Systecon

Developer Interview Task – Back-End

Introduction

The following set of sample tasks given in this document reflect a limited subset of the duties as a developer at Systecon North America. In the role as a developer you are expected (but not limited) to:

- Develop tools and methods to solve client specific problems.
- Develop tools and methods that integrate with Opus Suite
- Design and implement databases and other back end data stores
- Be an advanced user of the Opus Suite
- Develop front ends and UIs
- Process, analyze, and build models from large amounts of data
- Collect user requirements and transform these into specifications

Setup

For the practical tasks, you are free to select the language/tool you feel familiar with and that is appropriate for the task at hand. Feel free to use any resources available, including Google, StackOverflow, or any other reference material.

Task 1: Maintenance Data Analysis

The maintenance data lists time of failures for ten components installed in eight different systems. For each failure, the duration of the resulting repair time is also recorded. Your task is to analyze this data and calculate some metrics.

Data set

The data can be found in the file *maintenance_data.csv* in sub-folder MaintenanceData. The csv file consists of the following fields:

Field	Description
Item	The name of the component
System	The system in which the component is installed
Failure observation	The date of the failure observation.
Repair time	The time to repair the failure (in hours)

Data was recorded between 1/1/1990 and 12/31/2015.

Expected the result

For each question, a short answer is expected. In addition, you must be able to provide the code/script that generated the answer. The essence of the task is to look at and discuss the code.

Questions

1. Provide some high level statistics on the data set such as number of observations, and minimum and maximum values. Provide overall values (across the entire data set), values for each item, and values for each item and system combination.

2. Calculate the mean repair time for each item.

3. Calculate the failure rate (failures per hour) for each item and system combination.

4. Generate failure interarrival times (time between failures) for each item and system. What's the expected value of the interarrival times? Does it correspond to the failure rate?

5. Create a histogram of the interarrival times for a single item. Can you draw any conclusion on the underlying distribution?

Task 2: Parse LSA records

An LSA (Logistics Support Analysis) record contains data for a system and the parts installed in the system. Typical data is an identifier, the name of the part, failure rate, price, and where it is installed. LSA records come in a *fixed length text file*, where each record may span several rows. The schema (column names and width of each data element) is defined together with the task below (also as a csv file in the data set), and the task is to parse the content to a csv file, where each system/part is on a single row with all attributes as columns separated by commas.

The task is divided into two options. *Option 1* is a single line record, and *Option 2* is a multi-line record. It is recommended to start with *Option 1* and then proceed to *Option 2*.

Data set

The data for the task can be found in sub-folder LsaParsing. The following files are included.

Option 1

- lsa_single_line.txt: The fixed width data to be parsed.
- lsa_single_line_def.csv: The schema for the data set
- lsa_single_line_parsed.csv: The resulting csv file. Can be used to validate your result.

Option 2

- lsa_multi_line.txt: The fixed width data to be parsed.
- lsa_multi_line_def.csv: The schema for the data set
- lsa_multi_line_parsed.csv: The resulting csv file. Can be used to validate your result.

Schema and data

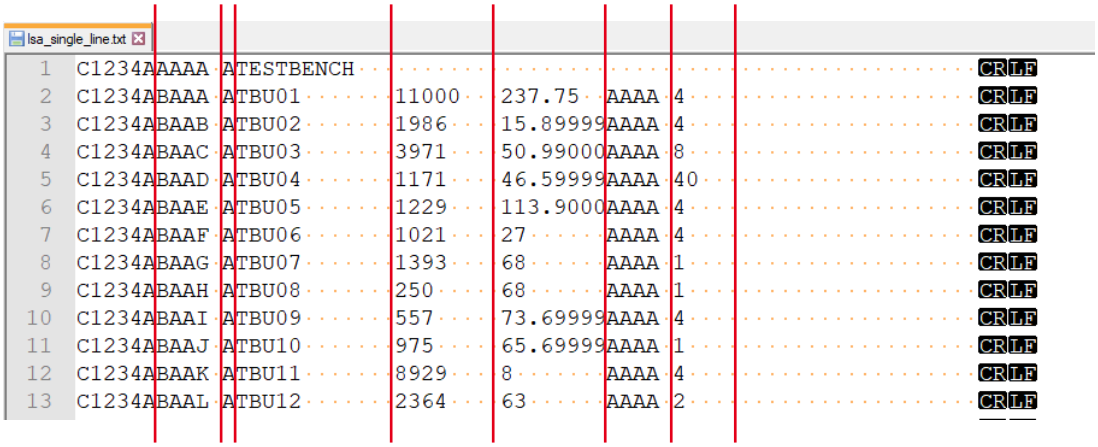
Option 1: Single line LSA

The schema for the single line LSA is:

Attribute	Width (Number of characters)	Comment
Pccn	6	Identical for all plisn
Plisn	5	Identifier

Cfi	1	This is going to be A for all records, as this is a single line
item_name	12	
unit_price	8	
failure_rate	8	
next_higher_plisn	5	
qty_per_assembly	5	

Sample data:



Option 2: Multi line LSA

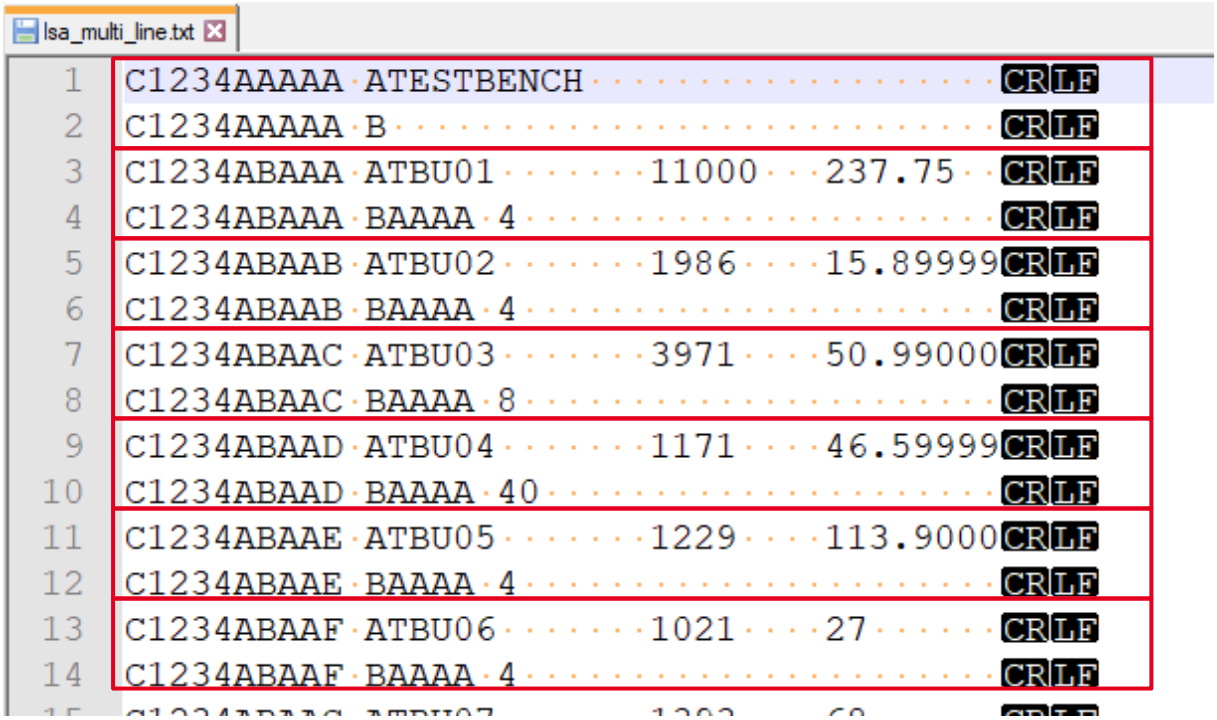
The schema for the multi-line LSA is:

Attribute	Width	cfi value	Comment
pccn	6	Identical on A and B	Identical for all plisn
plisn	5	Identical on A and B	Identifier
cfi	1		This defines if it is the first row (A) or second row (B) for the record.
item_name	12	A	
unit_price	8	A	
failure_rate	8	A	
next_higher_plisn	5	B	
qty_per_assembly	5	B	

(blank)	18	B	Filler to achieve constant width.

Thus, for a record (a combination of pccn and plisn), there will be two rows in the source data. Your task is to combine this to a single row.

Sample data:



Expected Result

The expected result is a csv-file with the parsed data. The result files are included for reference.

The final csv file is supposed to look like this:

lsa_single_line_parsed.csv

```
pccn,plisn,cfi,item_name,unit_price,failure_rate,next_higher_plisn,qty_per_assemblyCR
C1234A,AAAA,A,TESTBENCH,,,,CRLF
C1234A,BAAA,A,TBU01,11000,237.75,AAAA,4CRLF
C1234A,BAAB,A,TBU02,1986,15.89999,AAAA,4CRLF
C1234A,BAAC,A,TBU03,3971,50.99000,AAAA,8CRLF
C1234A,BAAD,A,TBU04,1171,46.59999,AAAA,40CRLF
C1234A,BAAE,A,TBU05,1229,113.9000,AAAA,4CRLF
C1234A,BAAF,A,TBU06,1021,27,AAAA,4CRLF
C1234A,BAAG,A,TBU07,1393,68,AAAA,1CRLF
C1234A,BAAH,A,TBU08,250,68,AAAA,1CRLF
C1234A,BAAI,A,TBU09,557,73.69999,AAAA,4CRLF
C1234A,BAAJ,A,TBU10,975,65.69999,AAAA,1CRLF
C1234A,BAAK,A,TBU11,8929,8,AAAA,4CRLF
C1234A,BAAL,A,TBU12,2364,63,AAAA,2CRLF
C1234A,BAAM,A,TBU13,650,35.70999,AAAA,1CRLF
C1234A,BAAN,A,TBU14,743,35,AAAA,5CRLF
```

Task 3: Database creation

Create a database, containing a single table that will hold the parsed data from Task 2.

Task 4: Data insertion

Create a program that will insert the data from Task 2 into the database in Task 3.