size is 64 MB. To store a file of 25 MB size, at least one block will be allocated. This means at least 64 MB will be allocated for the file of 25 MB size.

35. Why does HDFS store data in Block structure?

HDFS stores all the data in terms of Blocks. With Block structure there are some benefits that HDFS gets. Some of these are as follows:

1. **Fault Tolerance**: With Block structure, HDFS implements replication. By replicating same block in multiple locations, fault tolerance of the system increases. Even if some copy is not

accessible, we can get the data from another copy.

2. **Large Files**: We can store very large files that cannot be even stored one disk, in HDFS by using Block structure. We just divide the data of file in multiple Blocks. Each Block can be stored on same or different machines.

3. **Storage management**: With Block storage it is easier for Hadoop nodes to calculate the data storage as well as perform optimization in the algorithm to minimize data transfer across the network.

36. How will you create a custom Partitioner in a Hadoop job?

Partition phase runs between Map and Reduce phase. It is an optional phase. We can create a custom partitioner by extending the **org.apache.hadoop.mapreduce.Partitio** class in Hadoop. In this class, we have to override **getPartition(KEY key, VALUE value, int numPartitions)** method.

This method takes three inputs. In this method, numPartitions is same as the number of reducers in our job. We pass key and value to get the partition number to which this key,value record will be assigned. There will be a reducer corresponding to that partition. The reducer will further handle to summarizing of the data.

Once custom Partitioner class is ready, we have to set it in the Hadoop job. We can use following method to set it:

```
job.setPartitionerClass(CustomPartitione
```

37. What are the differences between RDBMS and HBase data model?

The main differences between RDBMS and HBase data model are as follows:

1. **Schema**: In RDBMS, there is a schema of tables, columns etc. In HBASE, there is no schema.

2. **Normalization**: RDBMS data model is normalized as per the rule of Relation DB normalization.

HBase data model does not require any normalization.

3. **Partition**: In RDBMS we can choose to do partitioning. In HBase, partitioning happens automatically.

38. What is a Checkpoint node in HDFS?

A Checkpoint node in HDFS periodically fetches fsimage and edits from NameNode, and merges them. This merge result is called a Checkpoint. Once a Checkpoint is created, Checkpoint Node uploads the Checkpoint to NameNode.

Secondary node also takes Checkpoint similar to Checkpoint Node. But it does not upload the Checkpoint to

NameNode.

Main benefit of Checkpoint Node is in case of any failure on NameNode. A NameNode does not merge its edits to fsimage automatically during the runtime. If we have long running task, the edits will become huge. When we restart NameNode, it will take much longer time, because it will first merge the edits. In such a scenario, Checkpoint node helps for a long running task. Checkpoint nodes performs the task of merging the edits with fsimage and then uploads these to NameNode. This saves time during the restart of NameNode.

39. What is a Backup Node in HDFS?

Backup Node in HDFS is similar to Checkpoint Node. It takes the stream of edits from NameNode. It keeps these edits in memory and also writes these to storage to create a new checkpoint. At any point of time, Backup Node is in sync with the Name Node.

The difference between Checkpoint Node and Backup Node is that Backup Node does not upload any checkpoints to

Name Node. Also Backup node takes a stream instead of periodic reading of edits from Name Node.

40. What is the meaning of term Data Locality in Hadoop?

In a Big Data system, the size of data is huge. So it does not make sense to move data across the network. In such a scenario, Hadoop tries to move computation closer to data.

So the Data remains local to the location wherever it was stored. But the computation tasks will be moved to data nodes that hold the data locally.

Hadoop follows following rules for Data Locality optimization:

1. Hadoop first tries to schedule the task on node that has an HDFS file on a local disk.
2. If it cannot be done, then Hadoop will try to schedule the task on a node on the same rack as the node that has data.
3. If this also cannot be done, Hadoop will schedule the task on the node with same data on a different rack.

The above method works well, when we work with the default replication factor of 3 in Hadoop.

41. What is the difference between Data science, Big Data and Hadoop?

The difference between Data Science, Big Data and Hadoop is as follows:

Data Science is an approach of handling problem with the help of Statistics, Mathematics, Computer Science, Machine learning, Data mining and predictive analysis. With Data Science we can extract knowledge from data in different forms. Current Data Science methods make very good use of Big Data processing systems.

Big Data is the solution for handling huge amount of data with the modern tools and algorithms. It is a way of processing and analyzing the data that is too big for a traditional database management system to handle. With the mobile devices, Internet of things and sharing applications, datasets have become huge. Big Data tools and techniques come handy in processing such a large data set.

Hadoop is a framework for handling Big Data. It uses commodity hardware and distributed storage to process large data sets. It is one of the most popular Big Data frameworks these days.

42. What is a Balancer in HDFS?

In HDFS, data is stored in blocks on a DataNode. There can be a situation when data is not uniformly spread into blocks on a DataNode. When we add a new DataNode to a cluster, we can face such a situation.

In such a case, HDFS provides a useful tool Balancer to analyze the placement of blocks on a DataNode. Some people call it as Rebalancer also. This is an administrative tool used by admin staff. We can use this tool to spread the blocks in a uniform manner on a DataNode.

43. What are the important points a NameNode considers before selecting the DataNode for placing a data block?

Some of the important points for selecting a DataNode by NameNode are as follows:

1. NameNode tries to keep at least one replica of a Block on the same node that is writing the block.

2. It tries to spread the different replicas of same block on different racks, so that in case of one rack failure, other rack has the data.

3. One replica will be kept on a node on the same node as the one that it writing it. It is different from point 1. In Point 1, block is written to same node. In this point block is written on a different node on same rack. This is important for minimizing the network I/O.

NameNode also tries to spread the blocks uniformly among all the DataNodes in a cluster.

44. What is Safemode in HDFS?

Safemode is considered as the read-only mode of NameNode in a cluster. During the startup of NameNode, it is in SafeMode. It does not allow writing to file-system in Safemode. At this time, it collects data and statistics from all the DataNodes. Once it has all the data on blocks, it leaves Safemode.

The main reason for Safemode is to

avoid the situation when NameNode starts replicating data in DataNodes before collecting all the information from DataNodes. It may erroneously assume that a block is not replicated well enough, whereas, the issue is that NameNode does not know about whereabouts of all the replicas of a block. Therefore, in Safemode, NameNode first collects the information about how many replicas exist in cluster and then tries to create replicas wherever the number of replicas is less than the policy.

45. How will you replace HDFS data volume before shutting down a DataNode?

In HDFS, DataNode supports hot swappable drives. With a swappable drive we can add or replace HDFS data volumes while the DataNode is still running.

The procedure for replacing a hot swappable drive is as follows:

- First we format and mount the new drive.
- We update the DataNode configuration dfs.datanode.data.dir to reflect the data volume directories.
- Run the "dfsadmin -reconfig datanode HOST:PORT start" command to start the reconfiguration process
- Once the reconfiguration is complete, we just unmount the old data volume
- After unmount we can physically remove the old disks.

46. What are the important configuration files in Hadoop?

There are two important configuration files in a Hadoop cluster:

1. **Default Configuration**: There are core-default.xml, hdfs-default.xml and mapred-default.xml files in which we specify the default configuration for Hadoop cluster. These are read only files.

2. **Custom Configuration**: We have site-specific custom files like core-site.xml, hdfs-site.xml, mapred-site.xml in which we can specify the site-specific configuration.

All the Jobs in Hadoop and HDFS implementation uses the parameters defined in the above-mentioned files. With customization we can tune these processes according to our use case.

In Hadoop API, there is a Configuration class that loads these files and provides the values at run time to different jobs.

47. How will you monitor memory used in a Hadoop cluster?

In Hadoop, TaskTracker is the one that uses high memory to perform a task. We can configure the TastTracker to monitor memory usage of the tasks it creates. It can monitor the memory usage to find the badly behaving tasks, so that these tasks do not bring the machine down with excess memory consumption.

In memory monitoring we can also limit

the maximum memory used by a tasks. We can even limit the memory usage per node. So that all the tasks executing together on a node do not consume more memory than a limit.

Some of the parameters for setting memory monitoring in Hadoop are as follows:

- mapred.cluster.map.memory.mb, mapred.cluster.reduce.memory.mb: This is the size of virtual memory of a single map/reduce slot in a cluster of Map-Reduce framework.
- mapred.job.map.memory.mb, mapred.job.reduce.memory.mb: This is the default limit of memory

set on each map/reduce task in Hadoop.

- mapred.cluster.max.map.memory.m
  mapred.cluster.max.reduce.memory
  This is the maximum limit of memory set on each map/reduce task in Hadoop.

48. Why do we need Serialization in Hadoop map reduce methods?

In Hadoop, there are multiple data nodes that hold data. During the processing of map and reduce methods data may transfer from one node to another node. Hadoop uses serialization to convert the data from Object structure to Binary format.

With serialization, data can be converted to binary format and with deserialization data can be converted back to Object format with reliability.

49. What is the use of Distributed Cache in Hadoop?

Hadoop provides a utility called Distributed Cache to improve the performance of jobs by caching the files used by applications.

An application can specify which file it wants to cache by using JobConf configuration.

Hadoop framework copies these files to the nodes one which a task has to be executed. This is done before the start of execution of a task.

DistributedCache supports distribution of simple read only text files as well as complex files like jars, zips etc.

50. How will you synchronize the changes made to a file in Distributed Cache in Hadoop?

It is a trick question. In Distributed Cache, it is not allowed to make any changes to a file. This is a mechanism to cache read-only data across multiple nodes.

