PRACTICA 1

SEMINARIO DE SISTEMAS 2

Bayron Romeo
Axpuac Yoc

201314474

JUNIO 2020

BAYRON ROMEO
AXPUAC YOC

MARKET 502

GUATEMALA JUNIO 2020

RESUMEN

En el presente documento se da a conocer una solución de Business Intelligence por medio de la herramienta Apache Spark, El concepto 'Business Intelligence', o inteligencia empresarial, se refiere a la utilización de datos en una empresa para facilitar la toma de decisiones dentro de la misma. Es un conjunto de estrategias y herramientas enfocadas al análisis de datos de una empresa mediante el análisis de datos existentes. El business intelligence se caracteriza por mostrar el estado pasado y presente de la organización. Así, pues, recae en las manos de los directivos utilizar este conocimiento para tomar decisiones estratégicas.Un proyecto de business intelligence puede ser una tarea compleja por lo que se requiere una buena organización. Para organizar nuestro sistema de Business Intelligence, consecuentemente, deberemos tener en cuenta los tres elementos:

1. Evaluar nuestras fuentes de datos, que pueden ser internas o externas:

- Accesibilidad
- Fiabilidad
- Calidad
- Posibilidades de integración

2. Convertirlas en información:

- Contextualizando: cuándo, cómo y por qué se han generado.
- Categorizando: cómo se pueden medir y clasificar.
- Calculando: procesarlos si es necesario.
- Corrigiendo: eliminar errores, duplicados e inconsistencias.
- Condensando: definir criterios de agregación, resumir

3. Definir el conocimiento que podremos obtener:

- Comparando la información con otra.
- Haciendo predicciones, buscando interrelaciones…

Para esta práctica tenemos en nuestras manos tres documentos con información a analizar, el primero documento contiene las ventas realizadas sobre videojuegos, el segundo las ventas realizadas por regiones y el último es un documento que posee información sobre diferentes incidentes policiacos. Estos documentos son analizados por medio de pyspark para obtener resultados que dan a conocer información de interés.

APACHE SPARK

Apache Spark es un framework de computación en clúster open-source. Fue desarrollada originariamente en la Universidad de California, en el AMPLab de Berkeley. El código base del proyecto Spark fue donado más tarde a la Apache Software Foundation que se encarga de su mantenimiento desde entonces. Spark proporciona una interfaz para la programación de clusters completos con Paralelismo de Datos implícito y tolerancia a fallos. Apache Spark es un framework de programación para procesamiento de datos distribuidos diseñado para ser rápido y de propósito general. Como su propio nombre indica, ha sido desarrollada en el marco del proyecto Apache, lo que garantiza su licencia Open Source. Además, podremos contar con que su mantenimiento y evolución se llevarán a cabo por grupos de trabajo de gran prestigio, y existirá una gran flexibilidad e interconexión con otros módulos de Apache como Hadoop, Hive o Kafka. Parte de la esencia de Spark es su carácter generalista. Consta de diferentes APIs y módulos que permiten que sea utilizado por una gran variedad de profesionales en todas las etapas del ciclo de vida del dato. Dichas etapas pueden incluir desde soporte para análisis interactivo de datos con SQL a la creación de complejos pipelines de machine learning y procesamiento en streaming, todo usando el mismo motor de procesamiento y las mismas APIs.

Respecto a su propósito general, la virtud de Spark es estar diseñado para cubrir una amplia gama de cargas de trabajo que previamente requerían sistemas distribuidos diferentes. Estos sistemas incluyen procesamiento batch, algoritmos iterativos, queries interactivas, procesamiento streaming,, todos ellos en un pipeline típico de análisis de datos. Por último, hemos dicho que Spark es flexible en su utilización, y es que ofrece una serie de APIs que permiten a usuarios con diferentes backgrounds poder utilizarlo. Incluye APIs de Python, Java, Scala, SQL y R, con funciones integradas y en general una performance razonablemente buena en todas ellas. Permite trabajar con datos más o menos estructurados (RDDs, dataframes, datasets) dependiendo de las necesidades y preferencias del usuario. Además, como hemos ido viendo a lo largo del post, se integra de manera cómoda con otras herramientas Big Data, procedentes del proyecto Apache.

BENEFICIOS DE APACHE SPARK

- Velocidad: Spark puede ser 100 veces más rápido que Hadoop para el procesamiento de datos a gran escala al explotar la computación en memoria y otras optimizaciones. También es rápido cuando los datos se almacenan en el disco, y actualmente tiene el récord mundial para la clasificación en disco a gran escala.
- Facilidad de uso: Spark tiene API fáciles de usar para operar en grandes conjuntos de datos. Esto incluye una colección de más de 100 operadores para transformar datos y APIs de marcos de datos familiares para manipular datos semiestructurados. APIs

como Java, Scala, Phyton y R. También es conocido por su facilidad de uso a la hora de crear algoritmos que adquieren todo el conocimiento de datos muy complejos.

- Un motor unificado: Spark viene empaquetado con bibliotecas de nivel superior, que incluyen soporte para consultas SQL, transmisión de datos, aprendizaje automático y procesamiento de gráficos. Estas bibliotecas estándar aumentan la productividad del desarrollador y se pueden combinar sin problemas para crear flujos de trabajo complejos.

Hadoop es una estructura de software de código abierto para almacenar datos y ejecutar aplicaciones en clústeres de hardware comercial. Proporciona almacenamiento masivo para cualquier tipo de datos, enorme poder de procesamiento y la capacidad de procesar tareas o trabajos concurrentes virtualmente ilimitados. Una de las grandes preguntas sobre Spark es su relación con Hadoop. ¿Se trata de otra tecnología competencia del famoso framework? En realidad, Spark es la evolución natural de Hadoop, cuya funcionalidad es muy rígida y limitada en el sentido de que no aprovecha al máximo las capacidades del procesamiento distribuido. Algunas de las evoluciones que supone Spark frente a su predecesor son el procesamiento en memoria que disminuye las operaciones de lectura/escritura, la posibilidad de análisis interactivo con SQL y la facilidad para interactuar con múltiples sistemas de almacenamiento persistente. La biblioteca de software Apache Hadoop es un marco que permite el procesamiento distribuido de grandes conjuntos de datos en grupos de computadoras utilizando modelos de programación simples. Está diseñado para escalar desde servidores individuales hasta miles de máquinas, cada una de las cuales ofrece computación y almacenamiento local. En lugar de confiar en el hardware para ofrecer alta disponibilidad, la biblioteca en sí está diseñada para detectar y manejar fallas en la capa de aplicación, por lo que ofrece un servicio de alta disponibilidad en la parte superior de un grupo de computadoras, cada una de las cuales puede ser propensa a fallas.

