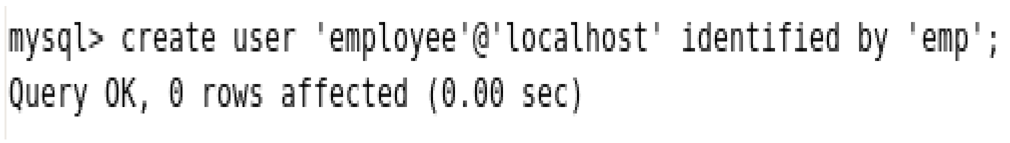
ASSIGNMENT 15.4

1. **Give a hands on demo on the below sqoop techniques**
   -- Protect your mysql password by saving in a file
   -- Protect your mysql password by reading it from a standard input

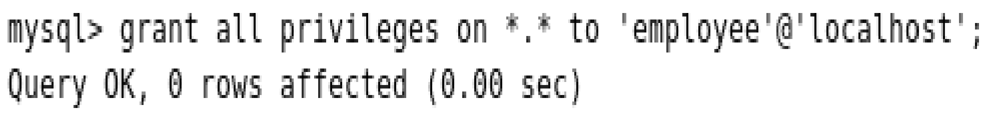
   Note: You can import any of the table present in your mysql for performing this sqoop operation

CREATING A MYSQL USER



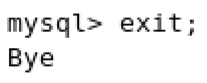
```
mysql> create user 'employee'@'localhost' identified by 'emp';
Query OK, 0 rows affected (0.00 sec)
```

GRANTING ALL PRIVILEGES



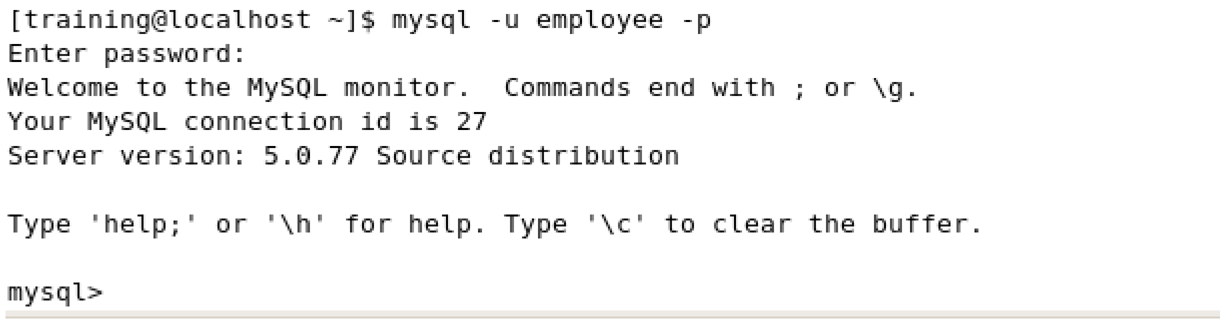
```
mysql> grant all privileges on *.* to 'employee'@'localhost';
Query OK, 0 rows affected (0.00 sec)
```

EXIT MYSQL



```
mysql> exit;
Bye
```

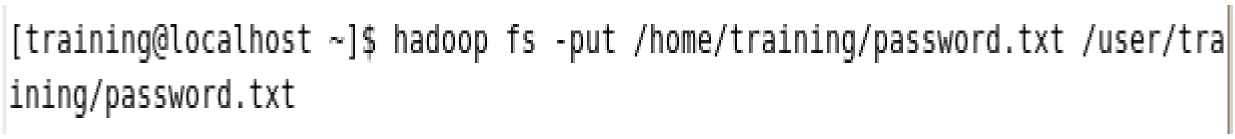
STARTING MYSQL WITH THIS NEW USER



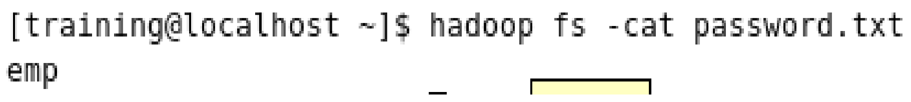
```
[training@localhost ~]$ mysql -u employee -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 27
Server version: 5.0.77 Source distribution

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

Protect your mysql password by saving in a file

CREATING A PASSWORD FILE

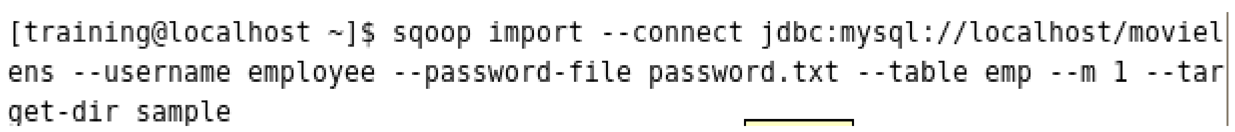


