Análisis e Implementación en SparkQL

- Entrega hasta el 24 de mayo en https://forms.gle/aRiWtjayausRFCTd8
- Los dos conjuntos de datos entregados, CSV separados por coma, Evaluación del Pitch (2021.04.13 ISoftware).csv y Asistencia del Pitch (2021.04.13 ISoftware).csv proceden de dos encuestas realizadas con Google Forms.
- Los archivos proceden de las evaluaciones de los pitch (exposiciones cortas de negocio) que realizan los estudiantes de la asignatura de Ingeniería de Software. Los estudiantes están organizados en equipos. Cada equipo realiza una presentación de máximo 15 minutos, una vez inicia la presentación todos los estudiantes deben registrar su asistencia lo cual queda registrado en 'Asistencia del Pitch (2021.04.13 ISoftware).csv' incluidos los miembros del equipo.
- El archivo `Asistencia del Pitch (2021.04.13 ISoftware).csv` contiene las columnas: `"Marca temporal"` es tiempo dado en fecha y hora, `"Nombre de usuario"` es el correo electrónico del estudiante y es texto, `"Equipo al que perteneces:"` equipo de trabajo al que pertenece el estudiante también es texto y `"Equipo que va a exponer:"` equipo que el estudiante va a tender a su presentación.
- El archivo `Evaluación del Pitch (2021.04.13 ISoftware).csv` contiene las columnas `"Marca temporal"` es tiempo dado en fecha y hora, `"Nombre de usuario"` es el correo electrónico del estudiante y es texto, `"Equipo que vas a evaluar:"` equipo que ha expuesto y que va a ser evaluado por cada estudiante que no sea integrante; a continuación, se tienen las siguientes columnas que corresponden a la evaluación de los respectivos ítems:

    1. "Introducción: El equipo responde adecuadamente ¿Quiénes son y por qué están aquí?",
    2. "Equipo: El equipo responde adecuadamente ¿Quiénes están detrás de la idea y cuál es su función?",
    3. "Problema: El equipo responde adecuadamente ¿Qué problema resolverá?, ¿es realmente un problema?",
    4. "Ventajas: El equipo responde adecuadamente ¿Por qué su solución es especial?, ¿qué la hace distinta de otras?",
    5. "Solución: El equipo responde adecuadamente ¿Cómo piensa resolver el problema?",
    6. "Producto: El equipo responde adecuadamente ¿Cómo funciona el producto o servicio? Muestra algunos ejemplos.",
    7. "Tracción: El equipo responde adecuadamente si cuenta con clientes que demuestran potencial.",
    8. "Mercado: El equipo responde conoce, o por lo menos intentar predecir, el tamaño del mercado que impactará.",

9. "Competencia: El equipo responde adecuadamente ¿Cuáles son las soluciones alternativas al problema que plantea?",
10. "Modelo de negocio: El equipo responde adecuadamente ¿Cómo hará dinero? ",
11. "Inversión: El equipo responde adecuadamente ¿Cuál es su presupuesto y cuánto espera ganar?",
12. "Contacto: El equipo deja los datos al cliente y muestra cómo pueden contactarle.",
13. "Exposición: ¿Qué tan coordinados estaban los expositores?",
14. "Exposición: ¿Los expositores se expresaron con claridad y se hicieron entender?",
15. "Exposición: Las diapositivas son claras y coherentes y apoyaron adecuadamente la exposición.",
    - "Suponiendo que eres inversionista, ¿Estarías dispuesto a invertir dinero en este equipo? (esta pregunta no se pondera en la nota)",
    - "Observaciones para el equipo, estas observaciones las debe considerar el equipo para mejorar la siguiente presentación."

- Cada ítem se evalúa con la siguiente escala: 0. Ausente; 1. Deficiente; 2. Regular; 3. Aceptable; 4. Bueno; 5. Excelente
- Carge los datos, cada archivo en una tabla SparkSQL y responda cada una de las consultas dadas en cada celda. Tenga en cuenta que algunas consultas pueden tener como resultado el vacío.