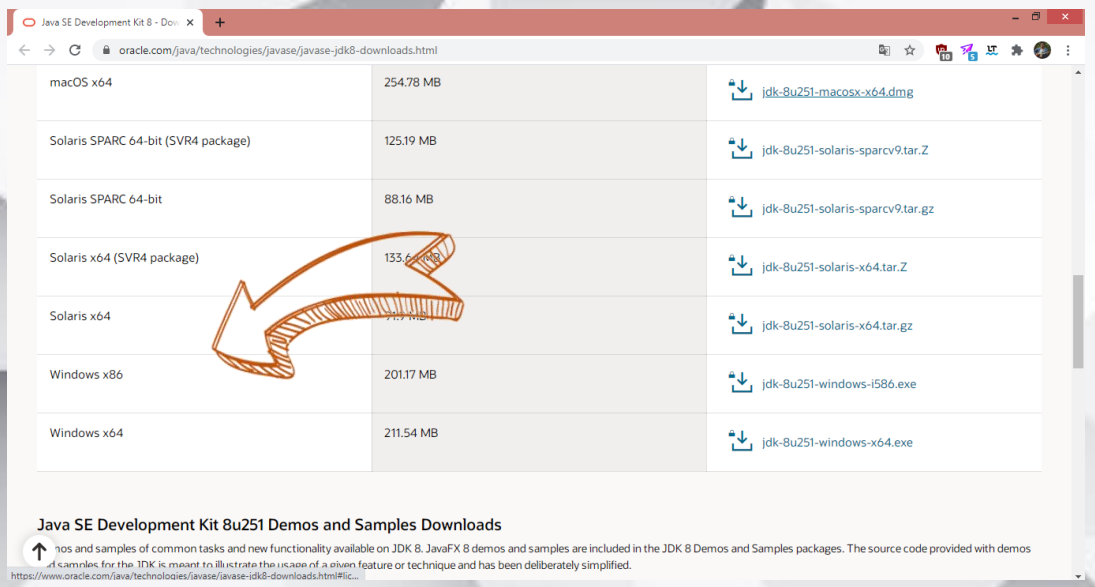
Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en la legibilidad de su código. Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional.

PYCHAR

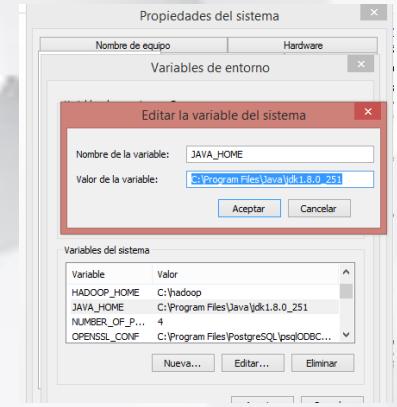
PyCharm es un entorno de desarrollo integrado (IDE) utilizado en la programación de computadoras , específicamente para el lenguaje Python . Está desarrollado por la empresa checa JetBrains . Proporciona análisis de código, un depurador gráfico, un probador de unidad integrado, integración con sistemas de control de versiones (VCS), y soporta el desarrollo web con Django y Data Science con Anaconda . PyCharm es multiplataforma , con versiones de Windows , macOS y Linux . La edición comunitaria se publica bajo la licencia de Apache , y también hay una edición profesional con características adicionales, lanzada bajo una licencia patentada .

INSTALACIÓN DE HERRAMIENTAS

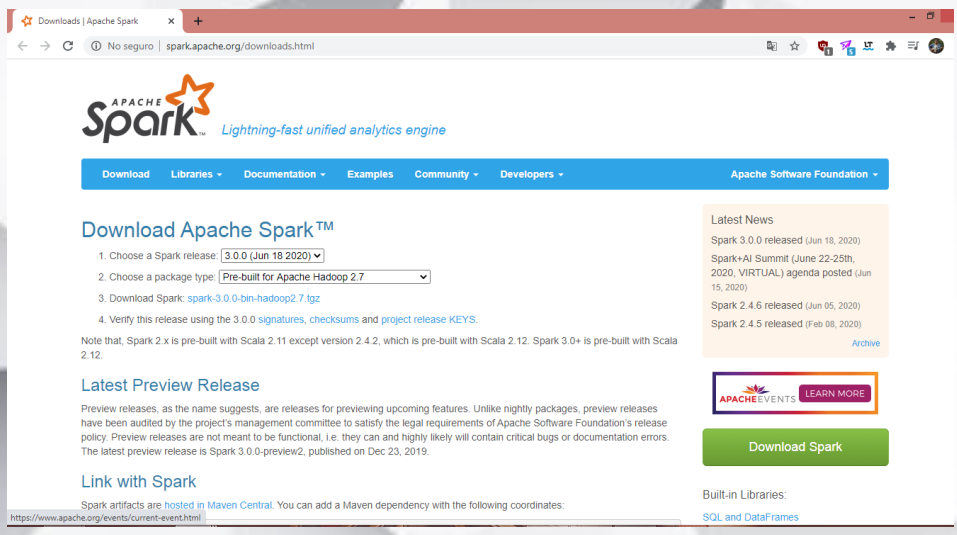
1. Instalación Java JDK: Java Development Kit es un software que provee herramientas de desarrollo para la creación de programas en Java. Puede instalarse en una computadora local o en una unidad de red. En la unidad de red se pueden tener las herramientas distribuidas en varias computadoras y trabajar como una sola aplicación. Para poder descargar esta herramientas se requieren de los siguientes pasos
    a. Tener una cuenta registrado en Oracle.
    b. Ingresar al siguiente enlace https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html y descargar la versión 8 del JDK para Windows.
    c. Ya descargado el archivo lo ejecutamos y seleccionaremos en donde se guardaran los archivos de este software.



