

Loading the datasets into an AWS RDS instance (Task 1)

Creation of RDS Instance.....	2
Downloading the datasets to master node of EMR cluster	2
Loading the datasets.....	4
Step 1. Initializing DNS names for RDS instance and EMR cluster	4
Step 2. Login to RDS instance from EMR cluster	5
Step 3. Creation of database.....	6
Step 4. Creation of table schema	6
Step 5. Global variable for importing data from file	9
Step 6. Loading the dataset	10
Step 7. Verifying the imported data.....	14
Annexure – I: Creating the EMR cluster.....	16
Annexure – II: Creating the RDS instance on AWS.....	22
Annexure – III: Connecting the RDS instance to EMR cluster.....	27
Edit the MySQL Global Variable through Parameter Group.....	29
Annexure IV: Login to EMR cluster’s master node through putty	31

Loading the datasets into an AWS RDS instance (Task 1)

Creation of RDS Instance

This document is purposely created for steps related to loading of datasets, however steps involved (including screenshots) for creating the EMR instance, creating the RDS instance and connecting RDS instance to EMR cluster are added in Annexure-I (at end of this document) for reference.

Downloading the datasets to master node of EMR cluster

Download the datasets using ‘wget’ command on EMR cluster’s shell. This will download the files to local filesystem. For ease all the files are downloaded to a new folder “tripdata” in home directory.

```
sudo -i
cd
pwd
mkdir tripdata
cd tripdata
wget https://nyc-tlc-upgrad.s3.amazonaws.com/yellow_tripdata_2017-01.csv
wget https://nyc-tlc-upgrad.s3.amazonaws.com/yellow_tripdata_2017-02.csv
wget https://nyc-tlc-upgrad.s3.amazonaws.com/yellow_tripdata_2017-03.csv
wget https://nyc-tlc-upgrad.s3.amazonaws.com/yellow_tripdata_2017-04.csv
wget https://nyc-tlc-upgrad.s3.amazonaws.com/yellow_tripdata_2017-05.csv
wget https://nyc-tlc-upgrad.s3.amazonaws.com/yellow_tripdata_2017-06.csv
ls -ltr
wc -l yellow_tripdata_2017-*.csv
```

Screenshot of commands for downloading dataset

```
[hadoop@ip-172-31-24-56 ~]$ sudo -i
EEEEEEEEEEEEEEEEEE MMMMMMM M:::::M M:::::::M R:::::R RRRRRRRRRRRRRR
E:::::::E E:::::E E M:::::M M:::::M M:::::M R:::::R R:::::R
EE:::::E EEEEEEE E M:::::M M:::::M M:::::M R:::::R RRRRRR:::::R
 E:::::E EEEEEEE M:::::M M:::::M M:::::M RR:::::R R:::::R
 E:::::E M:::::M M:::::M M:::::M M:::::M R:::::R R:::::R
 E:::::E EEEEEEEEEE M:::::M M:::::M M:::::M M:::::M R:::::R RRRRRR:::::R
 E:::::E M:::::M M:::::M M:::::M M:::::M R:::::R RRRRRR:::::R
 E:::::E EEEEEEE M:::::M M:::::M M:::::M R:::::R RRRRRR:::::R
 E:::::E M:::::M M:::::M M:::::M R:::::R R:::::R
 E:::::E EEEEEEE M:::::M MMM M:::::M R:::::R R:::::R
EE:::::E EEEEEEE E M:::::M M:::::M R:::::R R:::::R
 E:::::E EEEEEEE E M:::::M M:::::M RR:::::R R:::::R
 EEEEEEEEEEEEEEEEEE MMMMMMM RRRRRRR RRRRRR

[root@ip-172-31-24-56 ~]# cd
[root@ip-172-31-24-56 ~]# pwd
/root
[root@ip-172-31-24-56 ~]# mkdir tripdata
[root@ip-172-31-24-56 ~]# cd tripdata
```

```
[root@ip-172-31-24-56 tripdata]# wget https://nyc-tlc-upgrad.s3.amazonaws.com/yellow_tripdata_2017-01.csv
--2023-11-25 02:42:51-- https://nyc-tlc-upgrad.s3.amazonaws.com/yellow_tripdata_2017-01.csv
Resolving nyc-tlc-upgrad.s3.amazonaws.com (nyc-tlc-upgrad.s3.amazonaws.com)... 52.217.200.249, 52.217.226.145, 54.231.193.249, ...
Connecting to nyc-tlc-upgrad.s3.amazonaws.com (nyc-tlc-upgrad.s3.amazonaws.com)|52.217.200.249|:443...
connected.
HTTP request sent, awaiting response... 200 OK
Length: 914029540 (872M) [text/csv]
Saving to: 'yellow_tripdata_2017-01.csv'

100%[=====] 914,029,540 27.6MB/s in 37s

2023-11-25 02:43:28 (23.6 MB/s) - 'yellow_tripdata_2017-01.csv' saved [914029540/914029540]

[root@ip-172-31-24-56 tripdata]# wget https://nyc-tlc-upgrad.s3.amazonaws.com/yellow_tripdata_2017-02.csv
--2023-11-25 02:43:28-- https://nyc-tlc-upgrad.s3.amazonaws.com/yellow_tripdata_2017-02.csv
Resolving nyc-tlc-upgrad.s3.amazonaws.com (nyc-tlc-upgrad.s3.amazonaws.com)... 54.231.203.81, 54.231.225.1, 54.231.228.121, ...
Connecting to nyc-tlc-upgrad.s3.amazonaws.com (nyc-tlc-upgrad.s3.amazonaws.com)|54.231.203.81|:443...
connected.
HTTP request sent, awaiting response... 200 OK
Length: 863487050 (823M) [text/csv]
Saving to: 'yellow_tripdata_2017-02.csv'

100%[=====] 863,487,050 24.4MB/s in 33s

2023-11-25 02:44:01 (24.9 MB/s) - 'yellow_tripdata_2017-02.csv' saved [863487050/863487050]

[root@ip-172-31-24-56 tripdata]# wget https://nyc-tlc-upgrad.s3.amazonaws.com/yellow_tripdata_2017-03.csv
--2023-11-25 02:44:01-- https://nyc-tlc-upgrad.s3.amazonaws.com/yellow_tripdata_2017-03.csv
Resolving nyc-tlc-upgrad.s3.amazonaws.com (nyc-tlc-upgrad.s3.amazonaws.com)... 52.216.79.4, 52.217.86.20, 52.217.191.25, ...
Connecting to nyc-tlc-upgrad.s3.amazonaws.com (nyc-tlc-upgrad.s3.amazonaws.com)|52.216.79.4|:443...
connected.
HTTP request sent, awaiting response... 200 OK
Length: 969809025 (925M) [text/csv]
Saving to: 'yellow_tripdata_2017-03.csv'

100%[=====] 969,809,025 24.4MB/s in 37s

2023-11-25 02:44:38 (25.1 MB/s) - 'yellow_tripdata_2017-03.csv' saved [969809025/969809025]

[root@ip-172-31-24-56 tripdata]# wget https://nyc-tlc-upgrad.s3.amazonaws.com/yellow_tripdata_2017-04.csv
--2023-11-25 02:44:38-- https://nyc-tlc-upgrad.s3.amazonaws.com/yellow_tripdata_2017-04.csv
Resolving nyc-tlc-upgrad.s3.amazonaws.com (nyc-tlc-upgrad.s3.amazonaws.com)... 52.216.171.83, 52.216.
```

```

Connecting to nyc-tlc-upgrad.s3.amazonaws.com (nyc-tlc-upgrad.s3.amazonaws.com) |52.216.171.83|:443...
connected.
HTTP request sent, awaiting response... 200 OK
Length: 946349441 (903M) [text/csv]
Saving to: 'yellow_tripdata_2017-04.csv'

100%[=====] 946,349,441 23.9MB/s  in 37s

2023-11-25 02:45:16 (24.2 MB/s) - 'yellow_tripdata_2017-04.csv' saved [946349441/946349441]

[root@ip-172-31-24-56 tripdata]# wget https://nyc-tlc-upgrad.s3.amazonaws.com/yellow_tripdata_2017-05
.csv
--2023-11-25 02:45:16--  https://nyc-tlc-upgrad.s3.amazonaws.com/yellow_tripdata_2017-05.csv
Resolving nyc-tlc-upgrad.s3.amazonaws.com (nyc-tlc-upgrad.s3.amazonaws.com)... 52.216.221.33, 52.217.
171.145, 52.217.230.201, ...
Connecting to nyc-tlc-upgrad.s3.amazonaws.com (nyc-tlc-upgrad.s3.amazonaws.com) |52.216.221.33|:443...
connected.
HTTP request sent, awaiting response... 200 OK
Length: 951965526 (908M) [text/csv]
Saving to: 'yellow_tripdata_2017-05.csv'

100%[=====] 951,965,526 27.9MB/s  in 37s

2023-11-25 02:45:53 (24.8 MB/s) - 'yellow_tripdata_2017-05.csv' saved [951965526/951965526]

[root@ip-172-31-24-56 tripdata]# wget https://nyc-tlc-upgrad.s3.amazonaws.com/yellow_tripdata_2017-06
.csv
--2023-11-25 02:45:53--  https://nyc-tlc-upgrad.s3.amazonaws.com/yellow_tripdata_2017-06.csv
Resolving nyc-tlc-upgrad.s3.amazonaws.com (nyc-tlc-upgrad.s3.amazonaws.com)... 52.216.38.81, 52.216.5
1.25, 52.217.139.249, ...
Connecting to nyc-tlc-upgrad.s3.amazonaws.com (nyc-tlc-upgrad.s3.amazonaws.com) |52.216.38.81|:443...
connected.
HTTP request sent, awaiting response... 200 OK
Length: 910028408 (868M) [text/csv]
Saving to: 'yellow_tripdata_2017-06.csv'

100%[=====] 910,028,408 28.4MB/s  in 36s

2023-11-25 02:46:29 (24.0 MB/s) - 'yellow_tripdata_2017-06.csv' saved [910028408/910028408]

[root@ip-172-31-24-56 tripdata]# ls -ltr
total 5425468
-rw-r--r-- 1 root root 914029540 Nov 25 2022 yellow_tripdata_2017-01.csv
-rw-r--r-- 1 root root 863487050 Nov 25 2022 yellow_tripdata_2017-02.csv
-rw-r--r-- 1 root root 969809025 Nov 25 2022 yellow_tripdata_2017-03.csv
-rw-r--r-- 1 root root 946349441 Nov 25 2022 yellow_tripdata_2017-04.csv
-rw-r--r-- 1 root root 951965526 Nov 25 2022 yellow_tripdata_2017-05.csv
-rw-r--r-- 1 root root 910028408 Nov 25 2022 yellow_tripdata_2017-06.csv
[root@ip-172-31-24-56 tripdata]#

