*#!/usr/bin/env python3.6  
# -\*- coding: utf-8 -\*-  
  
""" (c) Disney. All rights reserved. """***import** sys  
  
sys.path.insert(0, **'../'**)  
**import** argparse  
**from** input.Config\_Reader **import** ConfigReader  
**from** source.teradata.TeradataConnector **import** TeradataConnector  
**from** monitoringSystem.ExecuteRules **import** ExecuteRules  
  
  
**def** validate\_arguments(args):  
 *"""  
 validate input arguments  
 :args:  
 input argument list* **:return***:  
 None  
 """* valid\_env = (**"test"**, **"prod"**)  
 **if** sum((x == args.env **for** x **in** valid\_env)) < 1:  
 **raise** ValueError(**"invalid input --environment parameter: {0}, "  
 "valid environment arguments are {1}"**.format(args.env, **" "**.join(valid\_env)))  
  
 **if not** args.Table\_Id:  
 args.Table\_Id = **"0"** *# if not args.Parameter:  
 # args.Parameter = "Taxonomy"* **if** int(args.Table\_Id) < 0:  
 **raise** ValueError(**"invalid input --Table can not be less than zero: {0}"**.format(args.env))  
 **return  
  
  
def** get\_arguments(parser):  
 *"""  
 used to parse and validate input arguments* **:param** *parser:  
 parses/validates input arguments i.e. env, Job\_Id, Table\_Id* **:return***:  
 valid arguments  
 """* args = parser.parse\_args()  
 validate\_arguments(args)  
 **return** args  
  
  
**if** \_\_name\_\_ == **"\_\_main\_\_"**:  
 **try**:  
 parser = argparse.ArgumentParser()  
 parser.add\_argument(**"--env"**, help=**"environment(test/prod"**)  
 parser.add\_argument(**"--Job\_Id"**, help=**"Job\_Id"**)  
 parser.add\_argument(**"--Table\_Id"**, help=**"Table\_Id"**)  
 *# parser.add\_argument("--Parameter", help="Parameter")* args = get\_arguments(parser)  
 config = ConfigReader(args.env)  
 validations = TeradataConnector(config=config, job\_id=args.Job\_Id, table\_id=args.Table\_Id)  
 rules = validations.get\_rules  
 ExecuteRules(rules=rules, config=config, job\_id=args.Job\_Id, table\_id=args.Table\_Id)  
 **except** Exception **as** e:  
 **raise** Exception(  
 **"DataQuality Framework has encountered an exception \n{0}"**.format(str(e)))  
 *# finally:  
 # print("data base exception during executing \n{0}\n{1}\n{2}".format(str(e)))  
 # sys.exit(1)*

*#!/usr/bin/env python3.6  
# -\*- coding: utf-8 -\*-  
  
""" (c) Disney. All rights reserved. """***import** json  
**import** resources.encrypt **as** encrypt  
  
  
**class** ConfigReader(object):  
 **def** \_\_init\_\_(self, env):  
 *"""  
 get configurations based on input arguments* **:param** *env:  
 input environment argument  
 """* self.\_\_env = env  
 self.\_\_config = self.set\_config  
 self.\_\_db = self.set\_db  
  
 @property  
 **def** get\_config(self):  
 *"""  
 config getter* **:return***:  
 retun configurations for input environment  
 """* **return** self.\_\_config[self.\_\_env]  
  
 @property  
 **def** get\_db(self):  
 *"""  
 get db config for given environment* **:return***:  
 return db configurations  
 """* **return** self.\_\_db[self.\_\_env]  
  
 @property  
 **def** set\_config(self):  
 *"""  
 setter method to create config dictionary based on input environment* **:param** *self:* **:return***:  
 return config dictionary based on input environment  
 """* \_config\_dict = json.loads(open(**"resources/application.json"**, **'r'**).read())  
 **for** elem **in** \_config\_dict[**"default"**].keys():  
 \_config\_dict[self.\_\_env][elem].update(\_config\_dict[**"default"**][elem])  
 **if** elem == **"teradata"**:  
 \_config\_dict[self.\_\_env][elem][**'password'**] = encrypt.decrypt(\_config\_dict[**"encryption\_key"**].encode(),  
 \_config\_dict[self.\_\_env][elem][**'password'**].encode())  
 **return** \_config\_dict  
  
 @property  
 **def** set\_db(self):  
 *"""  
 setter method to create db config dictionary based on input environment* **:param** *self:* **:return***:  
 return db config dictionary based on input environment  
 """* \_db\_dict = json.loads(open(**"resources/DBmanager.json"**, **'r'**).read())  
 **for** elem **in** \_db\_dict[**"default"**].keys():  
 \_db\_dict[self.\_\_env][elem].update(\_db\_dict[**"default"**][elem])  
 **return** \_db\_dict

*#!/usr/bin/env python3.6  
# -\*- coding: utf-8 -\*-  
  
""" (c) Disney. All rights reserved. """***import** teradatasql  
**import** pandas **as** pd  
**import** json  
  