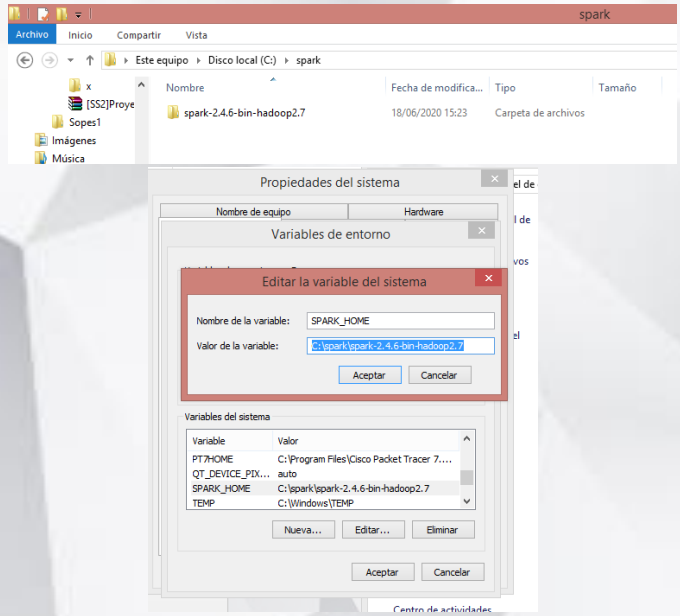
d. Agregamos el JDK como variable de entorno.



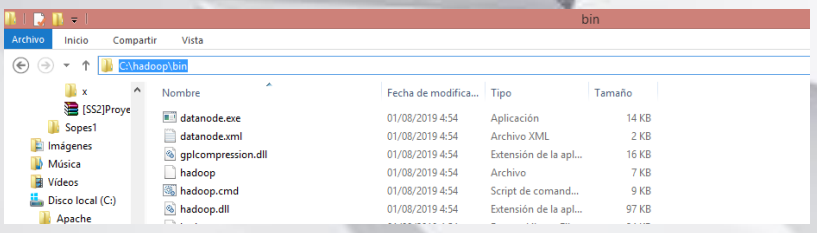
2. Instalación Apache Spark
    a. Ingresar al siguiente link y descargar la versión de Apache Spark que mejor se adapte a nuestro ordenador, debemos percatarnos en la versión de Hadoop que trae el paquete a descargar. La descarga se realiza en el siguiente enlace: http://spark.apache.org/downloads.html

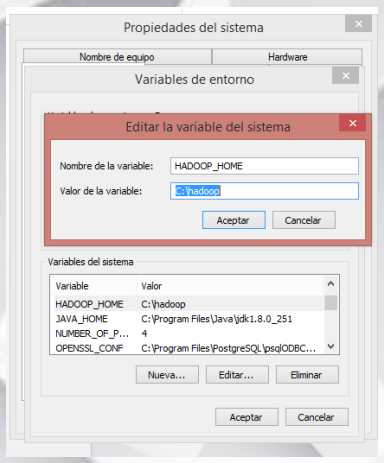


b. Creamos en nuestro disco duro una carpeta denominada spark en ella descomprimimos el archivo descargado y agregamos esta carpeta como variable de entorno.

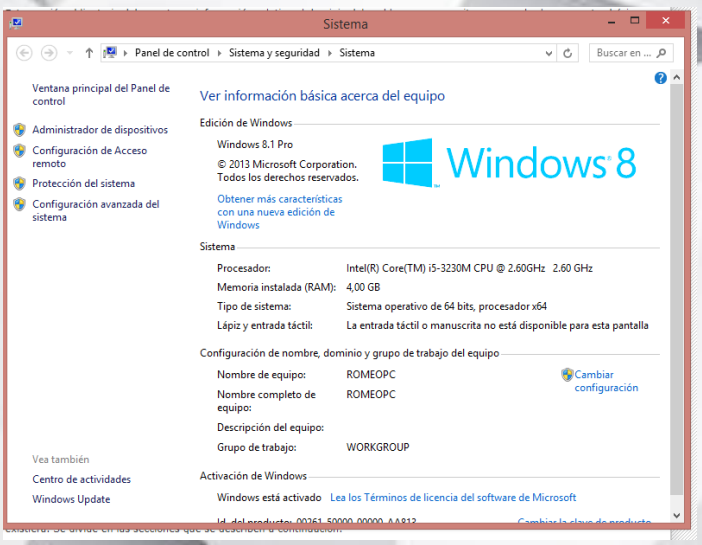


3. Por último falta agregar los archivos binarios de Hadoop, Debemos tener en cuenta la versión de Hadoop que se descargó ya que de esto depende la siguiente descarga. Ingresamos al siguiente enlace https://github.com/steveloughran/winutils y seleccionamos la carpeta que haga referencia a la versión Hadoop que se mencionó al descargar spark, creamos una carpeta en nuestro ordenador y lo agregamos como variable de entorno.





REQUISITOS DEL SISTEMA



Nuestro análisis de bigdata por medio Spark se realizado en un ordenador marca HP14 con un sistema operativo Windows 8.1 Pro, este ordenador cuenta con una memoria RAM de 4GB y un procesador Intel® Core ™ i5 de 2.60 GHz, con un tipo de sistema de 64 bits. Estas características son las mínimas que se recomiendan para la implementación del proyecto de análisis de bigdata de la empresa. Características del software:

Fácil instalación y Mantenimiento:

- Completamente basado en pyhton y spark
- Acceso inmediato
- No requiere instalación o mantenimiento continuo
- Apoyo y entrenamiento al usuario
- Guía de Inicio rápido para los usuarios
- Compatible con cualquier plataforma
- Funciona en Windows, Mac OS y Linux con un navegador web

Navegadores Oficiales:

- Internet Explorer 6.0 o versión más reciente (Windows)
- Apple Safari 5.0 o versión más reciente (Mac OS X)
- Mozilla Firefox 3.6 o versión más reciente (todas las del sistema)
- Google Chrome 11 o versión más reciente (todas las del sistema)

Requerimientos mínimos de hardware

- 2 GHz procesador de 64 bits
- 4 GB de memoria RAM
- 25 GB de Disco duro.