```

Loading the datasets

Step 1. Initializing DNS names for RDS instance and EMR cluster

For the purpose of assignment, EMR cluster and RDS instance is created temporarily and doesn't have permanent DMS. Therefore, for ease of development and execution of required commands, two shell local variables are created before executing any command,

for storing DNS of RDS instance and EMR cluster, by using below commands on shell of EMR cluster:

```
DNS_EMR="ec2-100-26-47-42.compute-1.amazonaws.com"
DNS_RDS="tripdb.cl3grgmbpn1n.us-east-1.rds.amazonaws.com"
echo $DNS_EMR
echo $DNS_RDS
```

This will also help to test the assignment code/commands, as there will not be any requirement of changing DNS in each of command separately.

Screenshot of local variable initialization

```
[root@ip-172-31-24-56 tripdata]# DNS_EMR="ec2-100-26-47-42.compute-1.amazonaws.com"
[root@ip-172-31-24-56 tripdata]# DNS_RDS="tripdb.cl3grgmbpn1n.us-east-1.rds.amazonaws.com"
[root@ip-172-31-24-56 tripdata]# echo $DNS_EMR
ec2-100-26-47-42.compute-1.amazonaws.com
[root@ip-172-31-24-56 tripdata]# echo $DNS_RDS
tripdb.cl3grgmbpn1n.us-east-1.rds.amazonaws.com
[root@ip-172-31-24-56 tripdata]#
```

Step 2. Login to RDS instance from EMR cluster

Login to RDS instance of MySQL using the credentials used to create the database service:

Username : admin
Password : user1234

```
mysql -h $DNS_RDS -P 3306 -u admin -p
```

Enter the password which is mentioned above, when prompt for it.

Screenshot for connecting to database

```
[root@ip-172-31-24-56 tripdata]# mysql -h $DNS_RDS -P 3306 -u admin -p
Enter password:
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MySQL connection id is 23
Server version: 8.0.33 Source distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]>
```

Step 3. Creation of database

Following actions are taken:

- Create new database in MySQL instance with name “tripdata”.
- Validate database has been created by using ‘show databases’ command. It should include “tripdata” as database along with few default databases came with MySQL for administration and demonstration purposes.
- Switch to new database created, to ensure that all DDLs and DMLs run on “tripdata” database.

```
create database tripdata;
show databases;
use tripdata;
```

Screenshot of commands for creating database

```
MySQL [(none)]> create database tripdata;
Query OK, 1 row affected (0.01 sec)

MySQL [(none)]> show databases;
+-----+
| Database      |
+-----+
| information_schema |
| mysql          |
| performance_schema |
| sys            |
| tripdata       |
+-----+
5 rows in set (0.01 sec)

MySQL [(none)]> use tripdata;
Database changed
MySQL [tripdata]>
```

Step 4. Creation of table schema

Few important observations/actions about datasets: -

- Data is not clean and many fields contain outliers. For example, fields like “fare_amount”, “total_amount” contains negative (< 0) values, which cannot be a valid amount. Again, for categorical variables like “payment_type” value contains other than 1 to 6, i.e., 99, which may be representing the missing value. However, As EDA analysis is not part of assignment, currently no data cleaning is performed, and data is imported in RDS instance without cleaning.
- For all decimal fields range is taken as (16, 4), which means it can store upto 4 decimal places and 12-digit decimal number. This is because the range of data field cannot be determined due to presence of outlier.
- Only columns “congestion_surcharge” and “airport_fee” contains null value.

- A field “RECORD_ID” has been added as auto incremental integer primary key, to uniquely identify each record.

A table with following columns and datatype is created:

#	Column Name	Data Type	Is Null	Column Description
1	RECORD_ID	VARCHAR	N	A Primary Key is created which is combination of "SOURCE" + "YEAR" + "MONTH" + "ROW_NUMBER".
2	VendorID	INT	N	A code indicating the TPEP provider that provided the record. 1= Creative Mobile Technologies, LLC; 2=VeriFone Inc.
3	tpep_pickup_datetime	DATETIME	N	The date and time when the meter was engaged.
4	tpep_dropoff_datetime	DATETIME	N	The date and time when the meter was disengaged.
5	passenger_count	INT	N	The number of passengers in the vehicle. This is a driver-entered value.
6	trip_distance	DECIMAL	N	The elapsed trip distance in miles reported by the taximeter.
7	RatecodeID	INT	N	The final rate code in effect at the end of the trip. 1= Standard rate 2=JFK 3=Newark 4=Nassau or Westchester 5=Negotiated fare 6=Group ride Also, it is found that it contains value 99.
8	store_and_fwd_flag	VARCHAR	N	This flag indicates whether the trip record was held in vehicle memory before sending to the vendor, aka “store and forward,” because the vehicle did not have a connection to the server. Y= store and forward trip N= not a store and forward trip
9	PULocationID	INT	N	TLC Taxi Zone in which the taximeter was engaged.
10	DOLocationID	INT	N	TLC Taxi Zone in which the taximeter was disengaged.
11	payment_type	INT	N	A numeric code signifying how the passenger paid for the trip. 1= Credit card 2= Cash 3= No charge 4= Dispute 5= Unknown 6= Voided trip

12	fare_amount	DECIMAL	N	The time-and-distance fare calculated by the meter. Few outliers like negative values are observed.
13	extra	DECIMAL	N	Miscellaneous extras and surcharges. Currently, this only includes the \$0.50 and \$1 rush hour and overnight charges.
14	mta_tax	DECIMAL	N	\$0.50 MTA tax that is automatically triggered based on the metered rate in use.
15	tip_amount	DECIMAL	N	Tip amount – This field is automatically populated for credit card tips. Cash tips are not included.
16	tolls_amount	DECIMAL	N	Total amount of all tolls paid in trip.
17	improvement_surcharge	DECIMAL	N	\$0.30 improvement surcharge assessed trips at the flag drop. The improvement surcharge began being levied in 2015.
18	total_amount	DECIMAL	N	The total amount charged to passengers. Does not include cash tips.
19	congestion_surcharge	DECIMAL	Y	Total amount collected in trip for NYS congestion surcharge.
20	airport_fee	DECIMAL	Y	\$1.25 for pick up only at LaGuardia and John F. Kennedy Airports

Below DDL is used to create the desired table to store data: -

```

CREATE TABLE yellow_tripdata
(
    RECORD_ID                VARCHAR(50)          NOT NULL PRIMARY KEY,
    VendorID                 INT                  NOT NULL,
    tpep_pickup_datetime     DATETIME            NOT NULL,
    tpep_dropoff_datetime    DATETIME            NOT NULL,
    passenger_count          INT                  NOT NULL,
    trip_distance             DECIMAL(16, 4)      NOT NULL,
    RatecodeID               INT                  NOT NULL,
    store_and_fwd_flag        VARCHAR(1)          NOT NULL,
    PULocationID             INT                  NOT NULL,
    DOLocationID             INT                  NOT NULL,
    payment_type              INT                  NOT NULL,
    fare_amount               DECIMAL(16, 4)      NOT NULL,
    extra                     DECIMAL(16, 4)      NOT NULL,
    mta_tax                   DECIMAL(16, 4)      NOT NULL,
    tip_amount                DECIMAL(16, 4)      NOT NULL,
    tolls_amount               DECIMAL(16, 4)      NOT NULL,
    improvement_surcharge    DECIMAL(16, 4)      NOT NULL,
    total_amount               DECIMAL(16, 4)      NOT NULL,
    congestion_surcharge     DECIMAL(16, 4)      NULL,
    airport_fee               DECIMAL(16, 4)      NULL
);