  
**class** TeradataConnector(object):  
 **def** \_\_init\_\_(self, config, job\_id, table\_id):  
 *"""* **:param** *config:  
 config dictionary from config reader* **:param** *job\_id:  
 input job\_id parameter* **:param** *table\_id:  
 input table\_id parameter  
 """* self.\_\_config = config.get\_config  
 self.\_\_db = config.get\_db  
 self.\_\_job\_id = {**"job\_id"**: int(job\_id)}  
 self.\_\_table\_id = {**"table\_id"**: int(table\_id)}  
 *# self.\_\_parameter = {"parameter": str(parameter)}* self.\_queries = json.loads(open(**"queries/queries.json"**, **'r'**).read())  
 self.\_\_query = **None** self.\_rules = **None** @property  
 **def** get\_rules(self):  
 *"""* **:return***:  
 get rules from teradata  
 """* **try**:  
 **with** teradatasql.connect(\*\*self.\_\_config[**"teradata"**]) **as** connect:  
 parameters = {\*\*self.\_\_db[**"teradata"**], \*\*self.\_\_job\_id, \*\*self.\_\_table\_id}  
 self.\_\_query = self.\_queries[**"queries"**][**"GetRules"**].format(\*\*parameters)  
 self.\_rules = pd.read\_sql(self.\_\_query, connect)  
 **except** Exception **as** e:  
 **raise** Exception(**"Teradata Connection can not be established. \n{0}\n{1}"**.format(self.\_\_query, str(e)))  
 **return** self.\_rules

