

1 - Introducción

En el área de datos de [Nubimetrics](#) tenemos múltiples desafíos. Los principales son: disponibilizar la información de manera coherente, consumible y rápida a las diferentes áreas que lo requieren.

Para poder cubrir las responsabilidades de un perfil semi senior pensamos que deberías poder tener un entendimiento básico sobre el manejo de APIS; conocer ciertas operaciones sobre diferentes tipos de datos y la realización de transformaciones a través de código a las fuentes de datos existentes.

Te queremos proponer un trabajo sobre las APIs de Mercado Libre, la deserialización de un json, el manejo de un csv y la realización de 3 operaciones sobre los DataFrames.

Forma de Entrega: link a [Github](#) con el código más un [documento de Google Drive](#) con la propuesta de solución.

Te pedimos que antes de arrancar con los desafíos leas las preguntas frecuentes hasta el final

2 - Desafíos

Desafío 1: Mail

Enviar un mail con el lenguaje a utilizar: desafio@nubimetrics.com

Desafío 2: Interactuar con la API de [Mercado Libre](#)

Nos interesa que puedas hacer lo siguiente:

- Ir a la documentación de developers de Mercado Libre: [Developers Mercadolibre](#)
- Descargar la información en formato JSON de un Request.
- Almacenar ese JSON en una carpeta llamada con el siguiente patrón: NombreApi+Formato+año+mes. Por ejemplo: searchjson202008

Ejemplo de URL de la API: Ingresando al siguiente link que te lleva a la categoría [MLA1000](#) podrás notar que esta API devuelve como resultado todos los productos ofertados en la categoría con nombre: [Electrónica, Audio y Video](#).

Si bien los resultados vienen paginados solo nos interesa que bajes el json del primer offset. Si necesitan más información sobre esta API pueden [consultar la documentación de Mercado Libre](#)

Desafío 3: Deserializar un JSON

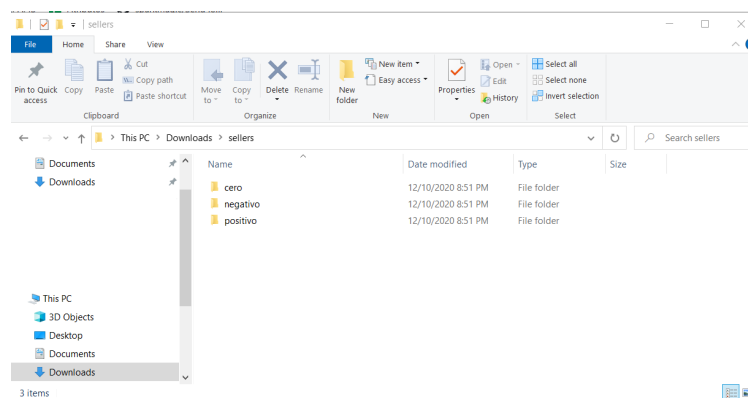
Nos interesa que puedas hacer lo siguiente:

- Ingresar a la carpeta [Input Ejemplos](#)
 - En ese lugar vas a encontrar el json: [sellers.json](#)
- Lee la información en un DataFrame.
- Seleccioná solamente los siguientes que hacen referencia a:
 - Id del site
 - Id del seller
 - Nickname del seller
 - Puntos del seller

El dataframe resultante debe ser como este:

siteId	sellerId	sellerNickname	sellerPoints
MPE	298734964	MARIELATAQUIRE	2
MPE	183049329	MURO8709951	-3
MPE	94592189	ILLARYPERU	-2
MPE	520133997	ISABELLADELPOZO	1
MPE	684964436	PHMO1747353	0
MPE	685079498	MELISSASUSANAARVALOSANTAYA	0
MPE	646068761	YOMIDELGADOSNCHEZ	0
MPE	685310649	DONATILDONATILDECORREATIMOTECO	0
MPE	685419864	VANESSAURNER	0
MPE	285674870	ERICKLOPEZUSMSYA	2
MPE	685275449	DANIELARUIZRIDRIGUES	0
MPE	48893023	MARCELASUSAN	1
MPE	603331827	COVA1031117	0
MPE	205264135	GOBR7283790	1
MPE	580279940	LORDVENCEDOR	0
MPE	300834652	DANIELAHILARIORAMOS	0
MPE	270322958	JHONANTHONYCAYLLAHUAMALASQU	0
MPE	684554092	OSORIOCOLQUIJESSMIGUEL	0
MPE	644166286	GOEL1504737	0
MPE	685449106	ISABELSNCHEZMEDINA	0

- Guardá la información deserializada en un archivo CSV con el siguiente patrón:
 - Sitio/Mes/Año/dia/**SEGMENTO**/(archivo).csv.
 - Ejemplo MPE/2020/08/28/**positivo**/[NoImporta].csv



Tené en cuenta que tenés que encontrar la forma de dividir los sellers por los puntos:

- Positivos: **Positivo**
- Cero puntos: **Cero**
- Negativos: **Negativo**
- **Bonus:**
 - **Opción 1:** Spark/Scala + IntelliJ
 - **Opción 2:** Pyspark + Pycharm

Desafío 4: Parseo de un Array de Structs en un dataframe

Nos interesa que puedas hacer lo siguiente:

- Ingresar a la carpeta [Input Ejemplos](#)
 - En ese lugar vas a encontrar el json: [MPE1004.json](#)
- Lee la información en un DataFrame.
- Generes un Dataframe que tenga la siguiente estructura:
 - rowId
 - itemId
 - soldQuantity
 - availableQuantity

El Dataframe resultante debe ser como el siguiente:

rowId	itemId	soldQuantity	availableQuantity
1	MPE433108265	6	9
2	MPE434382765	6	3
3	MPE433853177	3	17
4	MPE419883282	15	18
5	MPE431714651	15	1
6	MPE438492919	0	100
7	MPE429448587	0	50
8	MPE439307195	0	3
9	MPE439307251	0	3
10	MPE437503507	0	10
11	MPE438828260	0	3
12	MPE439307426	0	3
13	MPE440306037	0	1
14	MPE439307206	0	3
15	MPE431446248	2	23
16	MPE439307250	0	3
17	MPE439510012	0	1
18	MPE439307317	0	3
19	MPE439307286	0	3
20	MPE439307385	0	3
21	MPE440131689	0	7
22	MPE432990777	1	5
23	MPE440389411	1	9
24	MPE421767433	4	11
25	MPE432990779	0	1
26	MPE432439269	2	1
27	MPE431410374	0	1
28	MPE430002527	1	1
29	MPE432990781	0	1
30	MPE432202936	0	10
31	MPE428549066	0	5

32	MPE428549082 1	4	
33	MPE433933924 1	1	
34	MPE432291284 2	1	
35	MPE432728801 1	1	
36	MPE433252062 2	1	
37	MPE436649728 0	100	
38	MPE427140390 10	2	
39	MPE433046443 0	999	
+-----+-----+-----+-----+			

Desafío 5: Agregar las visitas al DataFrame con datos de ventas.

- Utilizando el DataFrame generado en el **desafío 4** (Parseo de un Array de Structs en un dataframe) Seleccionar:
 - itemId
 - soldQuantity
- Ingresar a la carpeta [Input Ejemplos](#)
 - En ese lugar vas a encontrar el csv: [visits.csv](#)
- Lee la información en un DataFrame.
- Generar un nuevo DataFrame utilizando el deserializado y [visits.csv](#) - El Join realizado por itemId. El DataFrame resultante debe tener los siguientes campos:
 - itemId
 - soldQuantity
 - visit
- Filtrar los elementos sin ventas.