```

After above SQL, table should be created on database. To validate checked through below SQL command:

```
show tables;
```

Screenshot of commands for creating table

```
MySQL [tripdata]> CREATE TABLE yellow_tripdata
-> (
->     RECORD_ID            VARCHAR(50)      NOT NULL PRIMARY KEY,
->     VendorID              INT             NOT NULL,
->     tpep_pickup_datetime  DATETIME        NOT NULL,
->     tolls_amount           DECIMAL(16, 4)    NOT NULL,
->     improvement_surcharge DECIMAL(16, 4)    NOT NULL,
->     total_amount           DECIMAL(16, 4)    NOT NULL,
->     congestion_surcharge  DECIMAL(16, 4)    NULL,
->     airport_fee            DECIMAL(16, 4)    NULL
->     tpep_dropoff_datetime  DATETIME        NOT NULL,
->     passenger_count        INT             NOT NULL,
->     trip_distance          DECIMAL(16, 4)    NOT NULL,
->     RatecodeID             INT             NOT NULL,
->     store_and_fwd_flag     VARCHAR(1)      NOT NULL,
->     PULocationID           INT             NOT NULL,
->     DOLocationID           INT             NOT NULL,
->     payment_type           INT             NOT NULL,
->     fare_amount             DECIMAL(16, 4)    NOT NULL,
->     extra                  DECIMAL(16, 4)    NOT NULL,
->     mta_tax                 DECIMAL(16, 4)    NOT NULL,
->     tip_amount              DECIMAL(16, 4)    NOT NULL,
->     tolls_amount            DECIMAL(16, 4)    NOT NULL,
->     improvement_surcharge  DECIMAL(16, 4)    NOT NULL,
->     total_amount            DECIMAL(16, 4)    NOT NULL,
->     congestion_surcharge   DECIMAL(16, 4)    NULL,
->     airport_fee             DECIMAL(16, 4)    NULL
-> );
Query OK, 0 rows affected (0.05 sec)

MySQL [tripdata]> show tables;
+-----+
| Tables_in_tripdata |
+-----+
| yellow_tripdata   |
+-----+
1 row in set (0.01 sec)

MySQL [tripdata]> █
```

Step 5. Global variable for importing data from file

By default, in MySQL importing data from local client is disabled. Before loading dataset, we need to ensure that flag “LOCAL_INFILE” in MySQL is set to TRUE. To check the value, following command is executed on MySQL:

```
SHOW GLOBAL VARIABLES LIKE 'LOCAL_INFILE';
```

If above command doesn't show the flag value as "TRUE" or "ON", then following command is run to enable the flag:

```
SET GLOBAL LOCAL_INFILE=TRUE;
```

Screenshot of global flag set

```
MySQL [tripdata]> SHOW GLOBAL VARIABLES LIKE 'LOCAL_INFILE';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| local_infile | ON   |
+-----+-----+
1 row in set (0.00 sec)

MySQL [tripdata]> █
```

Step 6. Loading the dataset

MySQL support loading a table using local CSV or other files. This will directly load the table. Any datatype conversion can also be performed inline.

Following command for setting MySQL variable for row number to 0 to help creating PK:

```
set @myrow_numb=0;
```

Following commands are used to load file in database:

```

LOAD DATA LOCAL INFILE '/root/tripdata/yellow_tripdata_2017-01.csv'
INTO TABLE yellow_tripdata
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
IGNORE 1 LINES
(
    @VendorID, @tpep_pickup_datetime, @tpep_dropoff_datetime,
    @passenger_count, @trip_distance, @RatecodeID, @store_and_fwd_flag,
    @PULocationID, @DOLocationID, @payment_type, @fare_amount, @extra,
    @mta_tax, @tip_amount, @tolls_amount, @improvement_surcharge,
    @total_amount, @congestion_surcharge, @airport_fee
)
SET
RECORD_ID          = CONCAT('MYSQL-', SUBSTRING(@tpep_pickup_datetime, 1, 8),
LPAD(@myrow_numb:=@myrow_numb+1, 10, '0')),
VendorID          = @VendorID,
tpep_pickup_datetime = STR_TO_DATE(@tpep_pickup_datetime, '%Y-%m-%d %H:%i:%s'),
tpep_dropoff_datetime= STR_TO_DATE(@tpep_dropoff_datetime, '%Y-%m-%d %H:%i:%s'),
passenger_count    = @passenger_count,
trip_distance      = CAST(@trip_distance AS CHAR)+0,
RatecodeID         = @RatecodeID,
store_and_fwd_flag = @store_and_fwd_flag,
PULocationID       = @PULocationID,
DOLocationID       = @DOLocationID,
payment_type        = @payment_type,
fare_amount         = CAST(@fare_amount AS CHAR)+0,
extra              = CAST(@extra AS CHAR)+0,
mta_tax             = CAST(@mta_tax AS CHAR)+0,
tip_amount          = CAST(@tip_amount AS CHAR)+0,
tolls_amount        = CAST(@tolls_amount AS CHAR)+0,
improvement_surcharge= CAST(@improvement_surcharge AS CHAR)+0,
total_amount        = CAST(@total_amount AS CHAR)+0,
congestion_surcharge = CASE WHEN @congestion_surcharge = '' THEN NULL ELSE
CAST(@congestion_surcharge AS CHAR)+0 END,
airport_fee         = CASE WHEN TRIM(REPLACE(REPLACE(@airport_fee, '\r', ''),
'\n', '')) = '' THEN NULL ELSE CAST(TRIM(REPLACE(REPLACE(@airport_fee, '\r',
''), '\n', '')) AS CHAR)+0 END;

```

```

LOAD DATA LOCAL INFILE '/root/tripdata/yellow_tripdata_2017-02.csv'
INTO TABLE yellow_tripdata
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
IGNORE 1 LINES
(
    @VendorID, @tpep_pickup_datetime, @tpep_dropoff_datetime,
    @passenger_count, @trip_distance, @RatecodeID, @store_and_fwd_flag,
    @PULocationID, @DOLocationID, @payment_type, @fare_amount, @extra,
    @mta_tax, @tip_amount, @tolls_amount, @improvement_surcharge,
    @total_amount, @congestion_surcharge, @airport_fee
)
SET
RECORD_ID          = CONCAT('MYSQL-', SUBSTRING(@tpep_pickup_datetime, 1, 8),
LPAD(@myrow_numb:=@myrow_numb+1, 10, '0')),
VendorID          = @VendorID,
tpep_pickup_datetime = STR_TO_DATE(@tpep_pickup_datetime, '%Y-%m-%d %H:%i:%s'),
tpep_dropoff_datetime= STR_TO_DATE(@tpep_dropoff_datetime, '%Y-%m-%d %H:%i:%s'),
passenger_count    = @passenger_count,
trip_distance      = CAST(@trip_distance AS CHAR)+0,
RatecodeID         = @RatecodeID,
store_and_fwd_flag = @store_and_fwd_flag,
PULocationID       = @PULocationID,
DOLocationID       = @DOLocationID,
payment_type        = @payment_type,
fare_amount         = CAST(@fare_amount AS CHAR)+0,
extra              = CAST(@extra AS CHAR)+0,
mta_tax             = CAST(@mta_tax AS CHAR)+0,
tip_amount          = CAST(@tip_amount AS CHAR)+0,
tolls_amount        = CAST(@tolls_amount AS CHAR)+0,
improvement_surcharge= CAST(@improvement_surcharge AS CHAR)+0,
total_amount        = CAST(@total_amount AS CHAR)+0,
congestion_surcharge = CASE WHEN @congestion_surcharge = '' THEN NULL ELSE
CAST(@congestion_surcharge AS CHAR)+0 END,
airport_fee         = CASE WHEN TRIM(REPLACE(REPLACE(@airport_fee, '\r', ''),
'\n', '')) = '' THEN NULL ELSE CAST(TRIM(REPLACE(REPLACE(@airport_fee, '\r',
''), '\n', '')) AS CHAR)+0 END;

