Data Engineering Solution

Requirement 1:

"There is file outliers.txt. It contains 690 records with 14 features. Load the file into a database of your choice and then detect outliers in the data set. For the latter, you are allowed to use any one of the following tools: R, Python, SQL. If you want to use something else, please contact us."

Solution:

Python is used here to develop the scripts to read data from the text file(Outliers.txt) loaded into MySQL server.

First, to connect MySQL server, python MySQL Connector is used.

```
15  sql_conn = mysql.connector.connect(user='root', password='Infosys1',host='localhost',database='srdb')
16  if sql_conn.is_connected():
17          db_Info = sql_conn.get_server_info()
18          print("Connected to MySQL database... MySQL Server version on ",db_Info)
```

Now, query the "outliers" table and store the result in a Pandas DataFrame ("df").

Data is converted to numeric format to fetch the basic statistics (Count, Mean, Standard Deviation, Median, 1st Quartile, 3rd Quartile, Min value, and Max Value) of the dataframe.

```
20  query = "SELECT * FROM outliers"
21  df = pd.read_sql(query, sql_conn)
22  df[df.columns]=df[df.columns].apply(pd.to_numeric,errors="coerce")
23  df.describe()
```

Check the column having the data-points distributed across a wide range. This is the columns with max outliers.

In this case, Column 14 - "col14" has the maximum outliers.

Determine the 1st Quartile, Median and 3rd Quartile values for "col14" to calculate the Inter-Quartile Range(IQR).

```
27  q75,median, q25 = np.percentile(df["col14"], [75,50,25])
28  iqr = q75 - q25
29  print(iqr)
30  print(q75)
31  print(q25)
32  print("Median:" , median)
```

Data-points having value more than One and Half times the IQR from the 1st Q or 3rd Q are considered to be outliers.

```
outlier = [] ;
i = 0
for a in df["col14"]:
    if (a < (q25 - 1.5 * iqr)):
        outlier.append(a)
    elif (a > (q75 + 1.5 * iqr)):
        outlier.append(a)
outliers = np.array(outlier)
```

Once the outliers are detected, store the result in a text file.

```
np.savetxt("solutions/outlierDetected.txt", outliers , fmt="%d")
```

Below are the files to support the above requirement.

Requirement 2:

“

File complains.json is a json file of consumer complains.

a. Write a script to load the file into a relational database of your choice. You are not allowed to use third party add-on or software packages.
b. Write a script to load the file to Hive. You are not allowed to use third party add-on or software packages. If you don’t have Hive, you should install a virtual image on your laptop.

”

Solution 2 – (a) :

Python is used here to develop the scripts to load data from any JSON file to any RDBMS (MySQL here)

First, setup the connection to the MySQL server

```
##MySQL Connection Configuration
sql_conn = mysql.connector.connect(user='root', password='Infosys1',host='localhost',database='srdb')
if sql_conn.is_connected():
        db_Info = sql_conn.get_server_info()
        print("Connected to MySQL database... MySQL Server version on ",db_Info)

##Cursor to execute SQL queries
cur = sql conn.cursor()
```

Now, as argument the JSON file path is provided. This is to create a list with the JSON documents as elements.

```
21  ##Parse the Arguments
22  filename = sys.argv[1]
23
24  ##Read the JSON file passed in argument with --file tag
25  with open(filename) as complain:
26      data=json.loads(complain.read())
27
```

Then, the python dictionary functions are used to extract the key values from the documents to be used as Column Names in the Table.

Post that, the Table Creation Query is executed using the cursor connecting to MySQL server.

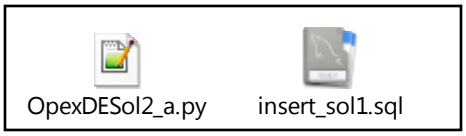
```
28  keys = []
29  for row in data:
30      for key in row.keys():
31          if key not in keys:
32              keys.append(key)
33  qry1 = """CREATE TABLE complains({0});""".format(", ".join(map(lambda key: "{0} TEXT".format(key), keys)))
34
35  cur.execute(qry1)
36  print("Table Created!!")
```

Also, a different approach has been adopted below to insert the data from JSON to the above created table in MySQL.

The insert statements are written to a ".sql" file which can be run to insert the data to the server.

```
38  f = open("../solutions/sol2/insert_sol1.sql", "a")
39  f.write("USE srdb;" + '\n')
40  for row in data:
41      columns = ', '.join("" + str(x).replace('/', '_') + "" for x in row.keys())
42      values = ', '.join("'" + str(x).replace("'", '') + "'" for x in row.values())
43      sql = "INSERT INTO %s ( %s ) VALUES ( %s );" % ('complains', columns, values)
44      f = open("../solutions/sol2/insert_sol1.sql", "a")
45      f.write(sql + '\n')
```

Below are the py and sql script.