TRANSFORMACIONES Y ACCIONES

1. EJERCICIO 1:
   En este ejercicio se analiza un archivo CSV, el cual contiene el registro de ventas globales de una empresa que distribuye video juegos de diferentes empresas. El primer paso para obtener los resultados del análisis de la data cruda, es tener nuestro archivo CVS separado por " | " ya que esto permitirá ser analizado de una mejor manera en nuestro sistema de pyspark.

```
Rank|Name|Platform|Year|Genre|Publisher|NA_Sales|EU_Sales|JP_Sales|Other_Sales|Global_Sales
1|Wii Sports|Wii|2006|Sports|Nintendo|41.49|29.02|3.77|8.46|82.74
2|Super Mario Bros.|NES|1985|Platform|Nintendo|29.08|3.58|6.81|0.77|40.24
3|Mario Kart Wii|Wii|2008|Racing|Nintendo|15.85|12.88|3.79|3.31|35.82
4|Wii Sports Resort|Wii|2009|Sports|Nintendo|15.75|11.01|3.28|2.96|33
5|Pokemon Red/Pokemon Blue|GB|1996|Role-Playing|Nintendo|11.27|8.89|10.22|1|31.37
6|Tetris|GB|1989|Puzzle|Nintendo|23.2|2.26|4.22|0.58|30.26
7|New Super Mario Bros.|DS|2006|Platform|Nintendo|11.38|9.23|6.5|2.9|30.01
8|Wii Play|Wii|2006|Misc|Nintendo|14.03|9.2|2.93|2.85|29.02
9|New Super Mario Bros. Wii|Wii|2009|Platform|Nintendo|14.59|7.06|4.7|2.26|28.62
10|Duck Hunt|NES|1984|Shooter|Nintendo|26.93|0.63|0.28|0.47|28.31
11|Nintendogs|DS|2005|Simulation|Nintendo|9.07|11|1.93|2.75|24.76
12|Mario Kart DS|DS|2005|Racing|Nintendo|9.81|7.57|4.13|1.92|23.42
13|Pokemon Gold/Pokemon Silver|GB|1999|Role-Playing|Nintendo|9|6.18|7.2|0.71|23.1
14|Wii Fit|Wii|2007|Sports|Nintendo|8.94|8.03|3.6|2.15|22.72
15|Wii Fit Plus|Wii|2009|Sports|Nintendo|9.09|8.59|2.53|1.79|22
16|Kinect Adventures!|X360|2010|Misc|Microsoft Game Studios|14.97|4.94|0.24|1.67|21.82
17|Grand Theft Auto V|PS3|2013|Action|Take-Two Interactive|7.01|9.27|0.97|4.14|21.4
18|Grand Theft Auto: San Andreas|PS2|2004|Action|Take-Two Interactive|9.43|0.4|0.41|10.57|20.81
19|Super Mario World|SNES|1990|Platform|Nintendo|12.78|3.75|3.54|0.55|20.61
20|Brain Age: Train Your Brain in Minutes a Day|DS|2005|Misc|Nintendo|4.75|9.26|4.16|2.05|20.22
21|Pokemon Diamond/Pokemon Pearl|DS|2006|Role-Playing|Nintendo|6.42|4.52|6.04|1.37|18.36
22|Super Mario Land|GB|1989|Platform|Nintendo|10.83|2.71|4.18|0.42|18.14
23|Super Mario Bros. 3|NES|1988|Platform|Nintendo|9.54|3.44|3.84|0.46|17.28
24|Grand Theft Auto V|X360|2013|Action|Take-Two Interactive|9.63|5.31|0.06|1.38|16.38
25|Grand Theft Auto: Vice City|PS2|2002|Action|Take-Two Interactive|8.41|5.49|0.47|1.78|16.15
26|Pokemon Ruby/Pokemon Sapphire|GBA|2002|Role-Playing|Nintendo|6.06|3.9|5.38|0.5|15.85
27|Pokemon Black/Pokemon White|DS|2010|Role-Playing|Nintendo|5.57|3.28|5.65|0.82|15.32
28|Brain Age 2: More Training in Minutes a Day|DS|2005|Puzzle|Nintendo|3.44|5.36|5.32|1.18|15.3
29|Gran Turismo 3: A-Spec|PS2|2001|Racing|Sony Computer Entertainment|6.85|5.09|1.87|1.16|14.98
```

   El primer inciso de este ejercicio nos indica que debemos crear un reporte el cual visualice el total de ventas globales de las siguientes categorías: Action Sports Fighting Shooter Racing Adventure Strategy, para ello primero creamos el siguiente codigo en Python junto a spark para el análisis de nuestra data.

```
sc = SparkContext("local","Practica1")
path1 = "C:\\Users\\Bayyron\\Desktop\\Junio2020\\Seminario2\\Laboratorio\\Archivos\\Libro1.csv"
texto = sc.textFile(path1)
Rdd = texto.map(lambda linea:linea.split("|"))\
    .filter(lambda linea:(linea[4] != "Genre"))\
    .filter(lambda linea:(linea[4] != ""))\
    .filter(lambda linea:(linea[4].upper() == "ACTION" \
                          or linea[4].upper() == "SPORTS"\
                          or linea[4].upper() == "Fighting".upper()\
                          or linea[4].upper() == "Shooter".upper()\
                          or linea[4].upper() == "Racing".upper()\
                          or linea[4].upper() == "Adventure".upper()\
                          or linea[4].upper() == "Strategy".upper()))\
    .map(lambda linea:(linea[4],float(linea[10])))\
    .reduceByKey(lambda x,y: x+y)\
    .sortBy(lambda linea: linea[1],ascending=False)
Ejex = []
EjeY = []
for datos in Rdd.collect():
    Ejex.append(datos[0])
    EjeY.append(round(datos[1],2))
    print("Categoria->" + datos[0] + "  Total->" + str(datos[1]))

GraficaBarras = go.Bar(
    x = Ejex,
    y = EjeY,
    text=EjeY,
    textposition='auto'
)

data = [GraficaBarras]
py.plot(data,filename="Video_Games_Sales_1.html")
print("EJERCICIO 2")
```

