2021

BIG DATA DOCUMENTATION

ZOMATO_PROJECT_AUTOMATION

SHRUTI GUPTA

INFOCEPTS TECHNOLOGIES PVT.LTD | PUNE

Setting Up Environment In HDFS

hdfs dfs -mkdir zomato_etl_shruti_gupta

hdfs dfs -mkdir zomato_etl_shruti_gupta/ zomato_ext

hdfs dfs -mkdir zomato_etl_shruti_gupta/zomato_ext/zomato

hdfs dfs -mkdir zomato_etl_shruti_gupta/zomato_ext/dim_country

My HDFS Folder Structure is as follows:

- zomato_etl_shruti_gupta
- zomato_ext
  - zomato
  - dim_country

OUTPUT:

```
File  Edit  View  Search  Terminal  Help
talentum@talentum-virtual-machine:~$ hdfs dfs -mkdir -p zomato_etl_shruti_gupta/zomato_ext/zomato
talentum@talentum-virtual-machine:~$ hdfs dfs -mkdir -p zomato_etl_shruti_gupta/zomato_ext/dim_country
talentum@talentum-virtual-machine:~$ hdfs dfs -ls -R
drwxr-xr-x   - talentum supergroup          0 2021-05-26 01:48 .sparkStaging
drwxr-xr-x   - talentum supergroup          0 2021-05-26 18:08 zomato_etl_shruti_gupta
drwxr-xr-x   - talentum supergroup          0 2021-05-26 18:09 zomato_etl_shruti_gupta/zomato_ext
drwxr-xr-x   - talentum supergroup          0 2021-05-26 18:09 zomato_etl_shruti_gupta/zomato_ext/dim_country
drwxr-xr-x   - talentum supergroup          0 2021-05-26 18:08 zomato_etl_shruti_gupta/zomato_ext/zomato
```

Setting Up Environment In Local File System (UNIX)

mkdir zomato_etl

mkdir zomato_raw_files

mkdir zomato_etl/source

mkdir zomato_etl/source/json

mkdir zomato_etl/source/csv

mkdir zomato_etl/archive mkdir zomato_etl/hive

mkdir zomato_etl/hive/ddl

mkdir zomato_etl/hive/dml

mkdir zomato_etl/spark

mkdir zomato_etl/spark/jars

mkdir zomato_etl/spark/scala

mkdir zomato_etl/script

mkdir zomato_etl/logs

<u>My Local FileSystem Folder Structure is as follows:</u>

zomato_etl

- source
    - json
    - csv
- archive
- hive
    - ddl
    - dml
- spark
    - jars
    - scala
- script (shell scripts and property files)

- logs (log_ddmmyyyy_hhmm.log)

Question 2

Copied file1.json, file2.json, file3.json into source/json folder for the spark application to access and operate on.

Spark Shell Module1 - Json To CSV Convertor

Coding

%pyspark

```
import pyspark.sql.functions as F

df=spark.read.format("json").option("inferSchema","true").load("file:///home/talentum/zomato_etl/source/json/file1.json")

df.printSchema()

new_df = df.select(F.explode(df.restaurants.restaurant))

final_df =new_df.select(new_df.col.R.res_id.alias('Restaurant ID'),new_df.col['name'].alias('Restaurant Name'),new_df.col.location.country_id.alias('Country Code'),new_df.col.location.city.alias("City"),new_df.col.location.address.alias('Address'),new_df.col.location.locality.alias('Locality'),new_df.col.location.locality_verbose.alias('Locality Verbose'),new_df.col.location.longitude.alias('Longitude'),new_df.col.location.latitude.alias('Latitude'),new_df.col.cuisines.alias("Cuisines"),new_df.col.average_cost_for_two.alias("Average Cost For Two"),new_df.col.currency.alias("Currency"),new_df.col.has_table_booking.alias("Has Table Booking"),new_df.col.has_online_delivery.alias("Has Online Delivery"),new_df.col.is_delivering_now.alias("Is Delivering Now"),new_df.col.switch_to_order_menu.alias("Switch To Order Menu"),new_df.col.price_range.alias("Price Range"),new_df.col.user_rating.aggregate_rating.alias("Aggregate Rating"),new_df.col.user_rating.rating_text.alias("Rating Text"),new_df.col.user_rating.votes.alias("Votes")

)

final_df.show()

print(len(final_df.columns))

final_df.write.format('csv').options(delimiter='\t').save('file:///home/talentum/zomato_etl/source/csv/file1.csv')
```

%pyspark

```
import pyspark.sql.functions as F
```

```
df=spark.read.format("json").option("inferSchema","true").load("file:///home/talentum/zomato_etl/so
urce/json/file2.json")

new_df = df.select(F.explode(df.restaurants.restaurant))

final_df =new_df.select(new_df.col.R.res_id.alias('Restaurant
ID'),new_df.col['name'].alias('Restaurant Name'),new_df.col.location.country_id.alias('Country
Code'),new_df.col.location.city.alias("City"),new_df.col.location.address.alias('Address'),new_df.col.l
ocation.locality.alias('Locality'),new_df.col.location.locality_verbose.alias('Locality
Verbose'),new_df.col.location.longitude.alias('Longitude'),new_df.col.location.latitude.alias('Latitude'
),new_df.col.cuisines.alias("Cuisines"),new_df.col.average_cost_for_two.alias("Average Cost For
Two"),new_df.col.currency.alias("Currency"),new_df.col.has_table_booking.alias("Has Table
Booking"),new_df.col.has_online_delivery.alias("Has Online
Delivery"),new_df.col.is_delivering_now.alias("Is Delivering
Now"),new_df.col.switch_to_order_menu.alias("Switch To Order
Menu"),new_df.col.price_range.alias("Price
Range"),new_df.col.user_rating.aggregate_rating.alias("Aggregate
Rating"),new_df.col.user_rating.rating_text.alias("Rating
Text"),new_df.col.user_rating.votes.alias("Votes")

)

print(len(final_df.columns))

final_df.show()

final_df.write.format('csv').options(delimiter='\t').save('file:///home/talentum/zomato_etl/source/csv/fi
le2.csv')
```

%pyspark

```
import pyspark.sql.functions as F

df=spark.read.format("json").option("inferSchema","true").load("file:///home/talentum/zomato_etl/so
urce/json/file3.json")

new_df = df.select(F.explode(df.restaurants.restaurant))

final_df =new_df.select(new_df.col.R.res_id.alias('Restaurant
ID'),new_df.col['name'].alias('Restaurant Name'),new_df.col.location.country_id.alias('Country
Code'),new_df.col.location.city.alias("City"),new_df.col.location.address.alias('Address'),new_df.col.l
ocation.locality.alias('Locality'),new_df.col.location.locality_verbose.alias('Locality
Verbose'),new_df.col.location.longitude.alias('Longitude'),new_df.col.location.latitude.alias('Latitude'
),new_df.col.cuisines.alias("Cuisines"),new_df.col.average_cost_for_two.alias("Average Cost For
Two"),new_df.col.currency.alias("Currency"),new_df.col.has_table_booking.alias("Has Table
Booking"),new_df.col.has_online_delivery.alias("Has Online
Delivery"),new_df.col.is_delivering_now.alias("Is Delivering
Now"),new_df.col.switch_to_order_menu.alias("Switch To Order
Menu"),new_df.col.price_range.alias("Price
Range"),new_df.col.user_rating.aggregate_rating.alias("Aggregate
Rating"),new_df.col.user_rating.rating_text.alias("Rating
Text"),new_df.col.user_rating.votes.alias("Votes")

)
```

print(len(final_df.columns))

final_df.show()

final_df.write.format('csv').options(delimiter='\t').save('file:///home/talentum/zomato_etl/source/csv/file3.csv')

OUTPUT:

```
root
 |-- code: long (nullable = true)
 |-- message: string (nullable = true)
 |-- restaurants: array (nullable = true)
 |    |-- element: struct (containsNull = true)
 |    |    |-- restaurant: struct (nullable = true)
 |    |    |    |-- R: struct (nullable = true)
 |    |    |    |    |-- res_id: long (nullable = true)
 |    |    |    |-- apikey: string (nullable = true)
 |    |    |    |-- average_cost_for_two: long (nullable = true)
 |    |    |    |-- book_url: string (nullable = true)
 |    |    |    |-- cuisines: string (nullable = true)
 |    |    |    |-- currency: string (nullable = true)
 |    |    |    |-- deeplink: string (nullable = true)
 |    |    |    |-- establishment_types: array (nullable = true)
 |    |    |    |    |-- element: string (containsNull = true)
 |    |    |    |-- events_url: string (nullable = true)
```

Took 3 sec. Last updated by anonymous at May 17 2021, 2:19:44 PM.

```
20
+-------------+--------------------+------------+---------------+--------------------+--------------------+--------------------+-------
--------+--------------+--------------------+--------------------+--------+-----------------+-------------------+-----------------+----
---------------+-----------+---------------+-----------+-----+
|Restaurant ID|     Restaurant Name|Country Code|           City|             Address|            Locality|    Locality Verbose|      L
ongitude|      Latitude|            Cuisines|Average Cost For Two|Currency|Has Table Booking|Has Online Delivery|Is Delivering Now|Switc
h To Order Menu|Price Range|Aggregate Rating|Rating Text|Votes|
+-------------+--------------------+------------+---------------+--------------------+--------------------+--------------------+-------
--------+--------------+--------------------+--------------------+--------+-----------------+-------------------+-----------------+----
---------------+-----------+---------------+-----------+-----+
|     16668008|       Arigato Sushi|          37|        Yorkton|14 Second Ave Nor...|             Yorkton|    Yorkton, Yorkton|-102.46
13173000| 51.2106824000|               Asian|                25|       $|               0|                 0|                0|
0|          2|             3.3|    Average|   26|
|       801690|               Mocha|           1|        Lucknow|CP-1, 2nd Floor, ...|         Gomti Nagar|Gomti Nagar, Lucknow|  81.00
11849000| 26.8528099000|Cafe, Italian, Co...|                800|     Rs.|               0|                 0|                0|
0|          3|             4.6|  Excellent|  567|
|     17558738|     Blue House Cafe|         216|       Vernonia|919 Bridge St, Ve...|            Vernonia|  Vernonia, Vernonia|-123.19
```