Therefore, it is not possible to update a cached file or run any synchronization in

Distributed Cache.

Apache Hive Questions

51. How will you improve the performance of a program in Hive?

There are many ways to improve the performance of a Hive program. Some of these are as follows:

1. **Data Structure**: We have to select the right data structure for our purpose in a Hive program.
2. **Standard Library**: Wherever possible, we

should use methods from standard library. Methods implemented in standard library have much better performance than user implementation.

3. **Abstraction**: At times, a lot of abstraction and indirection can cause slow performance of a program. We should remove the redundant abstraction in code.
4. **Algorithm**: Use of right algorithm can make a big difference in a program. We have to find and select the suitable algorithm to solve our problem with high

performance.

52. Can we use Hive for Online Transaction Processing (OLTP) systems?

No, Hive is not suitable for OLTP systems. It does not support insert and update at each row level. In OLTP systems, we have to update each order or each row.

53. How will you change the data type of a column in Hive?

We can use Alter statement to change the data type of a column in Hive.

Command will be as follows:

```
ALTER TABLE tableName REPLACE COLUMNS (columnName dataType)
```

54. What is Metastore in Hive?

Hive table definitions, mappings and other metadata are stored in Metastore. This can be stored in a RDBMS supported by JPOX.

By default only one user can see Metastore at a time.

Metastore has two parts.

1. **Metastore Service**: This service provides interface between Hive and users/processed.

2. **Metastore Database**: This database stores table definitions, data mappings etc.

In default configuration, Hive Metastore is stored in same JVM as Hive driver. This is also known as Embedded Metastore, which is good for Dev and Testing environments.

We can also use an external DB like MySQL for storing Metadata. This configuration is called Local Metastore.

55. What is SerDe in Hive?

SerDe is a short name for Serializer/Deserializer. In Hive, SerDe is used for input/output interface. For any communication we have to Serialize and Deserialize the data.

With SerDe, Hive can read data from a table and write it to HDFS in a custom format. We can also implement custom SerDe to handle our own data formats.

During map and reduce, Hive can

use serialize methods of LazyBinarySerDe and BinarySortableSerDe.

We can also use ObjectInspector, for a SerDe to serialize an object created by another SerDe.

56. What are the components in Hive data model?

There are following components in Hive data model:

1. **Tables**: These are tables similar to Relation Database (RDBMS). We can create filters on tables. We can run joins and union on tables. Table data is stored in HDFS. Rows in a table are organized like columns in a RDBMS. Each row can have

a datatype.

2. **Partitions**: In each table in Hive, we can specify partition keys that determine how the data is stored. With partition we can optimize the queries to only looks specific dataset instead of scanning whole table.

3. **Buckets**: In each partition, we can divide the data into buckets. Each bucket is based on the hash of a column in a table. There is a separate file in which each bucket of a partition is

stored. With buckets it is efficient to evaluate queries based on a sample of data.

57. What are the different modes in which we can run Hive?

We can run Hive in following modes:

1. **Local mode**: In Hive local mode, Map Reduce jobs related to Hive run locally on a user machine. This is the default mode in which Hadoop uses local file system.

2. **Distributed Mode**: In this mode, Hive as well as Hadoop is running in a fully distributed mode. NameNode, DataNode, JobTracker, TaskTracker etc run on different machines in this mode.

3. **Pseudo-distributed Mode**: This is the mode used by developers to test the code before deploying to production. In this mode, all the daemons run on same virtual machine. With this mode, we can quickly write

scripts and test on limited data sets.

58. What are the main components of Hive?

Main components of Hive are as follows:

1. **Hive Driver**: Hive Driver receives queries from users. It handles sessions and provides commands to execute and fetch APIs that are similar to JDBC/ODBC interface.

2. **Metastore**: All the Metadata information about Hive tables and partitions is stored in Metastore. It also maintains information about columns and column types. It knows about the serializers and deserializers that are used in reading and writing data to corresponding HDFS files.

3. **Hive Compiler**: The compiler in Hive parses queries and performs semantic analysis on different query blocks and query expressions. It also

generates the execution plan for looking up data from tables and partitions.

4. **Execution Engine**: This is the Hive component that executes the execution plan created by compiler. The Execution engine manages the dependencies between different stages of the execution plan. It executes different stages of Execution plan on the relevant system components.

5. **User Interface (UI):** Hive provides a UI for users to

interact with the system. Users can submit queries and execute other operations on the system.

59. What is the use of Hive in Hadoop eco-system?

Hive provides a SQL like interface to interact with data stored in Hadoop eco-system. We can create tables in Hive and store data into them.

Hive is also used for mapping and working with HBase tables.
Under the hood, Hive queries are converted into MapReduce jobs. So Hive hides the complexity associated with creating and

running of MapReduce jobs.

60. What Collection/Complex data types are supported by Hive?

Hive supports following Complex/Collection data types:

1. **Map**: This is a collection of key value pairs. We can specify [key] to get the value stored at key in the collection.

2. **Array**: This is an index based ordered sequence of

values in a collection. We can access it with an index like [i] where i is the location of an element.

3. **Struct**: This is a record type that has other named fields inside. It is similar to struct in C or object in Java.

4. **Uniontype**: This is similar to Union in C. Since support for this data type is not fully available, we should not use it.

61. What is the use of .hiverc file in Hive?

In Hive, .hiverc is the initialization file. This file is loaded when we start Command Line Interface for Hive. We can set the initial values of parameters in this file.

62. How will you run Unix commands from Hive?

We can run Unix commands from Hive shell by using ! sign. We have to append ! sign before a unix command.

E.g. For using date command we have to run it as !date in Hive shell.

63. What is the purpose of USE command in Hive?

In Hive we have to call USE command before running and hive queries.

USE command sets the current database on which query statements will be executed. We can call USE default to set the database to default database.

Or we can call USE database_name to set the database on which we want to run the queries.

64. What is the precedence order in Hive configuration?

In Hive we can use following precedence order to set the configurable properties.

1. Hive SET command has the highest priority
2. -hiveconf option from Hive Command Line
3. hive-site.xml file
4. hive-default.xml file

5. hadoop-site.xml file
6. hadoop-default.xml file

65. How will you display header row with the results of a Hive query?

Hive provides a property hive.cli.print.header to set the configuration in which header row is printed. It is as follows:

```
hive> set hive.cli.print.header=true;
```

66. Can we create multiple tables in Hive for a data file?

Yes, we can create multiple table schemas for a data file. Hive will save schema in Hive Metastore. Data will remain in same file. Based on the schema we can retrieve different results from same Data.

67. How does CONCAT function work in Hive?

Hive provides CONCAT function to concatenate the input strings. We can specify multiple strings in a comma-separated list.

There is another function CONCAT_WS in Hive that can be used to specify the custom delimiter.

68. How will you change settings of a Hive session?

We can use SET command to change the settings within a Hive Session. We can even change the settings of MapReduce job with SET command.

If we just call SET command with a property name, we will get the current value of the property.

With SET -v we get the list of all the properties including Hadoop default in Hive system.

69. How will you rename a table in Hive without using ALTER command?

We can use IMPORT/EXPORT statements to rename a table in Hive.

We first export the original table to a location by EXPORT statement.

```
hive> EXPORT TABLE tableName TO
'TEMP_LOCATION'
```

Then we create new table with IMPORT statement.

```
hive> IMPORT TABLE newTableName FROM ’TEMP_LOCATION’
```

70. What is the difference between SORT BY and ORDER BY in Hive?

In Hive we use, SORT BY to sort data in each Reducer. We can use multiple Reducers in SORT BY clause.

We can use ORDER BY to sort all the data that passes through one Reducer. So ORDER BY option can be use only with one Reducer.

ORDER BY guarantees total order in the output. SORT BY guarantees ordering only within the data of one reducer.

71. What is the use of strict mode in Hive?

We use strict mode in Hive to avoid performing full scan of a table in a query. It is very useful when we have a very large table. We can use **‘set hive.mapred.mode=strict’** command to put Hive into strict mode. In this mode, any query that needs full table scan will throw error to user.

Strict mode is a useful option in

Hive to improve performance and avoid any unexpected delays due to full table scan.

72. What is the use of IF EXISTS clause in Hive statements?

We use IF EXISTS clause in DROP TABLE statement. When we drop a table, there can be a case that table does not exist. In such a case DROP statement will give error.

In case we are not sure if the table exists or not in Hive, we use DROP TABLE IF EXISTS tableName statement to drop a table. This statement will not give error in case table does not exist.

73. What is the use of PURGE in DROP statement of Hive?

We use PURGE option when we are absolutely sure to delete the data of a table. With DROP TABLE tableName PURGE statement, table data will not go to trash folder. So it cannot be retrieved in case of an accidental DROP statement.

74. What are the main limitations of Apache Hive?

Apache Hive is a popular tool in Hadoop eco-system. Some of the limitations of Hive are as follows:

1. **OLTP**: Hive cannot be used for OLTP applications, since it does not support record level update, insert statements.

2. **Data Warehouse**: Hive is more suitable for data

warehouse applications on Big Data.

3. **Latency**: Hive queries have higher latency than SQL due to time overhead during start-up.

4. **Batch Oriented**: Hive is mostly used in Batch-oriented use cases.

5. **OLAP**: For Online Analytic Processing (OLAP) we can use Hive. We can use HBase also for OLAP processing.

75. What is the difference between HBase and Hive?

HBase is a NoSQL database. It supports update, insert and delete statements at record level. HBase does not support a query language like SQL. But we can use Hive with HBase for querying purpose.

Hive runs on MapReduce. HBase runs on HDFS.