El Dataframe resultante debe ser como el siguiente:

+-----+-----+-----+		
	itemId	soldQuantity visits
+-----+-----+-----+		
MPE433108265	6	203
MPE434382765	6	170
MPE433853177	3	1034
MPE419883282	15	1772
MPE431714651	15	33
MPE431446248	2	2242
MPE432990777	1	426
MPE440389411	1	158
MPE421767433	4	746
MPE432439269	2	42
MPE430002527	1	60
MPE428549082	1	352
MPE433933924	1	49
MPE432291284	2	6
MPE432728801	1	68
MPE433252062	2	92
MPE427140390	10	81
+-----+-----+-----+		

Desafío 6: Agregar métricas a un DataFrame.

Utilizando el DataFrame generado en el **desafío 5** (Agregar las visitas al DataFrame con datos de ventas):

- Generar una columna con la tasa de conversión (ventas/Visitas)
- Generar un ranking por la tasa de conversión. Mejor tasa de conversión mejor posición en el ranking.

El Dataframe resultante debe ser como el siguiente:

itemId	soldQuantity	visits	conversionRate	conversionRanking
MPE431714651	15	33	0.4545	1
MPE432291284	2	6	0.3333	2
MPE427140390	10	81	0.1235	3
MPE432439269	2	42	0.0476	4
MPE434382765	6	170	0.0353	5
MPE433108265	6	203	0.0296	6
MPE433252062	2	92	0.0217	7
MPE433933924	1	49	0.0204	8
MPE430002527	1	60	0.0167	9
MPE432728801	1	68	0.0147	10
MPE419883282	15	1772	0.0085	11
MPE440389411	1	158	0.0063	12
MPE421767433	4	746	0.0054	13
MPE433853177	3	1034	0.0029	14
MPE428549082	1	352	0.0028	15
MPE432990777	1	426	0.0023	16
MPE431446248	2	2242	9.0E-4	17

Desafío 7: Porcentaje de Stock

Tomando el DataFrame generado en el **desafío 4** (Parseo de un Array de Structs en un dataframe):

- Seleccionar los campos:
 - itemId
 - availableQuantity
- Generar un DataFrame donde se calcule el porcentaje que tiene cada uno de esos ítems en el total.

El Dataframe resultante debe ser como el siguiente:

itemId	availableQuantity	stockPercentage
MPE433046443	999	70.3
MPE438492919	100	7.04
MPE436649728	100	7.04
MPE429448587	50	3.52

MPE431446248 23	1.62	
MPE419883282 18	1.27	
MPE433853177 17	1.2	
MPE421767433 11	0.77	
MPE432202936 10	0.7	
MPE437503507 10	0.7	
MPE440389411 9	0.63	
MPE433108265 9	0.63	
MPE440131689 7	0.49	
MPE428549066 5	0.35	
MPE432990777 5	0.35	
MPE428549082 4	0.28	
MPE438828260 3	0.21	
MPE439307317 3	0.21	
MPE439307426 3	0.21	
MPE439307206 3	0.21	
MPE439307286 3	0.21	
MPE434382765 3	0.21	
MPE439307251 3	0.21	
MPE439307250 3	0.21	
MPE439307385 3	0.21	
MPE439307195 3	0.21	
MPE427140390 2	0.14	
MPE431714651 1	0.07	
MPE432439269 1	0.07	
MPE440306037 1	0.07	
MPE431410374 1	0.07	
MPE439510012 1	0.07	
MPE430002527 1	0.07	
MPE433933924 1	0.07	
MPE432990779 1	0.07	
MPE432990781 1	0.07	
MPE432291284 1	0.07	
MPE432728801 1	0.07	
MPE433252062 1	0.07	
+-----+-----+-----+		

Desafío 8: Paths 1

El objetivo del desafío es que podamos ver el desarrollo de una función. La cual, recibiendo un conjunto de strings devuelva una Lista de elementos que agrupe desde el primer día del mes hasta la fecha ingresada como parámetros inclusive. Por lo tanto, el objetivo será generar una función Path que devuelva **paths hasta el día solicitado**. El formato de la url es ficticio. Podés usar por ejemplo:

```
"https://importantdata@location/"+year+"/"+month+"/"+day+"/"
```