{  
 **"queries"** : {  
 **"GetRules"**: **"SELECT a.msjobId, b.msTableId, b.msSource, b.msSourceTable, b.msSourceGroup, b.msSourceColumn, b.msSourceFilter, b.msJoin,\n\tb.msDestination, b.msDestinationTable, b.msDestinationGroup, b.msDestinationColumn, b.msDestinationFilter,\n\tc.msruleId, c.msrule, c.msseverity, c.msUpperLT, c.msLowerLT, c.msmean, c.msstandardDeviation, c.percentage\nFROM {workdb}msJob AS a \nINNER JOIN {workdb}msTables AS b \n\tON a.msTableId = b.msTableId AND a.isActive = 1 \n\tand b.isActive = 1 \nINNER JOIN {workdb}msRules AS c on a.msruleId = c.msruleId \n\tAND a.msjobId = c.msjobId AND c.isActive = 1\n\tWHERE a.msjobId = {job\_id} and a.msTableId >= {table\_id}\n\tORDER BY a.msjobId ASC, b.msTableId ASC, c.msruleId ASC;"**,  
 **"OuterAgg"**: **"CREATE TABLE {tempdb}FailedValidations\_{Job\_Id} AS (\n\nSELECT a.col1, a.Source\_mes, b.Destination\_mes, {status} AS STATUS\nFROM (\n\tSELECT {msSourceGroup}, CAST({msrule}({msSourceColumn}) AS DECIMAL(38, 2)) AS Source\_mes\n\tFROM {msSourceTable}\n\tWHERE 1=1 {msSourceFilter}\n\tGROUP BY col1) AS a\nFULL OUTER JOIN (\n\tSELECT {msDestinationGroup}, CAST({msrule}({msDestinationColumn}) AS DECIMAL(38, 2)) AS Destination\_mes\n\tFROM {msDestinationTable}\n\tWHERE 1=1 {msDestinationFilter}\n\tGROUP BY col1) AS b\nON a.col1 = b.col1)\nWITH DATA;"**,  
 **"getFailCount"**: **"SELECT count(1) as failCount from {tempdb}FailedValidations\_{Job\_Id};"**,  
 **"updateRuleControl"**: **"UPDATE {workdb}msControl \nSET msJobCount = msJobCount + {failCount}\nwhere msTable = 'msRuleCounts';"**,  
 **"rule\_counts"**: **"INSERT INTO {workdb}msRuleCounts\n(\nmsRuleCountId,\nmsjobId\n,msruleId\n,msTableId\n,msGroup\n,sourceCount\n,destinationCount\n,description\n,\"timestamp\"\n)\nSELECT ROW\_NUMBER() OVER(ORDER BY 1) + (SELECT msJobCount - {failCount} - 1 FROM {workdb}msControl WHERE msTable = 'msRuleCounts'),{msjobId}, {msruleId}, {msTableId}, col1, Source\_mes, Destination\_mes, status\n, CURRENT\_TIMESTAMP(0) \nFROM {tempdb}FailedValidations\_{Job\_Id};"**,  
 **"status"**: **"WITH status\_{Job\_Id} as (\nSELECT CASE WHEN STATUS = 'error' THEN 1 ELSE 0 END AS errorCount, CASE WHEN STATUS = 'warning' THEN 1 ELSE 0 END AS warningCount\nFROM {tempdb}FailedValidations\_{Job\_Id}\n)\nSELECT CASE WHEN SUM(errorCount) > 0 THEN 'error' \nWHEN SUM(warningCount) > 0 THEN 'warning' ELSE 'good' END AS status FROM status\_{Job\_Id};"**,  
 **"drop\_Validations\_table"**: **"DROP TABLE {tempdb}FailedValidations\_{Job\_Id};"**,  
 **"updateJobControl"**: **"UPDATE {workdb}msControl\nSET msJobCount = msJobCount + 1\nwhere msTable = 'msjobStatus';"**,  
 **"jobStatus"**: **"INSERT INTO {workdb}ms \n(id,msjobId, msTableId ,msruleId, description, status, \"timestamp\")\nSelect msJobCount-1 ,{msjobId}, {msTableId}, {msruleId}, '', {message}, CURRENT\_TIMESTAMP(0)\nfrom {workdb}msControl\nwhere msTable = 'msjobStatus';"**,  
 **"updateErrorControl"**: **"UPDATE {workdb}msControl\nSET msJobCount = msJobCount + 1\nwhere msTable = 'aeErrorLog';"**,  
 **"errorLog"**: **"INSERT INTO {workdb}aeErrorLog \n(errorid,aeJobId, aeruleId, aeTableid, description, errorType, status, isEmailTriggered, \"timestamp\")\nSelect msJobCount-1 ,{msjobId}, {msruleId}, {msTableId}, '', {msseverity}, {message}, 1, CURRENT\_TIMESTAMP(0)\nfrom {workdb}msControl\nwhere msTable = 'aeErrorLog';"**,  
 **"drop\_missing"**: **"DROP TABLE {tempdb}MisssingData\_{Job\_Id};"**,  
 **"LeftMatch"**: **"CREATE TABLE {tempdb}MisssingData\_{Job\_Id} AS (\n\nSELECT a.SourceGroup, a.COL, CASE WHEN {msseverity} = 3 THEN 'error' ELSE 'warning' END AS STATUS\n\nFROM (\n\tSELECT {msColumns} AS SourceGroup, UPPER(TRIM({msSourceColumn})) AS COL\n\n\tFROM {msSourceTable}\n\tWHERE 1=1 {msSourceFilter}\n\tGROUP BY {msColumns}, UPPER(TRIM({msSourceColumn}))) AS a\nLEFT JOIN (\n\tSELECT {msColumns} AS DestinationGroup, UPPER(TRIM({msDestinationColumn})) AS COL\n\tFROM {msDestinationTable}\n\tWHERE 1=1 {msDestinationFilter}\n\tGROUP BY {msColumns}, UPPER(TRIM({msDestinationColumn}))) AS b\nON a.SourceGroup = b.DestinationGroup AND a.col = b.col\nWHERE b.col IS NULL and a.col IS NOT NULL)\nWITH DATA;"**,  
 **"getMissingCount"**: **"Select count(1) as missCount from {tempdb}MisssingData\_{Job\_Id};"**,  
 **"updateMissingControl"**: **"UPDATE {workdb}msControl\nSET msJobCount = msJobCount + {missCount}\nwhere msTable = 'msMissingData';"**,  
 **"insert\_missing"**: **"INSERT INTO {workdb}msMissingData\n(\nmsid\n,msjobId\n,msruleId\n,msTableId\n,\"data\"\n,\"timestamp\"\n,isActive\n)\nSELECT ROW\_NUMBER() OVER(ORDER BY 1) + (SELECT msJobCount - {missCount} -1 FROM {workdb}msControl WHERE msTable = 'msMissingData'),{msjobId}, {msruleId}, {msTableId}, col\n, CURRENT\_TIMESTAMP(0), 1\n\nFROM {tempdb}MisssingData\_{Job\_Id};"**,  
 **"updateRuleMissingControl"**: **"UPDATE {workdb}msControl \nSET msJobCount = msJobCount + 1 \nwhere msTable = 'msRuleCounts'"**,  
 **"missing\_rule\_counts"**: **"INSERT INTO {workdb}msRuleCounts\n(msRuleCountId\n,msjobId\n,msruleId\n,msTableId\n,msGroup\n,sourceCount\n,destinationCount\n,description\n,\"timestamp\"\n)\nSELECT\nROW\_NUMBER() OVER(ORDER BY 1) + (SELECT msJobCount - 1 - 1 FROM {workdb}msControl WHERE msTable = 'msRuleCounts') AS msRuleCountId ,a.