Apache Hive is a data warehousing and data analytics framework that is used for data warehouse applications. HBase is a NoSQL database on top of HDFS.

76. What is ObjectInspector in Hive?

ObjectInspector is used in Hive to look into the internal structure of a complex object. It is an interface that is implemented by classes like StandardListObjectInspector, StandardMapObjectInspector, StandardSetObjectInspector etc.

Main use of ObjectInspector is in SerDe package for Serialization and Deserialization.

We have to make sure that hashCode() and equals() methods of Object class work for ObjectInspector as well.

77. What are the main components of Query Processor in Apache Hive?

The main components of Apache Hive Query Processor are as follows:

- Parse and SemanticAnalysis (ql/parse)
- Optimizer (ql/optimizer)
- Plan Components (ql/plan)
- MetaData Layer (ql/metadata)

- Map/Reduce Execution Engine (ql/exec)
- Sessions (ql/session)
- Type interfaces (ql/typeinfo)
- Hive Function Framework (ql/udf)
- Tools (ql/tools)

78. How will you resolve an out of memory error while running a JOIN query?

In case of JOIN query with large tables, we have a lot of data in memory. This can cause out of memory error.

One simple solution for handling this error is to change the RIGHT OUTER JOIN to LEFT OUTER

JOIN. We should try to put the table with a large number of rows on the rightmost side in a JOIN query.

79. What are the different SerDe implementations in Hive?

There are many implementations of SerDe in Hive. We can also write our own custom SerDe implementation. Some of the popular implementations are as follows:

- ByteStreamTypedSerDe
- RegexSerDe
- OpenCSVSerde
- DelimitedJSONSerDe

80. What is the use of HCatalog?

Hadoop provides a table and storage management layer called HCatalog. With HCatalog users in Hive can use different data processing tools like- Pig, MapReduce, Hive, Streaming etc to read and write data on the grid.

With HCatalog we can read and write files in any format for which SerDe is available. Default options supported by HCatalog are: RCFile, CSV, JSON and

SequenceFile.

In HCatalog we can see the data in a relational view.

81. What is the Data Model of HCatalog?

HCatalog has same datatypes as Hive. We can create tables in HCatalog to store data. These tables can be placed in a Database.

We can also partition HCatalog tables based on a Hash. There are multiple dimensions in a HCatalog partition.

There are records in a partition. Once a partition is created, we cannot add or remove records into it.

Records are divided into columns. Each column has a name and a datatype.

Overall, HCatalog data model is similar to Relational Data Model.

82. What is RLIKE operator in Hive?

RLIKE is an operator similar to LIKE in SQL. We use LIKE to search for string with similar text.

E.g. user_name LIKE ‘%Smith’

Hive provides RLIKE operator that can be used for searching advanced Regular Expressions in Java.

E.g. user_name RLIKE ‘.*(Smith|Sam).*’

This will return user_name that has Smith or Sam in it.

83. Can we use same name for a TABLE and VIEW in Hive?

No, we cannot use same name for a table and view in Hive. So we have to select a unique name for a view in Hive.

84. How will you load data into a VIEW in Hive?

This is a trick question. We cannot use VIEW in Hive INSERT or LOAD statement. We can only load data into a Hive table.

85. What is Bucketing in Hive?

With Bucketing in Hive, we can divide datasets into manageable parts. It is similar to hashing. Even if the size of data set varies, we can still have fixed number of buckets.

Also with bucketing we can do map-side joins in Hive.

E.g. Let say we have a table with date as a first level partition and user_id as second level partition.

The date may have smaller number of partitions. But user_id may have a large number of partitions. If we have millions of users, there will be millions of second level partitions and files.

To avoid creating so many partitions, we can do bucketing instead of partitioning on user_id. With bucketing, we can use HASH function to put different user_ids in different buckets. We can create a manageable number of buckets for user_id values.

86. What are the pros and cons of archiving a partition in Hive?

We can archive some less used partitions in Hive. The main advantage of archiving is that it will decrease the number of files to be stored in NameNode. We can even query an archived partition in Hive.

The main disadvantage of archiving is that queries become slower and less efficient in Hive.

87. What are the table generating functions in Hive?

Table generating functions in Hive are user-defined functions like CONCAT that take a single row and give an output of multiple rows.

Some of the Table generating functions are:

- EXPLODE(array)
- EXPLODE(map)
- STACK()

- JSON_TUPLE()

88. How can we specify in Hive to load an HDFS file in LOAD DATA?

We have to remove the LOCAL clause from LOAD DATA statement to avoid loading the local file. Once LOCAL clause is removed, we can load HDFS file.

89. What is a Skewed table in Hive?

A Skewed tables is a special type of table in which some values in a column appear more often. Due to this the distribution in skewed. In Hive, when we specify a table as SKEWED during creation, then skewed values are written into separate files and remaining values go to another file.

E.g. CREATE TABLE tableName (column1 STRING, column2

STRING) SKEWED BY (column1) on (‘value1’)

During queries, we get better performance in Hive with SKEWED tables.

90. What is the use of CLUSTERED BY clause during table creation in Hive?

CLUSTERED BY in Hive is same as DISTRIBUTE BY and SORT BY. When we specify CLUSTERED BY, it will first distribute the data into different reducers by using a Hash. Once data is distributed, it will sort the data.

We have to specify CLUSTERED BY clause during table creation. But it is useful in querying of data in Hive.

91. What is a Managed table in Hive?

Managed tables are the tables in which files, metadata and statistics etc are managed by internal Hive processes. Hive creates Managed tables by default. When we drop a managed table or partition, then all the metadata and data associated with the table is also deleted.

We use Managed tables, when we want Hive to manage the lifecycle of a table. Even for temporary tables, we use managed tables in Hive.

When we run DESCRIBE FORMATTED tableName statement, it displays whether a table is MANAGED_TABLE or EXTERNAL_TABLE.

92. How will you prevent data to be dropped or queried from a partition in Hive?

We can use ALTER TABLE table_name ENABLE NO_DROP to prevent a table partition from being dropped.

We can use ALTER TABLE table_name ENABLE OFFLINE to prevent a table partition from being queried. In offline mode, we can still access metadata of a table partition.

93. What is the use of TOUCH in ALTER statement?

In Hive, TOUCH clause in ALTER statement is used to read the metadata and write it back. This operation will modify the last accessed time of a partition in Hive.

With TOUCH statement we can also execute the POST and PRE hooks on a table partition.

This statement cannot be used for

creating a table or partition if it does not exist yet.

94. How does OVERWRITE clause work in CREATE TABLE statement in Hive?

We use OVERWRITE clause in CREATE TABLE statement to delete the existing data and write new data in a Hive table. Essentially, as the name suggests, OVERWRITE helps in overwriting existing data in a Hive table.

95. What are the options to connect an application to a Hive server?

We can use following options to connect an application a Hive server:

JDBC Driver: We can use JDBC Driver with embedded as well as remote access to connect to HiveServer. This is for Java based connectivity.

Python Client: For Python language application there is Python client that can connect to Hive server.

Ruby Client: With Ruby client driver also we can connect to Hive server.

Thrift Client: We can use Beeline command line shell to connect to Hive server over Thrift. For production mode, this is one of the very good options. It is a secure option for production use. Also we do not need to grant HDFS access to users for using Thrift client.

96. How TRIM and RPAD functions work in Hive?

TRIM and RPAD functions are for processing String data type in Hive.

With TRIM function we can delete the spaces before and after a String. It is very useful for formatting user input in which user may have entered extra spaces. The other variations of TRIM function are LTRIM and RTRIM that remove spaces from left and right side of the string respectively.

E.g. TRIM(‘ Smith ’)
Smith

RPAD function is used to add padding (extra spaces) in a String on the right hand side. So that String reaches a specified length. LPAD function is same as RPAD but it pads on the left hand side of String.

E.g. Let say we have a String “Hello”.

LPAD(‘Hello’,8,’ ')
   Hello

We can also specify our optional padding character in RPAD and LPAD functions.

These functions are similar to the ones in SQL.

97. How will you recursively access sub-directories in Hive?

We can use following commands in Hive to recursively access sub-directories:

```
hive> Set mapred.input.dir.recursive=true;
```

```
hive> Set hive.mapred.supports.subdirectories=true
```

Once above options are set to true,

Hive will recursively access sub-directories of a directory in MapReduce.

98. What is the optimization that can be done in SELECT * query in Hive?

We can convert some of the SELECT queries in Hive into single FETCH task. With this optimization, latency of SELECT query is decreased.

To use this we have to set the value of hive.fetch.task.conversion parameter. The permissible values are:

- 0: It means FETCH is disabled.
- 1: It is **minimal** mode. SELECT *, FILTER on partition columns (WHERE and HAVING clauses), LIMIT only
- 2: It is **more** mode: SELECT, FILTER, LIMIT only (including virtual columns)

"more" can even take UDF expressions in the SELECT clause.

99. What is the use of ORC format tables in Hive?

We use Optimized Row Columnar (ORC) file format to store data efficiently in Hive. It is used for performance improvement in reading, writing and processing of data.

In ORC format, we can overcome the limitations of other Hive file formats. Some of the advantages of ORC format are:

- There is single file as the output of each task. This reduces load on NameNode.

- It supports date time, decimal, struct, map etc complex types.

- It stores light-weight indexes within the file.

- We can bound the memory used in read/write of data.

- It stores metadata with Protocol Buffers that supports add/remove of fields.

100. What are the main use cases for using Hive?

Hive is mainly used for Datawarehouse applications. Hive used Hadoop and MapReduce that put some restrictions on use cases for Hive.

