**Task1:**

Call a rest service from below end point and filter the destination from payload based on input passing from Query Parameter.

URL: <http://mu.mulesoft-training.com/essentials/united/flights>

**Task2:**

Call a rest service from **task1** end point and filter the payload based on price passing from Query Parameter.

**Task3:**

Call a rest service from **task1** end point and apply filter the destination from payload based on input passing from Query Parameter. If you are not passing any input from Query parameters, then it should give entire payload as response.

**Task 4:**

Implement a flow to Remove vowels from input (do not use replace keyword)

Input:::: “ABCDEFGHIJKLMNOPQRSTUVWXYZ”  
 output::: "BCDFGHJKLMNPQRSTVWXYZ”

**Task 5:**

**Input:**

{"flights":[

{"code":"er38sd",

"price":400,

"departureDate":"2015/03/20",

"planeType":" Boeing737 ",

"origin":"MUA"

}]}

* *First field should be in upper case*
* Second field should be in characters when it is equals to 400....
* Third field should be converted into required date format
* Fourth field should be remove unwanted spaces from string
* Fifth field should be in lower cases.

**Required output:**

[

{

"code": "ER38SD",

"price": "four hundred",

"departureDate": "2015-03-20",

"planeType": "Boeing737",

"origin": "mua"

}

]

**Task-6**

<http://mu.mulesoft-training.com/essentials/delta/flights?wsdl>

1. List all flights.
2. Find flights – filter the data as per destination.

**Task-7**

http://mu.mulesoft-training.com/essentials/united/flights

http://mu.mulesoft-training.com/essentials/delta/flights?wsdl

**Rest**

* Call a Rest service and filter the destination from payload based on input passing from Query Parameter (Use first link to call Rest service)
* If you are not passing any input from Query parameters then it should give entire payload as response.

**Soap**

* Call a Soap service and filter the destination from payload based on input passing from Query Parameter (Use second link to call Soap service)
* If you are not passing any input from Query parameters then it should give entire payload as response.

Combine the Soap and rest payloads and get output as shown below.

{

"unitedFlights":[{

"code": "ER38sd",

"price": 400,

"origin": "MUA",

"destination": "SFO",

"departureDate": "2015/03/20",

"airlineName": "United",

"planeType": "Boeing 737",

"emptySeats": 0

},

{

"code": "ER38sd",

"price": 400,

"origin": "MUA",

"destination": "SFO",

"departureDate": "2015/03/20",

"airlineName": "United",

"planeType": "Boeing 737",

"emptySeats": 0

}],

"deltFlights":[{

"code": "ER38sd",

"price": 400,

"origin": "MUA",

"destination": "SFO",

"departureDate": "2015/03/20",

"airlineName": "United",

"planeType": "Boeing 737",

"emptySeats": 0

},

{

"code": "ER38sd",

"price": 400,

"origin": "MUA",

"destination": "SFO",

"departureDate": "2015/03/20",

"airlineName": "United",

"planeType": "Boeing 737",

"emptySeats": 0

}]

}

**Task-8**

**Input :**

{

"name":"Raju",

"address":"Hyderabad",

"role":"Associate Trainee",

"company":"mouritech",

"id":"220200"

}

**output:**

If id is "220200"

{

"name":"Raju",

"company":"mouritech"

}

If id not equal to "220200"

{ "message":"not matched with ID" }

**Task-9**

**Input:**

{

"code": "ER38sd",

"price": 400,

"origin": "MUA",

"destination": "SFO",

"departureDate": "2015/03/20",

"airlineName": "United",

"planeType": "Boeing 737",

"emptySeats": 0

}

If destination matches with SFO all the fields should be mapped with csv format and write a csv file with current timestamp

**Example filename**: filename03-2021-12-02-04-41.csv

**Expected Output:**

code,price,origin,destination,departureDate,airlineName,planeType,emptySeats

ER38sd,400,MUA,SFO,2015/03/20,United,Boeing 737,0

If destination not matches with SFO all the fields should be mapped with xml format and write xml file with current timestamp

**Example filename:** filename03-2021-12-02-04-41.xml

**Expected output:**

<?xml version='1.0' encoding='UTF-8'?>

<data>

<code>ER38sd</code>

<price>400</price>

<origin>MUA</origin>

< destination >CLE</destination>

<departureDate>2015/03/20</departureDate>

<airlineName>United</airlineName>

<planeType>Boeing 737</planeType>

<emptySeats>0</emptySeats>

</data>

**Task10**

**Description:**

1. Input should be json object.
2. Sort the key, value pairs alphabetically on basis of ***key.***
3. Generate the output as mentioned in the example below.

**Example:**

**Input:**

{

"name":"<<your\_name>>",

"age":"<<your\_age>>",

"city":"<<your\_city>>",

"role":"<<your\_role>>"

}

**Output:**

age=<<your\_age>>&city=<<your\_city>>&name=<<your\_name>>&<<your\_role>>

**Note:** Please send the input using postman and your code should work for any number of key, value pairs.

**Task 11**

Please work on below POC and implement the logic based on given requirement.