```

Screenshot of loading the dataset in RDS

```
MySQL [tripdata]> LOAD DATA LOCAL INFILE '/root/tripdata/yellow_tripdata_2017-01.csv'  
-> INTO TABLE yellow_tripdata  
-> FIELDS TERMINATED BY ','  
-> LINES TERMINATED BY '\n'  
-> IGNORE 1 LINES  
-> (@VendorID, @tpep_pickup_datetime, @tpep_dropoff_datetime,  
-> @passenger_count, @trip_distance, @RatecodeID, @store_and_fwd_flag,  
-> @PULocationID, @DOLocationID, @payment_type, @fare_amount, @extra,  
-> @mta_tax, @tip_amount, @tolls_amount, @improvement_surcharge,  
-> @total_amount, @congestion_surcharge, @airport_fee  
-> )  
-> SET  
-> RECORD_ID = CONCAT('MYSQL-', SUBSTRING(@tpep_pickup_datetime,  
1, 8), LPAD(@myrow numb:=@myrow numb+1, 10, '0')),  
-> VendorID = @VendorID,  
-> tpep_pickup_datetime = STR_TO_DATE(@tpep_pickup_datetime, '%Y-%m-%d %H:%i:  
%s'),  
-> tpep_dropoff_datetime= STR_TO_DATE(@tpep_dropoff_datetime, '%Y-%m-%d %H:%i:  
%s'),  
-> passenger_count = @passenger_count,  
-> trip_distance = CAST(@trip_distance AS CHAR)+0,  
-> RatecodeID = @RatecodeID,  
-> store_and_fwd_flag = @store_and_fwd_flag,  
-> PULocationID = @PULocationID,  
-> DOLocationID = @DOLocationID,  
-> payment_type = @payment_type,  
-> fare_amount = CAST(@fare_amount AS CHAR)+0,  
-> extra = CAST(@extra AS CHAR)+0,  
-> mta_tax = CAST(@mta_tax AS CHAR)+0,  
-> tip_amount = CAST(@tip_amount AS CHAR)+0,  
-> tolls_amount = CAST(@tolls_amount AS CHAR)+0,  
-> improvement_surcharge= CAST(@improvement_surcharge AS CHAR)+0,  
-> total_amount = CAST(@total_amount AS CHAR)+0,  
-> congestion_surcharge = CASE WHEN @congestion_surcharge = '' THEN NULL ELSE  
CAST(@congestion_surcharge AS CHAR)+0 END,  
-> airport_fee = CASE WHEN TRIM(REPLACE(REPLACE(@airport_fee, '\r',  
''), '\n', '')) = '' THEN NULL ELSE CAST(TRIM(REPLACE(REPLACE(@airport_fee, '\r',  
''), '\n', '')) AS CHAR)+0 END;  
Query OK, 9710820 rows affected, 1 warning (6 min 6.68 sec)  
Records: 9710820 Deleted: 0 Skipped: 0 Warnings: 1  
  
MySQL [tripdata]>
```

```

MySQL [tripdata]> LOAD DATA LOCAL INFILE '/root/tripdata/yellow_tripdata_2017-02
.csv'
-> INTO TABLE yellow_tripdata
-> FIELDS TERMINATED BY ','
-> LINES TERMINATED BY '\n'
-> IGNORE 1 LINES
-> (    @VendorID, @tpep_pickup_datetime, @tpep_dropoff_datetime,
->      @passenger_count, @trip_distance, @RatecodeID, @store_and_fwd_flag,
->      @PULocationID, @DOLocationID, @payment_type, @fare_amount, @extra,
->      @mta_tax, @tip_amount, @tolls_amount, @improvement_surcharge,
->      @total_amount, @congestion_surcharge, @airport_fee
-> )
-> SET
-> RECORD_ID          = CONCAT('MYSQL-', SUBSTRING(@tpep_pickup_datetime,
1, 8), LPAD(@myrow_numb:=@myrow_numb+1, 10, '0')),
-> VendorID           = @VendorID,
-> tpep_pickup_datetime = STR_TO_DATE(@tpep_pickup_datetime, '%Y-%m-%d %H:%i
:%s'),
-> tpep_dropoff_datetime= STR_TO_DATE(@tpep_dropoff_datetime, '%Y-%m-%d %H:%i
:%s'),
-> passenger_count     = @passenger_count,
-> trip_distance       = CAST(@trip_distance AS CHAR)+0,
-> RatecodeID          = @RatecodeID,
-> store_and_fwd_flag  = @store_and_fwd_flag,
-> PULocationID        = @PULocationID,
-> DOLocationID        = @DOLocationID,
-> payment_type         = @payment_type,
-> fare_amount          = CAST(@fare_amount AS CHAR)+0,
-> extra                = CAST(@extra AS CHAR)+0,
-> mta_tax               = CAST(@mta_tax AS CHAR)+0,
-> tip_amount            = CAST(@tip_amount AS CHAR)+0,
-> tolls_amount          = CAST(@tolls_amount AS CHAR)+0,
-> improvement_surcharge= CAST(@improvement_surcharge AS CHAR)+0,
-> total_amount          = CAST(@total_amount AS CHAR)+0,
-> congestion_surcharge = CASE WHEN @congestion_surcharge = '' THEN NULL ELSE
CAST(@congestion_surcharge AS CHAR)+0 END,
-> airport_fee           = CASE WHEN TRIM(REPLACE(REPLACE(@airport_fee, '\r',
'\n'), '\n', '')) = '' THEN NULL ELSE CAST(TRIM(REPLACE(REPLACE(@airport_fee, '\r
', '\n'), '\n', '')) AS CHAR)+0 END;
Query OK, 9169775 rows affected, 1 warning (6 min 41.17 sec)
Records: 9169775 Deleted: 0 Skipped: 0 Warnings: 1

MySQL [tripdata]> █

```

Step 7. Verifying the imported data

The data in MySQL has been loaded, we can verify the record count of inserted data. Following commands to check the total rows imported in table:

```
SELECT COUNT(*) FROM yellow_tripdata;
```

```
quit;
```

Also, check the total records in two CSV files on local file systems:

```
wc -l /root/tripdata/yellow_tripdata_2017-01.csv  
wc -l /root/tripdata/yellow_tripdata_2017-02.csv
```

Screenshot of verification of the dataset

```
MySQL [tripdata]> SELECT COUNT(*) FROM yellow_tripdata;  
+-----+  
| COUNT(*) |  
+-----+  
| 18880595 |  
+-----+  
1 row in set (57.84 sec)
```

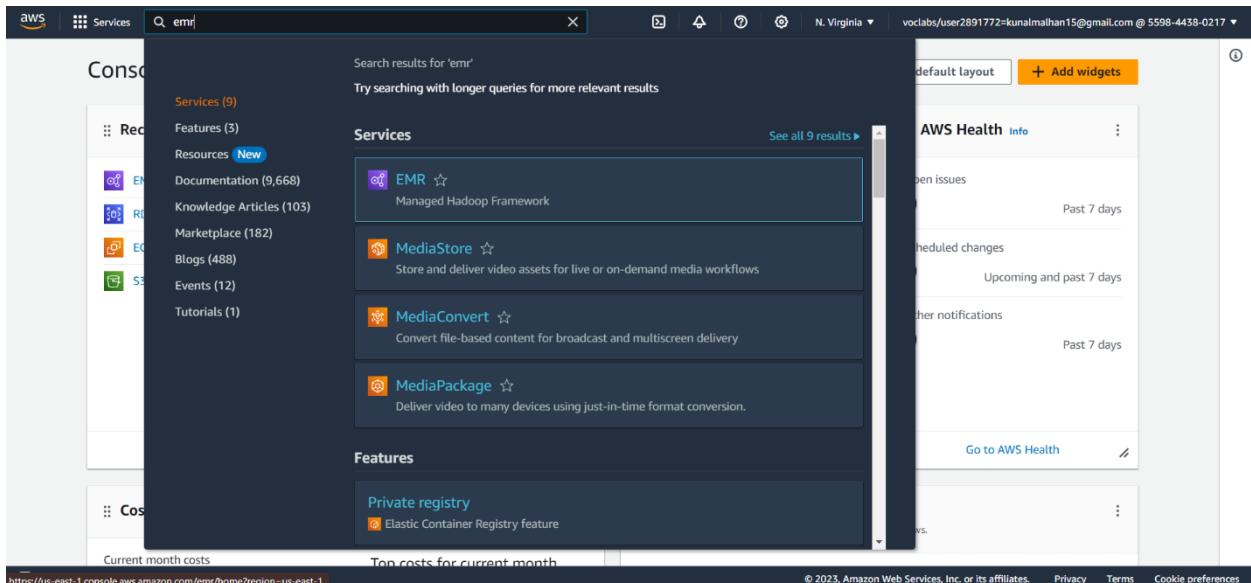
```
[root@ip-172-31-24-56 tripdata]# wc -l /root/tripdata/yellow_tripdata_2017-01.cs  
v  
9710821 /root/tripdata/yellow_tripdata_2017-01.csv  
[root@ip-172-31-24-56 tripdata]# wc -l /root/tripdata/yellow_tripdata_2017-02.cs  
v  
9169776 /root/tripdata/yellow_tripdata_2017-02.csv  
[root@ip-172-31-24-56 tripdata]#
```

Sum of lines (excluding header row) is same as the records added in database.

```
***** END OF TASK-1 *****
```