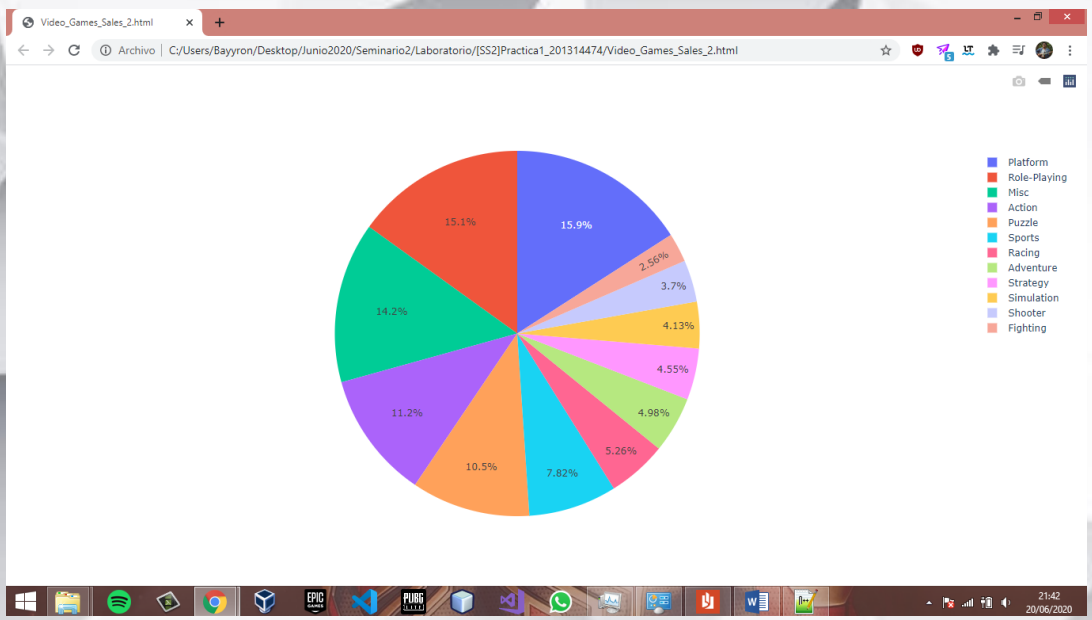
El código presentado anteriormente crear en primer lugar una variable denominada sc, esta variable nos permitirá crear un contexto para trabajar, seguidamente tenemos la variable path1, esta variable almacena la ruta del archivo que mencionamos, luego creamos nuestro RDD el cual almacenará la información del archivo, seguidamente empezamos a aplicar los filtros correspondientes para este RDD, lo primero que haremos con la función map creamos los datos de este RDD, seguidamente los resultados de esta función son filtrados por tres factores el primero elimina la columna inicial del archivo, el segundo que el dato a buscar no venga vacío y así mismo obtener los datos correspondiente por las categorías mencionadas.

Seguidamente volvemos a usar la función map para extraer únicamente la columna categoría y el total que se vendió, luego reducimos las llaves sumando todos las categorías y ordenamos de mayor a menor. Seguidamente creamos dos arreglos que almacenaran la información del reporte, estos arreglos guardan la información resultante del RDD y luego con la librería plotly, graficamos por medio de un objeto de tipo BAR los resultados obteniendo la gráfica siguiente.

El segundo inciso de este ejercicio nos indica que debemos mostrar el total de géneros publicados por la plataforma Nintendo. Para ello volvemos a utilizar nuestra variable path1, y por medio de la función texfile, guardamos la información en la variable texto como el ejercicio anterior, seguidamente creamos un RDD que hace referencia a los datos obtenidos para Nintento.

```
##PROCESO PARA EL EJERCICIO 2: Total de Generos publicados Por nintendo
RddNintendo = texto.map(lambda linea:linea.split('|'))\
    .filter(lambda linea:(linea[5].upper() == "Nintendo".upper()))\
    .map(lambda linea:(linea[4]))\
    .map(lambda variable: (variable,1))\
    .reduceByKey(lambda x,y: x+y)
Juegos = []
TotalJuegos = []
for datos in RddNintendo.collect():
    Juegos.append(datos[0])
    TotalJuegos.append(datos[1])
    print("Categoria->" + datos[0] + "  Total de Juegos->" + str(datos[1]))

GraficaPie = go.Pie(
    labels= Juegos,
    values= TotalJuegos
)
data =[GraficaPie]
py.plot(data,filename="Video_Games_Sales_2.html")
```

Para la obtención de los datos de Nintendo primero separamos los datos por el carácter “ | ”, luego filtramos la columna cinco del archivo y todos los que contengan la compañía Nintendo serán parte del RDD, nos quedamos únicamente con la columna cuatro del archivo a nuestro RDD, esta columna hace referencia a las categorías de los juego, añadimos esta columna a nuestro RDD, luego con otro map, añadimos el digito uno esto nos permitirá ir sumando cada categoría, sumamos todas las categorías. Dicho resultado se muestra por medio de una gráfica de pie.

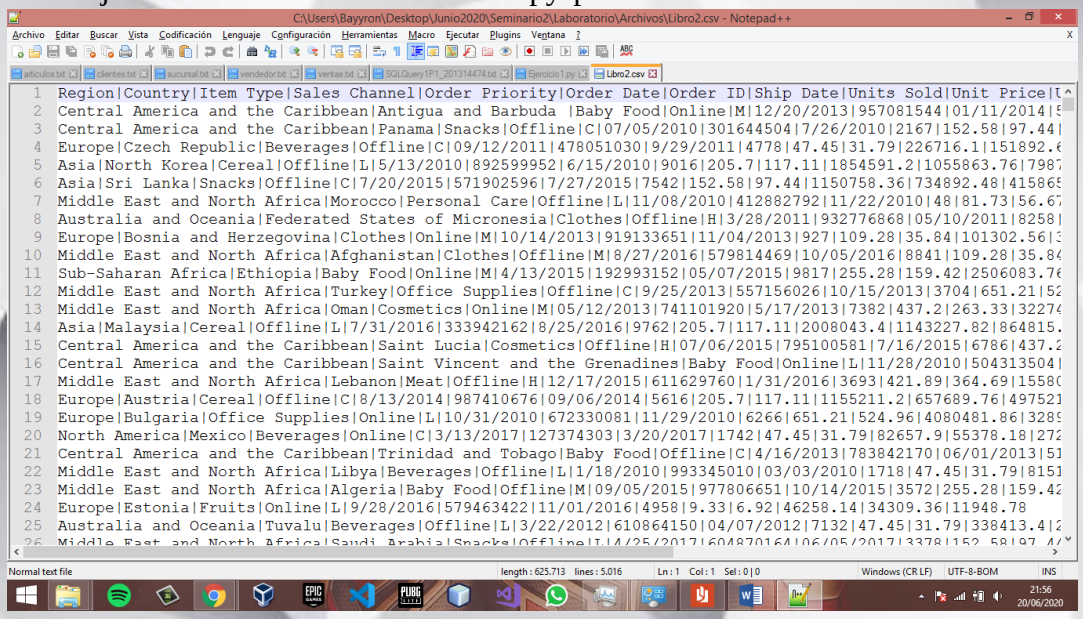


El tercer inciso de este ejercicio nos indica que debemos mostrar el top 5 de plataformas con más lanzamientos.

```
print("EJERCICIO 3")
Rdd3 = texto.map(lambda linea:linea.split('|'))\
    .filter(lambda linea:(linea[2].upper() != "".upper()))\
    .map(lambda linea:(linea[2]))\
    .map(lambda variable: (variable,1))\
    .reduceByKey(lambda x,y: x+y)\
    .sortBy(lambda linea: linea[1],ascending=False)
TOP5PLATAFORMAS = Rdd3.take(5)
PLATAFORMAS = []
TOTALJUEGOSPLATAFORMA = []
for datos in TOP5PLATAFORMAS:
    PLATAFORMAS.append(datos[0])
    TOTALJUEGOSPLATAFORMA.append(datos[1])
    print("Categoria->" + datos[0] + "  Total de Juegos->" + str(datos[1]))

GraficaBarras = go.Bar(
    x = PLATAFORMAS,
    y = TOTALJUEGOSPLATAFORMA,
    text=TOTALJUEGOSPLATAFORMA,
    textposition='auto'
)

data = [GraficaBarras]
py.plot(data,filename="Video_Games_Sales_3.html")
```

