**INSTITUTO TECNOLÓGICO DE TIJUANA**

**SUBDIRECCIÓN ACADÉMICA**

**DEPARTAMENTO DE SISTEMAS Y COMPUTACIÓN**

**SEMESTRE AGOSTO-DICIEMBRE 2018**

**CARRERA:**

**Ing. Informática**

**MATERIA Y SERIE**

**Datos Masivos BDD-1704 IF9A**

**TÍTULO**

**Ejercicio 6**

**UNIDAD A EVALUAR**

**Unidad 1**

**NOMBRE Y # DE CONTROL DEL ALUMNO:**

**Beltrán Bustos Claudio Israel 14212317**

**GRUPO**

**BDD-1704 IF9A**

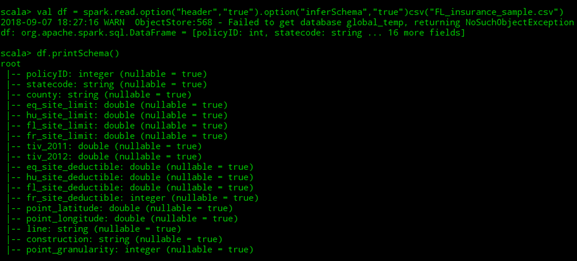
**NOMBRE DEL MAESTRO**

**José Christian Romero**

A continuación, se mostrarán screenshots de 25 funciones junto con su descripción con los dataframes usando spark apache (Scala):

**1.- printSchema()**

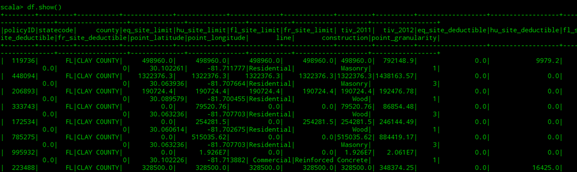
|  |
| --- |
| **df.printSchema()** |



Ésta función se utilizó para imprimir en la consola las columnas que conforman el dataframe. Ésta función es muy útil ya que nos permite analizar las columnas que tiene el dataframe así como el tipo de datos de dichas columnas.

**2.- .show()**

|  |
| --- |
| **df.show()** |



Ésta función nos permite ver los registros contenidos en nuestro dataframe, así podemos nosotros observar y analizar las columnas con sus respectivos datos para empezar a realizar operaciones con ellas. De ésta manera, podemos empezar a analizar y preprocesar los datos para que luego realicemos todos aquellos cálculos matemáticos que se requieran.

**3.- .count()**

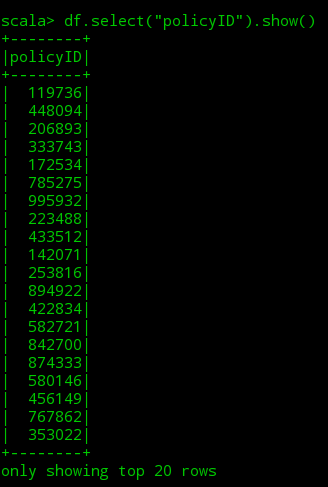
|  |
| --- |
| **df.count()** |



Ésta función nos permite a nosotros como programadores y analistas saber el total de registros que contiene nuestro dataframe, es muy útil ya que esto nos permitirá saber la cantidad de datos o el volumen del dataframe en el cual estamos trabajando.

**4.- .select()**

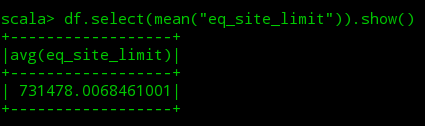
|  |
| --- |
| **df.select("policyID").show()** |



Ésta función nos permite seleccionar un conjunto de datos de nuestra elección, tomando como argumento algún condicionante seguido del nombre de la columna o del parámetro que deseemos. Podemos hacer combinaciones con sentencias de sql usando otra función o podemos ordenar la consulta de datos antes de mostrarlo. Podemos incluso guardar dicha consulta en una vista para que así solamente llamemos a la vista sin la necesidad de teclear toda la línea de código.