Ejemplo de la función en **Scala**:

```
def generateMonthlyPathList(year: String, month:String, day:String):
```

```
List[String] = {  
  //Implementación de tu solución  
}
```

Ejemplo de la función en **Python**:

```
def generateMonthlyPathList(year, month, day):  
    #Implementación de tu solución  
    return
```

Un ejemplo de resultado. Llamada a la función:

```
generateMonthlyPathList("2021", "05", "17")
```

Resultado si imprimo a cada string en una línea:

```
https://importantdata@location/2021/05/01/  
https://importantdata@location/2021/05/02/  
https://importantdata@location/2021/05/03/  
https://importantdata@location/2021/05/04/  
https://importantdata@location/2021/05/05/  
https://importantdata@location/2021/05/06/  
https://importantdata@location/2021/05/07/  
https://importantdata@location/2021/05/08/  
https://importantdata@location/2021/05/09/  
https://importantdata@location/2021/05/10/  
https://importantdata@location/2021/05/11/  
https://importantdata@location/2021/05/12/  
https://importantdata@location/2021/05/13/  
https://importantdata@location/2021/05/14/  
https://importantdata@location/2021/05/15/  
https://importantdata@location/2021/05/16/  
https://importantdata@location/2021/05/17/
```

Desafío 9: Paths 2

El objetivo del desafío es que podamos ver el desarrollo de una función. La cual, recibiendo un string para el día y un entero para la cantidad de días anteriores devuelva una Lista de elementos que agrupe desde el día pasado como parámetro hasta los n días anteriores. Por lo tanto, el objetivo será generar una función Path que devuelva paths para **N días corridos**. El formato de la url es ficticio. Podés usar por ejemplo:

```
"https://importantdata@location/"+year+"/"+month+"/"+day+"/"
```

Ejemplo de la función en **Scala**:

```
def generateLastDaysPaths(date: String, days: Int): List[String] = {  
    //Implementación de tu solución  
}
```

Ejemplo de la función en **Python**:

```
def generateLastDaysPaths(date, days):  
    #Implementación de tu solución  
    return
```

Un ejemplo de resultado. Llamada a la función: "20210410", 10

```
generateLastDaysPaths("20210410", 10)
```

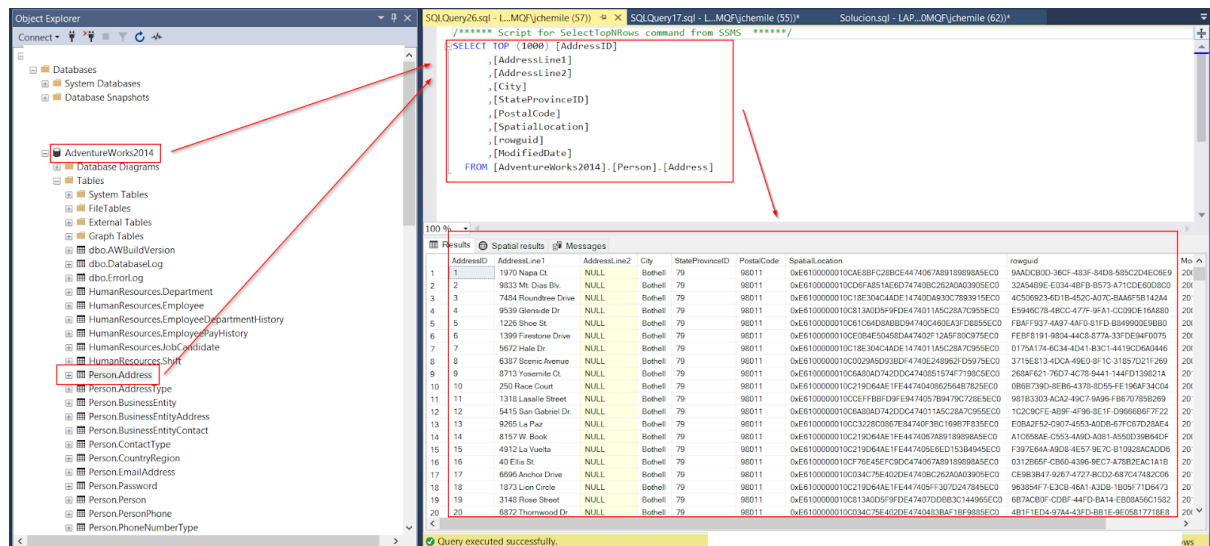
Resultado si imprimo cada path de la lista en una línea:

```
https://importantdata@location/2021/04/01/  
https://importantdata@location/2021/04/02/  
https://importantdata@location/2021/04/03/  
https://importantdata@location/2021/04/04/  
https://importantdata@location/2021/04/05/  
https://importantdata@location/2021/04/06/  
https://importantdata@location/2021/04/07/  
https://importantdata@location/2021/04/08/  
https://importantdata@location/2021/04/09/  
https://importantdata@location/2021/04/10/
```