Some of the main use cases for Hive are:

- Analysis of static Big data
- Applications in which less responsive time is acceptable
- Analysis of data that does not

change rapidly

Bonus Questions

101. What is STREAMTABLE in Hive?

In Hive, we can optimize a query by using STREAMTABLE hint. We can specify it in SELECT query with JOIN. During the map/reduce stage of JOIN, a table data can be streamed by using this hint.

E.g.

```
SELECT /*+
STREAMTABLE(table1) */
table1.val, table2.val
FROM table1 JOIN table2 ON
```

(table1.key = table2.key1)

In above query we are using table1 as a stream.

If we do not use STREAMTABLE hint then Hive will stream the right most table in the JOIN query.

Apache Spark Questions

102. What are the main features of Apache Spark?

Main features of Apache Spark are as follows:

1. **Performance**: The key feature of Apache Spark is its Performance. With Apache Spark we can run programs up to 100 times faster than Hadoop MapReduce in memory. On disk we can run it 10 times faster than Hadoop.

2. **Ease of Use**: Spark supports Java, Python, R, Scala etc. languages. So it makes it much easier to develop applications for Apache Spark.

3. **Integrated Solution**: In Spark we can create an integrated solution that combines the power of SQL, Streaming and data analytics.

4. **Run Everywhere**: Apache Spark can run on many platforms. It can run on Hadoop, Mesos, in Cloud or

standalone. It can also connect to many data sources like HDFS, Cassandra, HBase, S3 etc.

5. **Stream Processing:** Apache Spark also supports real time stream processing. With real time streaming we can provide real time analytics solutions. This is very useful for real-time data.

103. What is a Resilient Distribution Dataset in Apache Spark?

Resilient Distribution Dataset (RDD) is an abstraction of data in Apache Spark. It is a distributed and resilient collection of records spread over many partitions.

RDD hides the data partitioning and distribution behind the scenes.

Main features of RDD are as follows:

1. **Distributed**: Data in a RDD is distributed across multiple nodes.

2. **Resilient**: RDD is a fault-tolerant dataset. In case of node failure, Spark can re-compute data.

3. **Dataset**: It is a collection of data similar to collections in Scala.

4. **Immutable**: Data in RDD cannot be modified after

creation. But we can transform it using a Transformation.

104. What is a Transformation in Apache Spark?

Transformation in Apache Spark is a function that can be applied to a RDD. Output of a Transformation is another RDD.

Transformation in Spark is a lazy operation. It means it is not executed immediately. Once we call an action, transformation is executed.

A Transformation does not change

the input RDD.

We can also create a pipeline of certain Transformations to create a Data flow.

105. What are security options in Apache Spark?

Apache Spark provides following security options:

1. **Encryption**: Apache Spark supports encryption by SSL. We can use HTTPS protocol for secure data transfer. Therefore data is transmitted in encrypted mode. We can use spark.ssl parameters to set the SSL configuration.

2. **Authentication**: We can perform authentication by a shared secret in Apache Spark. We can use spark.authenticate to configure authentication in Spark.

3. **Event Logging**: If we use Event Logging, then we can set the permissions on the directory where event logs are stored. These permissions can ensure access control for Event log.

106. How will you monitor Apache Spark?

We can use the Web UI provided by SparkContext to monitor Spark. We can access this Web UI at port 4040 to get the useful information. Some of the information that we can monitor is:

1. Scheduler tasks and stages
2. RDD Sizes and Memory usage
3. Spark Environment Information

4. Executors Information

Spark also provides a Metrics library. This library can be used to send Spark information to HTTP, JMX, CSV files etc.

This is another option to collect Spark runtime information for monitoring another dashboard tool.

107. What are the main libraries of Apache Spark?

Some of the main libraries of Apache Spark are as follows:

1. **MLib**: This is Spark’s Machine Learning library. We can use it to create scalable machine learning system. We can use various machine learning algorithms as well as features like pipelining etc with this library.

2. **GraphX**: This library is used for computation of Graphs. It helps in creating a Graph abstraction of data and then use various Graph operators like- SubGraph, joinVertices etc.

3. **Structured Streaming**: This library is used for handling streams in Spark. It is a fault tolerant system built on top of Spark SQL Engine to process streams.

4. **Spark SQL:** This is another popular component that is

used for processing SQL queries on Spark platform.

5. **SparkR**: This is a package in Spark to use Spark from R language. We can use R data frames, dplyr etc from this package. We can also start SparkR from RStudio.

108. What are the main functions of Spark Core in Apache Spark?

Spark Core is the central component of Apache Spark. It serves following functions:

1. Distributed Task Dispatching
2. Job Scheduling
3. I/O Functions

109. How will you do memory tuning in Spark?

In case of memory tuning we have to take care of these points.

1. Amount of memory used by objects
2. Cost of accessing objects
3. Overhead of Garbage Collection

Apache Spark stores objects in memory for caching. So it becomes important to perform memory tuning in a Spark application.

First we determine the memory usage by the application. To do this we first create a RDD and put it in cache. Now we can see the size of the RDD in storage page of Web UI. This will tell the amount of memory consumed by RDD.

Based on the memory usage, we can estimate the amount of memory needed for our task.

In case we need tuning, we can follow these practices to reduce memory usage:

1. Use data structures like

Array of objects or primitives instead of Linked list or HashMap. Fastutil library provides convenient collection classes for primitive types compatible with Java.

2. We have to reduce the usage of nested data structures with a large number of small objects and pointes. E.g. Linked list has pointers within each node.

3. It is a good practice to use numeric IDs instead of Strings for keys.

4. We can also use JVM flag -XX:+UseCompressedOops to make pointers be four bytes instead of eight.

110. What are the two ways to create RDD in Spark?

We can create RDD in Spark in following two ways:

1. **Internal**: We can parallelize an existing collection of data within our Spark Driver program and create a RDD out of it.

2. **External**: We can also create RDD by referencing a Dataset in an external data

source like AWS S3, HDFS, HBASE etc.

111. What are the main operations that can be done on a RDD in Apache Spark?

There are two main operations that can be performed on a RDD in Spark:

1. **Transformation**: This is a function that is used to create a new RDD out of an existing RDD.

2. **Action**: This is a function that returns a value to Driver program after running a computation on RDD.

112. What are the common Transformations in Apache Spark?

Some of the common transformations in Apache Spark are as follows:

1. **Map(func):** This is a basic transformation that returns a new dataset by passing each element of input dataset through func function.

2. **Filter(func):** This transformation returns a new dataset of elements that return true for func function. It is used to filter elements in a dataset based on criteria in func function.

3. **Union(other Dataset):** This is used to combine a dataset with another dataset to form a union of two datasets.

4. **Intersection(other Dataset):** This transformation gives the elements common to two datasets.

5. **Pipe(command, [envVars]):** This transformation passes each partition of the dataset through a shell command.

113. What are the common Actions in Apache Spark?

Some of the commonly used Actions in Apache Spark are as follows:

1. **Reduce(func):** This Action aggregates the elements of a dataset by using func function.

2. **Count():** This action gives the total number of elements in a Dataset.

3. **Collect():** This action will return all the elements of a dataset as an Array to the driver program.

4. **First():** This action gives the first element of a collection.

5. **Take(n):** This action gives the first n elements of dataset.

6. **Foreach(func):** This action runs each element in dataset through a for loop and executes function func on each element.

114. What is a Shuffle operation in Spark?

Shuffle operation is used in Spark to re-distribute data across multiple partitions.

It is a costly and complex operation.

In general a single task in Spark operates on elements in one partition. To execute shuffle, we have to run an operation on all elements of all partitions. It is also called all-to-all operation.

115. What are the operations that can cause a shuffle in Spark?

Some of the common operations that can cause a shuffle internally in Spark are as follows:

1. Repartition
2. Coalesce
3. GroupByKey
4. ReduceByKey
5. Cogroup
6. Join

116. What is purpose of Spark SQL?

Spark SQL is used for running SQL queries. We can use Spark SQL to interact with SQL as well as Dataset API in Spark.

During execution, Spark SQL uses same computation engine for SQL as well as Dataset API.

With Spark SQL we can get more information about the structure of data as well as computation being performed.

We can also use Spark SQL to read data from an existing Hive installation.

Spark SQL can also be accessed by using JDBC/ODBC API as well as command line.

117. What is a DataFrame in Spark SQL?

A DataFrame in SparkSQL is a Dataset organized into names columns. It is conceptually like a table in SQL.

In Java and Scala, a DataFrame is a represented by a DataSet of rows.

We can create a DataFrame from an existing RDD, a Hive table or from other Spark data sources.

118. What is a Parquet file in Spark?

Apache Parquet is a columnar storage format that is available to any project in Hadoop ecosystem. Any data processing framework, data model or programming language can use it.

It is a compressed, efficient and encoding format common to Hadoop system projects.

Spark SQL supports both reading

and writing of parquet files. Parquet files also automatically preserves the schema of the original data.

During write operations, by default all columns in a parquet file are converted to nullable column.

119. What is the difference between Apache Spark and Apache Hadoop MapReduce?

Some of the main differences between Apache Spark and Hadoop MapReduce are follows:

1. **Speed:** Apache Spark is 10X to 100X faster than Hadoop due to its usage of in memory processing.

2. **Memory**: Apache Spark stores data in memory, whereas Hadoop MapReduce stores data in hard disk.

3. **RDD**: Spark uses Resilient Distributed Dataset (RDD) that guarantee fault tolerance. Where Apache Hadoop uses replication of data in multiple copies to achieve fault tolerance.