Took 5 sec. Last updated by anonymous at May 17 2021, 2:20:42 PM.

```
20
+-------------+--------------------+------------+---------------+--------------------+----------------+--------------------+------------
---+-------------+--------------------+--------------------+--------+----------------+-------------------+----------------+----------
----------+-----------+----------------+-----------+-----+
|Restaurant ID|     Restaurant Name|Country Code|           City|             Address|        Locality|    Locality Verbose|      Longit
ude|      Latitude|            Cuisines|Average Cost For Two|Currency|Has Table Booking|Has Online Delivery|Is Delivering Now|Switch To
Order Menu|Price Range|Aggregate Rating|Rating Text|Votes|
+-------------+--------------------+------------+---------------+--------------------+----------------+--------------------+------------
---+-------------+--------------------+--------------------+--------+----------------+-------------------+----------------+----------
----------+-----------+----------------+-----------+-----+
|      2100702|     Barbeque Nation|           1|       Guwahati|2nd Floor, Aditya...|         Ulubari|   Ulubari, Guwahati|  91.7598570
000| 26.1721190000|        North Indian|               1500|     Rs.|               0|                  0|                0|
0|          4|             4.9|  Excellent|  774|
|     16608059| 1918 Bistro & Grill|          14|        Tanunda|94 Murray St, Tan...|         Tanunda|    Tanunda, Tanunda| 138.9660640
000|-34.5196190000|Modern Australian...|                 30|       $|               0|                  0|                0|
0|          3|             4.4|  Very Good|  339|
|     17558684|Berry Patch Resta...|         216|     Clatskanie|49289 Us-30, West...|      Clatskanie|Clatskanie, Clats...|-123.3681510
```

Took 4 sec. Last updated by anonymous at May 17 2021, 2:22:20 PM.

Put all three converted files from local FileSystem to hdfs location

mv zomato_etl/source/csv/file1.csv/part-*.csv zomato_etl/source/csv/zomato_20190609.csv

mv zomato_etl/source/csv/file2.csv/part-*.csv zomato_etl/source/csv/zomato_20190610.csv

mv zomato_etl/source/csv/file3.csv/part-*.csv zomato_etl/source/csv/zomato_20190611.csv

hdfs dfs -put zomato_etl/source/csv/zomato_20190609.csv zomato_etl_shruti_gupta/zomato_ext/zomato

hdfs dfs -put zomato_etl/source/csv/zomato_20190610.csv zomato_etl_shruti_gupta/zomato_ext/zomato

hdfs dfs -put zomato_etl/source/csv/zomato_20190611.csv zomato_etl_shruti_gupta/zomato_ext/Zomato

<u>OUTPUT:</u>

```
talentum@talentum-virtual-machine:~$ hdfs dfs -ls -R /user/talentum/zomato_etl_shruti_gupta
drwxr-xr-x   - talentum supergroup          0 2021-05-26 18:09 /user/talentum/zomato_etl_shruti_gupta/zomato_ext
drwxr-xr-x   - talentum supergroup          0 2021-05-26 18:09 /user/talentum/zomato_etl_shruti_gupta/zomato_ext/dim_country
drwxr-xr-x   - talentum supergroup          0 2021-05-26 18:19 /user/talentum/zomato_etl_shruti_gupta/zomato_ext/zomato
-rw-r--r--   1 talentum supergroup     261702 2021-05-26 18:18 /user/talentum/zomato_etl_shruti_gupta/zomato_ext/zomato/zomato_201906
09.csv
-rw-r--r--   1 talentum supergroup    2271735 2021-05-26 18:19 /user/talentum/zomato_etl_shruti_gupta/zomato_ext/zomato/zomato_201906
10.csv
-rw-r--r--   1 talentum supergroup    1818655 2021-05-26 18:19 /user/talentum/zomato_etl_shruti_gupta/zomato_ext/zomato/zomato_201906
11.csv
talentum@talentum-virtual-machine:~$
```

Question 3

Create External table named “zomato” partitioned by filedate and load zomato_<filedate>.csv into respective partition.

Create Managed table named “zomato_summary_log” table as per given schema to log the job id, step, spark-submit used, start-time, end-time and job status.

Create a Managed table “dim_country” using country_code.csv file given

Creation Of External zomato Table

Coding

```
USE shruti_database;

CREATE external TABLE zomato(`Restaurant ID` INT,
                                                `Restaurant Name` STRING,
                                                `Country Code` INT,
                                                `City` STRING,
                                                `Address` STRING,
                                                `Locality` STRING,
                                                `Locality Verbose` STRING,
                                                `Longitude` STRING,
                                                `Latitude` STRING,
                                                `Cuisines` STRING,
                                                `Average Cost for two` INT,
                                                `Currency` STRING,
                                                `Has Table booking` INT,
                                                `Has Online delivery` INT,
                                                `Is delivering now` INT,
                                                `Switch to order menu` INT,
                                                `Price range` INT,
                                                `Aggregate rating` STRING,
```

```
`Rating text` STRING,
`Votes` STRING)
PARTITIONED BY ( filedate int )
ROW FORMAT DELIMITED
fields terminated by '\t'
stored as textfile;
```

```
hive>ALTER TABLE zomato
SET LOCATION '/user/talentum/zomato_etl_shruti_gupta/zomato_ext/zomato';
```

Load Data Into zomatoTable

Coding

```
hive>LOAD DATA INPATH
'zomato_etl_shruti_gupta/zomato_ext/zomato/zomato_20190609.csv'
OVERWRITE INTO TABLE zomato
PARTITION (filedate='20190609');
hive>LOAD DATA INPATH
'zomato_etl_shruti_gupta/zomato_ext/zomato/zomato_20190610.csv'
OVERWRITE INTO TABLE zomato
PARTITION (filedate='20190610')

hive>LOAD DATA INPATH
'zomato_etl_shruti_gupta/zomato_ext/zomato/zomato_20190611.csv'
OVERWRITE INTO TABLE zomato
PARTITION (filedate='20190611')
```

OUTPUT:

```
hive> ALTER TABLE zomato
    > SET LOCATION '/user/talentum/zomato_etl_shruti_gupta/zomato_ext/zomato';
OK
Time taken: 0.376 seconds
hive> LOAD DATA INPATH
    > 'zomato_etl_shruti_gupta/zomato_ext/zomato/zomato_20190609.csv'
    > OVERWRITE INTO TABLE zomato
    > PARTITION (filedate='20190609');
Loading data to table shruti_database.zomato partition (filedate=20190609)
OK
Time taken: 2.202 seconds
hive> LOAD DATA INPATH
    > 'zomato_etl_shruti_gupta/zomato_ext/zomato/zomato_20190610.csv'
    > OVERWRITE INTO TABLE zomato
    > PARTITION (filedate='20190610');
Loading data to table shruti_database.zomato partition (filedate=20190610)
OK
Time taken: 1.538 seconds
hive> LOAD DATA INPATH
    > 'zomato_etl_shruti_gupta/zomato_ext/zomato/zomato_20190611.csv'
    > OVERWRITE INTO TABLE zomato
    > PARTITION (filedate='20190611');
Loading data to table shruti_database.zomato partition (filedate=20190611)
OK
Time taken: 1.088 seconds
```

Create Managed dim _countrytableandloadcountry_code.csv into it

Coding

```
USE shruti_database;

CREATE TABLE dim_country(
                    `Country Code` int,
                    `Country` string)
                    ROW FORMAT DELIMITED
                    fields terminated by ','
                    stored as textfile;
```

hive>ALTER TABLE dim_country

SET LOCATION '/user/talentum/zomato_etl_shruti_gupta/zomato_ext/dim_country';

Importing into HDFS:

hdfs dfs -put /home/talentum/shared/country_code.csv /user/talentum/zomato_etl_shruti_gupta/zomato_ext/dim_country

```
hive> LOAD DATA INPATH
'/user/hive/warehouse/shruti_database.db/dim_country/country_code.csv'
OVERWRITE INTO TABLE dim_country;
```

Create Managedzomato_summary_log table

Coding

```
USE shruti_database;

CREATE TABLE zomato_summary_log(
                                `Job id` int,
                                `Job step` string,
                                `spark submit command` string,
                                `Job Start time` timestamp,
                                `Job End time` timestamp,
                                `Job status` string);
```

Question 4

Asparkapplicationtoloadzomato_summarytablefromzomatotableand applygiven transformations:

All the columnsfrom thezomato table,adding toit two partition columns “p_filedate” and “p_country_name”, twoderivedcolumns“m_rating_colour”and“m_cuisines”,twoauditcolumns “user_id”and“create_datetime” with relevantconstraints.

Load data in a historical or a filedate/country constraint

CreationOf Managed zomato_summary TableAndLoadDataFrom zomatoTable

Coding

```
USE shruti_database;
```

```
CREATE TABLE zomato_summary(
                    `Restaurant ID` INT,
                    `Restaurant Name` STRING,
                    `Country Code` INT,
                    `City` STRING,
                    `Address` STRING,
                    `Locality` STRING,
                    `Locality Verbose` STRING,
                    `Longitude` STRING,
                    `Latitude` STRING,
                    `Cuisines` STRING,
                    `Average Cost for two` INT,
                    `Currency` STRING,
                    `Has Table booking` INT,
                    `Has Online delivery` INT,
                    `Is delivering now` INT,
                    `Switch to order menu` INT,
                    `Price range` INT,
                    `Aggregate rating` STRING,
                    `Rating text` STRING,
                    `Votes` STRING,
                    `m_rating_colour` STRING,
                    `m_cuisines` STRING,
                    `create_datetime` TIMESTAMP,
                    `user_id` STRING)
                    PARTITIONED BY ( p_filedate INT,
                    p_country_name STRING )
                    stored as ORC;
```