Integrantes del equipo

1. Juliana Arias Ciro - 1038409725
2. Juan Pablo López Buitrago - 1037975877
3. Federico Cardona Salazar - 1053870065
4. Jhon Edwin Mejia Leon - 1111204157
5. Jorge Iván Gómez Restrepo - 9770450

```
# Es una libreria que da soporte para llamar funciones, clases y tipos del Lenguaje C
# Es un requerimiento para ts.flint
!pip3 install Cython

# Es una libreria especial para la serialización de arreglos basada en C++, permite la
# integración entre pandas, NumPy, Spark ...
# https://pypi.org/project/pyarrow/
# Es un requerimiento para ts.flint
!pip3 install pyarrow==0.9.0
```

```
# Es una colección de módulos para el análisis de series de tiempo con PySpark
# https://ts-flint.readthedocs.io/en/latest/
!pip3 install ts-flint

# Descargamos las librerias java de flint
# quedará en la carpeta /datalake
!wget https://repo1.maven.org/maven2/com/twosigma/flint/0.6.0/flint-0.6.0.jar
```

```
Requirement already satisfied: Cython in /usr/local/lib/python3.7/dist-packages (0.29.2
Collecting pyarrow==0.9.0
  Using cached https://files.pythonhosted.org/packages/be/2d/11751c477e4e7f4bb07ac7584a
Requirement already satisfied: numpy>=1.10 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: six>=1.0.0 in /usr/local/lib/python3.7/dist-packages (fr
Building wheels for collected packages: pyarrow
  Building wheel for pyarrow (setup.py) ... error
  ERROR: Failed building wheel for pyarrow
  Running setup.py clean for pyarrow
Failed to build pyarrow
Installing collected packages: pyarrow
  Found existing installation: pyarrow 0.16.0
    Uninstalling pyarrow-0.16.0:
      Successfully uninstalled pyarrow-0.16.0
    Running setup.py install for pyarrow ... error
  Rolling back uninstall of pyarrow
  Moving to /usr/local/bin/plasma_store
   from /tmp/pip-uninstall-_t4bega0/plasma_store
  Moving to /usr/local/lib/python3.7/dist-packages/pyarrow-0.16.0.dist-info/
   from /usr/local/lib/python3.7/dist-packages/~yarrow-0.16.0.dist-info
  Moving to /usr/local/lib/python3.7/dist-packages/pyarrow/
   from /usr/local/lib/python3.7/dist-packages/~yarrow
ERROR: Command errored out with exit status 1: /usr/bin/python3 -u -c 'import sys, setu
Requirement already satisfied: ts-flint in /usr/local/lib/python3.7/dist-packages (0.6.
--2021-05-08 21:03:40--  https://repo1.maven.org/maven2/com/twosigma/flint/0.6.0/flint-
Resolving repo1.maven.org (repo1.maven.org)... 199.232.192.209, 199.232.196.209
Connecting to repo1.maven.org (repo1.maven.org)|199.232.192.209|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2171677 (2.1M) [application/java-archive]
Saving to: ‘flint-0.6.0.jar.7’

flint-0.6.0.jar.7   100%[===================>]   2.07M  --.-KB/s    in 0.03s

2021-05-08 21:03:40 (61.4 MB/s) - ‘flint-0.6.0.jar.7’ saved [2171677/2171677]
```

```
# Agrege acá el código para importar las librerias
import os
# Requerido https://github.com/twosigma/flint/blob/master/python/README.md
os.environ['PYSPARK_SUBMIT_ARGS'] = '--jars /datalake/flint-0.6.0.jar --py-files /datalake/fl

# La libreria para "encontrar el sevicio" de Spark
import findspark
findspark.init()
```

```
# Librerias para "gestionar el servicio" de Spark
from pyspark import SparkConf, SparkContext
from pyspark.sql import SQLContext, SparkSession

# Creamos una aplicación Spark en el Servicio
# Tenga cuidado con las tildes o caracteres especiales en el nombre de la app
AppSpark = SparkConf().setAppName("Evaluacion iSofware")

# definimos un espacio o contexto para la App
ContextoSpark=SparkContext(conf=AppSpark)

# inicio una sesión en el espacio de la App
SesionSpark = SparkSession(ContextoSpark)

# inicio del espacio o contexto SQL
ContextoSql = SQLContext(sparkContext=ContextoSpark, sparkSession=SesionSpark)
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-9-8e24cc2eeee4> in <module>()
      7
      8 # La libreria para "encontrar el sevicio" de Spark
----> 9 import findspark
     10 findspark.init()
     11

ModuleNotFoundError: No module named 'findspark'

---------------------------------------------------------------------------
NOTE: If your import is failing due to a missing package, you can
manually install dependencies using either !pip or !apt.

To view examples of installing some common dependencies, click the
"Open Examples" button below.
---------------------------------------------------------------------------
```

SEARCH STACK OVERFLOW

```
# 1. Cargue los datos en la carpeta datalake y luego del /datalake al HDFS (Hadoop File Syste
# Recuerda usar ! para ejecutar el comando en el shell.
# Tu código a continuación...
!wget https://raw.githubusercontent.com/jariasci/Trabajo-Spark/main/Asistencia_Pitch.csv
!wget https://raw.githubusercontent.com/jariasci/Trabajo-Spark/main/Evaluacion_Pitch.csv

!hadoop fs -put Asistencia_Pitch.csv
!hadoop fs -put Evaluacion_Pitch.csv
!hadoop fs -ls


# 2. Cree dos tablas SparkSQL y almacene el csv en su correspondiente tabla.
# Observación: tenga especial cuidado con los encabezados de los archivos CSV.
# Usted puede considerar cambiar los encabezados de los CSV originales
# Tu código a continuación
```

```
# Tu código a continuación...

# Cargar los datos con sus correspondientes tipos
# Recordar que el CSV debe estar en el File System de Hadoop
ContextoSql.sql("""
CREATE TABLE IF NOT EXISTS
asistencia (
        creacion_registro STRING,
        correo STRING,
        equipo_pertenece STRING,
        equipo_exposicion STRING
        )
USING CSV OPTIONS (
    header='true',
    nullvalue='NA',
    timestampFormat=\"yyyy-MM-dd'T'HH:mm:ss\",
    path='Asistencia_Pitch.csv')
    """);

# Cargar los datos con sus correspondientes tipos
# Recordar que el CSV debe estar en el File System de Hadoop
ContextoSql.sql("""
CREATE TABLE IF NOT EXISTS
evaluacion (
        creacion_registro DATE,
        correo STRING,
        equipo_evaluar STRING,
        nota_introduccion DOUBLE,
        nota_equipo DOUBLE,
        nota_problema DOUBLE,
        nota_ventajas DOUBLE,
        nota_solucion DOUBLE,
        nota_producto DOUBLE,
        nota_traccion DOUBLE,
        nota_mercado DOUBLE,
        nota_competencia DOUBLE,
        nota_modelo DOUBLE,
        nota_inversion DOUBLE,
        nota_contacto DOUBLE,
        nota_exposicion1 DOUBLE,
        nota_exposicion2 DOUBLE,
        nota_exposicion3 DOUBLE,
        descripcion_inversionista STRING,
        observaciones STRING
        )
USING CSV OPTIONS (
    header='true',
    nullvalue='NA',
    timestampFormat=\"yyyy-MM-dd'T'HH:mm:ss\",
    path='Evaluacion_Pitch.csv')
    """);
```

```
Asistentes = ContextoSql.sql("""SELECT equipo_exposicion,count(equipo_pertenece)
                                FROM asistencia
                                WHERE SUBSTR(equipo_exposicion, 8) != SUBSTR(equipo_pertenece
                                GROUP BY equipo_exposicion""")
Asistentes.show(Asistentes.count())# 2. Consulte el listado total de estudiantes (correos ele
# curso de Ingeniería de Software, ordenados alfabéticamente
# Tu código a continuación...
consulta = ContextoSql.sql("SELECT DISTINCT correo FROM asistencia ORDER BY correo ASC")
consulta.show(consulta.count())


# 3. Consulte la cantidad de asistencias registradas por estudiante; además, la fecha y hora
# y la fecha y hora de la última asistencia
# Tu código a continuación...
consulta2 = ContextoSql.sql("""SELECT correo, count(equipo_exposicion) AS cantidad_de_asister
                            FROM asistencia
                            GROUP by correo
                            order by count(equipo_exposicion)""")
consulta2.show()


# 4. Consulte el listado de estudiantes que asistieron a 2 presentaciones o menos (una).
# Tu código a continuación...
consulta2 = ContextoSql.sql("""SELECT correo, count(equipo_exposicion)
                            FROM asistencia
                            GROUP by correo
                            order by count(equipo_exposicion)""")

consulta3 = consulta2.select(['correo']).filter((consulta2['count(equipo_exposicion)'] <= 2)&


# 5. Consulte el listado de estudiantes que no asistieron a ninguna presentación.
# Tu código a continuación...
consulta2 = ContextoSql.sql("""SELECT correo, count(equipo_exposicion)
                            FROM asistencia
                            GROUP by correo
                            order by count(equipo_exposicion)""")

consulta3 = consulta2.select(['correo']).filter(consulta2['count(equipo_exposicion)'] == 0).s



# 6. Consulte los integrantes por cada equipo al que pertenecen.
# Tu código a continuación...
consulta = ContextoSql.sql("SELECT equipo_pertenece,correo FROM asistencia GROUP BY equipo_pe
consulta.show(consulta.count())


# 7. Consulte la cantidad de asistentes por presentación, sin considerar los asistentes que p
# al equipo que realizó la presentación.
# Tu código a continuación...
Asistentes = ContextoSql.sql("""SELECT equipo_exposicion,count(equipo_pertenece)
                                FROM asistencia
```

```
                                FROM asistencia
                                WHERE SUBSTR(equipo_exposicion, 8) != SUBSTR(equipo_pertenece
                                GROUP BY equipo_exposicion""")
Asistentes.show(Asistentes.count())



# 8. Consutar cuáles integrantes evaluaron a su propio equipo. Estas evaluaciones no serán vá
# integrante no puede evaluar a su propio equipo.
# Tu código a continuación...
consulta2 = ContextoSql.sql("""SELECT asi.correo,asi.equipo_pertenece,eva.equipo_evaluar
                            FROM asistencia AS asi INNER JOIN evaluacion AS eva
                            ON asi.correo = eva.correo
                            WHERE SUBSTR (asi.equipo_pertenece, 15) = eva.equipo_evaluar
                            """)
consulta2.show()


# 9. Consultar la nota promedio por cada ítem (1 al 15), y la nota promedio total del cada ec
# son válidas las evaluaciones realizadas por los miembros del mismo equipo.
# Tu código a continuación...
!hadoop fs -rm spark-warehouse/notas_equipos/*
Notas = ContextoSql.sql("""SELECT asi.equipo_pertenece AS EQUIPO, ROUND(AVG(nota_introduccior
                            ROUND(AVG(nota_problema),2) AS NOTA3, ROUND(AVG(nota_ventajas),2)
                            ROUND(AVG(nota_solucion),2) AS NOTA5, ROUND(AVG(nota_producto),2)
                            ROUND(AVG(nota_traccion),2) AS NOTA7, ROUND(AVG(nota_mercado),2)
                            ROUND(AVG(nota_competencia),2) AS NOTA9, ROUND(AVG(nota_modelo),2
                            ROUND(AVG(nota_inversion),2) AS NOTA11, ROUND(AVG(nota_contacto),
                            ROUND(AVG(nota_exposicion1),2) AS NOTA13, ROUND(AVG(nota_exposici
                            ROUND(AVG(nota_exposicion3),2) NOTA15,ROUND((AVG(nota_exposicion3
                            AVG(nota_mercado)+AVG(nota_traccion)+AVG(nota_producto)+AVG(nota_
                            FROM asistencia AS asi INNER JOIN evaluacion AS eva
                            ON asi.correo = eva.correo
                            WHERE SUBSTR (asi.equipo_pertenece, 15) != eva.equipo_evaluar
                            GROUP BY asi.equipo_pertenece
                            """)
ContextoSql.sql("DROP TABLE IF EXISTS notas_equipos")
Notas.show()
Notas.write.format("orc").saveAsTable("notas_equipos")


# 10. Consulte el mejor equipo evaluado por cada ítem (según nota promedio. En caso de empate
# y el mejor equipo según el promedio total.
# Tu código a continuación...
Notas = ContextoSql.sql("""SELECT SUBSTR (asi.equipo_pertenece, 15) AS EQUIPO, ROUND(AVG(nota
                            ROUND(AVG(nota_problema),2) AS NOTA3, ROUND(AVG(nota_ventajas),2)
                            ROUND(AVG(nota_solucion),2) AS NOTA5, ROUND(AVG(nota_producto),2)
                            ROUND(AVG(nota_traccion),2) AS NOTA7, ROUND(AVG(nota_mercado),2)
                            ROUND(AVG(nota_competencia),2) AS NOTA9, ROUND(AVG(nota_modelo),2
                            ROUND(AVG(nota_inversion),2) AS NOTA11, ROUND(AVG(nota_contacto),
                            ROUND(AVG(nota_exposicion1),2) AS NOTA13, ROUND(AVG(nota_exposici
                            ROUND(AVG(nota_exposicion3),2) NOTA15,ROUND((AVG(nota_exposicion3
```

```
                            AVG(nota_mercado)+AVG(nota_traccion)+AVG(nota_producto)+AVG(nota_
                            FROM asistencia AS asi INNER JOIN evaluacion AS eva
                            ON asi.correo = eva.correo
                            WHERE SUBSTR (asi.equipo_pertenece, 15) != eva.equipo_evaluar
                            GROUP BY asi.equipo_pertenece
                            ORDER BY NOTA1 DESC, NOTA2 DESC, NOTA3 DESC, NOTA4 DESC, NOTA5 DE
                            """)

Notas.select(['NOTA1', 'EQUIPO']).sort(['NOTA1'], ascending=[False]).show(1)
Notas.select(['NOTA2', 'EQUIPO']).sort(['NOTA2'], ascending=[False]).show(1)
Notas.select(['NOTA3', 'EQUIPO']).sort(['NOTA3'], ascending=[False]).show(1)
Notas.select(['NOTA4', 'EQUIPO']).sort(['NOTA4'], ascending=[False]).show(1)
Notas.select(['NOTA5', 'EQUIPO']).sort(['NOTA5'], ascending=[False]).show(1)
Notas.select(['NOTA6', 'EQUIPO']).sort(['NOTA6'], ascending=[False]).show(1)
Notas.select(['NOTA7', 'EQUIPO']).sort(['NOTA7'], ascending=[False]).show(1)
Notas.select(['NOTA8', 'EQUIPO']).sort(['NOTA8'], ascending=[False]).show(1)
Notas.select(['NOTA9', 'EQUIPO']).sort(['NOTA9'], ascending=[False]).show(1)
Notas.select(['NOTA10', 'EQUIPO']).sort(['NOTA10'], ascending=[False]).show(1)
Notas.select(['NOTA11', 'EQUIPO']).sort(['NOTA11'], ascending=[False]).show(1)
Notas.select(['NOTA12', 'EQUIPO']).sort(['NOTA12'], ascending=[False]).show(1)
Notas.select(['NOTA13', 'EQUIPO']).sort(['NOTA13'], ascending=[False]).show(1)
Notas.select(['NOTA14', 'EQUIPO']).sort(['NOTA14'], ascending=[False]).show(1)
Notas.select(['NOTA15', 'EQUIPO']).sort(['NOTA15'], ascending=[False]).show(1)
Notas.select(['AVG_EQUIPO', 'EQUIPO']).sort(['AVG_EQUIPO'], ascending=[False]).show(1)


# 11. Consulte el peor equipo evaluado por cada ítem (según nota promedio. En caso de empate
# y el peor equipo según el promedio total.
# Tu código a continuación...
Notas = ContextoSql.sql("""SELECT SUBSTR (asi.equipo_pertenece, 15) AS EQUIPO, ROUND(AVG(nota
                            ROUND(AVG(nota_problema),2) AS NOTA3, ROUND(AVG(nota_ventajas),2)
                            ROUND(AVG(nota_solucion),2) AS NOTA5, ROUND(AVG(nota_producto),2)
                            ROUND(AVG(nota_traccion),2) AS NOTA7, ROUND(AVG(nota_mercado),2)
                            ROUND(AVG(nota_competencia),2) AS NOTA9, ROUND(AVG(nota_modelo),2
                            ROUND(AVG(nota_inversion),2) AS NOTA11, ROUND(AVG(nota_contacto),
                            ROUND(AVG(nota_exposicion1),2) AS NOTA13, ROUND(AVG(nota_exposici
                            ROUND(AVG(nota_exposicion3),2) NOTA15,ROUND((AVG(nota_exposicion3
                            AVG(nota_mercado)+AVG(nota_traccion)+AVG(nota_producto)+AVG(nota_
                            FROM asistencia AS asi INNER JOIN evaluacion AS eva
                            ON asi.correo = eva.correo
                            WHERE SUBSTR (asi.equipo_pertenece, 15) != eva.equipo_evaluar
                            GROUP BY asi.equipo_pertenece
                            ORDER BY NOTA1 DESC, NOTA2 DESC, NOTA3 DESC, NOTA4 DESC, NOTA5 DE
                            """)


Notas.select(['NOTA1', 'EQUIPO']).sort(['NOTA1'], ascending=[True]).show(1)
Notas.select(['NOTA2', 'EQUIPO']).sort(['NOTA2'], ascending=[True]).show(1)
Notas.select(['NOTA3', 'EQUIPO']).sort(['NOTA3'], ascending=[True]).show(1)
Notas.select(['NOTA4', 'EQUIPO']).sort(['NOTA4'], ascending=[True]).show(1)
Notas.select(['NOTA5', 'EQUIPO']).sort(['NOTA5'], ascending=[True]).show(1)
Notas.select(['NOTA6', 'EQUIPO']).sort(['NOTA6'], ascending=[True]).show(1)
```

```
Notas.select(['NOTA7', 'EQUIPO']).sort(['NOTA7'], ascending=[True]).show(1)
Notas.select(['NOTA8', 'EQUIPO']).sort(['NOTA8'], ascending=[True]).show(1)
Notas.select(['NOTA9', 'EQUIPO']).sort(['NOTA9'], ascending=[True]).show(1)
Notas.select(['NOTA10', 'EQUIPO']).sort(['NOTA10'], ascending=[True]).show(1)
Notas.select(['NOTA11', 'EQUIPO']).sort(['NOTA11'], ascending=[True]).show(1)
Notas.select(['NOTA12', 'EQUIPO']).sort(['NOTA12'], ascending=[True]).show(1)
Notas.select(['NOTA13', 'EQUIPO']).sort(['NOTA13'], ascending=[True]).show(1)
Notas.select(['NOTA14', 'EQUIPO']).sort(['NOTA14'], ascending=[True]).show(1)
Notas.select(['NOTA15', 'EQUIPO']).sort(['NOTA15'], ascending=[True]).show(1)
Notas.select(['AVG_EQUIPO', 'EQUIPO']).sort(['AVG_EQUIPO'], ascending=[True]).show(1)


# 12. Consulte la lista de estudiantes con la correspondiente nota obtenida en la presentació
# (nota promedio total de la evaluación obtenida por el equipo)
# Tu código a continuación...
consulta=ContextoSql.sql('''SELECT asi.correo,nota.AVG_EQUIPO
                        FROM asistencia AS asi INNER JOIN notas_equipos AS nota
                        ON asi.equipo_pertenece = nota.EQUIPO
                        group by asi.correo,nota.AVG_EQUIPO
                        order by asi.correo
                        ''')
consulta.show(consulta.count())
```