|  |  |
| --- | --- |
| **Project Code** | BCTFEB20-Big Data |
| **Project Name** | Zomato ETL |

|  |  |  |
| --- | --- | --- |
| **Prepared/Modified by** | **Role** | **Date of Preparation** |
| Avinash Khajure  Rajiv Tanna | Lead Analyst Lead Analyst | 06-SEP-2019 |
| **Reviewed by** | **Role** | **Date of Review** |
| Akash Jaiswal | Project Lead | 06-SEP-2019 |
| **Approved by** | **Role** | **Date of Approval** |
| Dhananjay Joshi | Senior Architect | 06-SEP-2019 |
| **Circulation List** |  | **Version Number of the template:** |
| **Version Number** |  | 1.0 |

1. External Source on boarding to HDFS
   1. Download Zomato restaurant data into zomato\_raw\_files folder



1. Converting un-structured data into csv data
   1. Copy first three files from **zomato\_raw\_files (Unix)** to **zomato\_etl/source/json** folder
   2. Write an **application** to convert each json file into csv file with suffix as \*\_filedate.csv and store them in **zomato\_etl/source/csv/** (Unix) folder

For example:

**json1🡪 zomato\_20190609.csv**

**json2🡪 zomato\_20190610.csv**

**json3🡪 zomato\_20190611.csv**

json4🡪 zomato\_20190612.csv

json5🡪 zomato\_20190613.csv

* 1. Note: “zomato\_\*.csv” should have below fields only

1. Restaurant ID
2. Restaurant Name
3. Country Code
4. City
5. Address
6. Locality
7. Locality Verbose
8. Longitude
9. Latitude
10. Cuisines
11. Average Cost for two
12. Currency
13. Has Table booking
14. Has Online delivery
15. Is delivering now
16. Switch to order menu
17. Price range
18. Aggregate rating
19. Rating text
20. Votes
21. Creation of External/Internal Hive table
    1. Move these csv files from **zomato\_etl/source/csv** folder to HDFS ***<HDFS LOCATION>***
    2. Create External Table named “**zomato**” partitioned by fildedate and load **zomato\_<filedate>.csv** into respective partition:
       1. Table should have all columns as per csv file
       2. New partition should be created whenever new file arrives
    3. Create Hive Managed Table named “**dim\_country**” using **country\_code.csv** file as per below details:
       1. Table should have all columns as per csv file
    4. Create zomato\_summary\_log table , schema given below:
       1. Job id
       2. Job Step
       3. Spark submit command
       4. Job Start time
       5. Job End time
       6. Job status
22. Transformation using Hive and Spark
    1. Write a **spark application** in scala to load summary table named “zomato\_summary” and apply the following transformation
       1. Create “zomato\_summary” table partitioned by **p\_filedate**,**p\_country\_name**
       2. **Schema for zomato\_summary table is mentioned below:**
          1. Restaurant ID
          2. Restaurant Name
          3. Country Code
          4. City
          5. Address
          6. Locality
          7. Locality Verbose
          8. Longitude
          9. Latitude
          10. Cuisines
          11. Average Cost for two
          12. Currency
          13. Has Table booking
          14. Has Online delivery
          15. Is delivering now
          16. Switch to order menu
          17. Price range
          18. Aggregate rating
          19. Rating text
          20. Votes
          21. m\_rating\_colour
          22. m\_cuisines
          23. p\_filedate
          24. p\_country\_name
          25. create\_datetime
          26. user\_id
       3. Data in this hive table should be in ORC format
       4. Add audit columns “create\_datetime” and “user\_id” in zomato\_summary table
       5. Derive a column “Rating Colour” based on the rule listed below

|  |  |  |
| --- | --- | --- |
| **Rating Text** | **Aggregate Rating** | **m\_rating\_colour** |
| Poor | 1.9-2.4 | Red |
| Average | 2.5-3.4 | Amber |
| Good | 3.5-.39 | Light Green |
| Very Good | 4.0-4.4 | Green |
| Excellent | 4.5-5 | Gold |

* + 1. Derive a column “**m\_cuisines**” and map the Indian (Andhra,Goan,Hyderabadi,North Indian etc.) cuisines to **“Indian”** and rest of the cuisines to **“World Cuisines”**
    2. Filter out the restaurants with NULL/BLANCK **Cuisines** values
    3. Populate “NA” in case of Null/blank values for string columns
    4. There should be no duplicate record in the summary table
  1. Spark application should be able to perform the following load strategies
     + 1. Manual 🡪 should be able to load the data for a Particular **filedate & country\_name**
       2. Historical 🡪 should be able to load the data historically for all the **filedate & country\_name**

1. Create a shell script wrapper to execute the complete flow as given
   1. ***Spark application and shell script should be parametrized (dbname, tablename, filters, arguments etc.)***
   2. ***Check if already another instances is running for the same application***
      1. ***Exit and send notification if already an instances is running or the previous application failed***
   3. ***User should be able to execute each module separately AND all the modules together***
   4. **Module 1 :**To call application that converts json file to csv
      1. Capture Logs in a file
      2. Check execution status
      3. If failed then add a failure entry into log table, send failure notification and exit
      4. If pass then add a success entry into log table and move to next step
   5. **Module 2 :**Execute command to load the csv files into Hive external/managed table (New partition should be created whenever new file is being loaded with new **filedate)**
      1. Capture Logs in a file
      2. Check execution status
      3. If failed then add a failure entry into log table, send failure notification and exit
      4. If pass then add a success entry into log table and move to next step
   6. **Module 3 :**To call spark application to load the zomato summary table
      1. Capture Logs in a file
      2. Check execution status
      3. If failed then add a failure entry into log table, send failure notification and exit
      4. If pass then add a success entry into log table and send a final notification
   7. Purge last 7 days of logs from the log directory
   8. Write a beeline command and insert log details for each successful and un-successful execution containing below detail
      1. Job id
      2. Job Step
      3. Spark submits that got triggered
      4. Job Start time
      5. Job End time
      6. Job status
2. Once the execution for 3 json file is completed, move these files into archive folder
3. Add new source file into source folder and execute complete workflow again
4. Schedule the job to run daily at 01:00 AM using crontab
5. Execute the complete job and perform the unit testing to check the complete ETL flow and data loading anomalies
6. Document execution statistics **(Start time,End time, Total time taken for execution,No.of executors, No. of Cores, Driver Memory)** along with application URLs
7. Create a unit test case document and reports bugs/observations.
8. Standard Coding guidelines e.g.
   1. Variable Names - Variable names will be all lower case, with individual words separated by an underscore.
   2. For each Function/Procedure add Comments in code
   3. Code Alignment and indentation should be proper
   4. Perform Exception Handling
   5. in-built functions, Code, location, table, database name, etc should be in lower cases
9. Folder Structure on Linux:

* zomato\_etl
  + source
    - json
    - csv
  + archive
  + hive
    - ddl
    - dml
  + spark
    - jars
    - scala
  + script (shell scripts and property files)
  + logs (log\_ddmmyyyy\_hhmm.log)
* zomato\_raw\_files

1. Folder Structure on HDFS

* zomato\_etl\_<username>
  + log
  + zomato\_ext
    - zomato
    - dim\_country