Output (data loaded to table in MySQL server):

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id	name	averageRating	createdAt	description	displayType	downloac
s6ew-h6mp	Consumer Complaints	0	1431450318	Each week we send thousands of consumers co…	table	8232
25ei-6bcr	Credit Card Complaints	0	1337199939	NULL	table	5182
wkue-ycpk	Beta Consumer Complaints Visualization	0	1434123722	Each week we send thousands of consum…	data_lens	9
nsyy-je5y	Consumer Complaints with Consumer Complaint …	0	1433252261	Each week we send thousands of consumers co…	fatrow	1959
gme7-gkkr	Survey of Credit Card Plans	0	1347900842	The CFPB maintains historical data from the sem…	table	714
r963-hvsf	College Credit Card Agreements	0	1350323610	As required by the Credit CARD Act of 2009, th…	table	774
gfmg-6ppu	Mortgage Complaints with Consumer Complaint …	0	1433953219	Each week we send thousands of consumers co…	fatrow	309
ytxv-uppu	Bank Account or Service Complaints with a Cons…	0	1434384433	Each week we send thousands of consumers co…	fatrow	286
g5qz-smft	Mortgage Complaints	0	1433952949	Each week we send thousands of consumers co…	table	242
ckyu-ku28	Debt Collection Complaints	0	1434384833	Each week we send thousands of consumers co…	table	230

Solution 2 – (b):

Python is used here to develop the scripts to load data from any JSON file to any Hive server

The idea here is to create the hive script file (with extension ".hql") to create the table in hive using the JSONSerDe and load the data from the provided JSON file (or any other file).

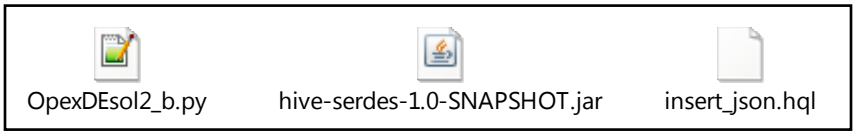
```
with open("complains.json") as complain:
    data=json.loads(complain.read())

keys = []
for row in data:
    for key in row.keys():
        if key not in keys:
            keys.append(key)
qry1 = """CREATE TABLE opex_json_hive({0}) ROW FORMAT SERDE 'com.cloudera.hive.serde.JSONSerDe';"""
                    .format(", ".join(map(lambda key: "{0} STRING".format(key), keys)))
f = open("solutions/sol2/insert_json.hql", "a")
f.write("USE opex;" + '\n')
f.write("add jar /hadoop/smruti/Hive/hive-serdes-1.0-SNAPSHOT.jar;" + '\n')
f.write(qry1 + '\n')
f.write("load data local inpath '/hadoop/smruti/Hive/complains.json' overwrite into table opex_json_hive;\n")
```

This .hql file can then be executed at any hive server using the below command.

```
beeline -u jdbc:hive2://serverName:10000/ -f /path/to/file.hql
```

Below is the py script, JSONSerde jar to be added, and the output hql script.



Requirement 3:

"Files arcana.txt, spor.txt and web.txt are ascii text file. Write a script in R or Python that creates the document-term matrix based on paragraphs in these files (a paragraph is a document – row – in the matrix; a paragraph in a document is terminated by newline "\n"). The script needs to store the matrix in a file in the sparse format. Name the rows as "para0, para1, …" and a column by the actual word it represents."

Solution:

Python is used here to develop the scripts to read text from any file and store its Document-Term Matrix in an output path.

The filename is passed as first argument and the output path as the second argument.

First, the file passed as argument is opened to read the lines and store in a Python List.

```
#Read the filename from argument
filename = sys.argv[1]

#Open the file and read the documents terminated by newline "\n"
lines = open(filename, "r").read().split('\n')
#lines = [line.rstrip('\n') for line in lines]
lines = [x.strip() for x in lines]
```

The Scikit-Learn CountVectorizer provides a simple way to tokenize a collection of text documents and create a matrix of counts of the words in the document in a sparse format.

```
#CountVectorizer is used to Convert a collection of documents to a matrix of counts
vec = CountVectorizer()
X = vec.fit_transform(lines)
df = pd.DataFrame(X.toarray(), columns=vec.get_feature_names())
```

Next, the index values were renamed as per requirement : para0,para1,… by using the dataframe rename function.

```
##Rename the index name to "para0", "para1"...
for i in range(len(lines)):
    df.rename(index={i: "para" + str(i)}, inplace=True)
```

Finally, the output sparse matrix was stored in the output path (specified as 2nd argument).

```
##store the output to a text file(Tab separated)
outputFilepath= sys.argv[2] + filename
df.to_csv(outputFilepath, sep='\t')
```

Below is the Py script and output results in an HTML format.

***