Para realizar el tercer ejercicio seguimos utilizando nuestra variable texto, recalcamos que esta variable contiene almacenado la información de los registros de las ventas de la empresa, primero separamos la información de este archivo por medio del carácter “ | ” cada valor resultando es filtrado, y eliminamos los que tengan la columna dos (Platform) del archivo igual a vacío, luego mapeamos nuestro RDD primero para que únicamente queden las plataformas dentro de nuestro RDD, seguidamente añadimos otra columna a nuestro RDD, esta columna tendrá para todos los registros el valor de uno, luego sumamos todas los registros y lo ordenamos de forma ascendente. Ya con los datos obtenidos procedemos a almacenarlo dentro de un objeto de la librería plotly y creamos la gráfica correspondiente obteniendo el siguiente resultado.

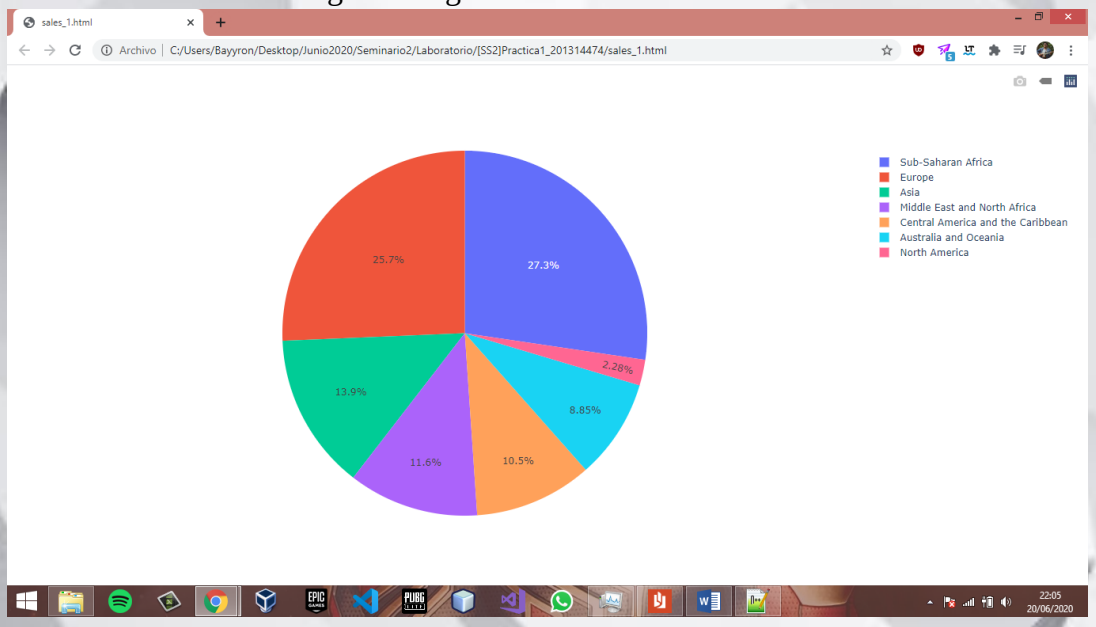
2. EJERCICIO 2: En este ejercicio se analiza un archivo CSV, el cual contiene el registro de ventas globales de una empresa que distribuye video juegos de diferentes empresas. El primer paso para obtener los resultados del análisis de la data cruda, es tener nuestro archivo CVS separado por " | " ya que esto permitirá ser analizado de una mejor manera en nuestro sistema de pyspark.



El primer inciso de este ejercicio nos indica que debemos crear un reporte el cual muestre una comparación de los ingresos de todas las regiones (Centro América, Europa, Asia, etc), para ello primero creamos el siguiente codigo en Python junto a spark para el análisis de nuestra data.

```
sc = SparkContext("local","Practica1")
path1 = "C:\\Users\\Bayyron\\Desktop\\Junio2020\\Seminario2\\Laboratorio\\Archivos\\Libro2.csv"
texto = sc.textFile(path1)
Rdd = texto.map(lambda linea:linea.split('|'))\
    .filter(lambda linea:(linea[0] != ""))\
    .filter(lambda linea:(linea[0] != "Region"))\
    .map(lambda linea:(linea[0],float(linea[11])))\
    .reduceByKey(lambda x,y: x+y)\
    .sortBy(lambda linea: linea[1],ascending=False)

Ejex = []
EjeY = []
print("EJERCICIO1")
for datos in Rdd.collect():
    Ejex.append(datos[0])
    EjeY.append((datos[1]))
    print("Region->" + datos[0] + "  Total->" + str(datos[1]))

GraficaPie = go.Pie(
    labels= Ejex,
    values= EjeY,
)
data =[GraficaPie]
py.plot(data,filename="sales_1.html")
```