Historically Loading.

```
set hive.exec.dynamic.partition=true;
set hive.exec.dynamic.partition.mode=nonstrict;
```

```
USE shruti_database;
INSERT INTO zomato_summary partition(p_filedate,p_country_name)(
                                        `restaurant id`,
                                        `restaurant name`,
                                        `country code`,
                                        `city`,
                                        `address`,
                                        `locality`,
                                        `locality verbose`,
                                        `longitude`,
                                        `latitude`,
                                        `cuisines`,
                                        `average cost for two`,
                                        `currency`,
                                        `has table booking`,
                                        `has online delivery`,
                                        `is delivering now`,
                                        `switch to order menu`,
                                        `price range`,
                                        `aggregate rating`,
                                        `rating text`,
                                        `votes`,
                                        `p_filedate`,
                                        `p_country_name`,
                                        `m_cuisines`,
                                        `m_rating_colour`,
                                        `create_datetime`,
                                        `user_id`)
                                        SELECT
                                        s.`restaurant id`,
```

```
nvl( s.`restaurant name`,'NA'),
s.`country code`,
nvl(s.`city`, 'NA'),
nvl(s.`address`, 'NA'),
nvl( s.`locality`,'NA'),
nvl(s.`locality verbose`,'NA'),
nvl( s.`longitude`, 'NA'),
nvl( s.`latitude`, 'NA'),
nvl(s.`cuisines`,'NA'),
s.`average cost for two`,
nvl(s.`currency`,'NA'),
s.`has table booking`,
s.`has online delivery`,
s.`is delivering now`,
s.`switch to order menu`,
s.`price range`,
nvl(s.`aggregate rating`,'NA'),
nvl(s.`rating text`,'NA'),
nvl(s.`votes`,'NA'),
s.`filedate`,
nvl(d.`country`,'NA'),
case when cuisines like '%Indian%'or cuisines like '%Andhra%' or cuisines like '%Hyderabadi%' or cuisines like '%Goan%' or cuisines like '%Bengali%' or cuisines like '%Bihari%' or cuisines like '%Chettinad%'or cuisines like '%Gujarati%'or cuisines like '%Rajasthani%'or cuisines like '%Kerala%' or cuisines like '%Maharashtrian%'or cuisines like '%Mangalorean%'or cuisines like '%Mithai%'or cuisines like '%Mughlai%' then 'Indian' else 'World Cuisines' end as m_cuisines, case when `aggregate rating` between 1.9 and 2.4 and
`rating text` = 'Poor' then 'Red' when `aggregate rating` between 2.5 and 3.4 and `rating text`='Average' then 'Amber' when `aggregate rating` between 3.5 and 3.9 and
`rating text`='Good' then 'Light Green' when `aggregate rating` between 4.0 and 4.4 and `rating text`='Very Good' then 'Green' when `aggregate rating` between 4.5 and 5 and
`rating text`='Excellent' then 'Gold' when `aggregate rating` = 0 and `rating text`='Not rated' then 'NA' end as m_rating_colour, from_unixtime(unix_timestamp()),
logged_in_user() from zomato s, dim_country d where s.`country code` = d.`country code`;
```

INSERT OVERWRITE TABLE zomato_summary SELECT DISTINCT * FROM zomato_summary;

FileDate and Country Name Constraint Loading.

```
set hive.exec.dynamic.partition=true;

set hive.exec.dynamic.partition.mode=nonstrict;

USE shruti_database;

INSERT INTO zomato_summary partition(p_filedate,p_country_name)(
                                        `restaurant id`,
                                        `restaurant name`,
                                        `country code`,
                                        `city`,
                                        `address`,
                                        `locality`,
                                        `locality verbose`,
                                        `longitude`,
                                        `latitude`,
                                        `cuisines`,
                                        `average cost for two`,
                                        `currency`,
                                        `has table booking`,
                                        `has online delivery`,
                                        `is delivering now`,
                                        `switch to order menu`,
                                        `price range`,
                                        `aggregate rating`,
                                        `rating text`,
                                        `votes`,
                                        `p_filedate`,
                                        `p_country_name`,
                                        `m_cuisines`,
```

```
`m_rating_colour`,
`create_datetime`,
`user_id`)
SELECT
s.`restaurant id`,
nvl( s.`restaurant name`,'NA'),
s.`country code`,
nvl(s.`city`, 'NA'),
nvl(s.`address`, 'NA'),
nvl( s.`locality`,'NA'),
nvl(s.`locality verbose`,'NA'),
nvl( s.`longitude`, 'NA'),
nvl( s.`latitude`, 'NA'),
nvl(s.`cuisines`,'NA'),
s.`average cost for two`,
nvl(s.`currency`,'NA'),
s.`has table booking`,
s.`has online delivery`,
s.`is delivering now`,
s.`switch to order menu`,
s.`price range`,
nvl(s.`aggregate rating`,'NA'),
nvl(s.`rating text`,'NA'),
nvl(s.`votes`,'NA'),
s.`filedate`,
nvl(d.`country`,'NA'),
case when cuisines like '%Indian%'or cuisines like '%Andhra%' or cuisines like '%Hyderabadi%' or cuisines like '%Goan%' or cuisines like '%Bengali%' or cuisines like '%Bihari%' or cuisines like '%Chettinad%'or cuisines like '%Gujarati%'or cuisines like '%Rajasthani%'or cuisines like '%Kerala%' or cuisines like '%Maharashtrian%'or cuisines like '%Mangalorean%'or cuisines like '%Mithai%'or cuisines like '%Mughlai%' then 'Indian' else 'World Cuisines' end as m_cuisines, case when `aggregate rating` between 1.9 and 2.4 and
`rating text` = 'Poor' then 'Red' when `aggregate rating` between 2.5 and 3.4 and `rating text`='Average' then 'Amber' when `aggregate rating` between 3.5 and 3.9 and
```

`rating text`='Good' then 'Light Green' when `aggregate rating` between 4.0 and 4.4 and `rating text`='Very Good' then 'Green' when `aggregate rating` between 4.5 and 5 and

`rating text`='Excellent' then 'Gold' when `aggregate rating` = 0 and `rating text`='Not rated' then 'NA' end as m_rating_colour, from_unixtime(unix_timestamp()),

logged_in_user() from zomato s, dim_country d and s.filedate = '20190610' and d.country = 'India' ;

INSERT OVERWRITE TABLE zomato_summary SELECT DISTINCT * FROM zomato_summary;

OUTPUT:

```
hive> show tables;
OK
dim_country
zomato
zomato_summary_log
Time taken: 0.114 seconds, Fetched: 3 row(s)
hive> show tables;
OK
dim_country
zomato
zomato_summary
zomato_summary_log
Time taken: 0.048 seconds, Fetched: 4 row(s)
hive> show tables;
OK
dim_country
zomato
zomato_summary
zomato_summary_log
Time taken: 0.051 seconds, Fetched: 4 row(s)
hive> show partitions zomato_summary;
OK
p_filedate=20190609/p_country_name=Brazil
p_filedate=20190609/p_country_name=India
p_filedate=20190609/p_country_name=Indonesia
p_filedate=20190609/p_country_name=New Zealand
p_filedate=20190609/p_country_name=Phillipines
p_filedate=20190609/p_country_name=Qatar
p_filedate=20190609/p_country_name=Singapore
p_filedate=20190609/p_country_name=South Africa
p_filedate=20190609/p_country_name=Sri Lanka
p_filedate=20190609/p_country_name=Turkey
p_filedate=20190609/p_country_name=UAE
p_filedate=20190609/p_country_name=United Kingdom
p_filedate=20190610/p_country_name=Australia
p_filedate=20190610/p_country_name=Canada
p_filedate=20190610/p_country_name=India
p_filedate=20190610/p_country_name=Singapore
p_filedate=20190610/p_country_name=United States
p_filedate=20190611/p_country_name=Australia
p_filedate=20190611/p_country_name=India
p_filedate=20190611/p_country_name=Indonesia
p_filedate=20190611/p_country_name=United States
```

Question 5

Asparkapplicationtoloadzomato_summarytablefromzomatotableand
applygiven transformations:

All the columnsfrom the zomato table, addingtoit two partitioncolumns “p_filedate” and “p_country_name”, twoderivedcolumns“m_rating_colour”and“m_cuisines”,twoauditcolumns “user_id”and“create_datetime” with relevantconstraints.

Load data in a historical or a filedate/country constraint

Module 1: JSON to CSV Convertor - Spark Application

Asparkapplicationthatretrievesa.jsonfilefromlocalfilesystemontoadesignatedhdfslocation.CreatesanRDDto readthenestedjsonfilebymethodsof‘explode’and‘write.option()’. Givinganoutput of the resultant .csv file and creates an hdfs file system configuration to move the file to a designated zomatotables’locationandrenameittothe required“zomato_*.csv”(*->filedate).Andalsocopiestheconverted/renamedcsvfileintolocalfilesystem.

CONNECTING HIVE AND SPARK

ln -s /home/talentum/hive/conf/hive-site.xml /home/talentum/spark/conf/hive-site.xml