4. **Streaming**: Apache Spark supports Streaming with very less administration. This

makes it much easier to use than Hadoop for real-time stream processing.

5. **API**: Spark provides a versatile API that can be used with multiple data sources as well as languages. It is more extensible than the API provided by Apache Hadoop.

120. What are the main languages supported by Apache Spark?

Some of the main languages supported by Apache Spark are as follows:

1. **Java**: We can use JavaSparkContext object to work with Java in Spark.

2. **Scala**: To use Scala with Spark, we have to create

SparkContext object in Scala.

3. **Python**: We also used SparkContext to work with Python in Spark.

4. **R**: We can use SparkR module to work with R language in Spark ecosystem.

5. **SQL**: We can also SparkSQL to work with SQL language in Spark.

121. What are the file systems supported by Spark?

Some of the popular file systems supported by Apache Spark are as follows:

1. HDFS
2. S3
3. Local File System
4. Cassandra
5. OpenStack Swift
6. MapR File System

122. What is a Spark Driver?

Spark Driver is a program that runs on the master node machine. It takes care of declaring any operation- Transformation or Action on a RDD.

With Spark Driver was can keep track of all the operations on a Dataset. It can also be used to rebuild a RDD in Spark.

123. What is an RDD Lineage?

Resilient Distributed Dataset (RDD) Lineage is a graph of all the parent RDD of a RDD. Since Spark does not replicate data, it is possible to lose some data. In case some Dataset is lost, it is possible to use RDD Lineage to recreate the lost Dataset.

Therefore RDD Lineage provides solution for better performance of Spark as well as it helps in building a resilient system.

124. What are the two main types of Vector in Spark?

There are two main types of Vector in Spark:

Dense Vector: A dense vector is backed by an array of double data type. This array contains the values.
E.g. {1.0 , 0.0, 3.0}

Sparse Vector: A sparse vector is backed by two parallel arrays. One array is for indices and the other array is for values.

E.g. {3, [0,2], [1.0,3.0]}

In this array, the first element is the number of elements in vector. Second element is the array of indices of non-zero values. Third element is the array of non-zero values.

125. What are the different deployment modes of Apache Spark?

Some of the popular deployment modes of Apache Spark are as follows:

1. **Amazon EC2**: We can use AWS cloud product Elastic Compute Cloud (EC2) to deploy and run a Spark cluster.

2. **Mesos**: We can deploy a Spark application in a private cluster by using Apache Mesos.

3. **YARN**: We can also deploy Spark on Apache YARN (Hadoop NextGen)

4. **Standalone**: This is the mode in which we can start Spark by hand. We can launch standalone cluster manually.

126. What is lazy evaluation in Apache Spark?

Apache Spark uses lazy evaluation as a performance optimization technique. In Laze evaluation as transformation is not applied immediately to a RDD. Spark records the transformations that have to be applied to a RDD. Once an Action is called, Spark executes all the transformations.

Since Spark does not perform immediate execution based on transformation, it is called lazy evaluation.

127. What are the core components of a distributed application in Apache Spark?

Core components of a distributed application in Apache Spark are as follows:

1. **Cluster Manager**: This is the component responsible for launching executors and drivers on multiple nodes.

We can use different types of cluster managers based on our requirements. Some of the common types are Standalone, YARN, Mesos etc.

2. **Driver**: This is the main program in Spark that runs the main() function of an application. A Driver program creates the SparkConetxt. Driver program listens and accepts incoming connections from its executors. Driver program can schedule tasks on the cluster. It runs closer

to worker nodes.

3. **Executor**: This is a process on worker node. It is launched on the node to run an application. It can run tasks and use data in memory or disk storage to perform the task.

128. What is the difference in cache() and persist() methods in Apache Spark?

Both cache() and persist() functions are used for persisting a RDD in memory across operations. The key difference between persist() and cache() is that in persist() we can specify the Storage level that we select for persisting. Where as in cache(), default strategy is used for persisting. The default storage strategy is MEMORY_ONLY.

129. How will you remove data from cache in Apache Spark?

In general, Apache Spark automatically removes the unused objects from cache. It uses Least Recently Used (LRU) algorithm to drop old partitions. There are automatic monitoring mechanisms in Spark to monitor cache usage on each node.

In case we want to forcibly

remove an object from cache in Apache Spark, we can use RDD.unpersist() method.

130. What is the use of SparkContext in Apache Spark?

SparkContext is the central object in Spark that coordinates different Spark applications in a cluster.

In a cluster we can use SparkContext to connect to multiple Cluster Managers that allocate resources to multiple applications.

For any Spark program we first create SparkContext object. We can access a cluster by using this object. To create a SparkContext object, we first create a SparkConf object. This object contains the configuration information of our application.

In Spark Shell, by default we get a SparkContext for the shell.

131. Do we need HDFS for running Spark application?

This is a trick question. Spark supports multiple file-systems. Spark supports HDFS, HBase, local file system, S3, Cassandra etc. So HDFS is not the only file system for running Spark application.

132. What is Spark Streaming?

Spark Streaming is a very popular feature of Spark for processing live streams with a large amount of data.

Spark Streaming uses Spark API to create a highly scalable, high throughput and fault tolerant system to handle live data streams.

Spark Streaming supports ingestion of data from popular sources like- Kafka, Kinesis, Flume etc.

We can apply popular functions like map, reduce, join etc on data processed through Spark Streams.

The processed data can be written to a file system or sent to databases and live dashboards.

133. How does Spark Streaming work internally?

Spark Streams listen to live data streams from various sources. On receiving data, it is divided into small batches that can be handled by Spark engine. These small batches of data are processed by Spark Engine to generate another output stream of resultant data.

Internally, Spark uses an abstraction called DStream or discretized stream. A DStream is a continuous stream of data. We can create DStream from Kafka, Flume, Kinesis etc.

A DStream is nothing but a sequence of RDDs in Spark.

We can apply transformations and actions on this sequence of RDDs to create further RDDs.

134. What is a Pipeline in Apache Spark?

Pipeline is a concept from Machine learning. It is a sequence of algorithms that are executed for processing and learning from data.

A Pipeline is similar to a workflow. There can be one or more stages in a Pipeline.

135. How does Pipeline work in Apache Spark?

A Pipeline is a sequence of stages. Each stage in Pipeline can be a Transformer or an Estimator. We run these stages in an order. Initially a DataFrame is passed as an input to Pipeline. This DataFrame keeps on transforming with each stage of Pipeline.

Most of the time, Runtime checking is done on DataFrame passing through the Pipeline. We can also save a Pipeline to a disk. It can be re-read from disk a later point of time.

136. What is the difference between Transformer and Estimator in Apache Spark?

A Transformer is an abstraction for feature transformer and learned model. A Transformer implements transform() method. It converts one DataFrame to another DataFrame. It appends one or more columns to a DataFrame.

In a feature transformer a DataFrame is the input and the output is a new DataFrame with a new mapped column.

An Estimator is an abstraction for a learning algorithm that fits or trains on data. An Estimator implements fit() method. The fit() method takes a DataFrame as input and results in a Model.

137. What are the different types of Cluster Managers in Apache Spark?

Main types of Cluster Managers for Apache Spark are as follows:

1. **Standalone**: It is a simple cluster manager that is included with Spark. We can start Spark manually by hand in this mode.

2. **Spark on Mesos**: In this

mode, Mesos master replaces Spark master as the cluster manager. When driver creates a job, Mesos will determine which machine will handle the task.

3. **Hadoop YARN**: In this setup, Hadoop YARN is used in cluster. There are two modes in this setup. In cluster mode, Spark driver runs inside a master process managed by YARN on cluster. In client mode, the Spark driver runs in the client process and application master is used

for requesting resources from YARN.

138. How will you minimize data transfer while working with Apache Spark?

Generally Shuffle operation in Spark leads to a large amount of data transfer. We can configure Spark Shuffle process for optimum data transfer. Some of the main points are as follows:

1. **spark.shuffle.compress**:

This configuration can be set to true to compress map output files. This reduces the amount of data transfer due to compression.

2. **ByKey operations**: We can minimize the use of ByKey operations to minimize the shuffle calls.

139. What is the main use of MLib in Apache Spark?

MLib is a machine-learning library in Apache Spark. Some of the main uses of MLib in Spark are as follows:

1. **ML Algorithms**: It contains Machine Learning algorithms such as classification, regression, clustering, and collaborative filtering.

2. **Featurization**: MLib

provides algorithms to work with features. Some of these are feature extraction, transformation, dimensionality reduction, and selection.

3. **Pipelines**: It contains tools for constructing, evaluating, and tuning ML Pipelines.

4. **Persistence**: It also provides methods for saving and load algorithms, models, and Pipelines.

5. **Utilities**: It contains utilities for linear algebra, statistics,

data handling, etc.

140. What is the Checkpointing in Apache Spark?

In Spark Streaming, there is a concept of Checkpointing to add resiliency in the application. In case of a failure, a streaming application needs a checkpoint to recover. Due to this Spark provides Checkpointing. There are two types of Checkpointing:

1. **Metadata Checkpointing**: Metadata is the configuration information and other

information that defines a Streaming application. We can create a Metadata checkpoint for a node to recover from the failure while running the driver application. Metadata includes configuration, DStream operations and incomplete batches etc.

2. **Data Checkpointing**: In this checkpoint we save RDD to a reliable storage. This is useful in stateful transformations where generated RDD depends on RDD of previous batch.

There can be a long chain of RDDs in some cases. To avoid such a large recovery time, it is easier to create Data Checkpoint with RDDs at intermediate steps.