**POC Details:**

1.Implement a logic using below Address,If the Address is

< 200 then assign Address to Address1 and populate only Address1 field

> 200 then assign Address to Address1,Address2 fields and populate both fields.

for **example** Address is 205 charsll  
 then we need to assign 200 chars to Address1 and 5 chars to Address2

**Address**="Mouritech Private Limited,Huda Techno Enclave,Survey No. 64,4th Floor,SBR Surya Pearl Sector III,IT Park,Madhapur,Hitech City Main Rd,Phase 2,Opposite to Mindspace,Top of Karachi Bakery and Burger king,HITEC City,Hyderabad,Telangana 500081"

**Task :: 12**

**Input::::**

[  
{  
"1000": "Alpha"  
},  
{  
"2000": "Bravo"  
},  
{  
"3000": "Charlie"  
}  
]

**Expected Output::**

[  
{  
"ID": "1000",  
"Name": "Alpha"  
},  
{  
"ID": "2000",  
"Name": "Bravo"  
},  
{  
"ID": "3000",  
"Name": "Charlie"  
}  
]

**TASK-13**

Please work on below POC and implement the logic based on given requirement.

1. Read the input from locations.csv file
2. Based on the country name export the .csv file into specified location.

**Task-14**

**Actual Date:**

**20200909080000**

**Required Date Format:**

2020-09-09T00:00:00

**Task-15**

create your Salesforce account by using https://login.salesforce.com.

1. create a custom object in Salesforce account and give the Name, Age, Address, Phone are the custom fields in created custom object.
2. Provide values for created custom fields in salesforce account.
3. Fetch the custom object from Salesforce account to mule project.
4. Use batch process to process the records and if the age is less than 30 from input payload send data into one file and if the age is greater than 30 from input payload send data into another file

**Task-16**

Given are the inputs input1 and input2 and their output transformation. Should provide the dataweave scripts that produce the transformation

First transformation lists all the price of books in an array. The price is appended with $ as currency

The next 2 transformations transform the json into csv format

**input1:**

<Lib>

<Book>

<Title>x</Title>

<price>12</price>

</Book>

<Book>

<Title>y</Title>

<price>13</price>

</Book>

</Lib>

**output:**

[12 INR, 13 INR]

**input2:**

[

{

"product": {

"id": 11,

"items": [

{

"name": "c1",

"quantity": 4,

"createdate": "20210101"

},

{

"name": "c2",

"quantity": 5,

"createdate": "20210103"

}

]

}

},

{

"product": {

"id": 12,

"items": [

{

"name": "c3",

"quantity": 6,

"createdate": "20210201"

},

{

"name": "c4",

"quantity": 1,

"createdate": "20210202"

}

]

}

}

]

**output1:**

product, item, quantity, create date

11, c1, 4, 01/01/2021

11, c2, 5, 03/01/2021

12, c3, 6, 01/02/2021

12, c4, 1, 02/02/2021

**output2:**

item, quantity

c1, 4

c2, 5

c3, 6

c4, 1

**Task-17:**

**Input:**

**{**

**"input": "MGC–Management,PAJ–Parking,COX–Convenience"**

**}**

**Output: Form 3 files based on input and write in folder**

**f**ileName: MG\_Service.csv

code,value

MG,Management

fileName: PA\_Service.json

{

"code": "PA",

"value": Parking

}

fileName: CO\_Service.xml

<service>

<code>CO</code>

<value>Convenience</value>

</service>

**Task-18:**

pass the query from url request. Get the required output without using dataweave language or dataweave funtions anywhere in the flow.

Do not use Transform Message Component.

**Input payload:**

[

{

"name": "John",

"age": 31,

"city": "New York",

"empId": "123",

"dept": "Mechancial"

},

{

"name": "Alp",

"age": 23,

"city": "Hyderabad",

"empId": "123",

"dept": "Computers"

},

{

"name": "Siva",

"age": 35,

"city": "London",

"empId": "123",

"dept": "Electrical"

}

]

When above payload is sent with query Parameter name=siva then below should be expected output,

**Expected Output:**

{

"name": "Siva",

"age": 35,

"city": "London",

"empId": "123",

"dept": "Electrical"

}

**Task 19 :-**

Convert current time[ now() ] to Australian Time zone. The output time format should be in any Australian Time zone format.

**Task 20 :-**

Read the input file from locations.csv and use batch processing to process the records and in batch processing make some records to fail. The failed records should be sent to error handling and failed records should be written to some file in error handling.

**Task 21: -**

Make a flow such that when two or more requests are sent at a time, only one request should be processed, and remaining requests should return custom response. ( Ex: - {

“message” : “ try after sometime”

} ).

Hint: - Make the flow to wait for specific time to complete the processing.