\*\nFROM \n(SELECT {msjobId} AS msjobId , {msruleId} AS msruleId, {msTableId} AS msTableId, {msColumns} AS msGroup, COUNT(col) AS sourceCount , 0 AS destinationCount, {message} AS description, CURRENT\_TIMESTAMP(0) AS \"timestamp\"\nFROM {tempdb}MisssingData\_{Job\_Id}\nGROUP BY {msColumns}) a;"**,  
 **"missing\_status"**: **"SELECT \nCASE WHEN (SELECT COUNT(col) AS COl FROM {tempdb}MisssingData\_{Job\_Id}) = 0 THEN 'good'\nWHEN {msseverity} = 3 THEN 'error' ELSE 'warning' END AS status;"**,  
 **"checkOuterAgg"**:**"SELECT COUNT(\*) AS \"COUNT\" FROM DBC.TABLES WHERE TRIM(databasename) || '.' || TRIM(TABLENAME) = '{tempdb}FailedValidations\_{Job\_Id}';"**,  
 **"checkLeftMatch"**: **"SELECT COUNT(\*) AS \"COUNT\" FROM DBC.TABLES WHERE TRIM(databasename) || '.' || TRIM(TABLENAME) = '{tempdb}MisssingData\_{Job\_Id}';"**,  
 **"collectstats1"**: **"COLLECT STATS COLUMN (msRuleCountId) ON {workdb}msRuleCounts;"**,  
 **"collectstats2"**: **"COLLECT STATS ON {workdb}msRuleCounts;"**,  
 **"collectstats3"**: **"COLLECT STATS COLUMN (msid) ON {workdb}msMissingData;"**,  
 **"collectstats4"**: **"COLLECT STATS ON {workdb}msMissingData;"**,  
 **"collectstats5"**: **"COLLECT STATS COLUMN (id) ON {workdb}msJobStatus;"**,  
 **"collectstats6"**: **"COLLECT STATS ON {workdb}msJobStatus;"**,  
 **"collectstats7"**: **"COLLECT STATS COLUMN (errorid) ON {workdb}aeErrorLog;"**,  
 **"collectstats8"**: **"COLLECT STATS ON {workdb}aeErrorLog;"**,  
 **"collectstats9"**: **"COLLECT STATS COLUMN (msTableId) ON {workdb}msTables;"**,  
 **"collectstats10"**: **"COLLECT STATS ON {workdb}msTables;"**,  
 **"Platform\_String\_Validation"**: **"INSERT INTO {workdb}platformStringValidation (review\_date,platform\_string,measure,data\_source,validity\_check) SELECT CURRENT\_TIMESTAMP(0),TRIM(UPPER(campaign\_nm)),CAST(sum(zeroifnull({msSourceColumn})) as BIGINT),'{media}',CASE WHEN REGEXP\_INSTR(campaign\_nm,{msDestinationFilter})>0 THEN 'Valid' ELSE 'Invalid' END FROM {msSourceTable} WHERE {msSourceFilter};"**,  
 **"Valid\_Platform\_String"**: **"INSERT INTO {workdb}validPlatformString (id,review\_date,platform\_string,media\_channel,media\_platform,media\_maui\_code,branded,creative\_maui\_code) SELECT row\_number() over(order by 1) + (SELECT msJobCount - {taxoValidCount} - 1 FROM {workdb}msControl WHERE msTable = 'validPlatformString') as id,\nreview\_date,\nplatform\_string,\nSUBSTR(platform\_string,1,3) as media\_channel,\nSTRTOK(CAST(platform\_string as VARCHAR(320)),'|',2) as media\_platform,\nSTRTOK(CAST(platform\_string as VARCHAR(320)),'|',4) as media\_maui\_code,\nSTRTOK(CAST(platform\_string as VARCHAR(320)),'|',5) as branded,\nSTRTOK(CAST(platform\_string as VARCHAR(320)),'|',6) as creative\_maui\_code\n \nFROM {workdb}platformStringValidation\nWHERE validity\_check = 'Valid'\n;"**,  
 **"Invalid\_Platform\_String"**: **"INSERT INTO {workdb}invalidPlatformString (id,review\_date,platform\_string,Fail\_Reason) SELECT row\_number() over(order by 1) + (SELECT msJobCount - {taxoInvalidCount} - 1 FROM {workdb}msControl WHERE msTable = 'invalidPlatformString') as id,\nreview\_date,\nplatform\_string,\nCASE\n WHEN REGEXP\_INSTR(CAST(platform\_string as VARCHAR(320)),'^({advertising\_channel})',1,1,0,'i') = 0 THEN 'Incorrect Media Channel Prefix'\n WHEN STRTOK(CAST(platform\_string as VARCHAR(320)),'|',2) IS NULL THEN 'Missing Media Platform Code'\n WHEN REGEXP\_INSTR(COALESCE(STRTOK(CAST(platform\_string as VARCHAR(320)),'|',2),0),'^{media\_platform\_code}',1,1,0,'i') = 0 THEN 'Incorrect Media Platform Code'\n WHEN STRTOK(CAST(platform\_string as VARCHAR(320)),'|',3) IS NULL THEN 'Missing WBS'\n WHEN STRTOK(CAST(platform\_string as VARCHAR(320)),'|',4) IS NULL THEN 'Missing MAUI Code Code'\n WHEN REGEXP\_INSTR(COALESCE(STRTOK(CAST(platform\_string as VARCHAR(320)),'|',4),0),'^{MAUI\_code}',1,1,0,'i') = 0 THEN 'Incorrect MAUI Code Match'\n WHEN STRTOK(CAST(platform\_string as VARCHAR(320)),'|',5) IS NULL THEN 'Missing Brand Code'\n WHEN