141. What is an Accumulator in Apache Spark?

An Accumulator is a variable in Spark that can be added only through an associative and commutative operation. An Accumulator can be supported in parallel.

It is generally used to implement a counter or cumulative sum.

We can create numeric type Accumulators by default in Spark.

An Accumulator variable can be named as well as unnamed.

142. What is a Broadcast variable in Apache Spark?

As per Spark online documentation, “A Broadcast variable allows a programmer to keep a read-only variable cached on each machine rather than shipping a copy of it with tasks.” Spark can also distribute broadcast variable with an efficient broadcast algorithm to reduce communication cost.

In Shuffle operations, there is a

need of common data. This common data is broadcast by Spark as a Broadcast variable. The data in these variables is serialized and de-serialized before running a task.

We can use SparkContext.broadcast(v) to create a broadcast variable.

It is recommended that we should use broadcast variable in place of original variable for running a function on cluster.

143. What is Structured Streaming in Apache Spark?

Structured Streaming is a new feature in Spark 2.1. It is a scalable and fault-tolerant stream-processing engine. It is built on Spark SQL engine. We can use Dataset or DataFrame API to express streaming aggregations, event-time windows etc. The computations are done on the optimized Spark SQL engine.

144. How will you pass functions to Apache Spark?

In Spark API, we pass functions to driver program so that it can be run on a cluster. Two common ways to pass functions in Spark are as follows:

1. **Anonymous Function Syntax**: This is used for passing short pieces of code in an anonymous function.

2. **Static Methods in a**

Singleton object: We can also define static methods in an object with only once instance i.e. Singleton. This object along with its methods can be passed to cluster nodes.

145. What is a Property Graph?

A Property Graph is a directed multigraph. We can attach an object on each vertex and edge of a Property Graph.

In a directed multigraph, we can have multiple parallel edges that share same source and destination vertex.

During modeling the data, the option of parallel edges helps in creating multiple relationships between same pair of vertices.

E.g. Two persons can have two relationships Boss as well as Mentor.

146. What is Neighborhood Aggregation in Spark?

Neighborhood Aggregation is a concept in Graph module of Spark. It refers to the task of aggregating information about the neighborhood of each vertex.
E.g. We want to know the number of books referenced in a book. Or number of times a Tweet is retweeted.

This concept is used in iterative graph algorithms. Some of the popular uses of this concept are in Page Rank, Shortest Path etc.

We can use aggregateMessages[] and mergeMsg[] operations in Spark for implementing Neighborhood Aggregation.

147. What are different Persistence levels in Apache Spark?

Different Persistence levels in Apache Spark are as follows:

1. **MEMORY_ONLY**: In this level, RDD object is stored as a de-serialized Java object in JVM. If an RDD doesn’t fit in the memory, it will be recomputed.

2. **MEMORY_AND_DISK**: In this level, RDD object is stored as a de-serialized Java object in JVM. If an RDD doesn’t fit in the memory, it will be stored on the Disk.

3. **MEMORY_ONLY_SER**: In this level, RDD object is stored as a serialized Java object in JVM. It is more efficient than de-serialized object.

4. **MEMORY_AND_DISK_SE** In this level, RDD object is stored as a serialized Java

object in JVM. If an RDD doesn’t fit in the memory, it will be stored on the Disk.

5. **DISK_ONLY**: In this level, RDD object is stored only on Disk.

148. How will you select the storage level in Apache Spark?

We use storage level to maintain balance between CPU efficiency and Memory usage.

If our RDD objects fit in memory, we use MEMORY_ONLY option. In this option, the performance is very good due to objects being in Memory only.

In case our RDD objects cannot fit in memory, we go for MEMORY_ONLY_SER option and select a serialization library that can provide space savings with serialization. This option is also quite fast in performance.

In case our RDD object cannot fit in memory with a big gap in memory vs. total object size, we go for MEMORY_AND_DISK option. In this option some RDD object are stored on Disk.

For fast fault recovery we use replication of objects to multiple partitions.

149. What are the options in Spark to create a Graph?

We can create a Graph in Spark from a collection of vertices and edges. Some of the options in Spark to create a Graph are as follows:

1. **Graph.apply**: This is the simplest option to create graph. We use this option to create a graph from RDDs of vertices and edges.

2. **Graph.fromEdges**: We can also create a graph from RDD of edges. In this option, vertices are created automatically and a default value is assigned to each vertex.

3. **Graph.fromEdgeTuples**: We can also create a graph from only an RDD of tuples.

150. What are the basic Graph operators in Spark?

Some of the common Graph operators in Apache Spark are as follows:

1. numEdges
2. numVertices
3. inDegrees
4. outDegrees
5. degrees
6. vertices

7. edges
8. persist
9. cache
10. unpersistVertices
11. partitionBy

151. What is the partitioning approach used in GraphX of Apache Spark?

GraphX uses Vertex-cut approach to distributed graph partitioning.

In this approach, a graph is not split along edges. Rather we partition graph along vertices. These vertices can span on multiple machines.

This approach reduces communication and storage overheads.

Edges are assigned to different partitions based on the partition strategy that we select.

SQL Tricky Questions

152. Write SQL query to get the second highest salary among all Employees?

Given a Employee Table with two columns

ID, Salary

10, 2000

11, 5000

12, 3000

Answer:

There are multiple ways to get the second highest salary among all Employees.

Option 1: Use Subquery

```
SELECT MAX(Salary)
FROM Employee
WHERE Salary NOT IN (SELECT
MAX(Salary) FROM Employee );
```

In this approach, we are getting the maximum salary in a subquery and then excluding this from the rest of the resultset.

Option 2: Use Not equals

```
select MAX(Salary) from Employee
WHERE Salary <> (select
MAX(Salary) from Employee )
```

This is same as option 1 but we are using <> instead of NOT IN.

153. How can we retrieve alternate records from a table in Oracle?

We can use rownum and MOD function to retrieve the alternate records from a table.

To get Even number records:

```
SELECT *
FROM (SELECT rownum, ID, Name
      FROM Employee)
WHERE MOD(rownum,2)=0
```

To get Odd number records:

```
SELECT *
FROM (SELECT rownum, ID, Name
      FROM Employee)
WHERE MOD(rownum,2)=1
```

154. Write a SQL Query to find Max salary and Department name from each department.

Given a Employee table with three columns

ID, Salary, DeptID

10, 1000, 2

20, 5000, 3

30, 3000, 2

Department table with two columns:

ID, DeptName

1, Marketing

2, IT

3, Finance

Answer:

This is a trick question. There can be some department without any employee. So we have to ask interviewer if they expect the name of such Department also in result.

If yes then we have to join Department table with Employee table by using foreign key DeptID. We have to use LEFT OUTER JOIN to print all the departments.

Query would be like as follows:

```
SELECT d.DeptName, MAX(e.Salary)
FROM Department d LEFT OUTER
JOIN Employee e
ON e.DeptId = d.ID
GROUP BY DeptName
```

155. Write a SQL query to find records in Table A that are not in Table B without using NOT IN operator.

Consider two tables

Table_A

10

20

30

Table_B

15

30

45

Answer: We can use MINUS operator in this case for Oracle and EXCEPT for SQL Server.

Query will be as follows:

```
SELECT * FROM Table_A
MINUS
SELECT * FROM Table_B
```

156. What is the result of following query?

```
SELECT
    CASE WHEN null = null
    THEN ‘True’
    ELSE ‘False’
END AS Result;
```

Answer: In SQL null can not be compared with itself. There fore null = null is not true. We can compare null with a non-null value to check whether a value is not null.

Therefore the result of above query is False.

The correct way to check for null is to use IS NULL clause.

Following query will give result True.

```
SELECT
    CASE WHEN null IS NULL
    THEN ‘True’
    ELSE 'False'
END AS Result;
```

157. Write SQL Query to find employees that have same name and email.

Employee table:

ID	NAME	EMAIL
10	John	jbaldwin
20	George	gadams
30	John	jsmith

Answer: This is a simple question with one trick. The trick here is to use **Group by on two columns** Name and Email.

Query would be as follows:

```
SELECT name, email, COUNT(*)
FROM Employee
GROUP BY name, email
HAVING
COUNT(*) > 1
```

158. Write a SQL Query to find Max salary from each department.

Given a Employee table with three columns

ID, Salary, DeptID

10, 1000, 2

20, 5000, 3

30, 3000, 2

Answer:

We can first use group by DeptID on Employee table and then get the Max salary from each Dept group.

```
SELECT   DeptID, MAX(salary)
FROM    Employee
GROUP BY   DeptID
```

159. Write SQL query to get the nth highest salary among all Employees.

Given a Employee Table with two columns

ID, Salary

10, 2000

11, 5000

12, 3000

Answer:

Option 1: Use Subquery

We can use following sub query approach for this:

```
SELECT *
FROM Employee emp1
WHERE (N-1) = (
SELECT
COUNT(DISTINCT(emp2.salary))
FROM Employee emp2
WHERE emp2.salary > emp1.salary)
```

Option 2: Using Rownum in Oracle

```
SELECT * FROM (
  SELECT emp.*,
row_number() OVER (ORDER BY
salary DESC) rnum
FROM Employee emp
)
WHERE rnum = n;
```

160. How can you find 10 employees with Odd number as Employee ID?

Answer:

In Oracle we can use Top to limit the number of records. We can also use Rownum < 11 to get the only 10 or less number of records.

To find the Odd number Employee ID, we can use % function.