Annexure – I: Creating the EMR cluster

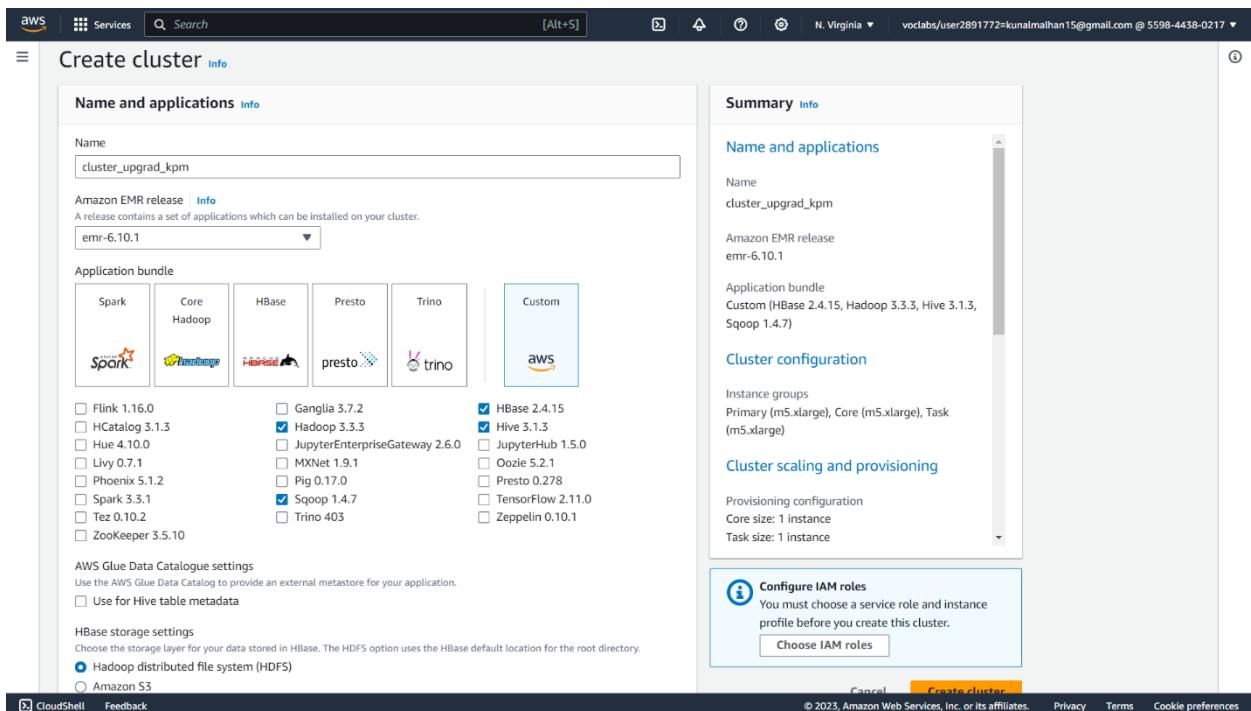
1. Search “EMR” or select “EMR” from recent visited on Console Home of AWS.



The screenshot shows the AWS Console Home page. A search bar at the top contains the query "emr". Below the search bar, there's a sidebar with sections for "Recent" (Documentation, Knowledge Articles, Marketplace, Blogs, Events, Tutorials), "Cost" (Current month costs, Top costs for current month), and "AWS Health" (Info, Open issues, Past 7 days, Scheduled changes, Upcoming and past 7 days, Other notifications, Past 7 days). The main content area displays search results for "emr" under "Services" and "Features". The "EMR" service is listed first under "Services" with the sub-description "Managed Hadoop Framework". Other services listed include MediaStore, MediaConvert, and MediaPackage. Under "Features", there are entries for "Private registry" and "Elastic Container Registry feature".

2. Creating “EMR” cluster with following configurations:

- a. Name : cluster_upgrad_kpm
- b. Amazon EMR release : emr-6.10.1
- c. Application bundle : Custom <Hadoop, HBase, Sqoop, Hive>



The screenshot shows the "Create cluster" wizard in the AWS EMR console. The current step is "Name and applications".
Name: cluster_upgrad_kpm
Amazon EMR release: emr-6.10.1
Application bundle: Custom (HBase 2.4.15, Hadoop 3.3.3, Hive 3.1.3, Presto 0.278, Trino 0.10.1, Zeppelin 0.10.1)
AWS Glue Data Catalogue settings: Use the AWS Glue Data Catalog to provide an external metastore for your application.
HBase storage settings: Choose the storage layer for your data stored in HBase. The HDFS option uses the HBase default location for the root directory.
• Hadoop distributed file system (HDFS)
○ Amazon S3
Summary: Shows the same configuration details as the "Name and applications" step.
Cluster configuration: Instance groups: Primary (m5.xlarge), Core (m5.xlarge), Task (m5.xlarge)
Cluster scaling and provisioning: Provisioning configuration: Core size: 1 instance, Task size: 1 instance
Configure IAM roles: You must choose a service role and instance profile before you create this cluster. [Choose IAM roles](#)

- d. Cluster Configuration:

- i. Primary EC2 instance – m4.xlarge with EBS size of 100 GB
- ii. Core Nodes – m4.xlarge with EBS size of 100 GB
- iii. Task Nodes – None (Removed Task Nodes)

The screenshot shows the 'Cluster configuration' section of the AWS EMR console. It highlights two options for primary node groups: 'Instance groups' (selected) and 'Instance fleets'. Under 'Primary', an m4.2xlarge instance type is chosen. Under 'Core', another m4.2xlarge instance type is chosen. A note indicates that multiple primary nodes can be used. A 'Node configuration - optional' section is present. On the right, a summary panel shows the cluster name 'cluster_upgrad_kpm', application bundle 'Custom (HBase 2.4.15, Hadoop 3.3.3, Hive 3.1.3, Sqoop 1.4.7)', and a 'Configure IAM roles' step.

- e. Cluster scaling and provisioning:
- i. Option : Set cluster size manually
- ii. Provisioning : 3 Instance(s) size
- f. Networking : Default <No Change>

The screenshot shows the 'Cluster scaling and provisioning' section. It highlights 'Set cluster size manually' under 'Choose an option'. In the 'Provisioning configuration' section, a core instance group is set to 3 m4.2xlarge instances. A 'Networking' section includes VPC and subnet configurations. A note about EC2 security groups is present. On the right, a summary panel shows the core size as 3 instances.

- g. Steps : Default <No Change>

- h. Cluster Termination : Manually terminate cluster (to avoid timeout problem)
- i. Bootstrap actions : Default <No Change>

The screenshot shows the AWS EMR Cluster Configuration page. In the left sidebar, under 'Bootstrap actions - optional', there is a table with columns: Name, Status, Type, Arguments, and Script location. A note says 'You don't have any steps added.' Below this is a 'Concurrency' section with a checkbox for 'Run multiple steps in parallel to improve cluster utilisation'. On the right side, there are sections for 'Summary', 'Name and applications' (cluster_upgrad_kpm), 'Amazon EMR release' (emr-6.10.1), 'Application bundle' (Custom (HBase 2.4.15, Hadoop 3.3.3, Hive 3.1.3, Sqoop 1.4.7)), 'Cluster configuration' (Instance groups: Primary (m4.2xlarge), Core (m4.2xlarge)), 'Cluster scaling and provisioning' (Provisioning configuration, Core size: 3 Instances), and 'Networking'. At the bottom right is a 'Create cluster' button.

- j. Cluster logs : Default <No Change>
- k. Tags : Default <No Change>
- l. Edit software settings : Default <No Change>

The screenshot shows the AWS EMR Cluster Configuration page. In the left sidebar, under 'Cluster logs - optional', there is a note about automatic archiving to Amazon S3 and a field for 'Amazon S3 location' (s3://aws-logs-559844380217-us-east-1/elasticmapreduce). Below this is a checkbox for 'Encrypt cluster-specific logs'. Under 'Tags - optional', there is a note about tags and a 'Add new tag' button. Under 'Edit software settings - optional', there is a note about configuration and a choice between 'Enter configuration' (selected) and 'Load JSON from Amazon S3'. The right side of the screen shows the same 'Summary', 'Name and applications', 'Amazon EMR release', 'Application bundle', 'Cluster configuration', 'Cluster scaling and provisioning', 'Networking', and 'Configure IAM roles' sections as the previous screenshot, along with the 'Create cluster' button at the bottom right.

m. Security configuration and EC2 key pair

- Amazon EC2 key pair for SSH : KPM_Upgrade_AWS (Select your own key)

The screenshot shows the AWS Management Console with the search bar set to "Security configuration and EC2 key pair – optional". Under "Amazon EC2 key pair for SSH to the cluster", the key pair "KPM_Upgrade_AWS" is selected. On the right, the "Summary" section shows the name "cluster_upgrad_kpm" and the Amazon EMR release "emr-6.10.1".

n. Identity and Access Management roles (IAM)

- Amazon EMR service Role : EMR_Default_Role
- EC2 instance profile for EMR : EMR_Default_Role

The screenshot shows the "Identity and Access Management (IAM) roles" page. Under "Amazon EMR service role", the "Choose an existing service role" option is selected with "EMR_DefaultRole" chosen. Under "EC2 instance profile for Amazon EMR", the "Choose an existing instance profile" option is selected with "EMR_EC2_DefaultRole" chosen. On the right, the "Summary" section shows the name "cluster_upgrad_kpm", the Amazon EMR release "emr-6.10.1", and the application bundle "Custom (HBase 2.4.15, Hadoop 3.3.3, Hive 3.1.3, Sqoop 1.4.7)". The "Cluster configuration" section shows "Primary (m4.2xlarge), Core (m4.2xlarge)". The "Cluster scaling and provisioning" section shows "Provisioning configuration, Core size: 3 Instances". The "Networking" section has a "Create cluster" button.

