BIG DATA PROJECT
UNIT
TESTING

PREPARED BY

Rahul Anand

29-05-2020

MODULE 1 - JSON TO CSV CONVERTER

A spark application that retrieves a .json file from local file system onto a designated hdfs location. Creates an RDD to read the nested json file by methods of 'explode' and 'write.option()'. Giving an output of the resultant .csv file and creates an hdfs file system configuration to move the file to a designated zomato tables' location and rename it to the required "zomato_*.csv" (* -> filedate). And also copies the converted/renamed csv file into local file system.

No.	Action	Inputs	Expected Output	Actual Output	Result	Observations
1	No json file in source folder	No file	No conversion	No conversion	Pass	Spark session starts, if condition checks for files and session closes
2	Csv file is kept in json folder	Csv file	No conversion	Read error	Fail	Spark.read tries to read csv file
3	Csv file kept in json folder	Csv file	No conversion	No conversion	Pass	If condition checks for file's extension
4	Rename part-* .csv to required nomenclature	part-*.csv location	zomato_(filedate) .csv file in destination location	Corrupted csv file	Fail	Hdfs configuration was not set correctly
5	Rename part-* .csv to required nomenclature	part-*.csv location	zomato_(filedate) .csv file in destination location	zomato_(filedate).cs v file in destination location	Pass	zomato_*.csv file stored in destination location
6	Application run without providing argument	file6.json	ArrayOutOfBound s Exception error	ArrayOutOfBounds Exception error	Fail	Argument is required
7	User quit application using CTRL-Z	CTRL-Z command	Kills App leaves the status_log file as "Running"	Kills App leaves the status_log file as "Running"	Fail	Cannot restart app until status_log file is deleted

MODULE 2 - LOAD DATA INTO ZOMATO TABLE

A spark application that loads all .csv file data into the given zomato table in hive database rahul_database and creates partition for each filedate.

No.	Action	Inputs	Expected Output	Actual Output	Result	Observations
1	No csv file in source folder	No file	No data loaded	No data loaded	Pass	Spark session starts, table created if not exists, if condition checks for files and session closes
2	Csv file present in a different location	No file	No data loaded	No data loaded	Pass	Spark session starts, table created if not exists, if condition checks for files and session closes
3	Partition already exists in source location	Csv file	Data loaded into table	IOException Error	Fail	Partition folder causes exception
4	Partition already exists in source location	Csv file	Data loaded into table	Data loaded into table	Pass	Check for extension and proceed with loading of data
5	Load data of a csv containing 1180 rows	Csv file	Data loaded into table	Data loaded into table with 2 extra rows	Anomaly	Data anomaly caused creation of 2 extra rows by splitting 2 restaurants into 4 rows
6	Csv file with wrong nomenclature	zomato_123.csv	Data loaded into table	Incorrect partition created with data	Fail	Filedate is extracted from the filename, hence here filedate = 123; i.e: incorrect
7	User quit application using CTRL-Z	CTRL-Z command	Kills App leaves the status_log file as “Running”	Kills App leaves the status_log file as “Running”	Fail	Cannot restart app until status_log file is deleted

MODULE 3 - LOAD DATA INTO ZOMATO SUMMARY TABLE FROM ZOMATO TABLE

A spark application that loads all data into the given zomato_summary table in hive database rahul_database, extracting it from the zomato table and removing duplicate rows. Audit column “User ID”, “Create Datetime” added with Partitions on “FileDate” and “Country Name”

No.	Action	Inputs	Expected Output	Actual Output	Result	Observations
1	User gives no choice argument in spark-submit	No arguments	Ask for options in runtime	ArrayOutOfBounds Exception	Fail	Application gets no arguments and runs into an exception while trying to instantiate the same
2	User gives no choice argument in spark-submit	No arguments	Ask for options in runtime	Ask for options in runtime	Pass	Application checks for length of arguments to work with user option choice
3	Incorrect filedate, country name value input	Incorrect filedate, country name	No execution	No execution, Invalid choice prompt shown	Pass	Default case switch occurs to handle invalid inputs
4	Summary table doesn’t exist	Historical arguments	Data loaded into table	Table not exists error	Fail	Table does not exist error, cannot load data into table
5	Summary table doesn’t exist	Historical arguments	Data loaded into table	Data loaded into table	Pass	Table created if not exists
6	Duplicate restaurant rows inserted on every execution	Zomato table data	Distinct data loaded into table	Duplicate values exists	Anomaly	Cannot distinct on “createdatetime” column to remove duplicates
7	User quit application using CTRL-Z	CTRL-Z command	Kills App leaves the status_log file as “Running”	Kills App leaves the status_log file as “Running”	Fail	Cannot restart app until status_log file is deleted

MODULE 1 Shell Script

A shell script that calls spark application “Module1” with an argument that returns the current date if no instance is already running. Handles previously failed instance by sending a mail to user with required information. Sends a mail to user with confirmation of completion of the module execution. Handles the beeline functionality to update zomato_summary_log table.

No.	Action	Inputs	Expected Output	Actual Output	Result	Observations
1	Hdfs commands to move the converted csv files into local file system	Converted csv files	Csv files into local file system	Csv files into local file system	Pass	Application was taking a lot more time than expected to run
2	Hdfs commands to move the converted csv files into local file system	Converted csv files	Csv files into local file system	Csv files into local file system	Pass	The hdfs commands are moved into the spark application from the shell script
3	Updating zomato_summary _log table after every execution	Log details	Log details getting updated in the log table	Null values getting added	Fail	Values with wrong datatype getting inserted in the table
4	Updating zomato_summary _log table after every execution	Log details	Log details getting updated in the log table	Log details getting updated in the log table	Pass	Changed the datatypes of all the columns to string
5	User quit application using CTRL-Z	CTRL-Z command	Kills App leaves the status_log file as “Running”	Kills App leaves the status_log file as “Running”	Fail	Cannot restart app until status_log file is deleted

MODULE 3 Shell Script

A shell script that calls spark application “Module3” with an argument that is passed onto the spark-submit argument to either run the historical update or the criteria based update on zomato_summary table from zomato table. If no argument is passed, the spark application handles the arguments on runtime. On success and failure, a mail is sent and on noticing a failed previous instance, a mail is sent with relevant information.

No.	Action	Inputs	Expected Output	Actual Output	Result	Observations
1	Passing argument to the spark submit command	Wrong Argument	Usage function gets printed specifying the correct choices	Usage function gets printed specifying the correct choices	Pass	Usage function will help the user in choosing the correct option
2	1 is passed as the argument	Filedate and country name	Data getting loaded in the table for the specified filedate and country name	Data getting loaded in the table for the specified filedate and country name	Pass	Desired filedate and country name data can be loaded in the table
3	2 is passed as the argument	No input	All the data gets loaded in the table for every partition	All the data gets loaded in the table for every partition	Pass	All the data gets loaded without having to specify partition values for each insertion
4	No argument is passed	No input	Inputs are taken from the user while the spark app is running	Inputs are taken from the user while the spark app is running	Pass	Useful when the user forgets to pass the argument in the beginning

Wrapper Shell Script

A shell script that calls all the spark applications using the argument “2” (Historical update) on Module3 spark application. This ensures total automation for all the applications and can be set on crontab for automatic execution every day at 1am. It calls all the three individual module shell script in a sequence and ensures all of them are updated with logs and mail to the users information.

No.	Action	Inputs	Expected Output	Actual Output	Result	Observations
1	Passing argument to the spark submit command	Wrong/no Argument	Usage function gets printed specifying the correct choices	Usage function gets printed specifying the correct choices	Pass	Usage function will help the user in choosing the correct option
2	Passing the right option to the wrapper script	Right argument	Modules get executed followed by purge	Modules get executed followed by purge	Anomaly	User is not able to view last 7 days logs including the most latest one
3	Passing the right option to the wrapper script	Right argument	Modules get executed followed by purge	Modules get executed followed by purge	Pass	Included a condition to delete only last 7 days of log not including the current day, leaving 1 days logs behind for reference of the user