```
[training@localhost ~]$ hadoop fs -put /home/training/password.txt /user/tra
ining/password.txt
```



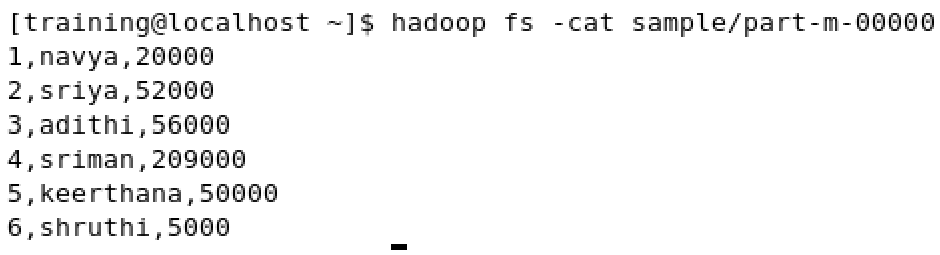
```
[training@localhost ~]$ hadoop fs -cat password.txt
emp
```

Protect your mysql password by reading it from a standard input



```
[training@localhost ~]$ sqoop import --connect jdbc:mysql://localhost/moviel
ens --username employee --password-file password.txt --table emp --m 1 --tar
get-dir sample
```

OUTPUT



```
[training@localhost ~]$ hadoop fs -cat sample/part-m-00000
1,navya,20000
2,sriya,52000
3,adithi,56000
4,sriman,209000
5,keerthana,50000
6,shruthi,5000
```

==

2_) EXPLAIN ABOUT SPEED TRANSFER IN SQOOP

Speed Transfer is also termed as performance tuning.
Sqoop is a tool designed to transfer data between Hadoop and relational databases or mainframes.
Sqoop automates most of this process, relying on the database to describe the schema for the data to be imported.
Sqoop uses MapReduce to import and export the data, which provides parallel operation as well as fault tolerance.

The best tuning practices involves

- Batch
- Split-by and boundary-query
- Direct
- Fetch-size
- Num-mapper

These arguments are used to optimise the functioning

INSERTING DATA IN BATCHES

Specifies that you can group the related SQL statements into a batch when you export data.

The JDBC interface exposes an API for doing batches in a prepared statement with multiple sets of values. With the --batch parameter, Sqoop can take advantage of this. This API is present in all JDBC drivers because it is required by the JDBC interface.

Enable JDBC batching using the --batch parameter.

```
sqoop export  --connect <<JDBC URL>>  --username
<<SQOOP_USER_NAME>>  --password <<SQOOP_PASSWORD>>
--table <<TABLE_NAME>>  --export-dir <<FOLDER_URI>>  --batch
```

CUSTOM BOUNDARY QUERIES

Specifies the range of values that you can import. You can use boundary-query if you do not get the desired results by using the split-by argument alone.

When you configure the boundary-query argument, you must specify the min(id) and max(id) along with the table name. If you do not configure the argument, Sqoop runs the following query.

```
sqoop import   --connect
<<JDBC URL>>   --username
<<SQOOP_USER_NAME>>   --password
<<SQOOP_PASSWORD>>   --query <<QUERY>>
  --split-by <<ID>>
  --target-dir <<TARGET_DIR_URI>>
 --boundary-query "select min(<<ID>>), max(<<ID>>)
from <<TABLE>>"
```

IMPORTING DATA DIRECTLY INTO HIVE

Specifies the direct import fast path when you import data from RDBMS.

Rather than using the JDBC interface for transferring data, the direct mode delegates the job of transferring data to the native utilities provided by the database vendor. In the case of MySQL, the mysqldump and mysqlimport will be used for retrieving data from the database server or moving data back. In the case of PostgreSQL, Sqoop will take advantage of the pg_dump utility to import data. Using native utilities will greatly improve performance, as they are optimized to provide the best possible transfer speed while putting less burden on the database server. There are several limitations that come with this faster import. For one, not all databases have available native utilities. This mode is not available for every supported database. Out of the box, Sqoop has direct support only for MySQL and PostgreSQL.

```
sqoop import   --connect
<<JDBC URL>>   --username
<<SQOOP_USER_NAME>>   --password
<<SQOOP_PASSWORD>>  --table
<<TABLE_NAME>>   --direct
```

IMPORTING DATA USING FETCH-SIZE

Specifies the number of entries that Sqoop can import at a time.

Use the following syntax:

--fetch-size=<n>
Where <n> represents the number of entries that Sqoop must fetch at a time. Default is 1000.

Increase the value of the fetch-size argument based on the volume of data that need to read. Set the value based on the available memory and bandwidth.

SPLIT-BY

Specifies the column name based on which Sqoop must split the work units.

syntax:

```
--split-by <column name>
```

```
sqoop import   --connect
<<JDBC URL>>   --username
<<SQOOP_USER_NAME>>   --password
<<SQOOP_PASSWORD>>   --query <<QUERY>>
  --split-by <<ID>>
```

USING NUM-MAPPERS

Specifies number of map tasks that can run in parallel. Default is 4. To optimize performance, set the number of map tasks to a value lower than the maximum number of connections that the database supports.

Use the parameter --num-mappers if you want Sqoop to use a different number of mappers. For example, to suggest 10 concurrent tasks, use the following Sqoop command:

```
sqoop import   --connect
jdbc:mysql://mysql.example.com/sqoop   --username
sqoop   --password
sqoop   --table
cities   --num-mappers 10
```

Controlling the amount of parallelism that Sqoop will use to transfer data is the main way to control the load on your database. Using more mappers will lead to a higher number of concurrent data transfer tasks, which can result in faster job completion. However, it will also increase the load on the database as Sqoop will execute more concurrent queries.

===