REGEXP\_INSTR(COALESCE(STRTOK(CAST(platform\_string as VARCHAR(320)),'|',5),0),'^({brand\_code})',1,1,0,'i') = 0 THEN 'Incorrect Brand Code'\n WHEN REGEXP\_INSTR(COALESCE(STRTOK('ZZ\_KNC-FY19\_ABD\_INS\_DOM\_APS\_LGN\_ABDG\_MONTANA-BMM-OLD|NB|B|4196700.AD.AM.01.01|MFGXEIV','|',6),0),'^({brand\_code})',1,1,0,'i') = 0 THEN 'Missing Creative Maui Code'\n ELSE 'SOMETHING ELSE'\n END as Fail\_Reason\nFROM {workdb}platformStringValidation\nWHERE validity\_check = 'Invalid'\n;"**,  
 **"MAI\_Taxonomy\_New\_Daily\_Strings"**: **"INSERT INTO {workdb}MAI\_Taxonomy\_New\_Daily\_Strings (id,review\_date,platform\_string,measure,media\_channel,media\_platform,data\_source,validity\_check,validity\_message,master\_append) SELECT\n ROW\_NUMBER() OVER(ORDER BY 1) + (SELECT msJobCount - {taxoStringCount} - 1 FROM {workdb}msControl WHERE msTable = 'platformStringValidation') as id ,\n P.review\_date ,\n P.platform\_string ,\n P.measure ,\n V.media\_channel ,\n V.media\_platform ,\n P.data\_source ,\n P.validity\_check ,\n I.Fail\_Reason as validity\_message ,\n '' as master\_append \n \nFROM {workdb}platformStringValidation P\nLEFT JOIN {workdb}validPlatformString V \nON P.platform\_string = V.platform\_string\nLEFT JOIN {workdb}invalidPlatformString I\nON P.platform\_string = I.platform\_string;\n"**,  
 **"media\_source\_catch"**: **"SELECT Media\_Source as media\_source from {workdb}msTables where msTableId={msTableId}"**,  
 **"truncate\_platformStringValidation"**:**"DELETE {workdb}platformStringValidation;"**,  
 **"get\_taxonomy\_string\_count"**: **"SELECT count(1) as taxonomy\_count from {workdb}platformStringValidation;"**,  
 **"get\_taxonomy\_valid\_string\_count"**: **"SELECT count(1) as valid\_taxonomy\_count from {workdb}platformStringValidation where validity\_check='Valid';"**,  
 **"get\_taxonomy\_invalid\_string\_count"**: **"SELECT count(1) as invalid\_taxonomy\_count from {workdb}platformStringValidation where validity\_check='Invalid';"**,  
 **"updateTaxoControl"**: **"UPDATE {workdb}msControl \nSET msJobCount = msJobCount + {taxoStringCount}\nwhere msTable = 'platformStringValidation';"**,  
 **"updateValidTaxoControl"**: **"UPDATE {workdb}msControl \nSET msJobCount = msJobCount + {taxoValidCount}\nwhere msTable = 'validPlatformString';"**,  
 **"updateInvalidTaxoControl"**: **"UPDATE {workdb}msControl \nSET msJobCount = msJobCount + {taxoInvalidCount}\nwhere msTable = 'invalidPlatformString';"**,  
 **"updateTaxoJobStatusControl"**: **"UPDATE {workdb}msControl \nSET msJobCount = msJobCount + 1\nwhere msTable = 'taxoJobStatus';"**,  
 **"taxoJobStatus"**: **"INSERT INTO {workdb}taxoJobStatus (id,msjobId,msTableId,msruleId,review\_date,data\_source,valid\_platformstring\_count,invalid\_platformstring\_count)\n VALUES ((SELECT msJobCount - 1 FROM {workdb}msControl WHERE msTable = 'taxoJobStatus'), {msjobId}, {msTableId}, {msruleId}, CURRENT\_TIMESTAMP(0),'{media}', {taxoValidCount} , {taxoInvalidCount});"** },  
 **"compare"**: {  
 **"thresholds"**: **""**,  
 **"mean"**: **""**,  
 **"percentage"**: **"CASE WHEN Source\_mes = 0 and Destination\_mes = 0 THEN 'good' WHEN Source\_mes <> 0 AND ABS(100.0\*(Destination\_mes - Source\_mes) / Source\_mes) <= {percentage} THEN 'good' \nWHEN {msseverity} = 3 THEN 'error' \nELSE 'warning' END"**,  
 **"exact"**: **"CASE WHEN a.Source\_mes = b.Destination\_mes THEN 'good' \nWHEN {msseverity} = 3 THEN 'error' \nELSE 'warning' END"** },  
 **"alert\_engine"**: {  
 **"OuterAgg"**: **"SELECT TOP 10 \* FROM {tempdb}FailedValidations\_{Job\_Id} WHERE STATUS <> 'good';"**,  
 **"LeftMatch"**: **"SELECT TOP 10 \* FROM {tempdb}MisssingData\_{Job\_Id} WHERE STATUS <> 'good';"**,  
 **"get\_Contact\_list"**: **"SELECT a.emailId \n\tFROM {workdb}aeContact AS a\n\tINNER JOIN {workdb}aeContactRule AS b \n\t\tON a.contactId = b.mscontactId \n\tINNER JOIN {workdb}msRules AS c\n\t\tON b.msruleId = c.msruleId AND b.msruleId = {msruleId} AND a.contactId IN (7,8,9,11);"**,  
 **"Daily\_Summary"**: **"SELECT msjobId,msruleId,msTableId,msSourceTable,msSourceColumn,msDestinationTable,msDestinationColumn,msRule,JobStatus\nFROM \n(SELECT msjobId,msruleId,msTableId,msSourceTable,msSourceColumn,msDestinationTable,msDestinationColumn,msRule,JobStatus,\nROW\_NUMBER() OVER (PARTITION BY msjobId,msruleId,msTableId ORDER BY ErrorLogTime desc) AS row\_id\nFROM DV\_CIMAPROD\_VMDB.dq\_vmReporting WHERE CAST(JobStatusTime AS DATE format'YYYY-MM-DD') = '{0}' \nQUALIFY row\_id = 1\n) AS a \nORDER BY CASE \nWHEN JobStatus= 'error' THEN 1\nWHEN JobStatus= 'warning' THEN 2\nELSE 3\nEND ASC\n, msTableId ASC, msjobId DESC, msruleId ASC;"** }  
}