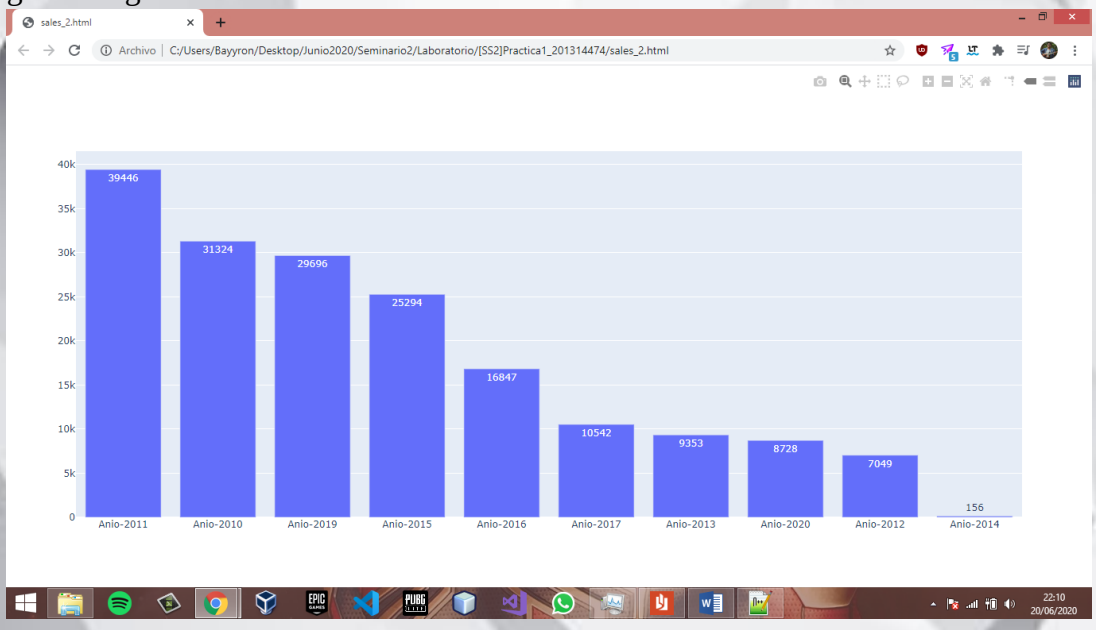
El código presentado anteriormente crear en primer lugar una variable denominada sc, esta variable nos permitirá crear un contexto para trabajar, seguidamente tenemos la variable path1, esta variable almacena la ruta del archivo que mencionamos, luego creamos nuestro RDD el cual almacenará la información del archivo, seguidamente empezamos a aplicar los filtros correspondientes para este RDD, lo primero que haremos con la función map creamos los datos de este RDD, seguidamente los resultados de esta función son filtrados por dos factores una elimina la columna inicial del archivo, y otro que el dato a buscar no venga vacío. Seguidamente volvemos a usar la función map para extraer únicamente la columna region y el total que se vendió, luego reducimos las llaves sumando todos las categorías y ordenamos de mayor a menor. Seguidamente creamos dos arreglos que almacenaran la información del reporte, estos arreglos guardan la información resultante del RDD y luego con la librería plotly, graficamos por medio de un objeto de tipo PIE los resultados obteniendo la gráfica siguiente.



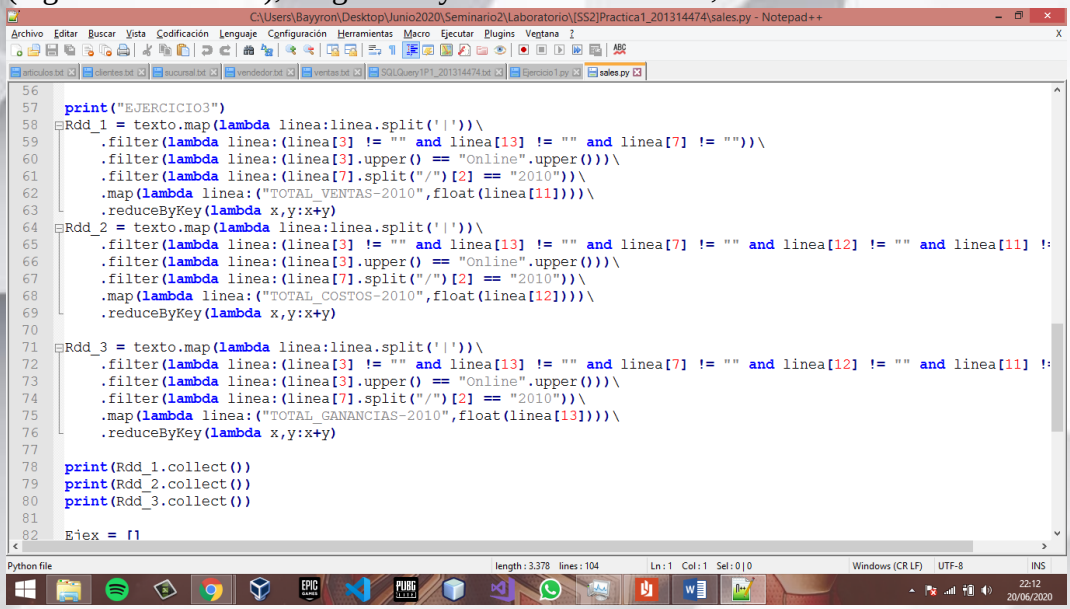
El segundo inciso de este ejercicio nos indica que debemos mostrar reporte cuál es el año con más unidades vendidas en Guatemala. Para ello volvemos a utilizar nuestra variable path1, y por medio de la función texfile, guardamos la información en la variable texto como el ejercicio anterior, seguidamente creamos un RDD que hace referencia a los datos obtenidos para Guatemala.

```
Rdd = texto.map(lambda linea:linea.split('|'))\
    .filter(lambda linea:(linea[1] != ""))\
    .filter(lambda linea:(linea[1] != "Country" and linea[7] != ""))\
    .filter(lambda linea:(linea[1].upper() == "GUATEMALA".upper()))\
    .map(lambda linea:(linea[7].split("/")[2],int(linea[8])))\
    .reduceByKey(lambda x,y: x+y)\
    .sortBy(lambda linea: linea[1],ascending=False)

print("EJERCICIO2")

Ejex = []
EjeY = []
for datos in Rdd.collect():
    Ejex.append("Anio-" + str(datos[0]))
    EjeY.append((datos[1]))
    print("Anio->" + str(datos[0]) + "  Unidades Vendidas->" + str(datos[1]))

GraficaBarras = go.Bar(
    x = Ejex,
    y = EjeY,
    text=EjeY,
    textposition='auto'
)

data = [GraficaBarras]
py.plot(data,filename="sales_2.html")
```

Para este ejercicio primero obtenmos los datos para nuestro RDD por medio de un mapeo de los datos obtenidos en el arhicov, luego fltramos los archivos que el país no venga vacio eliminamos la fila uno del archivo y filtramos los datos que si tengan registro para la columna de unidades vendidas, luego únicamente obtenemos por medio de otro filtro los datos correspondientes a Guatemala. Luego mapeamos nuestro RDD para obtener dos columnas, el año y el total unidades vendidas, Seguidamente creamos dos arreglos que almacenaran la información del reporte, estos arreglos guardan la información resultante del RDD y luego con la librería plotly, graficamos por medio de un objeto de tipo BAR los resultados obteniendo la gráfica siguiente.