**5.-mean()**

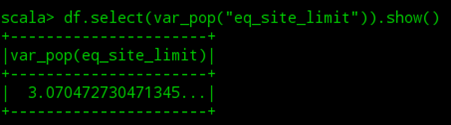
|  |
| --- |
| **df.select(mean("eq\_site\_limit")).show()** |



Ésta función nos permite a nosotros como analistas-estadísticos realizar el cálculo de la media de un conjunto de datos. Esto es útil ya que podemos realizar más operaciones de tipo estadística para poder graficar los datos que nos piden y así, podamos realizar comparaciones con otros conjuntos de datos. En éste caso se deseó calcular la media de la columna ‘eq\_site\_limit’.

**6.- var\_pop()**

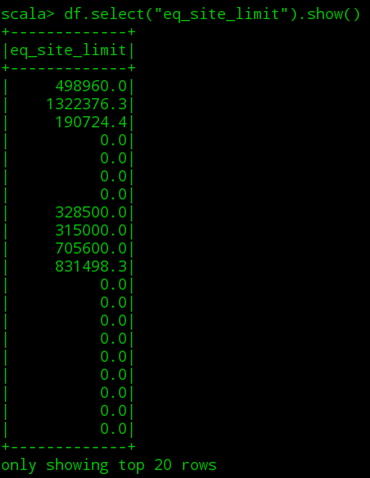
|  |
| --- |
| **df.select(var\_pop("eq\_site\_limit")).show()** |



Ésta función nos permite calcular la varianza poblacional de un conjunto de datos, recordando la fórmula de la varianza poblacional donde varianza = sumatoria de la resta de x menos la media al cuadrado sobre la cantidad de datos totales (población).

**7.- select()**

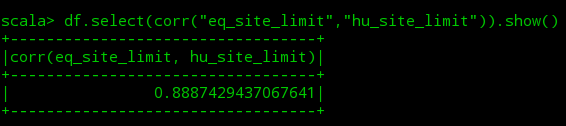
|  |
| --- |
| **df.select("eq\_site\_limit").show()** |



Ésta función nos permite seleccionar un conjunto de datos de nuestra elección, tomando como argumento algún condicionante seguido del nombre de la columna o del parámetro que deseemos. Podemos hacer combinaciones con sentencias de sql usando otra función o podemos ordenar la consulta de datos antes de mostrarlo. Podemos incluso guardar dicha consulta en una vista para que así solamente llamemos a la vista sin la necesidad de teclear toda la línea de código. Ésta vez, la función select() se usó para mostrar los registros de ‘eq\_site\_limit’.

**8.-corr()**

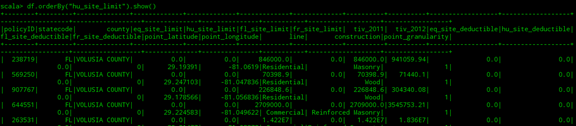
|  |
| --- |
| **df.select(corr("eq\_site\_limit","hu\_site\_limit")).show()** |



Ésta función nos permite a nosotros como analistas-estadísticos calcular la correlación lineal de dos grupos de datos x, y. Recordando la fórmula de la correlación: correlación = covarianza de x, y sobre el producto de la desviación estándar de x por la desviación estándar de y. Aquí se usó la función de correlación para calcular el grado de correlación de ‘eq\_site\_limit’ y ‘hu\_site\_limit’.

**9.- orderBy()**

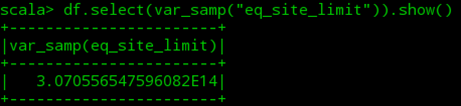
|  |
| --- |
| **df.orderBy("hu\_site\_limit").show()** |



Ésta función nos permite ordenar los datos de acuerdo al parámetro que nosotros queramos. En éste caso queremos que ordene los datos de acuerdo a ‘hu\_site\_limit’. Así que los datos aparecerán ordenados del menor al mayor de los valores de ‘hu\_site\_limit’. También se pueden hacer combinaciones de ordenamiento con otra columna, de tal manera que los ordene de acuerdo a esos dos parámetros.