Sample Query with TOP:

```
SELECT TOP 10 ID FROM Employee
WHERE ID % 2 = 1;
```

Sample Query with ROWNUM:

```
SELECT ID FROM Employee WHERE
ID % 2 = 1 AND ROWNUM < 11;
```

161. Write a SQL Query to get the names of employees whose date of birth is between 01/01/1990 to 31/12/2000.

This SQL query appears a bit tricky. We can use BETWEEN clause to get all the employees whose date of birth lies between two given dates.

Query will be as follows:

```
SELECT EmpName
FROM Employees
WHERE birth_date  BETWEEN
‘01/01/1990’ AND ‘31/12/2000’
```

Remember BETWEEN is always inclusive of both the dates.

162. Write a SQL Query to get the Quarter from date.

Answer: We can use to_char function with ‘Q’ option for quarter to get quarter from a date.

Use TO_CHAR with option ‘Q’ for Quarter

```
SELECT TO_CHAR(TO_DATE('3/31/2016',
'MM/DD/YYYY'), 'Q')
AS quarter
```

FROM DUAL

163. Write Query to find employees with duplicate email.

Employee table:

ID	NAME	EMAIL
10	John	jsmith
20	George	gadams
30	Jane	jsmith

Answer: We can use Group by clause on the column in which we want to find duplicate values.

Query would be as follows:

```
SELECT name, COUNT(email)
FROM Employee
GROUP BY email
HAVING ( COUNT(email) > 1 )
```

164. Write a Query to find all

Employee whose name contains the word "Rich", regardless of case. E.g. Rich, RICH, rich.

Answer:

We can use UPPER function for comparing the both sides with uppercase.

```
SELECT *
FROM Employees
WHERE  UPPER(emp_name) like
'%RICH%'
```

165. Is it safe to use ROWID to locate a record in Oracle SQL queries?

ROWID is the physical location of a row. We can do very fast lookup based on ROWID. In a transaction where we first search a few rows and then update them one by one, we can use ROWID.

But ROWID of a record can change over

time. If we rebuild a table a record can get a new ROWID.

If a record is deleted, its ROWID can be given to another record.

So it is not recommended to store and use ROWID in long term. It should be used in same transactions.

166. What is a Pseudocolumn?

A Pseudocolumn is like a table column, but it is not stored in the same table. We can select from a Pseudocolumn, but we can not insert, update or delete on a Pseudocolumn.

A Pseudocolumn is like a function with no arguments.

Two most popular Pseudocolumns in Oracle are ROWID and ROWNUM.

NEXTVAL and CURRVAL are also pseudo columns.

167. What are the reasons for de-normalizing the data?

We de-normalize data when we need better performance. Sometimes there are many joins in a query due to highly normalized data.

In that case, for faster data retrieval it becomes essential to de-normalize data.

168. What is the feature in SQL for writing If/Else statements?

In SQL, we can use CASE statements to write If/Else statements.

We can also use DECODE function in Oracle SQL for writing simple If/Else logic.

169. What is the difference between DELETE and TRUNCATE in SQL?

Main differences between DELETE and TRUNCATE commands are:

1. **DML vs. DDL**: DELETE is a Data Manipulation Language (DML) command. TRUNCATE is a Data Definition Language (DDL)

command.

2. **Number of Rows**: We can use DELETE command to remove one or more rows from a table. TRUNCATE command will remove all the rows from a table.

3. **WHERE clause**: DELETE command provides support for WHERE clause that can be used to filter the data that we want to delete. TRUNCATE command can only delete all the rows. There is no WHERE clause in TRUNCATE command.

4. **Commit**: After DELETE command we have to issue COMMIT or ROLLBACK command to confirm our changes. After TRUNCATE command there is no need to run COMMIT. Changes done by TRUNCATE command can not be rolled back.

170. What is the difference between DDL and DML commands in SQL?

Main differences between Data Definition Language (DDL) and Data Manipulation Language (DML) commands are:

1. **DDL vs. DML**: DDL statements are used for creating and defining the Database structure. DML statements are used for managing data within Database.

2. **Sample Statements**: DDL statements are CREATE, ALTER, DROP, TRUNCATE, RENAME etc. DML statements are SELECT, INSERT, DELETE, UPDATE, MERGE, CALL etc.

3. **Number of Rows**: DDL statements work on whole table. CREATE will a create a new table. DROP will remove the

whole table. TRUNCATE will delete all records in a table. DML statements can work on one or more rows. INSERT can insert one or more rows. DELETE can remove one or more rows.

4. **WHERE clause**: DDL statements do not have a WHERE clause to filter the data. Most of DML statements support filtering the data by WHERE clause.

5. **Commit**: Changes done by a DDL statement can not be rolled back. So there is no need to issue a COMMIT or ROLLBACK

command after DDL statement. We need to run COMMIT or ROLLBACK to confirm our changed after running a DML statement.

6. **Transaction**: Since each DDL statement is permanent, we can not run multiple DDL statements in a group like Transaction. DML statements can be run in a Transaction. Then we can COMMIT or ROLLBACK this group as a transaction. E.g. We can insert data in two tables and commit it together in a transaction.

7. **Triggers**: After DDL statements no triggers are fired. But after DML statements relevant triggers can be fired.

171. Why do we use Escape characters in SQL queries?

In SQL, there are certain special characters and words that are reserved for special purpose. E.g. & is a reserved character.

When we want to use these special characters in the context of our data, we have to use Escape characters to pass the message to database to interpret these as non Special / non Reserved characters.

172. What is the difference between Primary key and Unique key in SQL?

Main differences between Primary key and Unique key in SQL are:

1. **Number**: There can be only one Primary key in a table. There can be more than one Unique key in a table.

2. **Null value**: In some DBMS Primary key cannot be NULL. E.g. MySQL adds NOT NULL to Primary key. A Unique key can have null values.

3. **Unique Identifier**: Primary Key is a unique identifier of a record in database table. Unique key can be null and we may not be able to identify a record in a unique way by a unique key

4. **Changes**: It is not recommended to change a Primary key. A Unique key can be changed much easily.

5. **Usage**: Primary Key is used to identify a row in a table. A Unique key is used to prevent duplicate non-null values in a column.

173. What is the difference between INNER join and OUTER join in SQL?

Let say we have two tables X and Y.

The result of an INNER JOIN of X and Y is X intersect. It is the INNER overlapping intersection part of a Venn diagram.

The result of an OUTER JOIN of X and Y is X union Y. It is the OUTER parts of a Venn diagram.

E.g.

Consider following two tables, with just one column x and y:

x	y
10	30
20	40

30 | 50

40 | 60

In above tables (10,20) are unique to table X, (30,40) are common, and (50,60) are unique to table Y.

INNER JOIN

An INNER JOIN by using following query will give the intersection of the two tables X and Y. The intersection is the common data between these tables.

select * from X INNER JOIN Y on X.x =

```
Y.y;

x   | y
--- +--
30  | 30
40  | 40
```

OUTER JOIN

A full OUTER JOIN by using following query will us the union of X and Y. It will have all the rows in X and all the rows in Y. If some row in X has not corresponding value in Y, then Y side will be null, and vice versa.

```
select * from X FULL OUTER JOIN Y
on X.x = Y.y;

x    | y
----- + -----
  10 | null
  20 | null
  30 |   30
  40 |   40
null |   60
null |   50
```

174. What is the difference between Left OUTER Join and Right OUTER Join?

Let say we have two tables X and Y.

The result of an LEFT OUTER JOIN of X and Y is all rows of X and common rows between X and Y.

The result of an RIGHT OUTER JOIN of X and Y is all rows of Y and common rows between X and Y.

E.g.

Consider following two tables, with just one column x and y:

```
x  | y
---|--
10 | 30
20 | 40
```

```
30 | 50
40 | 60
```

In above tables (10,20) are unique to table X, (30,40) are common, and (50,60) are unique to table Y.

LEFT OUTER JOIN

A left OUTER JOIN by using following query will give us all rows in X and common rows in X and Y.

```
select * from X LEFT OUTER JOIN Y
on X.x = Y.y;
```

```
x   | y
-- -+-----
10  |null
20  |null
30  |   30
40  |   40
```

RIGHT OUTER JOIN

A right OUTER JOIN by using following query will give all rows in Y and common rows in X and Y.

```
select * from X RIGHT OUTER JOIN Y
on X.x = Y.y;
```

```
x     | y
----- +----
30    | 30
40    | 40
null  | 50
null  | 60
```

175. What is the datatype of ROWID?

ROWID Pseudocolumn in Oracle is of ROWID datatype. It is a string that represents the address of a row in the database.

176. What is the difference between where clause and having clause?

We use where clause to filter elements based on some criteria on individual records of a table.

E.g. We can select only employees with first name as John.

```
SELECT ID, Name
```

```
FROM Employee
WHERE name = ‘John’
```

We use having clause to filter the groups based on the values of aggregate functions.

E.g. We can group by department and only select departments that have more than 10 employees.

```
SELECT deptId, count(1)
FROM Employee
GROUP BY deptId HAVING count(*) >
10.
```

MySQL Questions

177. How will you calculate the number of days between two dates in MySQL?

We can use DATEDIFF function for this purpose.

The query to get the number of days between two dates in MySQL is as follows:

```
SELECT DATEDIFF('2016-12-31', '2015-01-01');
```

178. What are the different types of Triggers in MySQL?

MySQL supports six types of triggers. These are as follows:

1. **Before Insert**: This trigger runs before inserting a new row in a table.
2. **After Insert**: This trigger

runs after inserting a new row in a table.