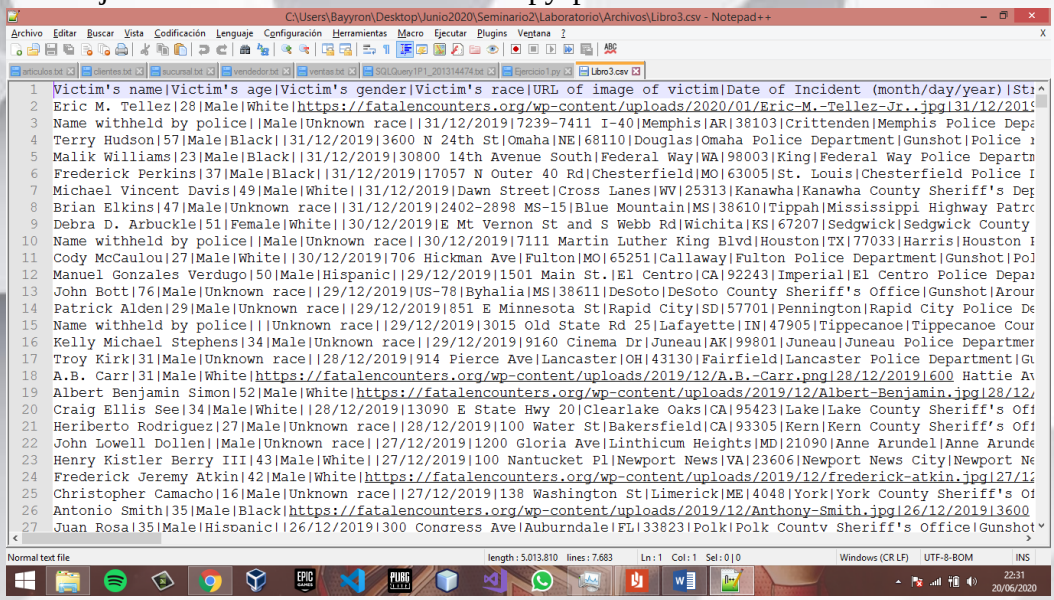
El tercer inciso de este ejercicio nos indica que debemos mostrar las ganancias (ingresos - costos), ingresos y costos del 2010, de las ventas en línea.



Para este reporte se crearon tres RDD los tres con los mismos filtros, la única diferencia son las columnas finales dentro del RDD el primer RDD tiene el total de ventas, el segundo el total de los costos y el tercero las ganancias de la empresa. Todos estos resultados se obtuvieron desde el archivo que se utilizó para este ejercicio.

3. EJERCICIO 3: En este ejercicio se analiza un archivo CSV, el cual contiene el registro de ventas globales de una empresa que distribuye video juegos de diferentes empresas. El primer paso para obtener los resultados del análisis de la data cruda, es

tener nuestro archivo CVS separado por “ | ” ya que esto permitirá ser analizado de una mejor manera en nuestro sistema de pyspark.



Este documento da a conocer registros policiacos sobre incidentes que han ocurrido en diferentes lugares, tiene información de los implicados, su raza, edad y muchos datos de interés, pero para el primer reporte se nos solicita el top 3 de las razas de las víctimas.

```
sc = SparkContext("local","Practica1")
path1 = "C:\\Users\\Bayyron\\Desktop\\Junio2020\\Seminario2\\Laboratorio\\Archivos\\Libro3.csv"
texto = sc.textFile(path1)
Rdd = texto.map(lambda linea:linea.split('|'))\
    .filter(lambda linea:(linea[3] != ""))\
    .filter(lambda linea:(linea[3] != "Victim's race"))\
    .map(lambda linea:(linea[3].upper(),1)) \
    .reduceByKey(lambda x, y: x + y) \
    .sortBy(lambda linea: linea[1], ascending=False)

print("EJERCICIO1")
TOP = Rdd.take(3)
Ejex = []
EjeY = []
for datos in TOP:
    Ejex.append((datos[0]))
    EjeY.append((datos[1]))
    print("Raza->" + str(datos[0]) + "  Total Victimas->" + str(datos[1]))

GraficaBarras = go.Bar(
    x = Ejex,
    y = EjeY,
    text=EjeY,
    textposition='auto'
)

data = [GraficaBarras]
py.plot(data,filename="police_killings_1.html")
```

El código anterior primero guarda la ubicación del archivo que será utilizado, luego obtenemos la data, procesamos la data cruda dentro de un RDD, primero obtenemos los datos por medio de mapeo por medio de un Split a cada línea, seguidamente determinamos que ninguno de los campos de la columna que determina la raza de los individuos este vacía y eliminamos la fila una del archivo, luego por medio de otro mapeo obtenemos las razas, ubicadas en la columna tres y añadimos un número a cada raza y procedimos a sumar las razas y ordenaras de forma ascendente.

Seguidamente creamos dos arreglos que almacenaran la información del reporte, estos arreglos guardan la información resultante del RDD y luego con la librería plotly, graficamos por medio de un objeto de tipo BAR los resultados obteniendo la gráfica siguiente.

Para el segundo reporte de este ejercicio se nos solicita el top 5 de años con más incidentes.

```
Rdd = texto.map(lambda linea:linea.split('|'))\
    .filter(lambda linea:(linea[5] != ""))\
    .filter(lambda linea:(linea[3] != "Victim's race"))\
    .filter(lambda linea:(len(linea[5]) == 10))\
    .map(lambda linea:(linea[5].split("/")[2],1))\
    .reduceByKey(lambda x, y: x + y) \
    .sortBy(lambda linea: linea[1], ascending=False)

print("EJERCICIO2")
TOP = Rdd.take(5)
Ejex = []
EjeY = []
for datos in TOP:
    Ejex.append(("Anio-" + str(datos[0])))
    EjeY.append((datos[1]))
    print("Anio->" + str(datos[0]) + "  Total casos->" + str(datos[1]))

GraficaBarras = go.Bar(
    x = Ejex,
    y = EjeY,
    text=EjeY,
    textposition='auto'
)
data = [GraficaBarras]
py.plot(data,filename="police_killings_2.html")
```

Primero obtenemos los datos por medio de mapeo por medio de un Split a cada línea de la información obtenida del archivo, seguidamente determinamos que ninguno de los campos de la columna que contiene las fechas de los incidentes de los individuos este vacía y eliminamos la fila una del archivo, luego por medio de otro mapeo obtenemos los años, ubicadas en la columna cinco y añadimos un número a cada año y procedimos a sumar las coincidencias de años y ordenaras de forma ascendente. Seguidamente creamos dos arreglos que almacenaran la información del reporte, estos arreglos guardan la información resultante del RDD y luego con la librería

plotly, graficamos por medio de un objeto de tipo BAR los resultados obteniendo la gráfica siguiente.