{  
 **"default"**: {  
 **"teradata"**: {  
 }  
 },  
 **"test"**: {  
 **"teradata"**: {  
 **"workdb"**: **"DV\_CIMATEMP\_DB.neeld009\_qa\_"**,  
 **"tempdb"**: **"DV\_CIMATEMP\_DB.neeld009\_qa\_"** }  
 },  
 **"QA"**: {  
 **"teradata"**: {  
 **"workdb"**: **"DV\_CIMAUSER\_DB.rajp007\_qa\_"**,  
 **"tempdb"**: **"DV\_CIMATEMP\_DB.rajp007\_qa\_"** }  
 },  
 **"prod"**: {  
 **"teradata"**: {  
 **"workdb"**: **"DV\_CIMAPROD\_DB.DQ\_"**,  
 **"tempdb"**: **"DV\_CIMATEMP\_DB.DQ\_"** }  
 }  
}

*#!/usr/bin/env python3.6  
# -\*- coding: utf-8 -\*-  
  
""" (c) Disney. All rights reserved. """***import** teradatasql  
**import** pandas **as** pd  
**import** json  
**from** alertEngine.AlertEngine **import** AlertEngine  
  
  
**def** filter\_rules(x, filter):  
 *"""* **:param** *x:  
 input rules* **:param** *filter:  
 filter rules based on the rule type being executed* **:return***:  
 dataframe after filtering  
 """* mask = (x[**"msSource"**] == **"teradata"**) & (x[**"msDestination"**] == **"teradata"**) & (x[**"msJoin"**] == filter)  
 rules = x[mask]  
 **return** rules.copy(deep=**True**)  
  
  
**def** generate\_mscolumn\_fields(x):  
 *"""* **:param** *x:  
 msSourceGroup* **:return***:  
 create msColumns based on input msSourceGroup  
 """* **if** x **is None**:  
 **return " 1"** n = len(x.split(**","**))  
 columns = [**"col{0}"**.format(i) **for** i **in** range(1, n + 1)]  
 **return " ,"**.join(columns), \  
 **" AND "**.join(**"a.$ = b.$"**.replace(**"$"**, col) **for** col **in** columns)  
  
  
**def** generate\_compare(rule):  
 *"""* **:param** *rule:  
 input rules from db* **:return***:  
 add new fields for comparision  
 """* \_compare = json.loads(open(**"queries/queries.json"**, **'r'**).read())[**"compare"**]  
 **if** rule[**"msUpperLT"**] != 0 **and** rule[**"msLowerLT"**] != 0:  
 **return** \_compare[**"thresholds"**].format(\*\*rule)  
 **if** rule[**"msmean"**] != 0 **and** rule[**"msstandardDeviation"**] != 0:  
 **return** \_compare[**"mean"**].format(\*\*rule)  
 **if** rule[**"percentage"**] != 0:  
 **return** \_compare[**"percentage"**].format(\*\*rule)  
 **return** \_compare[**"exact"**].format(\*\*rule)  
  
  
**def** populate\_status\_tables(cursor, connect, queries, rule, status):  
 *"""* **:param** *cursor:  
 used to connect to db* **:param** *connect:  
 used to connect to db* **:param** *queries:  
 queries from json* **:param** *rule:  
 input rule* **:param** *status:  
 status of rule comparision* **:return***:  
 None  
 """* **if** list(status[**'status'**])[0] == **"good"**:  
 query = queries[**"updateJobControl"**].format(\*\*rule)  
 cursor.execute(query)  
 query = queries[**"jobStatus"**].format(\*\*rule)  
 cursor.execute(query)  
  
 **else**:  
 AlertEngine(rule=rule, connect=connect)  
 query = queries[**"updateJobControl"**].format(\*\*rule)  
 cursor.execute(query)  
 query = queries[**"jobStatus"**].format(\*\*rule)  
 cursor.execute(query)  
 query = queries[**"updateErrorControl"**].format(\*\*rule)  
 cursor.execute(query)  
 query = queries[**"errorLog"**].format(\*\*rule)  
 cursor.execute(query)  
 **return  
  
  
class** ExecuteRules(object):  
 **def** \_\_init\_\_(self, rules, config, job\_id, table\_id):  
 *"""* **:param** *rule:* **:param** *config:* **:param** *job\_id:* **:param** *table\_id:  
 """* self.\_rules = rules  
 self.\_cur\_rules = **None** self.\_query = **None** self.\_queries = json.loads(open(**"queries/queries.json"**, **'r'**).read())  
 self.\_\_config = config.get\_config  
 self.\_\_db = config.get\_db  
 self.\_\_job\_id = job\_id  
 self.\_\_table\_id = int(table\_id)  
 *# self.\_\_parameter = str(parameter)* self.\_status = **None** self.\_agg\_temp\_delete = **None** self.\_match\_temp\_delete = **None** *#self.collect\_stats* self.execute\_teradata\_agg\_rules  
 self.execute\_teradata\_match\_rules  
 self.execute\_taxonomy\_rules  
 *#self.collect\_stats* @property  
 **def** execute\_taxonomy\_rules(self):  
 *"""* **:return***:  
 execute\_teradata\_taxonomy\_rules  
 """  
 # temp\_filter=self.\_\_parameter* self.\_cur\_rules = filter\_rules(self.\_rules, filter=**"Taxonomy"**)  
 print(self.\_cur\_rules)  
 **if** self.\_cur\_rules.empty **is not True**:  
 self.add\_columns  
  
 **for** k, rule **in** self.\_cur\_rules.iterrows():  
  
 **try**:  
 **with** teradatasql.connect(\*\*self.\_\_config[**"teradata"**]) **as** connect:  
 **with** connect.cursor() **as** cursor:  
  
 rule[**'media\_source'**] = str(list(pd.read\_sql(self.\_queries[**"queries"**][**"media\_source\_catch"**]  
 .format(\*\*rule), connect)[**'media\_source'**])[0])  
 **if**(rule[**'media\_source'**]==**"BING"**): *#discuss with keith* rule[**'media'**]=**"Bing Ad Center"** *#parameterise all the below values from a table* rule[**'media\_platform\_code'**]=**"B"** rule[**'advertising\_channel'**]=**"KNC|KAC"** rule[**'MAUI\_code'**]=**"M[^\|]+"** rule[**'brand\_code'**]=**"BR|NB"  
 else**:  
 print(**"NOT BING!!"**)  
  
 print(**"Appending msSourceFilter with msSourceGroup"**)  
 print(**"Before"**,rule.msSourceFilter)  
 rule.msSourceFilter=rule.msSourceFilter.format(\*\*rule)  
 print(**"After"**,rule.msSourceFilter)  
  
 cnt=int((pd.read\_sql(self.\_queries[**"queries"**][**"get\_taxonomy\_string\_count"**]  
 .format(\*\*rule), connect)[**'taxonomy\_count'**])[0])  
 print(**"Truncating platformStringValidation table with "**,cnt,**"rows"**)  
 self.\_query = self.\_queries[**"queries"**][**"truncate\_platformStringValidation"**].format(\*\*rule)  
 cursor.execute(self.\_query)  
  
 print(**"Populating platformValidationString table with latest strings!"**)  
 self.\_query = self.\_queries[**"queries"**][**"Platform\_String\_Validation"**].format(\*\*rule)  
 print(self.\_query)  
 cursor.execute(self.\_query)  
 self.\_query = **None** print(**"Updating counts in msControl"**)  
 rule[**'taxoStringCount'**] = **None** rule[**'taxoStringCount'**] = int(list(pd.read\_sql(self.\_queries[**"queries"**][**"get\_taxonomy\_string\_count"**]  
 .format(\*\*rule), connect)[**'taxonomy\_count'**])[0])  
 self.\_query = self.\_queries[**"queries"**][**"updateTaxoControl"**].format(\*\*rule)  
 cursor.execute(self.\_query)  
 self.\_query = **None** rule[**'taxoValidCount'**] = **None** rule[**'taxoValidCount'**] = int(list(pd.read\_sql(self.\_queries[**"queries"**][**"get\_taxonomy\_valid\_string\_count"**]  
 .format(\*\*rule), connect)[**'valid\_taxonomy\_count'**])[0])  
 self.\_query = self.\_queries[**"queries"**][**"updateValidTaxoControl"**].format(\*\*rule)  
 cursor.execute(self.\_query)  
 self.\_query = **None** rule[**'taxoInvalidCount'**] = **None** rule[**'taxoInvalidCount'**] = int(list(pd.read\_sql(self.\_queries[**"queries"**][**"get\_taxonomy\_invalid\_string\_count"**]  
 .format(\*\*rule), connect)[**'invalid\_taxonomy\_count'**])[0])  
 self.\_query = self.\_queries[**"queries"**][**"updateInvalidTaxoControl"**].format(\*\*rule)  
 cursor.execute(self.\_query)  
 self.\_query = **None** print(rule[**'taxoInvalidCount'**] , rule[**'taxoValidCount'**],rule[**'taxoStringCount'**])  
  
 **for** i **in** [**"Valid\_Platform\_String"**,**"Invalid\_Platform\_String"**,**"MAI\_Taxonomy\_New\_Daily\_Strings"**]:  
 print(i)  
 self.\_query = **None** self.\_query = self.\_queries[**"queries"**][i].format(\*\*rule)  
 cursor.execute(self.\_query)  
 self.\_query = **None** self.\_query = self.\_queries[**"queries"**][**"updateTaxoJobStatusControl"**].format(\*\*rule)  
 cursor.execute(self.\_query)  
 self.\_query = **None** self.\_query = self.\_queries[**"queries"**][**"taxoJobStatus"**].format(\*\*rule)  
 cursor.execute(self.\_query)  
 self.\_query = **None  
  
 except** (teradatasql.OperationalError, teradatasql.DatabaseError) **as** e:  
 **raise** Exception(  
 **"data base exception during executing \n{0}\n{1}\n{2}"**.format(self.\_query, rule, str(e)))  
 **except** Exception **as** e:  
 **raise** Exception(  
 **"data base exception during executing \n{0}\n{1}\n{2}"**.format(self.\_query, rule, str(e)))  
 **return** @property  
 **def** add\_columns(self):  
 *'''* **:return***:  
 add new columns needed for dynamic query creation  
 '''* self.\_cur\_rules[**"msColumns"**] = self.\_cur\_rules[**"msSourceGroup"**].map(generate\_mscolumn\_fields)  
 self.\_cur\_rules[**"tempdb"**] = self.\_\_db[**"teradata"**][**"tempdb"**]  
 self.\_cur\_rules[**"workdb"**] = self.\_\_db[**"teradata"**][**"workdb"**]  
 self.\_cur\_rules[**"Job\_Id"**] = self.\_\_job\_id  
 self.\_cur\_rules[**"Table\_Id"**] = self.\_\_table\_id  
 self.\_cur\_rules[**"status"**] = self.\_cur\_rules.apply(generate\_compare, axis=1)  
 **return** @property  
 **def** collect\_stats(self):  
 *"""* **:return***:  
 """* **with** teradatasql.connect(\*\*self.\_\_config[**"teradata"**]) **as** connect:  
 **with** connect.cursor() **as** cursor:  
 **for** query **in** (self.\_queries[**"queries"**][**"collectstats1"**], self.\_queries[**"queries"**][**"collectstats2"**]  
 , self.\_queries[**"queries"**][**"collectstats3"**], self.\_queries[**"queries"**][**"collectstats4"**]  
 , self.\_queries[**"queries"**][**"collectstats5"**], self.\_queries[**"queries"**][**"collectstats6"**]  
 , self.\_queries[**"queries"**][**"collectstats7"**], self.\_queries[**"queries"**][**"collectstats8"**]  
 , self.\_queries[**"queries"**][**"collectstats9"**], self.\_queries[**"queries"**][**"collectstats10"**]):  
 query = query.format(workdb=self.\_\_db[**"teradata"**][**"workdb"**])  
 cursor.execute(query)  
  
 **return** @property  
 **def** execute\_teradata\_agg\_rules(self):  
 *"""* **:return***:  
 execute\_teradata\_aggregate\_rules  
 """* self.\_cur\_rules = filter\_rules(self.\_rules, filter=**"OuterAgg"**)  
 **if** self.\_cur\_rules.empty **is not True**:  
 self.add\_columns  
  
 **for** k, rule **in** self.\_cur\_rules.iterrows():  
 **try**:  
 **with** teradatasql.connect(\*\*self.\_\_config[**"teradata"**]) **as** connect:  
 **with** connect.cursor() **as** cursor:  
  
 **if not** self.\_agg\_temp\_delete:  
 self.\_query = self.\_queries[**"queries"**][**"checkOuterAgg"**].format(\*\*rule)  
 **if** list(pd.read\_sql(self.\_query, connect)[**"COUNT"**])[0]:  
 self.\_query = self.\_queries[**"queries"**][**"drop\_Validations\_table"**].format(\*\*rule)  
 cursor.execute(self.\_query)  
 **else**:  
 self.\_agg\_temp\_delete = **True** self.\_query = self.\_queries[**"queries"**][rule[**"msJoin"**]].format(\*\*rule)  
 cursor.execute(self.\_query)  
 self.\_query = **None** rule[**'failCount'**] = int(list(pd.read\_sql(self.\_queries[**"queries"**][**"getFailCount"**]  
 .format(\*\*rule), connect)[**'failCount'**])[0])  
 self.\_query = self.\_queries[**"queries"**][**"updateRuleControl"**].format(\*\*rule)  
 cursor.execute(self.\_query)  
 self.\_query = self.\_queries[**"queries"**][**"rule\_counts"**].format(\*\*rule)  
 cursor.execute(self.\_query)  
 self.\_query = **None** self.\_status = pd.read\_sql(self.\_queries[**"queries"**][**"status"**].format(\*\*rule), connect)  
 rule[**"message"**] = **"'{0}'"**.format(list(self.\_status[**'status'**])[0])  
 populate\_status\_tables(cursor, connect, self.\_queries[**"queries"**], rule, self.\_status)  
 self.\_query = self.\_queries[**"queries"**][**"drop\_Validations\_table"**].format(\*\*rule)  
 cursor.execute(self.\_query)  
 self.\_query = **None  
 except** (teradatasql.OperationalError, teradatasql.DatabaseError) **as** e:  
 **raise** Exception(  
 **"data base exception during executing \n{0}\n{1}\n{2}"**.format(self.\_query, rule, str(e)))  
 **except** Exception **as** e:  
 **raise** Exception(  
 **"data base exception during executing \n{0}\n{1}\n{2}"**.format(self.\_query, rule, str(e)))  
 **return** @property  
 **def** execute\_teradata\_match\_rules(self):  
 *"""* **:return***:  
 execute\_teradata\_match\_rules  
 """* self.\_cur\_rules = filter\_rules(self.\_rules, filter=**"LeftMatch"**)  
  
 **if** self.\_cur\_rules.empty **is not True**:  
 self.add\_columns  
  
 **for** k, rule **in** self.\_cur\_rules.iterrows():  
 **try**:  
 **with** teradatasql.connect(\*\*self.\_\_config[**"teradata"**]) **as** connect:  
 **with** connect.cursor() **as** cursor:  
  
 **if not** self.\_match\_temp\_delete:  
 self.\_query = self.\_queries[**"queries"**][**"checkLeftMatch"**].format(\*\*rule)  
 **if** list(pd.read\_sql(self.\_query, connect)[**"COUNT"**])[0]:  
 self.\_query = self.\_queries[**"queries"**][**"drop\_missing"**].format(\*\*rule)  
 cursor.execute(self.\_query)  
 **else**:  
 self.\_match\_temp\_delete = **True  
  
 for** query **in** (rule[**"msJoin"**], **"getMissingCount"**, **"updateMissingControl"**,  
 **"insert\_missing"**, **"updateRuleMissingControl"**, **"missing\_rule\_counts"**):  
 **if** query == **"getMissingCount"**:  
 rule[**'missCount'**] = int(list(pd.read\_sql(self.\_queries[**"queries"**][query]  
 .format(\*\*rule), connect)[**'missCount'**])[0])  
 self.\_status = pd.read\_sql(self.\_queries[**"queries"**][**"missing\_status"**].format(\*\*rule),  
 connect)  
 rule[**"message"**] = **"'{0}'"**.format(list(self.\_status[**'status'**])[0])  
 **else**:  
 self.\_query = self.\_queries[**"queries"**][query].format(\*\*rule)  
 cursor.execute(self.\_query)  
  
 populate\_status\_tables(cursor, connect, self.\_queries[**"queries"**], rule, self.\_status)  
  
 self.\_query = self.\_queries[**"queries"**][**"drop\_missing"**].format(\*\*rule)  
 cursor.execute(self.\_query)  
 **except** (teradatasql.OperationalError, teradatasql.DatabaseError) **as** e:  
 **raise** Exception(  
 **"data base exception during executing \n{0}\n{1}\n{2}"**.format(self.\_query, rule, str(e)))  
 **except** Exception **as** e:  
 **raise** Exception(  
 **"data base exception during executing \n{0}\n{1}\n{2}"**.format(self.\_query, rule, str(e)))  
 **return**