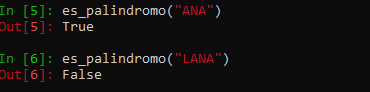
**Ejercicios de PySpark v2**:

PARTE 0:

1. Crear una función **construir\_nombre** que concatene nombre y apellido:



1. Crear una función llamada es palíndromo que verifique si una palabra puede ser leída al revés de la misma forma.

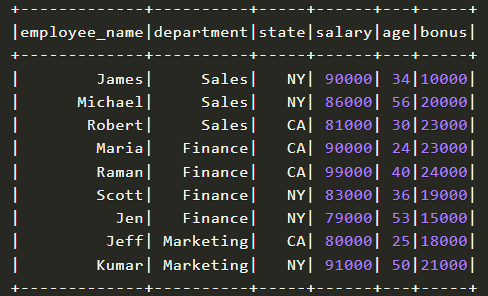


PARTE 1:

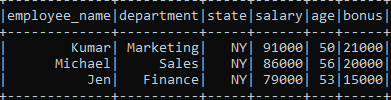
1. Crear el siguiente **dataframe** y guardarlo de forma local.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **D** | **E** | **F** |
| James | Sales | NY | 90000 | 34 | 10000 |
| Michael | Sales | NY | 86000 | 56 | 20000 |
| Robert | Sales | CA | 81000 | 30 | 23000 |
| Maria | Finance | CA | 90000 | 24 | 23000 |
| Raman | Finance | CA | 99000 | 40 | 24000 |
| Scott | Finance | NY | 83000 | 36 | 19000 |
| Jen | Finance | NY | 79000 | 53 | 15000 |
| Jeff | Marketing | CA | 80000 | 25 | 18000 |
| Kumar | Marketing | NY | 91000 | 50 | 21000 |

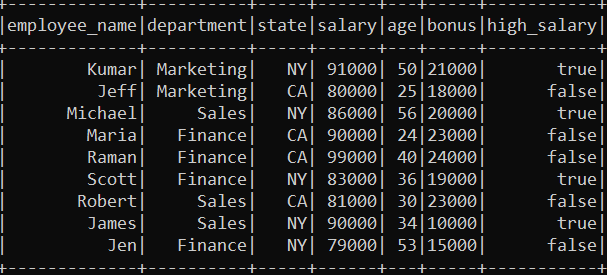
1. Renombrar las columnas A, B, C, D, E, F con employee\_name, department, state, salary, age, bonus. Debería quedar de la siguiente manera:



1. Filtrar las personas cuya edad sea mayor a 40 años:

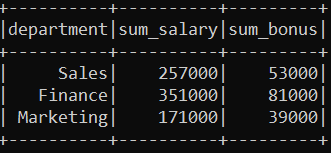


1. Crear una columna de tipo Boolean (True / False) llamada **high\_salary** que sea True cuando el dato del campo **salary** sea mayor a **80000** Y state sea **NY.** Se debe obtener el siguiente resultado:



Resolver utilizando **withColumn** y **select respectivamente (dos versiones)**.

1. Resolver el agrupamiento por departamento para las columnas salary y bonus. La tabla debe ser acondicionada de la siguiente manera **(notar el nombre de las columnas debe ser tal cual la imagen)**.



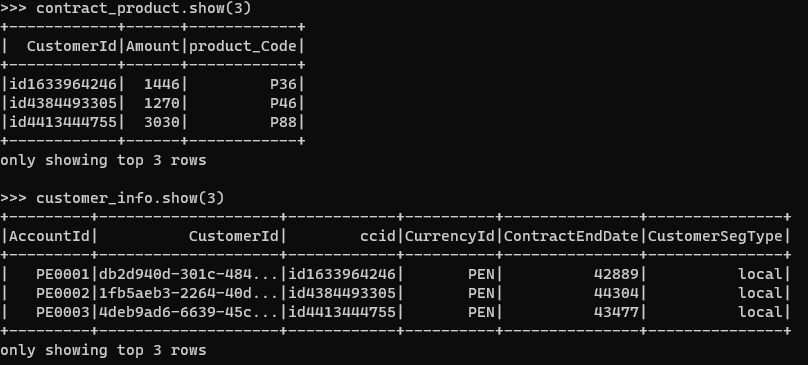
1. Escribir el resultado anterior en un directorio cuyo nombre sea “examen”:

PARTE 2:

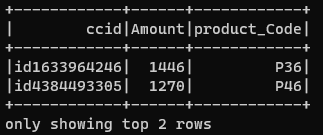
Descargar las siguientes tablas en un directorio local y leerlas con pyspark



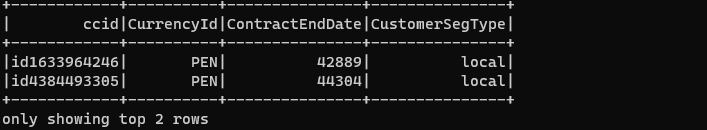
Leer cada tabla asignándolas a variables cuyo nombre coincida con el nombre de la carpeta de ubicación. Un show debería mostrar lo siguiente:



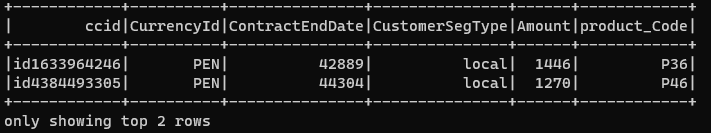
1. Para la tabla contract\_product, corregir el nombre de la columna “**CustomerId**” por “**ccid**”, el resultado debe lucir de la siguiente manera, sobre escribir la misma variable con la corrección:



1. Para la tabla **customer\_info**, sobreescribirla con una selección de las columnas ccid , CurrencyId , ContractEndDate y CustomerSegType.



1. Realizar un cruce tipo inner entre la tabla **customer\_info** y **contract\_product**, bajo la llave **ccid**, por último, guardar el resultado en una variable llamada **resultado\_cruce**:



1. Pegar la siguiente pieza de código:

**from datetime import datetime,timedelta**

**def days\_to\_date(days\_str):**

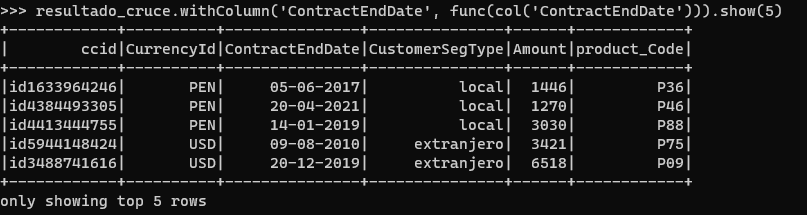
**days = int(days\_str)**

**return(timedelta(days = days) + datetime(1900,1,1,0,0)).strftime('%d-%m-%Y')**

Como se puede apreciar se ha creado una función que convierte los días transcurridos desde 1900 por ejemplo:



Utilizar dicha función para convertir la columna ContractEndDate de a fecha sobre la tabla **resultado\_cruce**, el resultado debe lucir así:



1. Reemplazar la columna ccid por la misma pero cuyo contenido sea invertido, por ejemplo, para ccid = id1633964246 debe ser ahora ccid = 6424693361di.
2. Guardar la tabla resultado\_cruce en un folder llamado t\_temp\_output, con una sola partición.