3. Allow access to EMR through Security Groups:

- On EMR – Properties – Network and Security, click on EC2 security groups for Primary Node.

The screenshot shows the AWS EMR Cluster Properties page. In the 'Network and security' section, under 'Network', it lists a Virtual Private Cloud (VPC) named 'vpc-0abeac30c3fb16e2b'. Under 'Subnet ID and Availability Zone (AZ)', it shows 'subnet-014b57fb4a7eba732 us-east-1c'. Below this, there's a section for 'EC2 security groups (firewall)' which includes a 'Primary node' entry for 'EMR-managed security group sg-0baef32f65f58f98fa'. There are also sections for 'Additional security groups', 'Core and task nodes', and 'EMR-managed security group sg-00155a72a333a40a'. The 'Security configuration' section shows 'none'. The 'Permissions' section lists the 'Service role for Amazon EMR EMR_DefaultRole' and the 'EC2 instance profile EMR_EC2_DefaultRole'. The 'Auto-scaling role' is listed as 'Not configured'.

The screenshot shows the AWS EC2 Security Groups page. It displays the details for the security group 'sg-0baef32f65f58f98fa - ElasticMapReduce-master'. The 'Details' section shows the security group name 'ElasticMapReduce-master', security group ID 'sg-0baef32f65f58f98fa', a description indicating it's a master group for Elastic MapReduce created on 2023-10-23T20:39:41.505Z, owner '559844380217', inbound rules count (8 permission entries), and outbound rules count (1 permission entry). The 'Inbound rules' tab is selected, showing a table with 8 entries. The table columns are: Name, Security group rule..., IP version, Type, Protocol, and Port range. The entries are:

Name	Security group rule...	IP version	Type	Protocol	Port range
-	sgr-052b0aac134e26e89	-	Custom TCP	TCP	8443
-	sgr-065a2af282f8f47df	-	All UDP	UDP	0 - 65535
-	sgr-036836f66ec1191e4	-	All TCP	TCP	0 - 65535
-	sgr-036836f66ec1191e4	-	All TCP	TCP	0 - 65535
-	sgr-036836f66ec1191e4	-	All TCP	TCP	0 - 65535
-	sgr-036836f66ec1191e4	-	All TCP	TCP	0 - 65535
-	sgr-036836f66ec1191e4	-	All TCP	TCP	0 - 65535
-	sgr-036836f66ec1191e4	-	All TCP	TCP	0 - 65535

- b. Enter and/or ensure Inbound rule with following configuration exists:
- IP version : IPv4
 - Type : SSH
 - Protocol : TCP
 - Port range : 22
 - Source : 0.0.0.0/0 (this will allow all source to connect through)

SSH. One can choose to enter own IP if its fixed IP.)

The screenshot shows the AWS EC2 Security Groups interface. On the left, there's a sidebar with navigation links like EC2 Dashboard, Services, Events, and Instances. Under Instances, there are sub-links for Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, and Images. Below Images are AMIs and AMI Catalog. Under Elastic Block Store are Volumes, Snapshots, and Lifecycle Manager. Under Network & Security are Security Groups and Elastic IPs. At the bottom of the sidebar are CloudShell and Feedback links.

The main content area displays a security group named "ElasticMapReduce-master". It shows the following details:

- Owner: 559844380217
- Inbound rules count: 8
- Permission entries: 8
- Outbound rules count: 1
- Permission entry: vpc-0abeac30c3fb16e2b

Below this, there are tabs for Inbound rules, Outbound rules, and Tags. The Inbound rules tab is selected, showing a table with 8 rows of rules. The columns are: Security group rule..., IP version, Type, Protocol, Port range, and Source. The last row, which is the SSH rule, has the following values:

Security group rule...	IP version	Type	Protocol	Port range	Source
sgr-0453de47f6a58a486	IPv4	SSH	TCP	22	0.0.0.0/0

Successful Installation:

Annexure – II: Creating the RDS instance on AWS

1. Search “RDS” or select “RDS” from recent visited on Console Home of AWS.

The screenshot shows the AWS Console Home page. A search bar at the top contains the query 'rds'. Below it, a sidebar lists various services under 'Services (12)' and 'Features (35)'. The main content area displays search results for 'rds', with the 'RDS' service card being the most prominent. This card includes a star icon, the service name, and a brief description: 'Managed Relational Database Service'. Below the card are links for 'Top features', 'Dashboard', 'Databases', 'Query Editor', 'Performance Insights', and 'Schemas'. To the right of the search results, there is a 'AWS Health' info panel showing 'Open issues' and 'Past 7 days' information. At the bottom of the page, there are links for 'Current month costs' and 'Top costs for current month', along with copyright and privacy information.

2. Click on Create Database.

The screenshot shows the Amazon RDS Dashboard. On the left, a sidebar provides navigation links for 'Dashboard', 'Databases', 'Query Editor', 'Performance insights', 'Schemas', 'Exports in Amazon S3', 'Automated backups', 'Reserved instances', 'Proxies', 'Subnet groups', 'Parameter groups', 'Option groups', 'Custom engine versions', 'Zero-ETL integrations', 'Events', 'Event subscriptions', 'Recommendations (0)', and 'Certificate update'. The main content area is titled 'Resources' and shows a summary of resources in the US East (N. Virginia) region. It includes sections for 'DB Instances (0/40)', 'DB Clusters (0/40)', and 'Automated'. Below these are 'Recent events (9)' and 'Event subscriptions (0/20)'. To the right, a 'Recommended for you' section lists several links: 'Test Your DR Strategy in Minutes', 'Migrate SSRS to RDS for SQL Server', 'Build RDS Operational Tasks', and 'Implementing Cross-Region DR'. At the bottom, an 'Additional information' section provides links to 'Getting started with RDS', 'Overview and features', 'Documentation', 'Articles and tutorials', and 'Data import guide for MySQL'.

3. Creating “RDS” instance with following configurations:

- a. Choose a database creation method : Standard Create <No Change>
- b. Engine Options : MySQL

Screenshot of the AWS RDS 'Create database' wizard showing the 'MySQL' edition selected.

Create database

Choose a database creation method [Info](#)

- Standard create: You set all of the configuration options, including ones for availability, security, backups, and maintenance.
- Easy create: Use recommended best-practice configurations. Some configuration options can be changed after the database is created.

Engine options

Engine type [Info](#)

- Aurora (MySQL Compatible)
- Aurora (PostgreSQL Compatible)
- MySQL
- MariaDB
- PostgreSQL
- Oracle
- Microsoft SQL Server

[CloudShell](#) [Feedback](#) © 2023, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

- c. Edition : MySQL Community
- d. Engine Version : 8.0.33 <No Change>
- e. Templates : Dev/Test

Screenshot of the AWS RDS 'Edition' selection screen showing the 'MySQL' edition selected.

Edition MySQL Community

Known issues/limitations
Review the [Known Issues/limitations](#) to learn about potential compatibility issues with specific database versions.

Hide filters

Show versions that support the Multi-AZ DB cluster [Info](#)
Create a Multi-AZ DB cluster with one primary DB instance and two readable standby DB instances. Multi-AZ DB clusters provide up to 2x faster transaction consistency and automatic failover in typically under 5 seconds.

Show versions that support the Amazon RDS Optimized Writes [Info](#)
Amazon RDS Optimized Writes improves write throughput by up to 2x at no additional cost.

Engine Version

Templates
Choose a sample template to meet your use case.

- Production: Use defaults for high availability and fast, consistent performance.
- Dev/test: This instance is intended for development and testing environments.
- Free tier: Use RDS Free Tier to develop new applications, test existing applications, or gain hands-on experience with Amazon RDS.

Availability and durability

[CloudShell](#) [Feedback](#) © 2023, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

- f. Availability and durability : Single DB instance
- g. Settings

- i. DB instance identifier : tripdb
- ii. Master Username : admin
- iii. Master password : user1234

The screenshot shows the AWS RDS MySQL settings page. On the left, there's a sidebar with 'Settings' selected. The main area has sections for 'DB instance identifier' (set to 'tripdb'), 'Master username' (set to 'admin'), and 'Master password'. A note says 'If you manage the master user credentials in Secrets Manager, some RDS features aren't supported.' Below these are fields for 'Auto generate a password' and 'Master password'. On the right, a panel titled 'MySQL' provides information about the MySQL engine, including its popularity, rich features, and various supported instance classes.