Coding

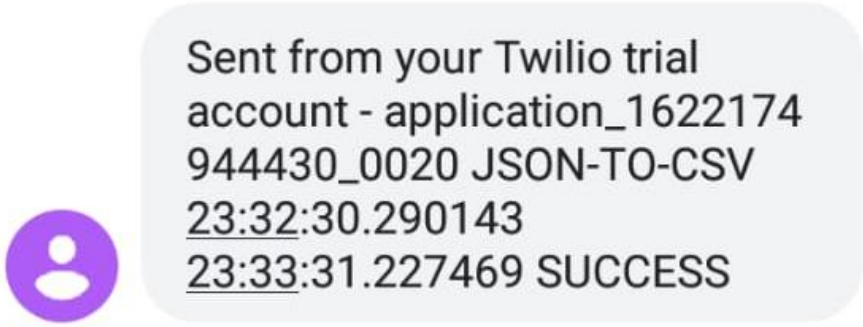
```
from datetime import datetime, timedelta
import os
from pyspark.sql import *
from twilio.rest import Client
import pyspark.sql.functions as F
import logging

account_sid = "ACca96d4741af7faf82e41bfda77ff8c99"
auth_token = "df7e28f820ef9c89a4f902b5c4b4e790"



def result_log(log_data):
    logr = logging.getLogger('mod1')
    logging.basicConfig(level=logging.INFO, format='%(message)s',
filename='/home/talentum/zomato_etl/logs/module_1_status.log', filemode='w')
    logr.info(log_data)
    logging.shutdown()

if __name__ =='__main__':
    spark = SparkSession.builder.master('yarn').enableHiveSupport().getOrCreate()


    start_time = str(datetime.now().time())
    job_step = "JSON-TO-CSV"
    app_id = spark.sparkContext.applicationId  # sc.application_id
    user = spark.sparkContext.sparkUser()

    # hdfs access
```

```
# Gateway
hadoop = spark.sparkContext._jvm.org.apache.hadoop
fs = hadoop.fs.FileSystem
conf = hadoop.conf.Configuration()
path = hadoop.fs.Path
hdfs = fs.get(conf)

# hdfs_path
shruti_host = "/user/talentum/"
shruti_home_hdfs = shruti_host + "zomato_etl_shruti_gupta"

try:
    if not (hdfs.exists(path(shruti_home_hdfs))):
        hdfs.mkdirs(path(shruti_home_hdfs))
    if not (hdfs.exists(path(shruti_home_hdfs + "/zomato_ext"))):
        hdfs.mkdirs(path(shruti_home_hdfs + "/zomato_ext"))
    if not (hdfs.exists(path(shruti_home_hdfs + "/zomato_ext/zomato"))):
        hdfs.mkdirs(path(shruti_home_hdfs + "/zomato_ext/zomato"))

    print("""************************Hello, "+{}+"!**********************
        *************** Beginning Processing!**************""".format(user))

    #  log_file location
    jsontocsv_status_location = "/home/talentum/zomato_etl/logs"

    # log file writing
    log = app_id + " " + job_step + " " + start_time + " " + "NA" + " " + "RUNNING"
    result_log(log)

    # file_writer = open(jsontocsv_status_location + "/module_1_status.log", 'w')
    # file_writer.write(app_id + "\t" + job_step + "\t" + start_timestamp + "\t" + end_timestamp + "\t" + "RUNNING")
    # file_writer.close()
    # json file_path
    file_path = "/home/talentum/zomato_etl/source/json/"

    # Date Incrementation

    date_increment = datetime.now().date()
    # reading and writing files

    file_path_loop = os.listdir(file_path)

    if len(file_path_loop) != 0:

        for i in file_path_loop:
            print("**************CONVERSION {} to CSV FORMAT********************".format(i))
            df = spark.read.format("json").option("inferSchema", "true").load(
                "file:///home/talentum/zomato_etl/source/json/{}".format(i))
            new_df = df.select(F.explode(df.restaurants.restaurant))
            final_df = new_df.select(new_df.col.R.res_id.alias('Restaurant_ID'),
                                     new_df.col['name'].alias('Restaurant_Name'),
                                     new_df.col.location.country_id.alias('Country_Code'),
                                     new_df.col.location.city.alias("City"),
                                     new_df.col.location.address.alias('Address'),
                                     new_df.col.location.locality.alias('Locality'),
                                     new_df.col.location.locality_verbose.alias('Locality_Verbose'),
                                     new_df.col.location.longitude.alias('Longitude'),
                                     new_df.col.location.latitude.alias('Latitude'),
                                     new_df.col.cuisines.alias("Cuisines"),
                                     new_df.col.average_cost_for_two.alias("Average_Cost_For_Two"),
                                     new_df.col.currency.alias("Currency"),
                                     new_df.col.has_table_booking.alias("Has_Table_Booking"),
                                     new_df.col.has_online_delivery.alias("Has_Online_Delivery"),
                                     new_df.col.is_delivering_now.alias("Is_Delivering_Now"),
```

```
                                new_df.col.switch_to_order_menu.alias("Switch_To_Order_Menu"),
                                new_df.col.price_range.alias("Price_Range"),
                                new_df.col.user_rating.aggregate_rating.alias("Aggregate_Rating"),
                                new_df.col.user_rating.rating_text.alias("Rating_Text"),
                                new_df.col.user_rating.votes.alias("Votes")
                                )

            final_df.write.format('csv').options(delimiter='\t').save(
                'file:///home/talentum/zomato_etl/source/csv/{}'.format(i))
            os.system(
                'mv /home/talentum/zomato_etl/source/csv/{}/part*
/home/talentum/zomato_etl/source/csv/zomato_{}'.format(
                    i,
                    str(date_increment)))

            os.system(
                'mv /home/talentum/zomato_etl/source/json/{} /home/talentum/zomato_etl/archive'.format(i))
            # copy from local to hdfs
            hdfs.copyFromLocalFile(path('/home/talentum/zomato_etl/source/csv/zomato_{}'.format(date_increment)),
                                   path('/user/talentum/zomato_etl_shruti_gupta/zomato_ext/zomato'))
            # date incrementation
            date_increment = date_increment + timedelta(1)
        else:
            print("**** OOPS!!! NO FILES FOUND FOR CONVERSION. KINDLY UPLOAD DATA FIRST!!! ****")
            pass
    except(IOError, IndexError, EOFError):
        print("**************************OOPS FAILED!!!!****************************************")
        log = app_id + " " + job_step + " " + start_time + " " + "NA" + " " + "FAILED"
        result_log(log)
        client = Client(account_sid, auth_token)

        message = client.messages.create(
            body=log,
            from_='+12056508193',
            to='+919149014430'
        )
        print(message.status)

    else:
        log = app_id + " " + job_step + " " + start_time + " " + str(datetime.now().time()) + " " + "SUCCESS"
        result_log(log)
        client = Client(account_sid, auth_token)

        message = client.messages.create(
            body=log,
            from_='+12056508193',
            to='+919149014430'
        )
        print(message.status)

        print("""**********************TASK ACCOMPLISHED!!!*******************""")
```

OUTPUT SCREENSHOTS:

```
Automated Execution Begins
Task1 - CONVERSION OF JSON TO CSV Launching
Task Accomplished!!!!
**********************Hello, "+talentum+"!*********************
            ***************     Beginning Processing!*************
*************CONVERSION file2.json to CSV FORMAT********************
*************CONVERSION file4.json to CSV FORMAT********************
*************CONVERSION file1.json to CSV FORMAT********************
*************CONVERSION file5.json to CSV FORMAT********************
*************CONVERSION file3.json to CSV FORMAT********************
queued
********************TASK ACCOMPLISHED!!!*******************
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/talentum/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class
SLF4J: Found binding in [jar:file:/home/talentum/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/Stat
Binder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Connecting to jdbc:hive2://localhost:10000
Connected to: Apache Hive (version 2.3.6)
Driver: Hive JDBC (version 2.3.6)
Transaction isolation: TRANSACTION_REPEATABLE_READ
Error: Error while compiling statement: FAILED: ParseException line 1:316 mismatched input 'To' expecting ) near ''{'' in val
onstructor (state=42000,code=40000)
Closing: 0: jdbc:hive2://localhost:10000
https://api.twilio.com/2010-04-01/Accounts/ACca96d4741af7faf82e41bfda77ff8c99/Messages.json PAYLOAD:
```



Module2: Load Data From CSV To zomato Table-SparkApplication

Aspark applicationthatretrieves a.csvfilefromhdfslocationintozomatotable.Creatingalooptoexecuteonly.csv files and catch all exceptions. Connecting to hive using JDBC connect statement and loadingdataandcreatingpartitions intozomatotableusingthefiledatementionedinthe.csvfilename

Coding

```
from datetime import datetime
from twilio.rest import Client
from pyspark.sql import SparkSession
```

```
import logging

account_sid = "ACca96d4741af7faf82e41bfda77ff8c99"
auth_token = "df7e28f820ef9c89a4f902b5c4b4e790"



def result_log(log_data):
    logr = logging.getLogger('mod2')
    logging.basicConfig(level=logging.INFO, format='%(message)s',
filename='/home/talentum/zomato_etl/logs/module_2_status.log', filemode='w')
    logr.info(log_data)
    logging.shutdown()


if __name__ =='__main__':

    spark = SparkSession.builder.master('yarn').enableHiveSupport().getOrCreate()

    start_time = str(datetime.now().time())
    job_step = "creating-and-loading-in-zomato-table"
    app_id = spark.sparkContext.applicationId  # sc.application_id
    user = spark.sparkContext.sparkUser()

    #  log_file location
    jsontocsv_status_location = "/home/talentum/zomato_etl/logs"

    # # log file writing
    try:
        # log file writing
        log = app_id + " " + job_step + " " + start_time + " " + "NA" + " " + "RUNNING"
        result_log(log)

        # file_writer = open(jsontocsv_status_location + "/module_2_status.log", 'w')
        # file_writer.write(app_id + "\t" + job_step + "\t" + start_timestamp + "\t" + end_timestamp + "\t"
+ "RUNNING")
```

```
# file_writer.close()

# hdfs access

# Gateway

hadoop = spark.sparkContext._jvm.org.apache.hadoop
fs = hadoop.fs.FileSystem
conf = hadoop.conf.Configuration()
path = hadoop.fs.Path
hdfs = fs.get(conf)

# hdfs_path

file_path = path("/user/talentum/zomato_etl_shruti_gupta/zomato_ext/zomato")

# creation of tables

spark.sql("use shruti_database")

spark.sql("""CREATE external TABLE IF NOT EXISTS  shruti_database.zomato
(Restaurant_ID INT,
Restaurant_Name STRING,Country_Code INT,City STRING, Address STRING,Locality STRING,Locality_Verbose STRING,Longitude
STRING,Latitude STRING,Cuisines STRING,Average_Cost_for_two INT,Currency STRING,Has_Table_booking INT,Has_Online_delivery
INT,Is_delivering_now INT,Switch_to_order_menu INT,Price_range INT,Aggregate_rating STRING,Rating_text STRING,Votes STRING)
PARTITIONED BY ( filedate int ) ROW FORMAT DELIMITED fields terminated by '\t' LOCATION '/user/talentum/zomato_etl_shruti_gupta/zomato_ext/zomato'
stored as textfile""")

# hdfs files (Loading data)

hdfs_file_path = hdfs.listStatus(file_path)
files = [i.getPath().getName() for i in hdfs_file_path]
```

```
        if not "filedate" in hdfs_file_path[0].getPath().getName():
            files_in_hdfs = [str(i.getPath().getName().split('_')[1]) for i in hdfs_file_path]
            print(files_in_hdfs)
            # print(files_in_hdfs)
            for i in files_in_hdfs:
                j = ''.join(i.split('-'))
                spark.sql("""LOAD DATA INPATH
'/user/talentum/zomato_etl_shruti_gupta/zomato_ext/zomato/zomato_{}'
                    OVERWRITE INTO TABLE zomato PARTITION (filedate={})""".format(i, j))
        else:
            print("**** OOPS!!! CANNOT PARTITIONED ON EXISTED PARTITION  ****")
    except(IOError, IndexError, EOFError):

        print("************************OOPS FAILED!!!!****************************************")
        log = app_id + " " + job_step + " " + start_time + " " + "NA" + " " + "FAILED"
        result_log(log)
        client = Client(account_sid, auth_token)

        message = client.messages.create(
            body=log,
            from_='+12056508193',
            to='+919149014430'
        )
        print(message.status)

    else:
        log = app_id + " " + job_step + " " + start_time + " " + str(datetime.now().time()) + " " +
"SUCCESS"
        result_log(log)
        client = Client(account_sid, auth_token)

        message = client.messages.create(
            body=log,
            from_='+12056508193',
            to='+919149014430'
        )
```

```
print(message.status)

print("""********************TASK ACCOMPLISHED!!!*******************""")
```

OUTPUT SCREENSHOTS:

```
Task2- CREATING AND LOADING DATA IN TABLE Launching
Task Ccomplished!!!
['2021-05-30', '2021-05-31', '2021-06-01', '2021-06-02', '2021-06-03']
queued
********************TASK ACCOMPLISHED!!!*******************
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/talentum/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/talentum/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLogger
Binder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Connecting to jdbc:hive2://localhost:10000
Connected to: Apache Hive (version 2.3.6)
Driver: Hive JDBC (version 2.3.6)
Transaction isolation: TRANSACTION_REPEATABLE_READ
Error: Error while compiling statement: FAILED: ParseException line 1:316 mismatched input 'To' expecting ) near ''{'' in value row c
onstructor (state=42000,code=40000)
Closing: 0: jdbc:hive2://localhost:10000
https://api.twilio.com/2010-04-01/Accounts/ACca96d4741af7faf82e41bfda77ff8c99/Messages.json PAYLOAD:
```

Module3: Load Data From zomato Table To zomato_summary table - Spark Application

Asparkapplication thatcombinesall thedatafrom zomato tableandmatchesthe country idwiththe datafrom dim_country table to add a column with country name from dim_country while creating audit column and derived column as required by constraints. It also gives the user an option to choose between historical loading and criteria/manual loading ofdata.

Coding

```
from datetime import datetime
from pyspark.sql import SparkSession
import sys
import logging
from twilio.rest import Client

account_sid = "ACca96d4741af7faf82e41bfda77ff8c99"
auth_token = "df7e28f820ef9c89a4f902b5c4b4e790"


def result_log(log_data):
    logr = logging.getLogger('mod3')
```

```
    logging.basicConfig(level=logging.INFO, format='%(message)s',
filename='/home/talentum/zomato_etl/logs/module_3_status.log', filemode='w')
    logr.info(log_data)
    logging.shutdown()


if __name__ =='__main__':

    spark = SparkSession.builder.master('yarn').enableHiveSupport().getOrCreate()
    start_time= str(datetime.now().time())
    job_step = "Creating-dim_country-and-creating-zomato_summary-with-loading-data "
    app_id = spark.sparkContext.applicationId  # sc.applicatio_id
    user = spark.sparkContext.sparkUser()

    #  log_file location
    jsontocsv_status_location = "/home/talentum/zomato_etl/logs"

    # # log file writing
    try:
        log = app_id + " " + job_step + " " + start_time + " " + "NA" + " " + "RUNNING"
        result_log(log)

        # Gateway

        hadoop = spark.sparkContext._jvm.org.apache.hadoop
        fs = hadoop.fs.FileSystem
        conf = hadoop.conf.Configuration()
        path = hadoop.fs.Path
        hdfs = fs.get(conf)



        spark.sql("use shruti_database")

        spark.sql("""CREATE TABLE  IF NOT EXISTS shruti_database.dim_country(Country_Code
int,Country string) ROW FORMAT DELIMITED
        fields terminated by '\t' stored as textfile
        """)

        spark.sql("""ALTER TABLE dim_country
        SET LOCATION '/user/talentum/zomato_etl_shruti_gupta/zomato_ext/zomato/dim_country'
        """)

        spark.sql(
            "LOAD DATA LOCAL INPATH '/home/talentum/zomato_etl/source/csv/country_code1.csv'
OVERWRITE INTO TABLE dim_country")

        spark.sql("""CREATE TABLE IF NOT EXISTS zomato_summary(Restaurant_ID
INT,Restaurant_Name STRING,Country_Code INT,City STRING,
        Address STRING,Locality STRING,Locality_Verbose STRING,Longitude STRING,Latitude
STRING,Cuisines STRING,Average_Cost_for_two INT,
        Currency STRING,Has_Table_booking INT,Has_Online_delivery INT,Is_delivering_now
INT,Switch_to_order_menu INT,Price_range INT,
        Aggregate_rating STRING,Rating_text STRING,Votes STRING, m_rating_colour STRING, m_cuisines
STRING, create_datetime TIMESTAMP,
         user_id STRING)  PARTITIONED BY ( p_filedate INT, p_country_name STRING ) stored as ORC""")

        if (len(sys.argv)>1):
            spark.sql("""INSERT OVERWRITE table shruti_database.zomato_summary
partition(p_filedate='{}',p_country_name='{}')SELECT s.restaurant_id,
        nvl( s.restaurant_name,'NA'),s.country_code,nvl(s.city, 'NA'),nvl(s.address, 'NA'),nvl(
s.locality,'NA'),nvl(s.locality_verbose,'NA'),
        nvl( s.longitude, 'NA'), nvl( s.latitude, 'NA'),
nvl(s.cuisines,'NA'),s.average_cost_for_two,nvl(s.currency,'NA'),s.has_table_booking,
```

```
s.has_online_delivery,s.is_delivering_now,s.switch_to_order_menu,s.price_range,nvl(s.aggregate_rating,'NA'
),nvl(s.rating_text,'NA'),
        nvl(s.votes,'NA'),case when s.cuisines like '%Indian%'or s.cuisines like '%Andhra%' or s.cuisines like
'%Hyderabadi%' or
        s.cuisines like '%Goan%' or s.cuisines like '%Bengali%' or s.cuisines like '%Bihari%' or s.cuisines like
'%Chettinad%'
        or s.cuisines like  '%Gujarati%' or s.cuisines like '%Rajasthani%' or s.cuisines like  '%Kerala%' or
s.cuisines like
        '%Maharashtrian%' or s.cuisines like '%Mangalorean%' or s.cuisines like '%Mithai%'or s.cuisines like
'%Mughlai%' then
        'Indian' else 'World Cuisines' end as m_cuisines, case when s.aggregate_rating between 1.9 and 2.4 and
s.rating_text = 'Poor'
        then 'Red' when s.aggregate_rating between 2.5 and 3.4 and s.rating_text='Average' then 'Amber' when
s.aggregate_rating
        between 3.5 and 3.9 and s.rating_text='Good' then 'Light Green' when s.aggregate_rating between 4.0
and 4.4 and
        s.rating_text='Very Good' then 'Green' when s.aggregate_rating between 4.5 and 5 and
s.rating_text='Excellent' then
        'Gold' when s.aggregate_rating = 0 and s.rating_text='Not rated' then 'NA'  end as m_rating_colour,
        from_unixtime(unix_timestamp()),'talentum' from shruti_database.zomato s,
        shruti_database.dim_country d where s.country_code = d.country_code""".format(sys.argv[1],
sys.argv[2]))

        # ,s.filedate='20210526',d.country='India'


        else:

            spark.sql("set hive.exec.dynamic.partition=true")
            spark.sql("set hive.exec.dynamic.partition.mode=nonstrict")
            spark.sql("""INSERT OVERWRITE table shruti_database.zomato_summary
partition(p_filedate,p_country_name)SELECT s.restaurant_id,
            nvl( s.restaurant_name,'NA'),s.country_code,nvl(s.city, 'NA'),nvl(s.address, 'NA'),nvl(
s.locality,'NA'),nvl(s.locality_verbose,'NA'),
            nvl( s.longitude, 'NA'), nvl( s.latitude, 'NA'),
nvl(s.cuisines,'NA'),s.average_cost_for_two,nvl(s.currency,'NA'),s.has_table_booking,

s.has_online_delivery,s.is_delivering_now,s.switch_to_order_menu,s.price_range,nvl(s.aggregate_rating,'NA'
),nvl(s.rating_text,'NA'),
            nvl(s.votes,'NA'),case when s.cuisines like '%Indian%'or s.cuisines like '%Andhra%' or s.cuisines like
'%Hyderabadi%' or
            s.cuisines like '%Goan%' or s.cuisines like '%Bengali%' or s.cuisines like '%Bihari%' or s.cuisines like
'%Chettinad%'
            or s.cuisines like  '%Gujarati%' or s.cuisines like '%Rajasthani%' or s.cuisines like  '%Kerala%' or
s.cuisines like
            '%Maharashtrian%' or s.cuisines like '%Mangalorean%' or s.cuisines like '%Mithai%'or s.cuisines like
'%Mughlai%' then
            'Indian' else 'World Cuisines' end as m_cuisines, case when s.aggregate_rating between 1.9 and 2.4
and s.rating_text = 'Poor'
            then 'Red' when s.aggregate_rating between 2.5 and 3.4 and s.rating_text='Average' then 'Amber' when
s.aggregate_rating
            between 3.5 and 3.9 and s.rating_text='Good' then 'Light Green' when s.aggregate_rating between 4.0
and 4.4 and
            s.rating_text='Very Good' then 'Green' when s.aggregate_rating between 4.5 and 5 and
s.rating_text='Excellent' then
            'Gold' when s.aggregate_rating = 0 and s.rating_text='Not rated' then 'NA'  end as m_rating_colour,
            from_unixtime(unix_timestamp()),'talentum',s.filedate,nvl(d.country,'NA') from
shruti_database.zomato s,
            shruti_database.dim_country d where s.country_code = d.country_code""")
        spark.sql("INSERT OVERWRITE TABLE zomato_summary SELECT DISTINCT * FROM
zomato_summary")
    except(IOError, IndexError, EOFError):
```

```
        print("************************OOPS EXECUTION FAILED, TRY
AGAIN!!!!*****************************************")
        log = app_id + " " + job_step + " " + start_time + " " + "NA" + " " + "FAILED"
        result_log(log)
        client = Client(account_sid, auth_token)

        message = client.messages.create(
            body=log,
            from_='+12056508193',
            to='+919149014430'
        )
        print(message.status)


    else:
        log = app_id + " " + job_step + " " + start_time + " " + str(datetime.now().time()) + " " + "SUCCESS"
        result_log(log)
        client = Client(account_sid, auth_token)

        message = client.messages.create(
            body=log,
            from_='+12056508193',
            to='+919149014430'
        )
        print(message.status)

        print("""********************CONGRATS!!!!! TASK EXECUTED!!!*******************""")
```

OUTPUT SCREENSHOTS:

```
Task Accomplished!!!
queued
*********************CONGRATS!!!!! TASK EXECUTED!!!********************
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/talentum/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/talentum/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticL
Binder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Connecting to jdbc:hive2://localhost:10000
Connected to: Apache Hive (version 2.3.6)
Driver: Hive JDBC (version 2.3.6)
Transaction isolation: TRANSACTION_REPEATABLE_READ
Error: Error while compiling statement: FAILED: ParseException line 1:318 mismatched input 'To' expecting ) near ''{'' in value
onstructor (state=42000,code=40000)
Closing: 0: jdbc:hive2://localhost:10000
https://api.twilio.com/2010-04-01/Accounts/ACca96d4741af7faf82e41bfda77ff8c99/Messages.json PAYLOAD:
Purging Begins
Deleting Log : modulex_log_28052021_1155.log
Deleting Log : modulex_log_28052021_1156.log
Deleting Log : modulex_log_28052021_1159.log
Deleting Log : modulex_log_28052021_1200.log
Deleting Log : modulex_log_28052021_1204.log
Deleting Log : modulex_log_28052021_1205.log
Deleting Log : modulex_log_28052021_1208.log
Deleting Log : modulex_log_28052021_1209.log
Deleting Log : modulex_log_28052021_1210.log
Deleting Log : modulex_log_28052021_1211.log
Deleting Log : modulex_log_28052021_1212.log
Deleting Log : modulex_log_30052021_2331.log
Deleting Log : modulex_log_30052021_2332.log
Deleting Log : modulex_log_30052021_2333.log
Deleting Log : modulex_log_30052021_2334.log
Deleting Log : modulex_log_30052021_2335.log
Deleting Log : modulex_log_30052021_2336.log
Deleting Log : modulex_log_30052021_2337.log
Deleting Log : modulex_log_30052021_2338.log
Purge Complete!
```

log4j-spark.properties

Defaultlog4j.propertiesfileasacommonfileforallthemodulesthatspecifiesthedetailsandcategoryofinformation that needsto be stored in the log file of thespecific module_log

Coding

log_location = /home/talentum/zomato_etl/logs

app_log_name = modulex_log

```
log4j.rootCategory=INFO,FILE


log4j.appender.FILE=org.apache.log4j.rolling.RollingFileAppender

log4j.appender.FILE.RollingPolicy=org.apache.log4j.rolling.TimeBasedRollingPolicy

log4j.appender.FILE.RollingPolicy.FileNamePattern=${log_location}/${app_log_name}_%d{ddMM
yyyy_HHmm}.log

log4j.appender.FILE.layout=org.apache.log4j.PatternLayout

log4j.appender.FILE.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %-5p %c{1}:%L -
%m%n
```

Module1 Shell Script: Call Module1 Spark Application

Coding

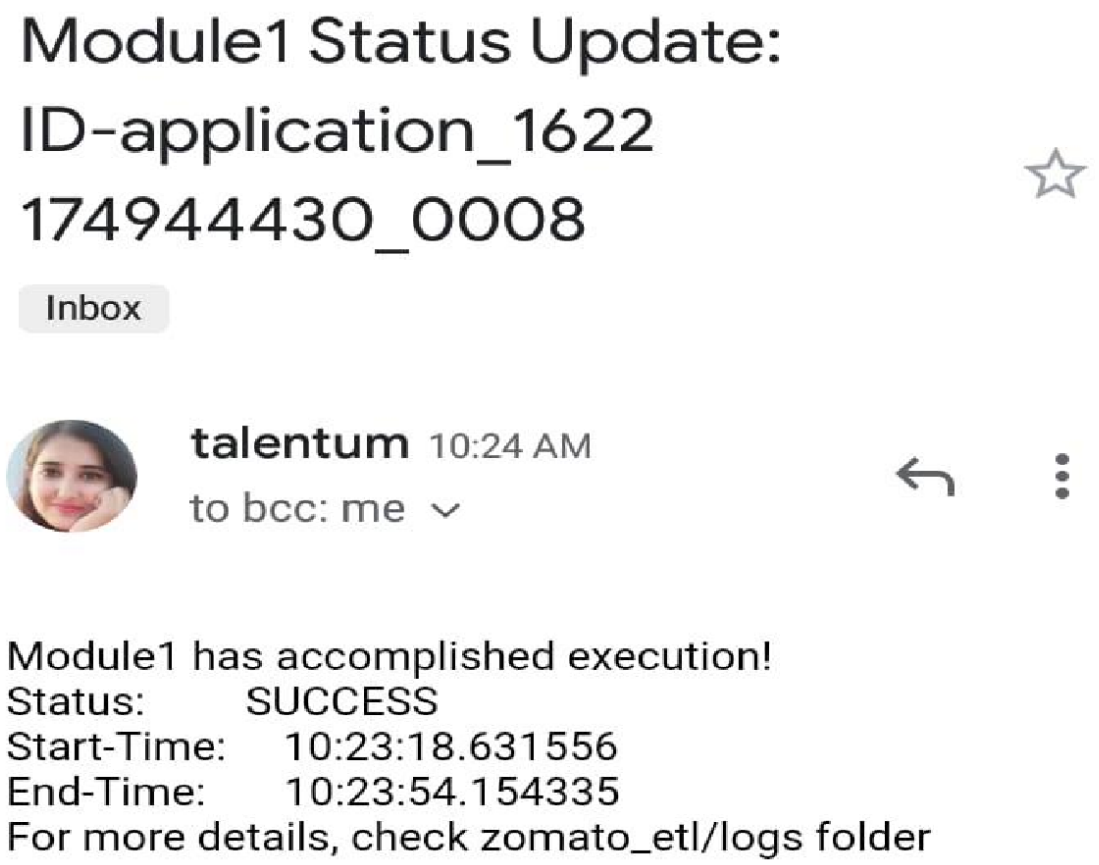
```
#!/bin/bash

#Initialising default variables
file_status_path=/home/talentum/zomato_etl/logs/module_1_status.log
mailing_address="shruti.gupta.training@gmail.com"
module_tag="Module1"

#Mail function to send mail on status update_connect
function user_notification(){
        module_tag=$1
        module_status=$2
        module_initialising=$3
        module_completiontime=$4
        module_id=$5
        echo -e "Subject: $module_tag Status Update: ID-$module_id\n\n$module_tag has accomplished execution!\nStatus:\t\t$module_status\nStart-Time:\t$module_initialising\nEnd-Time:\t$module_completiontime\nFor more details, check zomato_etl/logs folder" | /usr/sbin/sendmail $mailing_address
}

#Spark submit function to call the spark submit command and update the status

function execution_section() {

        echo "Task1 - CONVERSION OF JSON TO CSV Launching"
        spark_submit_value='/home/talentum/spark/bin/spark-submit --master yarn --num-executors 2 --executor-memory 1g /home/talentum/zomato_etl/spark/module1.py'
      echo "Task Accomplished!!!!"
        current_date=$(date +"%Y%m%d")
        $spark_submit_value
```

```
	update_connect "$spark_submit_value"
}

#Update function to update the status log and call the mail function
function update_connect() {

	spark_value=$1

	declare -a revised_value
	if test -f "$file_status_path"; then

		revised_value=(`cat $file_status_path`)

		#Beeline command to load status log into the zomato_summary_log table
		beeline -u jdbc:hive2://localhost:10000 -n hiveuser -p Hive@123
org.apache.hive.jdbc.HiveDriver -e "insert into shruti_database.zomato_summary_log
values('${revised_value[0]}','${revised_value[1]}','$spark_value','${revised_value[2]}','${revised_val
ue[3]}','${revised_value[4]}');"
echo ${revised_value[2]} ${revised_value[3]}

		if [ ${revised_value[4]}="SUCCESSFUL" ]; then {
			user_notification $module_tag "SUCCESS" ${revised_value[2]}
${revised_value[3]} ${revised_value[0]}
		}
		elif [ ${revised_value[4]}="FAILED" ]; then {
			user_notification $module_tag "FAILED" ${revised_value[2]}
${revised_value[3]} ${revised_value[0]}
		}
		elif [ ${revised_value[4]}="RUNNING" ]; then {
			user_notification $module_tag "Unsuccessfully RUNNING"
${revised_value[2]} ${revised_value[3]} ${revised_value[0]}
		}
		else {
```

```
                        user_notification $module_tag "Unknown" ${revised_value[2]} ${revised_value[3]} ${revised_value[0]}
                }
                fi

                else
                        echo "Unable to get updated instance"
                        echo "Could not update json to csv"
                fi
}

#Array to hold status.log file tag
declare -a file_value
if test -f "$file_status_path"; then

        file_value=(`cat $file_status_path`)

        #Case to run application based on running instance check
        case "${file_value[4]}" in
                "SUCCESS")
                        execution_section
                        ;;
                "FAILED")
                        echo "Previous Instance failure occured"
                        user_notification $module_tag "Previous Instance failure occured" ${file_value[2]} ${file_value[3]} ${file_value[0]}
                        execution_section
                        ;;
                "RUNNING")
                        echo "Running in progress"
                        echo "Terminating"
                        user_notification $module_tag "Running in progress" ${file_value[2]} ${file_value[3]} ${file_value[0]}
```

```
                        ;;
                *)
                        user_notification $module_tag "Something went wrong, Removing status
logs!" ${file_value[2]} ${file_value[3]} ${file_value[0]}
                        echo "Something went wrong, restarting this module!"
                        echo "Removing corrupted status log"
                        rm "$file_status_path"
                        execution_section
                        ;;
        esac
else
        echo "module_1_status file not found, Running Spark Application"
        execution_section
fi

echo $? $current_date $module_tag >> execution_status.txt
```

OUTPUT MAIL SCREENSHOTS:



Module2 Shell Script: Call Module2 Spark Application

Coding

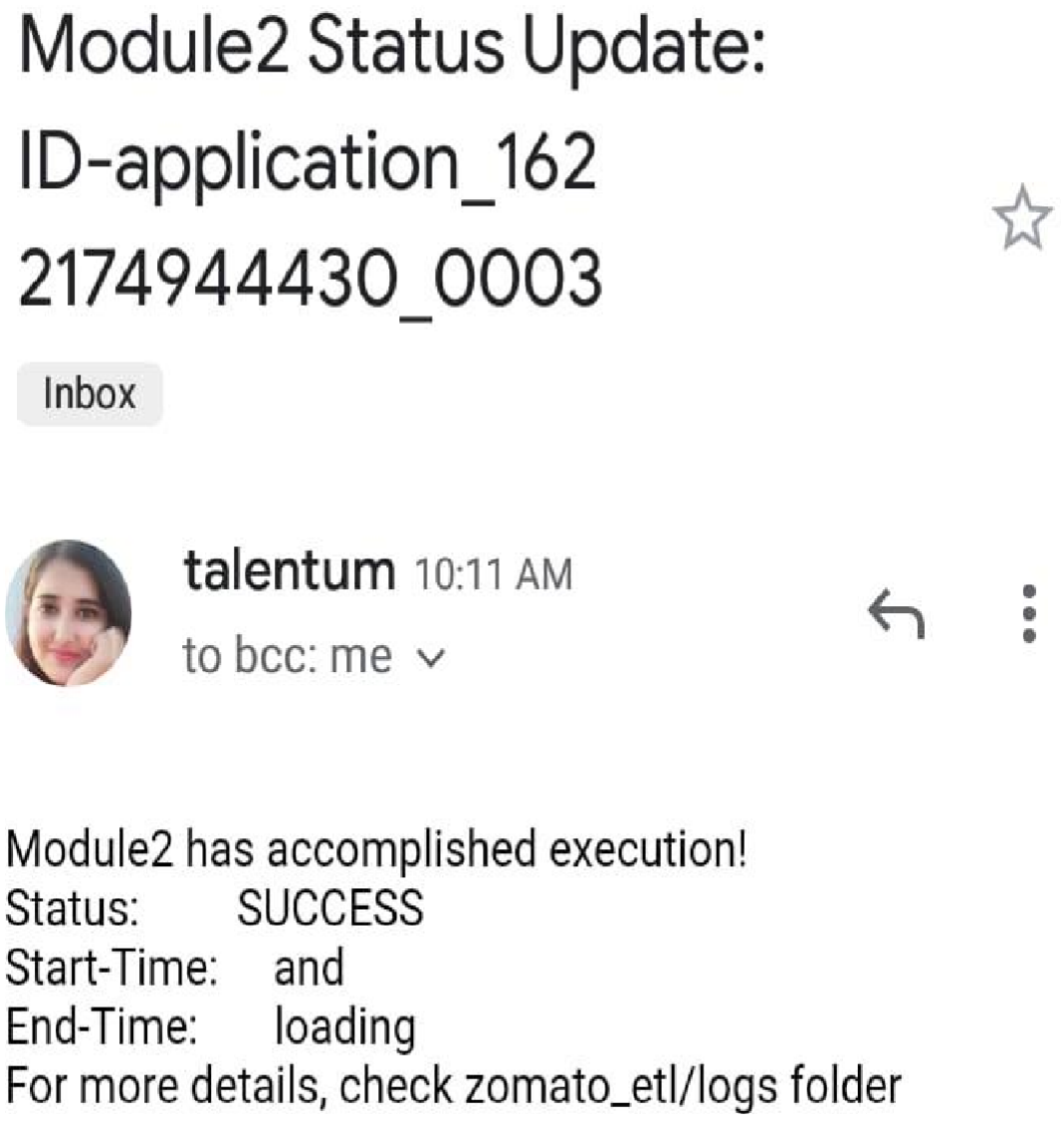
```
#!/bin/bash

#Initialising default variables
file_status_path=/home/talentum/zomato_etl/logs/module_2_status.log
mailing_address="shruti.gupta.training@gmail.com"
module_tag="Module2"

#Mail function to send mail on status update_connect
function mail_notify(){
        module_tag=$1
        module_status=$2
        module_initialising=$3
        module_completiontime=$4
        module_id=$5
        echo -e "Subject: $module_tag Status Update: ID-$module_id\n\n$module_tag has accomplished execution!\nStatus:\t\t$module_status\nStart-Time:\t$module_initialising\nEnd-Time:\t$module_completiontime\nFor more details, check zomato_etl/logs folder" | /usr/sbin/sendmail $mailing_address
}

#Spark submit function to call the spark submit command and update the status

function execution_section() {

        echo "Task2- CREATING AND LOADING DATA IN TABLE Launching "
        spark_submit_value='/home/talentum/spark/bin/spark-submit --master yarn --num-executors 2 --executor-memory 1g /home/talentum/zomato_etl/spark/module2.py'
      echo "Task Ccomplished!!!"
        current_date=$(date +"%Y%m%d")
        $spark_submit_value
```

```
	update_connect "$spark_submit_value"
}

#Update function to update the status log and call the mail function
function update_connect() {

	spark_value=$1

	declare -a revised_value
	if test -f "$file_status_path"; then

		revised_value=(`cat $file_status_path`)

		#Beeline command to load status log into the zomato_summary_log table
		beeline -u jdbc:hive2://localhost:10000 -n hiveuser -p Hive@123
org.apache.hive.jdbc.HiveDriver -e "insert into shruti_database.zomato_summary_log
values('${revised_value[0]}','${revised_value[1]}','$spark_value','${revised_value[2]}','${revised_val
ue[3]}','${revised_value[4]}');"
echo ${revised_value[2]} ${revised_value[3]}

		if [ ${revised_value[4]}="SUCCESSFUL" ]; then {
			mail_notify $module_tag "SUCCESS" ${revised_value[2]}
${revised_value[3]} ${revised_value[0]}
		}
		elif [ ${revised_value[4]}="FAILED" ]; then {
			mail_notify $module_tag "FAILED" ${revised_value[2]}
${revised_value[3]} ${revised_value[0]}
		}
		elif [ ${revised_value[4]}="RUNNING" ]; then {
			mail_notify $module_tag "Unsuccessfully RUNNING" ${revised_value[2]}
${revised_value[3]} ${revised_value[0]}
		}
		else {
```

```
                        mail_notify $module_tag "Unknown" ${revised_value[2]}
${revised_value[3]} ${revised_value[0]}
                }
                fi

                else
                        echo "Unable to get updated instance"
                        echo "Could not update zomato table"
                fi
}

#Array to hold status.log file tag
declare -a file_value
if test -f "$file_status_path"; then

        file_value=(`cat $file_status_path`)

        #Case to run application based on running instance check
        case "${file_value[4]}" in
                "SUCCESS")
                        execution_section
                        ;;
                "FAILED")
                        echo "Previous Instance failure occured"
                        mail_notify $module_tag "Previous Instance failure occured"
${file_value[2]} ${file_value[3]} ${file_value[0]}
                        execution_section
                        ;;
                "RUNNING")          echo "Running in progress"
                        echo "Terminating"
                        mail_notify $module_tag "Running in progress" ${file_value[2]}
${file_value[3]} ${file_value[0]}
                        ;;                              *)
```

```
                        mail_notify $module_tag "Something went wrong, Removing status logs!"
${file_value[2]} ${file_value[3]} ${file_value[0]}
                        echo "Something went wrong, restarting this module!"
                        echo "Removing corrupted status log"
                        rm "$file_status_path"
                        execution_section
                        ;;
        esac
else
        echo "module_2_status file not found, Running Spark Application"
        execution_section
fi
echo $? $current_date $module_tag >> execution_status.txt
```

OUTPUT MAILSCREENSHOTS:



Module3 Shell Script: Call Module3 Spark Application

Coding

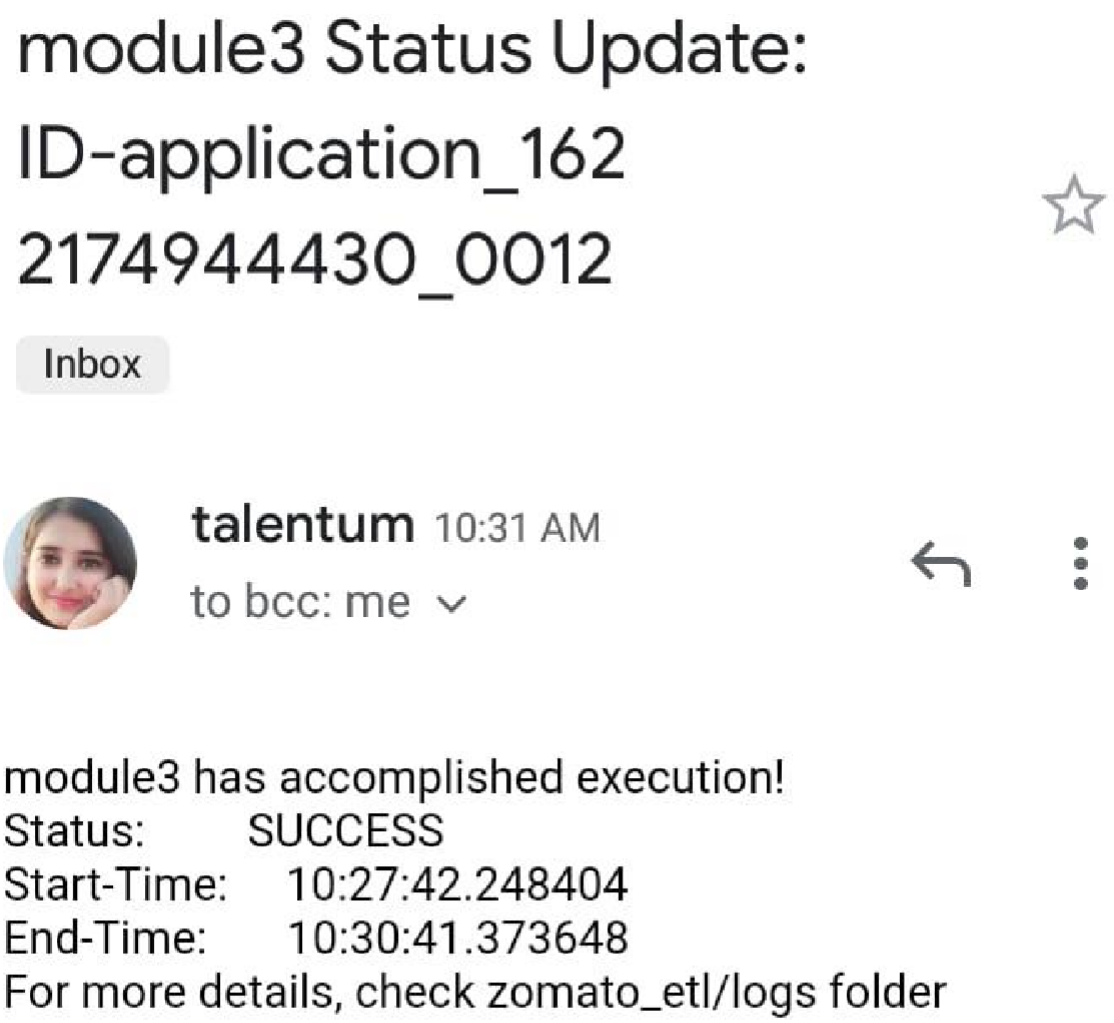
```
.txt#!/bin/bash

#Initialising default variables
file_status_path=/home/talentum/zomato_etl/logs/module_3_status.log
mailing_address="shruti.gupta.training@gmail.com"
module_tag="module3"

a=$1
b=$2

#Mail function to send mail on status update_connect
function user_notification(){
	module_tag=$1
	module_status=$2
	module_initialising=$3
	module_completiontime=$4
	module_id=$5
	echo -e "Subject: $module_tag Status Update: ID-$module_id\n\n$module_tag has accomplished execution!\nStatus:\t\t$module_status\nStart-Time:\t$module_initialising\nEnd-Time:\t$module_completiontime\nFor more details, check zomato_etl/logs folder" | /usr/sbin/sendmail $mailing_address
}

#Spark submit function to call the spark submit command and update the status

function execution_section() {

	echo "TASK 3 Launching"
	spark_submit_value="/home/talentum/spark/bin/spark-submit --master yarn --num-executors 2 --executor-memory 1g /home/talentum/zomato_etl/spark/module3.py $a $b"
```

```
        echo "Task Accomplished!!!"
            current_date=$(date +"%Y%m%d")
            $spark_submit_value

            update_connect "$spark_submit_value"
}

#Update function to update the status log and call the mail function
function update_connect() {

            spark_value=$1

            declare -a revised_value
            if test -f "$file_status_path"; then

                        revised_value=(`cat $file_status_path`)

                        #Beeline command to load status log into the zomato_summary_log table
                        beeline -u jdbc:hive2://localhost:10000 -n hiveuser -p Hive@123
org.apache.hive.jdbc.HiveDriver -e "insert into shruti_database.zomato_summary_log
values('${revised_value[0]}','${revised_value[1]}','$spark_value','${revised_value[2]}','${revised_val
ue[3]}','${revised_value[4]}');"
echo ${revised_value[2]} ${revised_value[3]}

                        if [ ${revised_value[4]}="SUCCESSFUL" ]; then {
                                    user_notification $module_tag "SUCCESS" ${revised_value[2]}
${revised_value[3]} ${revised_value[0]}
                        }
                        elif [ ${revised_value[4]}="FAILED" ]; then {
                                    user_notification $module_tag "FAILED" ${revised_value[2]}
${revised_value[3]} ${revised_value[0]}
                        }
                        elif [ ${revised_value[4]}="RUNNING" ]; then {
```

```
                        user_notification $module_tag "Unsuccessfully RUNNING"
${revised_value[2]} ${revised_value[3]} ${revised_value[0]}
                }
                else {
                        user_notification $module_tag "Unknown" ${revised_value[2]}
${revised_value[3]} ${revised_value[0]}
                }
                fi

                else
                        echo "Unable to get updated instance"
                        echo "Could not update zomato_summary table"
                fi
}

#Array to hold status.log file tag
declare -a file_value
if test -f "$file_status_path"; then

        file_value=(`cat $file_status_path`)

        #Case to run application based on running instance check
        case "${file_value[4]}" in
                "SUCCESS")
                        execution_section
                        ;;
                "FAILED")
                        echo "Previous Instance failure occured"
                        user_notification $module_tag "Previous Instance failure occured"
${file_value[2]} ${file_value[3]} ${file_value[0]}
                        execution_section
                        ;;
                "RUNNING")
```

```
                        echo "Running in progress"
                        echo "Terminating"
                        user_notification $module_tag "Running in progress" ${file_value[2]} ${file_value[3]} ${file_value[0]}
                        ;;
                *)
                        user_notification $module_tag "Something went wrong, Removing status logs!" ${file_value[2]} ${file_value[3]} ${file_value[0]}
                        echo "Something went wrong, restarting this module!"
                        echo "Removing corrupted status log"
                        rm "$file_status_path"
                        execution_section
                        ;;
        esac
else
        echo "module_3_status file not found, Running Spark Application"
        execution_section
fi

echo $? $current_date $module_tag >> execution_status.txt
```

OUTPUT MAIL SCREENSHOTS:



Purge_Logs Shell Script:

A shell script that traversesthrough all the files present in the logs folder and checks the file date difference between thecreationdateofthelogfilebasedonthenomenclatureofthelogfileandthecurrentdateandrunsanifcondition to delete the files that are under 7 days old and more than 1 day old in order to save the logs of the most recent execution for reference purposes.

Coding

```
#!/bin/bash

#Initialising variable with default data
log_path=/home/talentum/zomato_etl/logs/
current_date=$(date +"%d%m%Y")

echo "Purging Begins"

#Looping through the list of all files in the path
for file in `ls $log_path`;
do
	#Retrieving only the file name from the list
	file_name="$(basename "$file")"

	#Searching of the specific naming pattern log
	file_name_pattern=".{7}_log_.{8}_.*.log" # no.of characters
	if [[ $file_name =~ $file_name_pattern ]]; then

		#Retreiving the creation date from the name of the log file
		log_date=$(awk -F '_' '{print $3}' <<< $file_name)
		#Calculating difference
		difference="$(($current_date - $log_date))"
		#Checking the constraint
		if (($difference>=0 && $difference<70000000)); then
			echo "Deleting Log :" $file_name
```

```
                        #Deleting the log file
                        rm "$log_path$file_name"
                fi
        fi

done

echo "Purge Complete!"
```

Wrapper Shell Script:

A wrapper shell script that gets one argument from the user with a default case that calls the usage function to help theuser choosetheargumentcorrectly incaseotherwise. Itcallsthemodulesandthe purge_logs shell script to clean the logs directory after the execution of the module.

Coding

```
#!/bin/bash

#Gets option from User
opt=$1

#Default usage function to help the user incase of a wrong argument
function usage() {
        echo "usage:${0} OPTION"
        echo "1 <- Execute Module 1 [TASK- Conversion JSON TO CSV]"
        echo "2 <- Execute Module 2 [TASK -Create and load into zomato table]"
        echo "3 <- Execute Module 3 [TASK -Load data into zomato_summary table]"
        echo "4 <- Execute Module 4 [Automation Magic!!!]"
        exit 1
}
#Runs a case statement on the argument given by user
case $opt in
```

```
#Module 1 is called along with purge_logs
1)
        bash /home/talentum/zomato_etl/script/module_1.sh
        bash /home/talentum/zomato_etl/script/purge_logs_update.sh
        ;;

#Module 2 is called along with purge_logs
2)
        bash /home/talentum/zomato_etl/script/module_2.sh
        bash /home/talentum/zomato_etl/script/purge_logs_update.sh
        ;;

#Module 3 is called along with purge_logs
3)
        bash /home/talentum/zomato_etl/script/module_3.sh
        bash /home/talentum/zomato_etl/script/purge_logs_update.sh
        ;;

#All the modules are called along with purge_logs
4)
        echo "Automated Execution Begins"
        bash /home/talentum/zomato_etl/script/module_1.sh
        bash /home/talentum/zomato_etl/script/module_2.sh
        bash /home/talentum/zomato_etl/script/module_3.sh
        bash /home/talentum/zomato_etl/script/purge_logs_update.sh


        ;;

#Default Module is called with usage function
*)
```

usage

exit 1

;;

Esac

OUTPUT:

```
talentum@talentum-virtual-machine:~/zomato_etl/script$ ./wrapper_script.sh 4
Automated Execution Begins
Status file not found, Running Spark Application
Module 1 script has begun processing
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 2.4.5
      /_/

Using Python version 3.6.9 (default, Jan 26 2021 15:33:00)
SparkSession available as 'spark'.
**********************Hello, "+talentum+"!****************************
        ********************************************************************
        *****************      Beginning Processing!*************
********************SUCCESSFUL!!!*******************
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/talentum/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/talentum/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLogger
Binder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Connecting to jdbc:hive2://localhost:10000
21/05/26 20:16:04 [main]: WARN jdbc.HiveConnection: Failed to connect to localhost:10000
Could not open connection to the HS2 server. Please check the server URI and if the URI is correct, then ask the administrator to che
ck the server status.
Error: Could not open client transport with JDBC Uri: jdbc:hive2://localhost:10000: java.net.ConnectException: Connection refused (Co
nnection refused) (state=08S01,code=0)
Cannot run commands specified using -e. No current connection
20:15:57.685040 20:15:57.843333
Status file not found, Running Spark Application
Module 2 script has begun processing
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 2.4.5
      /_/

Using Python version 3.6.9 (default, Jan 26 2021 15:33:00)
SparkSession available as 'spark'.
[]
********************SUCCESSFUL!!!*******************
```

```
Purging Begins
Deleting Log : modulex_log_26052021_2015.log
Deleting Log : modulex_log_26052021_2016.log
Deleting Log : modulex_log_26052021_2017.log
Purge Complete!
```

Question 6 - Crontab cronjob setting

Creating a cronjob using crontab to run all the three modules automatically at 1am everyday.

Coding

sudo apt-get update
sudo apt-get install cron
systemctl enable crontab

```
>> crontab -e

0    1    *    *    *    /bin/bash /home/talentum/zomato_etl/script/wrapper_script.sh 4
```