3. **Before Update**: This trigger runs before updating an existing row in a table.
4. **After Update**: This trigger runs after updating an existing row in a table.
5. **Before Delete**: This trigger runs before deleting an existing row in a table.
6. **After Delete**: This trigger runs after deleting an existing row in a table.

179. What are the differences between Heap table and temporary table in MySQL?

1. **Duration**: Heap tables are stored in memory. Therefore a Heap table remains in existence even if the session is disconnected. When we restart Database, Heap tables get cleaned up.

2. Temporary tables are valid only during a session. Once the session is disconnected, temporary table is cleaned up.

3. **Privilege**: We need special privilege to create a Temporary table. Heap tables are just another form of storage in MySQL.

4. **Sharing**: Temporary tables are not shared between clients. Each connection will have a unique temporary table. But Heap tables can be shared between clients.

180. What is a Heap table in MySQL?

In MySQL there are tables that are present in memory. These are called Heap tables. During table creation we specify TYPE as HEAP for HEAP tables.

Heap tables provide very fast access to data.

We can not store BLOB or TEXT datatype in a HEAP table. These tables also do not support

AUTO_INCREMENT.

Once we restart the Database, data in HEAP tables is lost.

181. What is the difference between BLOB and TEXT data type in MySQL?

BLOB is a Binary large Object. We can store a large amount of binary data in a BLOB data type column. TEXT is non-binary, character based string data type. We can store text data in TEXT column. We have to define a character set with a TEXT column.

TEXT can be easily converted into plain text.

BLOB has four types: TINYBLOB, BLOB, MEDIUMBLOB and LONGBLOB. Where as, TEXT has its own four types: TINYTEXT, TEXT, MEDIUMTEXT, LONGTEXT.

182. What will happen when AUTO_INCREMEN on an INTEGER column reaches MAX_VALUE in MySQL?

Once a column reaches the MAX_VALUE, the AUTO_INCREMENT stops working. It gives following error in log:

ERROR: 1467 (HY000): Failed to read auto-increment value from storage engine

183. What are the advantages of MySQL as compared with Oracle DB?

Some of the main advantages of MySQL over Oracle DB are as follows:

1. **Cost**: MySQL is an Open Source and free RDBMS software. Oracle is usually a paid option for RDBMS.

2. **Space**: MySQL uses around 1 MB to run whereas Oracle may need as high as 128 MB to run the database server.

3. **Flexibility**: MySQL can be used to run a small website as well as very large scale systems. Oracle is generally used in medium to large scale systems.

4. **Management**: In MySQL, database administration is much easier due to self-management features like-automatic space expansion,

auto-restart and dynamic configuration changes. In Oracle dedicated DBA has to work on managing the Database.

5. **Portable**: MySQL is easily portable to different hardware and operating system. Migrating Oracle from one platform to another is a tougher task.

184. What are the disadvantages of MySQL?

Some of the main disadvantages of MySQL are as follows:

Dependent on Additional S/W: MySQL has less number of features in standard out-of-box version. So we have to add additional software to get more features. It gets difficult to find, decide and use the additional software with MySQL.

SQL Compliance: MySQL is not full SQL compliant. Due to this

developers find it difficult to cope with the syntax of SQL in MySQL.
Transaction handling: Some users complain that DB transactions are not handled properly in MySQL.

185. What is the difference between CHAR and VARCHAR datatype in MySQL?

Some of the main differences between CHAR and VARCHAR datatypes in MySQL are as follows:

1. **Size**: In a CHAR type column, length is fixed. In a

VARCHAR type column length can vary.

2. **Storage**: There are different mechanisms to store and retrieve CHAR and VARCHAR data types in MySQL.

3. **Maximum Size**: A CHAR data type can hold maximum 255 characters. A VARCHAR datatype can store up to 4000 characters.

4. **Speed**: CHAR datatype is 50% faster than VARCHAR datatype in MySQL.

5. **Memory Allocation**: A CHAR datatype column uses static memory allocation. Since the length of data stored in a VARCHAR can vary, this datatype uses dynamic memory allocation.

186. What is the use of 'i_am_a_dummy flag' in MySQL?

In MySQL, there is falg "ia_am_a_dummy" that can be used to save beginner developers from erroneous query like 'DELETE FROM table_name'. If we run this query it will delete all the data from table names table_name.

With 'i_am_a_dummy flag',

MySQL will not permit running such a query. It will prompt user to create a query with WHERE clause so that only specific data is deleted.

We can achieve similar functionality with 'safe_updates' option in MySQL.

This flag also works on UPDATE statement to restrict updates on a table without WHERE clause.

187. How can we get current date and time in MySQL?

We can use following query in MySQL to get the current date:

```
SELECT CURRENT_DATE();
```

We can use following query in MySQL to get the current time as well as date:

```
SELECT NOW();
```

188. What is the difference between timestamp in Unix and MySQL?

In Unix as well as in MySQL, timestamp is stored as a 32-bit integer.

A timestamp is the number of seconds from the Unix Epoch on January 1st, 1970 at UTC.

In MySQL we can represent the timestamp in a readable format.

Timestamp format in MySQL is YYYY-MM-DD HH:MM:SS

189. How will you limit a MySQL query to display only top 10 rows?

We can use LIMIT clause in MySQL to limit a query to a range of rows.

Following query will give top 10 rows from the table with table_name:

```
SELECT * FROM <table_name>
LIMIT 0,10;
```

Following query will give 6 rows starting from the 4th row in table with table_name:

```
SELECT * FROM <table_name>
LIMIT 3,6;
```

190. What is automatic initialization and updating for TIMESTAMP in a MySQL table?

In MySQL, there is a TIMESTAMP datatype that provides features like automatic initialization and updating to current time and date.

If a column is auto-initialized, then it will be set to current timestamp on inserting a new row with no value for the column.

If a column is auto-updated, then its value will be updated to current timestamp when the value of any other column in the same row is updated.

We can mark a column as DEFAULT to prevent this auto-initialize and auto-update behavior.

191. How can we get the list of all the indexes on a table?

We can use following command to get the list of all the indexes on a table in MySQL:

```
SHOW INDEX FROM table_name;
```

At maximum we can use 16 columns in a multi-column index of table.

192. What is SAVEPOINT in MySQL?

SAVEPOINT is a statement in SQL. We can use SAVEPOINT <savepoint_name> statement to create a point of time in a Database transaction with a name. Later we can use this savepoint to rollback the transaction upto that point of time.

193. 17. What is the difference between ROLLBACK TO SAVEPOINT and RELEASE SAVEPOINT?

We use ROLLBACK TO SAVEPOINT statement to undo the effect of a transaction upto the SAVEPOINT mentioned in ROLLBACK statement.

RELEASE SAVEPOINT is simply used to delete the SAVEPOINT with a name from a transaction. There is commit or rollback for SAVEPOINT in RELEASE statement.

In both the cases we should have first created a SAVEPOINT. Else we will get the error while doing ROLLBACK or RELEASE of a SAVEPOINT.

194. How will you search for a String in MySQL column?

We can use REGEXP operator to search for a String in MySQL column. It is regular expression search on columns with text type value.

We can define different types of regular expressions and search them in a text with the REGEXP expression that can match our crietria.

195. How can we find the version of the MySQL server and the name of the current database by SELECT query?

We can use built in functions VERSION() and DATABASE() in MySQL to get the version of MySQL server and the name of database in MySQL.

Query is as follows:

```
SELECT VERSION(),
DATABASE();
```

196. What is the use of IFNULL() operator in MySQL?

We use IFNULL operator in MySQL to get a non-null value for a column with null value.

IFNULL(expr1, expr2)
If expr1 is not null then expr1 is returned. If expr1 is null then expr2 is returned.

Eg. SELECT name, IFNULL(id,'Unknown') AS 'id' FROM user;

If id is not null then id is returned. If id is null then Unknown is returned.

197. How will you check if a table exists in MySQL?

We can use CHECK TABLE query to see the existence of a table in MySQL.

Query is as follows:

```
CHECK TABLE <table_name>;
```

198. How will you see the structure of a table in MySQL?

We can use DESC query to see the structure of a table in MySQL. It will return the name of columns and their datatype in a table.

Query is as follows:

```
DESC <table_name>;
```

199. What are the objects that can be created by CREATE statement in MySQL?

We can create following objects by CREATE statement in MySQL:

DATABASE
USER
TABLE
INDEX

VIEW
TRIGGER
EVENT
FUNCTION
PROCEDURE

200. 24. How will you see the current user logged into MySQL connection?

We can use USER() command to get the user logged into MySQL connection.

Command is as follows:

```
SELECT USER();
```

201. How can you copy the structure of a table into another table without copying the data?

It is a trick question. But it has practical use in day to day work.

Query for this is as follows:
CREATE TABLE table_name AS SELECT * FROM USER WHERE 1 > 2;

In this example condition in WHERE clause will be always false. Due to this no data is retrieved by SELECT query.

202. What is the difference between Batch and Interactive modes of MySQL?

In Interactive mode, we use command line interface and enter queries one by one. MySQL will execute the query and return the result in command line interface.

In Batch mode of MySQL we can write all the queries in a SQL file. Then we can run this SQL file from

MySQL command line or from Scheduler Job. MySQL will execute all the queries and return the result.

203. How can we get a random number between 1 and 100 in MySQL?

In MySQL we have a RAND() function that returns a random number between 0 and 1.

```
SELECT RAND();
```

If we want to get a random number between 1 and 100, we can use following query:

```
SELECT RAND() * 100;
```

REFERENCES

- http://spark.apache.org/
- http://hadoop.apache.org/
- http://hive.apache.org/
- https://www.mysql.com/