- h. DB Instance Class : Burstable Classes (includes t classes)
- i. Instance configuration : db.t3.medium
- j. Storage
 - i. Storage Type : General Purpose SSD (gp2) <No Change>
 - ii. Allocated storage : 20 GB <No Change>

The screenshot shows the AWS RDS MySQL instance configuration page. It displays 'Instance configuration' options for the selected 'db.t3.micro' instance. Under 'Storage', it shows 'Allocated storage' set to 20 GiB. A note states that after modifying storage, the DB instance will be in storage optimization mode. On the right, a panel titled 'MySQL' provides information about the MySQL engine, including its popularity, rich features, and various supported instance classes.

k. Connectivity <will perform after creation of instance>

- i. Compute resource : Default <No Change>
- ii. Virtual private cloud (VPC) : Default <No Change>
- iii. DB subnet group : Default <No Change>
- iv. Public access : No <No Change>

The screenshot shows the AWS RDS MySQL configuration page under the 'Connectivity' tab. It includes sections for Compute resource, Virtual private cloud (VPC), DB subnet group, and Public access. The Compute resource section has two options: 'Don't connect to an EC2 compute resource' (selected) and 'Connect to an EC2 compute resource'. The VPC section shows 'Default VPC (vpc-0abeac30c3fb16e2b)' selected. The DB subnet group section shows 'default-vpc-0abeac30c3fb16e2b' selected. The Public access section has two options: 'Yes' (selected) and 'No'. A note states: 'After a database is created, you can't change its VPC.' On the right side, there is a summary of MySQL features.

MySQL

- MySQL is the most popular open source database in the world.
- MySQL on RDS offers the rich features of the MySQL community edition with the flexibility to easily scale compute resources or storage capacity for your database.
- Supports database size up to 64 TiB.
- Supports General Purpose, Memory Optimized, and Burstable Performance instance classes.
- Supports automated backup and point-in-time recovery.
- Supports up to 15 Read Replicas per instance, within a single Region or 5 read replicas cross-region.

- v. VPC security group : Default <No Change>
- vi. Availability Zone : <Same as of EMR cluster to save inter AZ NW cost>
- vii. RDS proxy : Default <No Change>

The screenshot shows the AWS RDS MySQL configuration page under the 'RDS proxy' tab. It includes sections for VPC security group (firewall), Availability Zone, and RDS Proxy. The VPC security group section has two options: 'Choose existing' (selected) and 'Create new'. The Availability Zone section shows 'us-east-1c' selected. The RDS Proxy section has a checkbox for 'Create an RDS Proxy' which is unchecked. The Certificate authority - optional section shows 'rds-ca-2019 (default)' selected with an expiry date of 'Aug 22, 2024'. A note says: 'If you don't select a certificate authority, RDS chooses one for you.' On the right side, there is a summary of MySQL features.

MySQL

- MySQL is the most popular open source database in the world.
- MySQL on RDS offers the rich features of the MySQL community edition with the flexibility to easily scale compute resources or storage capacity for your database.
- Supports database size up to 64 TiB.
- Supports General Purpose, Memory Optimized, and Burstable Performance instance classes.
- Supports automated backup and point-in-time recovery.
- Supports up to 15 Read Replicas per instance, within a single Region or 5 read replicas cross-region.

- i. Database authentication
- m. Monitoring

- : Password authentication <No change>
- : Default <No Change>

n. Additional Configuration

: Default <No Change>

The screenshot shows the AWS RDS MySQL configuration interface. On the left, there are several tabs: 'Database authentication' (selected), 'Monitoring', 'Additional configuration', and 'Estimated monthly costs'. Under 'Database authentication', three options are listed: 'Password authentication' (selected), 'Password and IAM database authentication', and 'Password and Kerberos authentication'. Under 'Monitoring', there is a checkbox for 'Enable Enhanced monitoring'. Under 'Additional configuration', there is a note about database options like encryption and backup. Under 'Estimated monthly costs', it states that the Amazon RDS Free Tier is available for 12 months and lists free resources: 750 hrs of Amazon RDS in a Single-AZ db.t2.micro, db.t3.micro or db.t4g.micro Instance, and 20 GB of General Purpose Storage (SSD). On the right, a sidebar titled 'MySQL' provides a brief overview of the service and a bulleted list of features.

MySQL

MySQL is the most popular open source database in the world. MySQL on RDS offers the rich features of the MySQL community edition with the flexibility to easily scale compute resources or storage capacity for your database.

- Supports database size up to 64 TiB.
- Supports General Purpose, Memory Optimized, and Burstable Performance instance classes.
- Supports automated backup and point-in-time recovery.
- Supports up to 15 Read Replicas per instance, within a single Region or 5 read replicas cross-region.

Annexure – III: Connecting the RDS instance to EMR cluster

1. Go to RDS – Databases on AWS console, and select the database with DB identifier as “tripdb”.
2. Click on Actions – Setup EC2 connection

The screenshot shows the AWS RDS Databases page. A green success message at the top states "Successfully created database tripdb". Below it, a tooltip suggests creating a Blue/Green Deployment. The main table lists one database: "tripdb" (DB Identifier, Available, Instance, MySQL Community, us-east-1c). A context menu is open over the "tripdb" row, with "Set up EC2 connection" highlighted in orange. Other options in the menu include "Quick Actions - New", "Convert to Multi-AZ deployment", "Stop temporarily", "Reboot", "Delete", "Set up Lambda connection", "Create read replica", "Create Aurora read replica", "Create Blue/Green Deployment - new", "Promote", "Take snapshot", "Restore to point in time", "Migrate snapshot", "Create RDS Proxy", and "Create ElastiCache cluster - new".

3. Select EC2 instance one by one (Primary and all core nodes):

The screenshot shows the "Set up EC2 connection" step in the RDS setup wizard. It displays a list of EC2 instances: i-0726b8651cd00d90a, i-04bbcd54e612102be, i-00d007d30a4b01211, i-07d854049882c6a60, and i-05f8531c. A search bar is at the top left, and a "Choose an EC2 instance" dropdown is at the bottom left. Buttons for "Cancel" and "Continue" are at the bottom right.

4. Click on Setup Review and confirm.

Screenshot of the AWS RDS console showing the "Review and confirm" step for setting up an EC2 connection. The "Connection summary" section shows a diagram where an RDS instance (tripdb) and an EC2 instance (i-0726b8651cd00d90a) are connected via a VPC security group (rds-ec2-6). Both instances have their respective security groups (rds-ec2-6 and ec2-rds-6) added as connection rules. A note indicates that bold text denotes additions being made to set up a connection.

Changes to RDS database: tripdb

Attribute	Current value	New value
Security group	default	default, rds-ec2-6

Changes to EC2 instance: i-0726b8651cd00d90a

Attribute	Current value	New value
Security group	ElasticMapReduce-master	ElasticMapReduce-master, ec2-rds-6

Buttons: Cancel, Previous, Set up (highlighted)

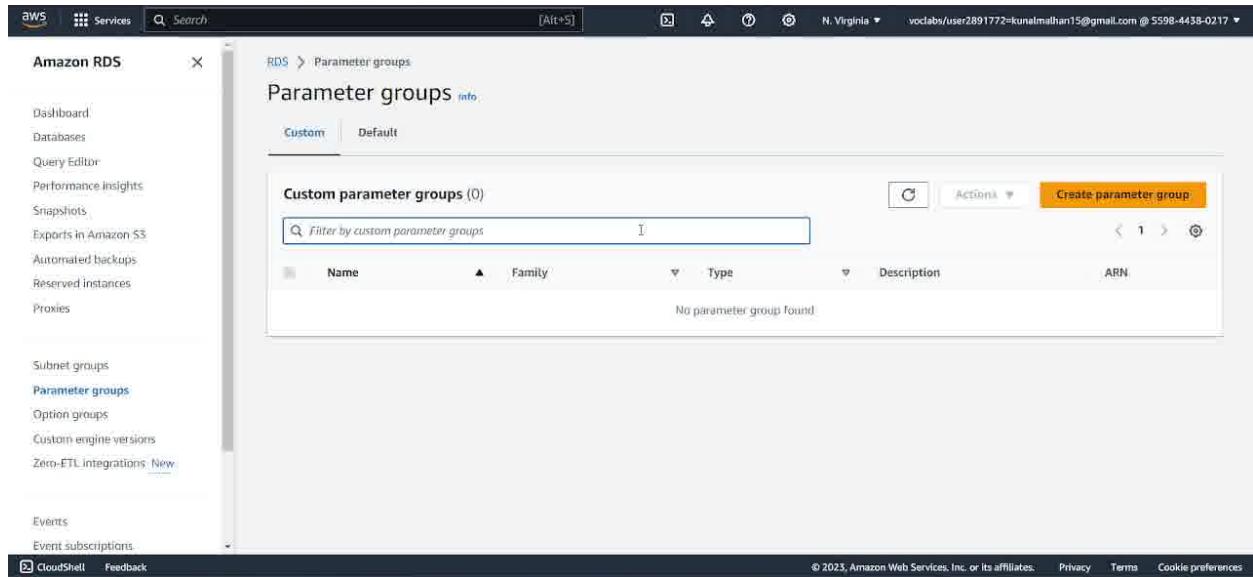
5. Database status will changed to “Modifying”, and then “Available” once completed.

Screenshot of the AWS RDS console showing the "Successfully modified parameter group custompg" message. Below it, a notification encourages creating a Blue/Green deployment. The main view displays a table of databases, showing one entry: tripdb (Status: Available, Instance: MySQL Community, Region & AZ: us-east-1c, Size: db.t3.medium, CPU: 5.09%).