**10.- var\_samp()**

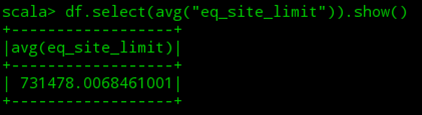
|  |
| --- |
| **df.select(var\_samp("eq\_site\_limit")).show()** |



Ésta función nos permite a nosotros como programadores o analistas-estadísticos calcular la varianza de la muestra, recordando la fórmula de la varianza donde varianza = sumatoria de la resta de x – la media al cuadrado sobre la cantidad total de datos menos 1. En éste caso, se deseó calcular la varianza de la muestra de la columna ‘eq\_site\_limit’.

**11.-avg()**

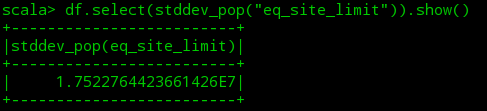
|  |
| --- |
| **df.select(avg("eq\_site\_limit")).show()** |



Ésta función nos permite a nosotros como analistas-matemáticos obtener el valor promedio de un conjunto de datos cualquiera, en éste caso queremos calcular el valor promedio de ‘eq\_site\_limit’. Al igual que las demás operaciones, una consulta u operación como ésta se puede guardar en una variable para futuros usos.

**12.- stddev\_pop()**

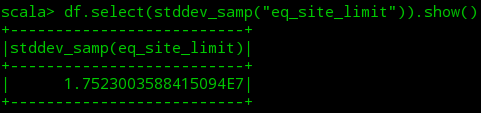
|  |
| --- |
| **df.select(stddev\_pop("eq\_site\_limit")).show()** |



Ésta función nos permite a nosotros como analistas-estadísticos calcular la desviación estándar poblacional de un conjunto de datos. Recordando la fórmula para la desviación estándar poblacional donde desviación estándar = raíz de la varianza.

**13.-stddev\_samp()**

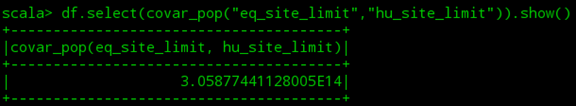
|  |
| --- |
| **df.select(stddev\_samp("eq\_site\_limit")).show()** |



Ésta función nos permite a nosotros como analistas-estadísticos calcular la desviación estándar de la muestra de un conjunto de datos. Recordando la fórmula para la desviación estándar de la muestra donde desviación estándar de la muestra = raíz de la varianza de la muestra.

**14.- covar\_pop()**

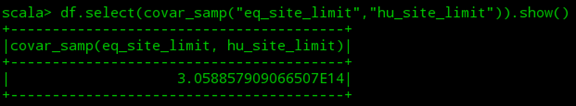
|  |
| --- |
| **df.select(covar\_pop("eq\_site\_limit","hu\_site\_limit")).show()** |



Ésta función nos permite a nosotros como analistas-estadísticos calcular la covarianza poblacional de un conjunto de datos x, y. Recordando la fórmula para la covarianza poblacional donde covarianza poblacional = sumatoria de los productos x por y sobre el total de la población. Menos el producto de la multiplicación de la media de x por la media de y. En éste caso, se quiere saber la covarianza poblacional de ‘eq\_site\_limit’ y ‘hu\_site\_limit’

**15.- covar\_samp()**

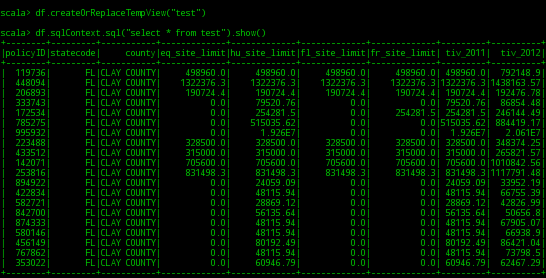
|  |
| --- |
| **df.select(covar\_samp("eq\_site\_limit","hu\_site\_limit")).show()** |



Ésta función nos permite a nosotros como analistas-estadísticos calcular la covarianza de la muestra de un conjunto de datos x, y. Recordando la fórmula para la covarianza de la muestra donde covarianza de la muestra = sumatoria de los productos x por y sobre el total de la población menos 1, menos el producto de la multiplicación de la media de x por la media de y. En éste caso, se quiere saber la covarianza de la muestra de ‘eq\_site\_limit’ y ‘hu\_site\_limit’

**16,17.- createOrReplaceTempview(), sql()**

|  |
| --- |
| **df.createOrReplaceTempView("test") df.sqlContext.sql("select \* from test").show()** |



La función createOrReplaceTempView() crea una vista temporal a partir de nuestro de dataframe, esto permite referenciarla para futuras operaciones con otras tablas (joins), sin la preocupación de modificar el nombre original del dataframe (si es que tiene una), por ende el “TempView”. La vista temporal es eliminada al finalizar el contexto de sql.

Se usa .sql() para usar sentencias tipo sql. En éste caso, hacemos un ‘select \* from test’, es decir, consultamos todos los elementos de la vista temporal ‘test’ que acabamos de crear.

**18.- .sql()**

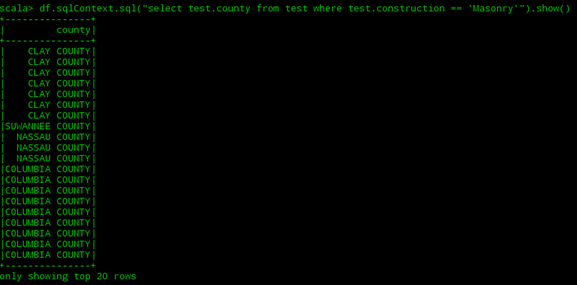
|  |
| --- |
| **df.sqlContext.sql("select test.construction from test").show()** |



Usando la función sqlContext.sql() nos permite insertar una sentencia de tipo sql, es decir, podemos realizar consultas sql en la terminal de scala usando el contexto sql. Esto es muy útil ya que nos brinda la libertad de usar los comandos familiares de sql como select \* from, where, as, orderBy, etc. Lo cual hace mucho más fácil la forma de hacer consultar. Un experto en SQL podría utilizar eficientemente éste comando. En éste caso, queremos que nos traiga todos los registros contenidos en la tabla temporal ‘test’ en la columna ‘construction’

**19.- sql()**

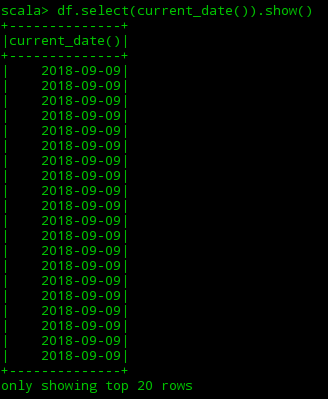
|  |
| --- |
| **df.sqlContext.sql("select test.county from test where test.construction == 'Masonry'").show()** |



Usando la función sqlContext.sql() nos permite insertar una sentencia de tipo sql, es decir, podemos realizar consultas sql en la terminal de scala usando el contexto sql. Esto es muy útil ya que nos brinda la libertad de usar los comandos familiares de sql como select \* from, where, as, orderBy, etc. Lo cual hace mucho más fácil la forma de hacer consultar. Un experto en SQL podría utilizar eficientemente éste comando. En éste caso, queremos que nos traiga todos los registros contenidos en la tabla temporal ‘test’ en la columna ‘county’ con la condición de que ‘construction’ tiene que ser igual a ‘Masonry’.

**20.- current\_date()**

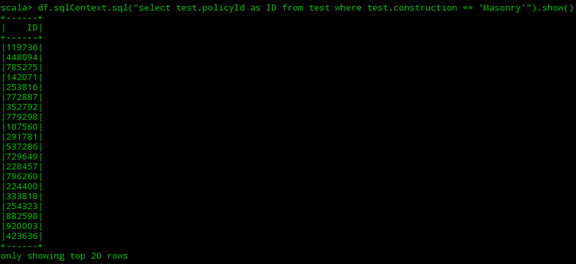
|  |
| --- |
| **df.select(current\_date()).show()** |



La función currentDate() nos regresa la fecha actual de acuerdo a la fecha de nuestra computadora. Se crea una columna temporal en donde despliega la fecha en formato año-mes-día. Como se ve en la imagen, la consulta de la fecha se hizo el día 09 del mes 09 del 2018. Esto es importante ya que al momento de hace reportes uno puede guardar la fecha actual de cuando abrió o realizó operaciones en el dataframe.

**21.- sql()**

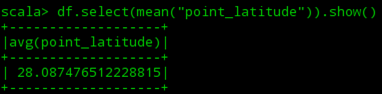
|  |
| --- |
| **df.sqlContext.sql("select test.policyId as ID from test where test.construction == 'Masonry'").show()** |



Usando la función sqlContext.sql() nos permite insertar una sentencia de tipo sql, es decir, podemos realizar consultas sql en la terminal de scala usando el contexto sql. Esto es muy útil ya que nos brinda la libertad de usar los comandos familiares de sql como select \* from, where, as, orderBy, etc. Lo cual hace mucho más fácil la forma de hacer consultar. Un experto en SQL podría utilizar eficientemente éste comando. En éste caso, queremos que nos traiga todos los registros contenidos en la tabla temporal ‘test’ en la columna ‘policyId’, cuyo alias fue ‘ID’ con la condición de que ‘construction’ tiene que ser igual a ‘Masonry’.

**22.-mean()**

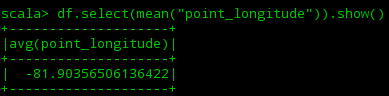
|  |
| --- |
| **df.select(mean("point\_latitude")).show()** |



Ésta función nos permite a nosotros como analistas-estadísticos realizar el cálculo de la media de un conjunto de datos. Esto es útil ya que podemos realizar más operaciones de tipo estadística para poder graficar los datos que nos piden y así, podamos realizar comparaciones con otros conjuntos de datos. En éste caso, se desea calcular la media de ‘point\_latitude’.

**23.-mean()**

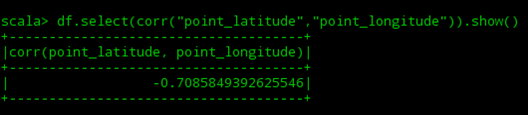
|  |
| --- |
| **df.select(mean("point\_longitude")).show()** |



Ésta función nos permite a nosotros como analistas-estadísticos realizar el cálculo de la media de un conjunto de datos. Esto es útil ya que podemos realizar más operaciones de tipo estadística para poder graficar los datos que nos piden y así, podamos realizar comparaciones con otros conjuntos de datos. En éste caso, se desea calcular la media de ‘point\_longitude’.

**24.- corr()**

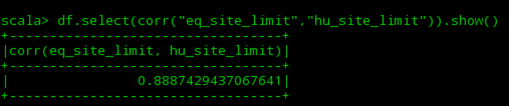
|  |
| --- |
| **df.select(corr("point\_latitude","point\_longitude")).show()** |



Ésta función nos permite a nosotros como analistas-estadísticos calcular la correlación lineal de dos grupos de datos x, y. Recordando la fórmula de la correlación: correlación = covarianza de x, y sobre el producto de la desviación estándar de x por la desviación estándar de y. Aquí se usó la función de correlación para calcular el grado de correlación de ‘point\_latitude’ y ‘point\_longitude’.

**25.-corr()**

|  |
| --- |
| **df.select(corr("eq\_site\_limit","hu\_site\_limit")).show()** |



Ésta función nos permite a nosotros como analistas-estadísticos calcular la correlación lineal de dos grupos de datos x, y. Recordando la fórmula de la correlación: correlación = covarianza de x, y sobre el producto de la desviación estándar de x por la desviación estándar de y. Aquí se usó la función de correlación para calcular el grado de correlación de ‘eq\_site\_limit’ y ‘hu\_site\_limit’.