Desafío 10: Restaurar BBDD

Restaurar la base de datos **AdventureWorks2014.bak** que se encuentra en el sitio oficial de microsoft [AdventureWorks sample databases](#)

Una vez restaurada en tu cliente de base de datos deberías verla de una forma similar a esta:



Desafío 11: SQL 1

Generar un listado con el promedio del tax Rate por país. Las tablas que se utilizaron para realizar el resultado de la consulta son las siguientes: **[Sales].[SalesTaxRate]** y **[Person].[StateProvince]**

El resultado debería ser similar al siguiente:

	country_region_code	average_taxRate
1	AU	10.00
2	CA	8.4333
3	DE	16.00
4	FR	19.60
5	GB	17.50
6	US	7.405

Desafío 12: SQL 2

Generar un listado en el que se pueda ver la **tasa de cambio** y la **tasa impositiva** por país. Sólo se deberán mostrar los países que la posean. Las métricas deberán estar redondeadas a 2 decimales. El currency rate debe ser el máximo para cada país. Recordá que sólo se deben mostrar aquellos resultados que tengan tanto tasa de cambio como tasa impositiva. Las tablas que intervienen son las siguientes:

- [Sales].[SalesTaxRate]
- [Person].[StateProvince]
- [Sales].[CountryRegionCurrency]

- [Person].[CountryRegion]
- [Sales].[CurrencyRate]
- [Sales].[Currency]

Y el resultado debe ser similar al siguiente:

	country_name	currency_name	currency_rate	average_tax_rate
1	Australia	Australian Dollar	2.09	10.00
2	Canada	Canadian Dollar	1.62	8.43
3	France	EURO	1.21	19.60
4	France	French Franc	7.37	19.60
5	Germany	Deutsche Mark	2.20	16.00
6	Germany	EURO	1.21	16.00
7	United Kingdom	United Kingdom Pound	0.73	17.50
8	United States	US Dollar	1.00	7.41

3 - Preguntas Frecuentes

1. ¿Puedo usar el lenguaje y/o framework que quiera?

Tenés que usar **Spark** para la resolución del problema.

2. ¿Cómo lo envío?

Link a [Github](#) con el código más un [documento de Google Drive](#) con la propuesta de solución.

3. ¿Tengo que hacerlos todos?

Idealmente si, pero nos interesa ver cómo resolviste lo que hayas llegado a hacer.

4. ¿Dónde me puedo comunicar si tengo dudas?

Si tenés una duda podés preguntar en: desafio@nubimetrics.com

5. ¿De verdad tengo que mandar el mail que se indica en el **desafío 1**?

No. **No envíes el mail del desafío 1**, era para sacarte una sonrisa en éste momento. De hecho podés hacerlo en más de un lenguaje.