Edit the MySQL Global Variable through Parameter Group

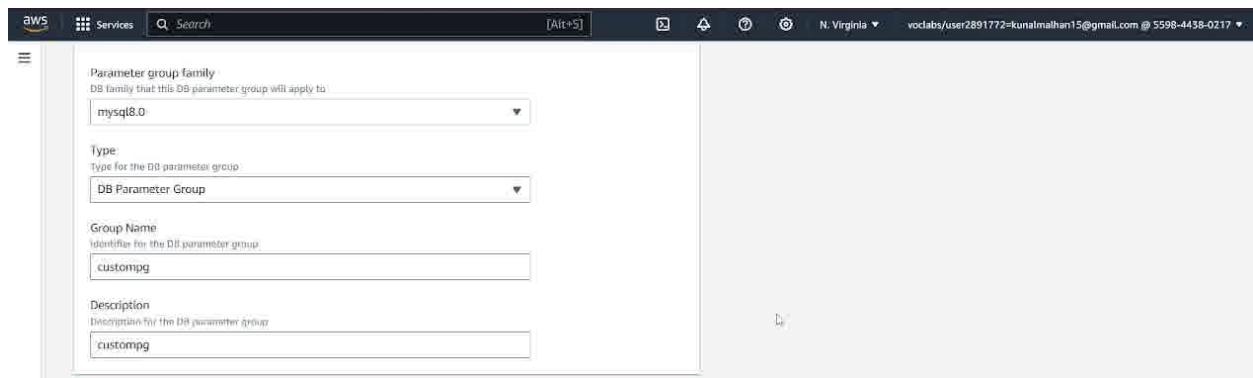
This is required as we are testing using DB instance with only 2 cores and 4 GB memory due to restriction on academy account. To avoid read timeout in parallel fetch, 'net_read_timeout' is set to higher value. This step also provide learning curve for setting any global database variable in similar manner.

1. Go To RDS > Parameter groups



The screenshot shows the AWS RDS Parameter Groups page. The left sidebar has 'Amazon RDS' selected under 'Databases'. The main area shows 'Parameter groups' with a 'Custom' tab selected. A table titled 'Custom parameter groups (0)' is displayed, showing columns for Name, Family, Type, Description, and ARN. A search bar at the top says 'Filter by custom parameter groups'. A prominent orange button at the top right says 'Create parameter group'.

2. Click on 'Create parameter group' and provide required basic details:



The screenshot shows the 'Create parameter group' wizard. Step 1: Set basic parameters. It asks for the 'Parameter group family' (MySQL 8.0), 'Type' (DB Parameter Group), 'Group Name' (custompg), and 'Description' (custompg). The 'Next Step' button is visible at the bottom right.

3. Select new custom parameter group, and click on Edit under Actions:

Parameter groups									
Custom		Default							
Custom parameter groups (1)									
<input type="text" value="Filter by custom parameter groups"/> Actions Create parameter group									
Name	Family	Type	Description	ARN					
custompg	mysql8.0	DB instance parameter group	custompg	arn:aws:rds:us-east-1:559844380217:pg:custompg					

4. Filter and enter modified value of “Global Variable” ‘net_read_timeout’:

Modifiable parameters (393)					
<input style="width: 200px; margin-right: 10px;" type="text" value="net_read"/> 1 match					
Name	Value	Apply type	Data type	Source	
net_read_timeout	28800	Dynamic	Integer	Engine default	

5. Click on Save Changes to complete.

Annexure IV: Login to EMR cluster's master node through putty

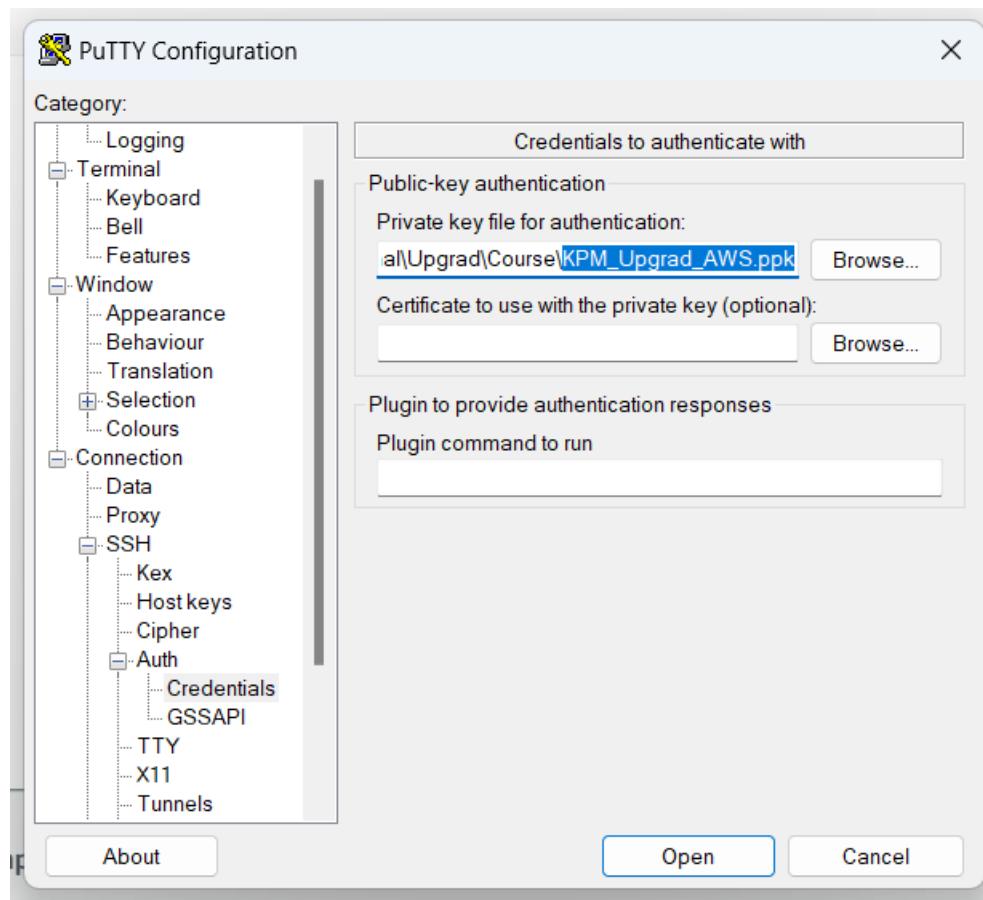
1. Copy the 'Primary node public DNS' from EMR cluster's summary.

The screenshot shows the AWS EMR Cluster Summary page for a cluster named 'cluster_upgrad_kpm'. The 'Summary' tab is selected. In the 'Status and time' section, there is a green notification box stating 'Primary node public DNS copied'. Below this, the copied URL is shown as 'ec2-34-236-158-76.compute-1.amazonaws.com'. There are also links to 'Connect to the Primary node using SSH' and 'Connect to the Primary node using SSM'.

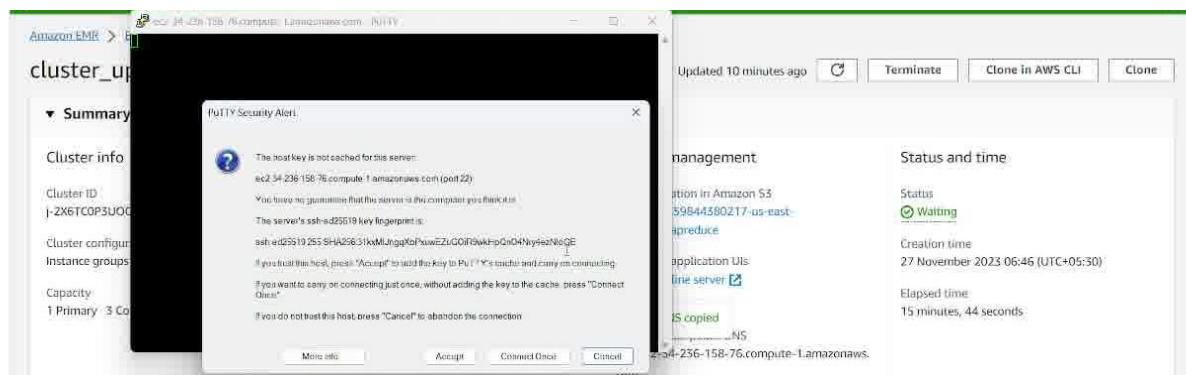
2. Use the Host Name as same public DNS of Primary Node:

The screenshot shows the AWS EMR Cluster Summary page for the same cluster. A Putty Configuration window is overlaid on the page. In the 'Basic options for your PuTTY session' section, the 'Host Name (or IP address)' field contains the value '34-236-158-76.compute-1.amazonaws.com'. The 'Connection type' is set to 'SSH'. The 'Status and time' section on the right shows the cluster status as 'Waiting'.

3. Use the ppk file used for creating the EMR cluster:



4. Click on Accept:



5. Default username for EMR cluster is 'hadoop'. Use to connect successfully:

