0. Prepara o spark e java, encontra o spark, cria contexto

modifique para o seu caso...

In [1]:
```
import os;


# os.environ["SPARK_HOME"] = "Users/guibs/AppData/Local/Packages/PythonSoftwareFoundation.Pyth
# os.environ["JAVA_HOME"] ="Program Files/Java/jre1.8.0_202"

#os.environ["SPARK_HOME"] = "/Users/pedro/servers/spark-3.1.1-bin-hadoop2.7"
#os.environ["JAVA_HOME"]="/Library/Java/JavaVirtualMachines/adoptopenjdk-8.jdk/Contents/Home"
!java -version
```

```
java version "1.8.0_202"
Java(TM) SE Runtime Environment (build 1.8.0_202-b08)
Java HotSpot(TM) 64-Bit Server VM (build 25.202-b08, mixed mode)
```

Prepara o pyspark e spark context

In [2]:
```
# import findspark

# findspark.init()

import pyspark
```

In [3]:
```
from pyspark.sql import SparkSession

spark = SparkSession.builder.master("local[*]").getOrCreate()
print ("Running Spark Version ", spark.version)

#OLD deprecated:
#from pyspark import SparkContext

#sc = SparkContext('local')
#print ("Running Spark Version ", sc.version)
```

```
Running Spark Version  3.1.3
```

NOTA: EU tive de resolver um erro estranho, qdo corria no Mac OS sem rede:

NOTA - ERROR: Spark fails to start in local mode when disconnected [Possible bug in handling IPv6 in Spark??] am not sure if this will help you, but it solved my problem on Mac. 1) Get your hostname. (In terminal, this is usually the first part of the line (before the @ in Linux, before the : in Mac)) (In Mac, you can also type hostname in terminal to get your hostname) 2) In /etc/hosts add: 127.0.0.1 whatever-your-hostname-is For me, I originally had 127.0.0.1 localhost but I changed it to 127.0.0.1 my-hostname Save this change and retry pyspark. O que fiz: Macs-MacBook-Pro-4:~ pedro $hostnameMacs-MacBook-Pro-4.localMacs-MacBook-Pro-4:spark-3.2.1-bin-hadoop3.2pedro$ cp /etc/hosts /etc/hostsBKUP cp: /etc/hostsBKUP: Permission denied Macs-MacBook-Pro-4:spark-3.2.1-bin-hadoop3.2 pedro$ sudo pico /etc/hosts O hosts estava: 127.0.0.1 localhost 255.255.255.255 broadcasthost ::1 localhost O hosts ficou: 127.0.0.1 Macs-MacBook-Pro-4.local 255.255.255.255 broadcasthost ::1 Macs-MacBook-Pro-4.local (nota: nao esquecr de repor localhost mais tarde, pode depois falhar com outras appls (?)

Continuacao

ver o SPark a correr no endereco: http://localhost:4040/

In [4]:
```
from pyspark.conf import SparkConf
conf = SparkConf()
print (conf.toDebugString())
```

```
spark.app.name=pyspark-shell
spark.master=local[*]
spark.submit.deployMode=client
spark.submit.pyFiles=
spark.ui.showConsoleProgress=true
```

PART 1- My first pyspark app

What does the next code do?

what changes if you increase the number of samples?

why?

In [5]:
```
# useful to have this code snippet to avoid getting an error in case forgeting
# to close spark

try:
    spark.stop()
except:
    pass

# Using findspark to find automatically the spark folder
import findspark
findspark.init()

# import python libraries
import random

# initialize
from pyspark.sql import SparkSession
spark = SparkSession.builder.master("local[*]").getOrCreate()
num_samples = 1000
#num_samples = 100000000

def inside(p):
    x, y = random.random(), random.random()
    return x*x + y*y < 1

count = spark.sparkContext.parallelize(range(0, num_samples)).filter(inside).count()
pi = 4 * count / num_samples
print(pi)
```

```
3.08
```

PART 1- SQL

Now lets do some basic SQL with tables emp and dep

In [6]:
```
employees = spark.read.json('C:/Users/guibs/Documents/GitHub/SGD/Labs/lab9_pyspark/employee.js

# Print the schema in a tree format

employees.printSchema()

employees.select("name").show(20)
```

```
root
 |-- age: string (nullable = true)
 |-- id: string (nullable = true)
 |-- job: string (nullable = true)
 |-- name: string (nullable = true)
 |-- ndep: string (nullable = true)

+-------+
|   name|
+-------+
| satish|
|krishna|
|  amith|
|  javed|
| prudvi|
|   arya|
|    joy|
|   jack|
|  brown|
+-------+
```

In [7]:
```
dep = spark.read.json('C:/Users/guibs/Documents/GitHub/SGD/Labs/lab9_pyspark/dep.json')
# Print the schema in a tree format

dep.printSchema()

dep.select("*").show(20)
```

```
root
 |-- dname: string (nullable = true)
 |-- location: string (nullable = true)
 |-- ndep: string (nullable = true)

+----------+--------+----+
|     dname|location|ndep|
+----------+--------+----+
|     SALES| Coimbra|   1|
| MARKETING| Coimbra|   2|
| LOGISTICS|  Lisbon|   3|
|MANAGEMENT|   Porto|   4|
+----------+--------+----+
```

#OLD DEPRECATED: #dep.registerTempTable("dep") #employees.registerTempTable("employees")

In [8]:
```
dep.createOrReplaceTempView("dep")
employees.createOrReplaceTempView("employees")
```

write a query to select all employees aged > 23

In [9]:
```
spark.sql("SELECT * FROM employees WHERE age > 23 ORDER BY AGE DESC").show()
```

```
+---+----+---------+-------+----+
|age|  id|      job|   name|ndep|
+---+----+---------+-------+----+
| 39|1203|LOGISTICS|  amith|   3|
| 29|1206|    SALES|   arya|   1|
| 28|1202|MARKETING|krishna|   2|
| 25|1201|    SALES| satish|   1|
+---+----+---------+-------+----+
```

write a query to show number of employees for each age

In [10]:
```
ageGroups = spark.sql("SELECT COUNT (AGE),age FROM employees GROUP BY age")
ageGroups.show()
```

```
+----------+---+
|count(AGE)|age|
+----------+---+
|         1| 29|
|         1| 28|
|         5| 23|
|         1| 25|
|         1| 39|
+----------+---+
```

Show all info of departments of each employee together with all the employee info

In [11]:
```
employees.registerTempTable("emp")
empdep = spark.sql("SELECT * FROM employees,dep WHERE employees.ndep == dep.ndep")
empdep.show()
```

```
+---+----+---------+-------+----+---------+--------+----+
|age|  id|      job|   name|ndep|    dname|location|ndep|
+---+----+---------+-------+----+---------+--------+----+
| 25|1201|    SALES| satish|   1|    SALES| Coimbra|   1|
| 28|1202|MARKETING|krishna|   2|MARKETING| Coimbra|   2|
| 39|1203|LOGISTICS|  amith|   3|LOGISTICS|  Lisbon|   3|
| 23|1204|    SALES|  javed|   1|    SALES| Coimbra|   1|
| 23|1205|    SALES| prudvi|   1|    SALES| Coimbra|   1|
| 29|1206|    SALES|   arya|   1|    SALES| Coimbra|   1|
| 23|1207|MARKETING|    joy|   2|MARKETING| Coimbra|   2|
| 23|1208|MARKETING|   jack|   2|MARKETING| Coimbra|   2|
| 23|1209|LOGISTICS|  brown|   3|LOGISTICS|  Lisbon|   3|
+---+----+---------+-------+----+---------+--------+----+
```

Same, but restrict to name of employee and of dep

In [12]:
```
empdep = spark.sql("SELECT * FROM employees,dep WHERE employees.ndep == dep.ndep AND dep.dname
empdep.show()
```

```
+---+----+-----+-----+----+-----+--------+----+
|age|  id|  job| name|ndep|dname|location|ndep|
+---+----+-----+-----+----+-----+--------+----+
| 23|1204|SALES|javed|   1|SALES| Coimbra|   1|
+---+----+-----+-----+----+-----+--------+----+
```

Show employee and department name but only for department SALES

In [13]:
```
empVendas = spark.sql("SELECT * FROM employees,dep WHERE employees.ndep == dep.ndep AND dep.dn
empVendas.show()
```

```
+---+----+-----+------+----+-----+--------+----+
|age|  id|  job|  name|ndep|dname|location|ndep|
+---+----+-----+------+----+-----+--------+----+
| 25|1201|SALES|satish|   1|SALES| Coimbra|   1|
| 23|1204|SALES| javed|   1|SALES| Coimbra|   1|
| 23|1205|SALES|prudvi|   1|SALES| Coimbra|   1|
| 29|1206|SALES|  arya|   1|SALES| Coimbra|   1|
+---+----+-----+------+----+-----+--------+----+
```

Show how many employees there are for each age

In [14]:
```
ageGroups = spark.sql("SELECT COUNT(AGE) as Count,AGE FROM employees GROUP BY AGE")
ageGroups.show()
```

```
+-----+---+
|Count|AGE|
+-----+---+
|    1| 29|
|    1| 28|
|    5| 23|
|    1| 25|
|    1| 39|
+-----+---+
```

In [15]:
```
employees
```

Out[15]:
```
DataFrame[age: string, id: string, job: string, name: string, ndep: string]
```

Another way to read json to view directly

In [16]:
```
spark.sql("CREATE OR REPLACE TEMPORARY VIEW emp1b USING json OPTIONS" +
      " (path 'C:/Users/guibs/Documents/GitHub/SGD/Labs/lab9_pyspark/employee.json')")
spark.sql("select * from emp1b").show()
```

```
+---+----+---------+-------+----+
|age|  id|      job|   name|ndep|
+---+----+---------+-------+----+
| 25|1201|    SALES| satish|   1|
| 28|1202|MARKETING|krishna|   2|
| 39|1203|LOGISTICS|  amith|   3|
| 23|1204|    SALES|  javed|   1|
| 23|1205|    SALES| prudvi|   1|
| 29|1206|    SALES|   arya|   1|
| 23|1207|MARKETING|    joy|   2|
| 23|1208|MARKETING|   jack|   2|
| 23|1209|LOGISTICS|  brown|   3|
+---+----+---------+-------+----+
```

Writing it back to file ...

In [17]:
```
# employees.write.json("C:/Users/guibs/Documents/GitHub/SGD/Labs/lab9_pyspark/employees1c.json
```

PART 2 - postgres: now try to write into postgres and see in pgadmin if it worked...

In [18]:
```
from sqlalchemy import create_engine
engine = create_engine('postgresql://postgres:68296829@localhost:5432/empdep2')
employees.toPandas().to_sql('emp', engine)
dep.toPandas().to_sql('dep', engine)
```

Out[18]: 4

Now read from postgres and show contents

In [19]:
```
import pandas as pd

dbConnection = engine.connect();

emp2pd = pd.read_sql("select * from \"emp\"", dbConnection);
```

In [20]:
```
emp2=spark.createDataFrame(emp2pd)
emp2.show()
```

```
+-----+---+----+---------+-------+----+
|index|age|  id|      job|   name|ndep|
+-----+---+----+---------+-------+----+
|    0| 25|1201|    SALES| satish|   1|
|    1| 28|1202|MARKETING|krishna|   2|
|    2| 39|1203|LOGISTICS|  amith|   3|
|    3| 23|1204|    SALES|  javed|   1|
|    4| 23|1205|    SALES| prudvi|   1|
|    5| 29|1206|    SALES|   arya|   1|
|    6| 23|1207|MARKETING|    joy|   2|
|    7| 23|1208|MARKETING|   jack|   2|
|    8| 23|1209|LOGISTICS|  brown|   3|
+-----+---+----+---------+-------+----+
```

There are some difficulties currently with data types. Try to read from postgres and see employees whose age is larger than 25... does it work? why?

In [21]:
```
#DOES NOT WORK, BECAUSE COLUMNS ARE TEXT:

empABOVE25 = spark.createDataFrame(pd.read_sql("select * from \"emp\" where age > 25", dbConne

empABOVE25.show()
```

```
---------------------------------------------------------------------------
UndefinedFunction                         Traceback (most recent call last)
File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\sqlalchemy\engine\base.py:1819, in Connection._execute_context(self, dialect, constructor, statement, parameters, execution_options, *args, **kw)
   1818     if not evt_handled:
-> 1819         self.dialect.do_execute(
   1820             cursor, statement, parameters, context
   1821         )
   1823 if self._has_events or self.engine._has_events:

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\sqlalchemy\engine\default.py:732, in DefaultDialect.do_execute(self, cursor, statement, parameters, context)
    731 def do_execute(self, cursor, statement, parameters, context=None):
--> 732     cursor.execute(statement, parameters)

UndefinedFunction: operator does not exist: text > integer
LINE 1: select * from "emp" where age > 25
                                      ^
HINT:  No operator matches the given name and argument types. You might need to add explicit type casts.


The above exception was the direct cause of the following exception:

ProgrammingError                          Traceback (most recent call last)
Input In [21], in <cell line: 3>()
      1 #DOES NOT WORK, BECAUSE COLUMNS ARE TEXT:
----> 3 empABOVE25 = spark.createDataFrame(pd.read_sql("select * from \"emp\" where age > 25", dbConnection));
      5 empABOVE25.show()

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\pandas\io\sql.py:592, in read_sql(sql, con, index_col, coerce_float, params, parse_dates, columns, chunksize)
    583     return pandas_sql.read_table(
    584         sql,
    585         index_col=index_col,
   (...)
    589         chunksize=chunksize,
    590     )
    591 else:
--> 592     return pandas_sql.read_query(
    593         sql,
    594         index_col=index_col,
    595         params=params,
    596         coerce_float=coerce_float,
    597         parse_dates=parse_dates,
    598         chunksize=chunksize,
    599     )

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\pandas\io\sql.py:1557, in SQLDatabase.read_query(self, sql, index_col, coerce_float, parse_dates, params, chunksize, dtype)
   1509 """
   1510 Read SQL query into a DataFrame.
   1511
   (...)
   1553
   1554 """
   1555 args = _convert_params(sql, params)
-> 1557 result = self.execute(*args)
   1558 columns = result.keys()
   1560 if chunksize is not None:

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\pandas\io\sql.py:1402, in SQLDatabase.execute(self, *args, **kwargs)
   1400 def execute(self, *args, **kwargs):
   1401     """Simple passthrough to SQLAlchemy connectable"""
```

```
-> 1402     return self.connectable.execution_options().execute(*args, **kwargs)

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\LocalCache\lo
cal-packages\Python310\site-packages\sqlalchemy\engine\base.py:1291, in Connection.execute(sel
f, statement, *multiparams, **params)
   1282 if isinstance(statement, util.string_types):
   1283     util.warn_deprecated_20(
   1284         "Passing a string to Connection.execute() is "
   1285         "deprecated and will be removed in version 2.0.  Use the "
   (...)
   1288         "driver-level SQL string."
   1289     )
-> 1291     return self._exec_driver_sql(
   1292         statement,
   1293         multiparams,
   1294         params,
   1295         _EMPTY_EXECUTION_OPTS,
   1296         future=False,
   1297     )
   1299 try:
   1300     meth = statement._execute_on_connection

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\LocalCache\lo
cal-packages\Python310\site-packages\sqlalchemy\engine\base.py:1595, in Connection._exec_drive
r_sql(self, statement, multiparams, params, execution_options, future)
   1585         (
   1586             statement,
   1587             distilled_params,
   (...)
   1591             statement, distilled_parameters, execution_options
   1592         )
   1594 dialect = self.dialect
-> 1595 ret = self._execute_context(
   1596     dialect,
   1597     dialect.execution_ctx_cls._init_statement,
   1598     statement,
   1599     distilled_parameters,
   1600     execution_options,
   1601     statement,
   1602     distilled_parameters,
   1603 )
   1605 if not future:
   1606     if self._has_events or self.engine._has_events:

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\LocalCache\lo
cal-packages\Python310\site-packages\sqlalchemy\engine\base.py:1862, in Connection._execute_co
ntext(self, dialect, constructor, statement, parameters, execution_options, *args, **kw)
   1859             branched.close()
   1861 except BaseException as e:
-> 1862     self._handle_dbapi_exception(
   1863         e, statement, parameters, cursor, context
   1864     )
   1866 return result

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\LocalCache\lo
cal-packages\Python310\site-packages\sqlalchemy\engine\base.py:2043, in Connection._handle_dba
pi_exception(self, e, statement, parameters, cursor, context)
   2041     util.raise_(newraise, with_traceback=exc_info[2], from_=e)
   2042 elif should_wrap:
-> 2043     util.raise_(
   2044         sqlalchemy_exception, with_traceback=exc_info[2], from_=e
   2045     )
   2046 else:
   2047     util.raise_(exc_info[1], with_traceback=exc_info[2])

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\LocalCache\lo
cal-packages\Python310\site-packages\sqlalchemy\util\compat.py:207, in raise_(***failed resolv
ing arguments***)
    204     exception.__cause__ = replace_context
    206 try:
--> 207     raise exception
    208 finally:
```

```
    209     # credit to
    210     # https://cosmicpercolator.com/2016/01/13/exception-leaks-in-python-2-and-3/
    211     # as the __traceback__ object creates a cycle
    212     del exception, replace_context, from_, with_traceback

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\sqlalchemy\engine\base.py:1819, in Connection._execute_context(self, dialect, constructor, statement, parameters, execution_options, *args, **kw)
   1817                 break
   1818     if not evt_handled:
-> 1819         self.dialect.do_execute(
   1820             cursor, statement, parameters, context
   1821         )
   1823 if self._has_events or self.engine._has_events:
   1824     self.dispatch.after_cursor_execute(
   1825         self,
   1826         cursor,
   (...)
   1830         context.executemany,
   1831     )

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\sqlalchemy\engine\default.py:732, in DefaultDialect.do_execute(self, cursor, statement, parameters, context)
    731 def do_execute(self, cursor, statement, parameters, context=None):
--> 732     cursor.execute(statement, parameters)

ProgrammingError: (psycopg2.errors.UndefinedFunction) operator does not exist: text > integer
LINE 1: select * from "emp" where age > 25
                                      ^
HINT:  No operator matches the given name and argument types. You might need to add explicit type casts.

[SQL: select * from "emp" where age > 25]
(Background on this error at: https://sqlalche.me/e/14/f405)
```

In [22]:
```
#that's because it is defined as text...
employees.schema
```

Out[22]:
```
StructType(List(StructField(age,StringType,true),StructField(id,StringType,true),StructField(job,StringType,true),StructField(name,StringType,true),StructField(ndep,StringType,true)))
```

Look in pgadmin for the type of the columns of the table... is there something wrong?

Even if you specify a schema when you read the json file, it still does not work ... why is the following not working?

In [23]:
```
#Define custom schema
from pyspark.sql.types import *

schemaemp = StructType([StructField("age", IntegerType(), True),
                        StructField("id", LongType(), True),
                        StructField("job", StringType(), True),
                        StructField("name", StringType(), True),
                        StructField("ndep", IntegerType(), True)])

emp_with_schema = spark.read.schema(schemaemp) \
        .json("C:/Users/guibs/Documents/GitHub/SGD/Labs/lab9_pyspark/employee.json")

emp_with_schema.printSchema()
emp_with_schema.show()
```

```
root
 |-- age: integer (nullable = true)
 |-- id: long (nullable = true)
 |-- job: string (nullable = true)
 |-- name: string (nullable = true)
 |-- ndep: integer (nullable = true)

+----+----+---------+-------+----+
| age|  id|      job|   name|ndep|
+----+----+---------+-------+----+
|null|null|    SALES| satish|null|
|null|null|MARKETING|krishna|null|
|null|null|LOGISTICS|  amith|null|
|null|null|    SALES|  javed|null|
|null|null|    SALES| prudvi|null|
|null|null|    SALES|   arya|null|
|null|null|MARKETING|    joy|null|
|null|null|MARKETING|   jack|null|
|null|null|LOGISTICS|  brown|null|
+----+----+---------+-------+----+
```

Schemas, data types: now create employeesTYPES.json by removing double quotes in non-strings in the json file

The TYPES json file DOES NOT have double quotes in number, such as:"1" {"id" : "1201", "name" : "satish", "age" : "25","job":"SALES","ndep": "1"}} into {"id" : 1201, "name" : "satish", "age" : 25,"job":"SALES","ndep": 1}}

Did this solve the problem with data types?

In [24]:
```
employeesTYPED = spark.read.json('C:/Users/guibs/Documents/GitHub/SGD/Labs/lab9_pyspark/employ
employeesTYPED.schema
```

Out[24]:
```
StructType(List(StructField(age,LongType,true),StructField(id,LongType,true),StructField(job,S
tringType,true),StructField(name,StringType,true),StructField(ndep,LongType,true)))
```

and we can specify the schema when loading...

In [25]:
```
emp_with_schema = spark.read.schema(schemaemp) \
        .json("/Users/guibs/Documents/GitHub/SGD/Labs/lab9_pyspark/employeeParsed.json")

emp_with_schema.printSchema()
emp_with_schema.show()
```

```
root
 |-- age: integer (nullable = true)
 |-- id: long (nullable = true)
 |-- job: string (nullable = true)
 |-- name: string (nullable = true)
 |-- ndep: integer (nullable = true)

+---+----+---------+-------+----+
|age|  id|      job|   name|ndep|
+---+----+---------+-------+----+
| 25|1201|    SALES| satish|   1|
| 28|1202|MARKETING|krishna|   2|
| 39|1203|LOGISTICS|  amith|   3|
| 23|1204|    SALES|  javed|   1|
| 23|1205|    SALES| prudvi|   1|
| 29|1206|    SALES|   arya|   1|
| 23|1207|MARKETING|    joy|   2|
| 23|1208|MARKETING|   jack|   2|
| 23|1209|LOGISTICS|  brown|   3|
+---+----+---------+-------+----+
```

Now write the new typed dataset into a new table emp2. See in pgadmin if the data types are ok now....

In [26]:
```
employeesTYPED.toPandas().to_sql('emp2', engine)
```

Out[26]: 9

Finally, redo the query for age above 25 and show that it works....

In [27]:
```
empABOVE25 = spark.createDataFrame(pd.read_sql("select * from \"emp2\" where age > 25", dbConn

empABOVE25.show()
```

```
+-----+---+----+---------+-------+----+
|index|age|  id|      job|   name|ndep|
+-----+---+----+---------+-------+----+
|    1| 28|1202|MARKETING|krishna|   2|
|    2| 39|1203|LOGISTICS|  amith|   3|
|    5| 29|1206|    SALES|   arya|   1|
+-----+---+----+---------+-------+----+
```

What about if I read the original employee with strings but replace the contents of columns with cast? Does it work? why?

In [28]:
```
emp3 = spark.read.json(path="/Users/guibs/Documents/GitHub/SGD/Labs/lab9_pyspark/employee.json
emp3.withColumn("age", emp3["age"].cast("integer"))\
  .withColumn("id", emp3["id"].cast("long"))\
  .withColumn("ndep", emp3["ndep"].cast("integer"))

emp3.show()
```

```
+---+----+---------+-------+----+
|age|  id|      job|   name|ndep|
+---+----+---------+-------+----+
| 25|1201|    SALES| satish|   1|
| 28|1202|MARKETING|krishna|   2|
| 39|1203|LOGISTICS|  amith|   3|
| 23|1204|    SALES|  javed|   1|
| 23|1205|    SALES| prudvi|   1|
| 29|1206|    SALES|   arya|   1|
| 23|1207|MARKETING|    joy|   2|
| 23|1208|MARKETING|   jack|   2|
| 23|1209|LOGISTICS|  brown|   3|
+---+----+---------+-------+----+
```

In [29]:
```
#I think THE SCHEMA TYPES ARE STILL INCORRECT....
emp3.schema
```

Out[29]:
```
StructType(List(StructField(age,StringType,true),StructField(id,StringType,true),StructField(job,StringType,true),StructField(name,StringType,true),StructField(ndep,StringType,true)))
```

Now use the schema but also replace the field contents using casts. Did this work? Why?

In [30]:
```
#now correct IT USING SCHEMA...

#emp3 = spark.read.schema(schemaemp).json(path="/Users/pedro/Documents/Aulas/SGD/2022SGD/prati
emp3 = spark.read.json(path="/Users/guibs/Documents/GitHub/SGD/Labs/lab9_pyspark/employee.json

emp3b=emp3.withColumn("age", emp3["age"].cast("integer"))\
  .withColumn("id", emp3["id"].cast("long"))\
  .withColumn("ndep", emp3["ndep"].cast("integer"))

emp3b.show()
```

```
+---+----+---------+-------+----+
|age|  id|      job|   name|ndep|
+---+----+---------+-------+----+
| 25|1201|    SALES| satish|   1|
| 28|1202|MARKETING|krishna|   2|
| 39|1203|LOGISTICS|  amith|   3|
| 23|1204|    SALES|  javed|   1|
| 23|1205|    SALES| prudvi|   1|
| 29|1206|    SALES|   arya|   1|
| 23|1207|MARKETING|    joy|   2|
| 23|1208|MARKETING|   jack|   2|
| 23|1209|LOGISTICS|  brown|   3|
+---+----+---------+-------+----+
```

In [31]:
```
emp3b.schema
```

Out[31]:
```
StructType(List(StructField(age,IntegerType,true),StructField(id,LongType,true),StructField(job,StringType,true),StructField(name,StringType,true),StructField(ndep,IntegerType,true)))
```