```
{
 "cells": [
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "### Assignment 2\n",
    "Welcome to Assignment 2. This will be fun. It is the first time you actually access external data from ApacheSpark. \n",
    "\n",
    "#### You can also submit partial solutions\n",
    "\n",
    "Just make sure you hit the play button on each cell from top to down. There are three functions you have to implement. Please also make sure than on each change on a function you hit the play button again on the corresponding cell to make it available to the rest of this notebook.\n"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "This notebook is designed to run in a IBM Watson Studio default runtime (NOT the Watson Studio Apache Spark Runtime as the default runtime with 1 vCPU is free of charge). Therefore, we install Apache Spark in local mode for test purposes only. Please don't use it in production.\n",
    "\n",
    "In case you are facing issues, please read the following two documents first:\n",
    "\n",
    "https://github.com/IBM/skillsnetwork/wiki/Environment-Setup\n",
    "\n",
    "https://github.com/IBM/skillsnetwork/wiki/FAQ\n",
    "\n",
    "Then, please feel free to ask:\n",
    "\n",
    "https://coursera.org/learn/machine-learning-big-data-apache-spark/discussions/all\n",
    "\n",
    "Please make sure to follow the guidelines before asking a question:\n",
    "\n",
    "https://github.com/IBM/skillsnetwork/wiki/FAQ#im-feeling-lost-and-confused-please-help-me\n",
    "\n",
    "\n",
    "If running outside Watson Studio, this should work as well. In case you are running in an Apache Spark context outside Watson Studio, please remove the Apache Spark setup in the first notebook cells."
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "from IPython.display import Markdown, display\n",
    "def printmd(string):\n",
    "    display(Markdown('# <span style=\"color:red\">'+string+'</span>'))\n",
    "\n",
    "\n",
    "if ('sc' in locals() or 'sc' in globals()):\n",
    "    printmd('<<<<<!!!!! It seems that you are running in a IBM Watson Studio Apache Spark Notebook. Please run it in an IBM Watson Studio Default Runtime (without Apache Spark) !!!!!>>>>>')\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
```

```
     "!pip install pyspark==2.4.5"
    ]
   },
   {
    "cell_type": "code",
    "execution_count": null,
    "metadata": {},
    "outputs": [],
    "source": [
     "try:\n",
     "    from pyspark import SparkContext, SparkConf\n",
     "    from pyspark.sql import SparkSession\n",
     "except ImportError as e:\n",
     "    printmd('<<<<<!!!!! Please restart your kernel after installing Apache Spark
!!!!!>>>>>')"
    ]
   },
   {
    "cell_type": "code",
    "execution_count": null,
    "metadata": {},
    "outputs": [],
    "source": [
     "sc = SparkContext.getOrCreate(SparkConf().setMaster(\"local[*]\"))\n",
     "\n",
     "spark = SparkSession \\\n",
     "    .builder \\\n",
     "    .getOrCreate()"
    ]
   },
   {
    "cell_type": "markdown",
    "metadata": {},
    "source": [
     "This is the first function you have to implement. You are passed a dataframe object. We've
also registered the dataframe in the ApacheSparkSQL catalog - so you can also issue queries
against the \"washing\" table using \"spark.sql()\". Hint: To get an idea about the contents of
the catalog you can use: spark.catalog.listTables().\n",
     "So now it's time to implement your first function. You are free to use the dataframe API, SQL
or RDD API. In case you want to use the RDD API just obtain the encapsulated RDD using \"df.rdd\".
You can test the function by running one of the three last cells of this notebook, but please make
sure you run the cells from top to down since some are dependant of each other..."
    ]
   },
   {
    "cell_type": "code",
    "execution_count": null,
    "metadata": {},
    "outputs": [],
    "source": [
     "#Please implement a function returning the number of rows in the dataframe\n",
     "def count():\n",
     "    #TODO Please enter your code here, you are not required to use the template code
below\n",
     "    #some reference:
https://spark.apache.org/docs/latest/api/python/pyspark.sql.html#pyspark.sql.DataFrame\n",
     "    #some more help: https://www.w3schools.com/sql/sql_count_avg_sum.asp\n",
     "    return spark.sql('select ### as cnt from washing').first().cnt"
    ]
   },
   {
    "cell_type": "markdown",
    "metadata": {},
    "source": [
     "Now it's time to implement the second function. Please return an integer containing the
number of fields (columns). The most easy way to get this is using the dataframe API. Hint: You
might find the dataframe API documentation useful:
http://spark.apache.org/docs/latest/api/python/pyspark.sql.html#pyspark.sql.DataFrame"
    ]
```

```
   },
   {
    "cell_type": "code",
    "execution_count": null,
    "metadata": {},
    "outputs": [],
    "source": [
     "def getNumberOfFields():\n",
     "    #TODO Please enter your code here, you are not required to use the template code
below\n",
     "    #some reference:
https://spark.apache.org/docs/latest/api/python/pyspark.sql.html#pyspark.sql.DataFrame\n",
     "    return len(df.###)"
    ]
   },
   {
    "cell_type": "markdown",
    "metadata": {},
    "source": [
     "Finally, please implement a function which returns a (python) list of string values of the
field names in this data frame. Hint: Just copy&past doesn't work because the auto-grader will
create a random data frame for testing, so please use the data frame API as well. Again, this is
the link to the documentation:
http://spark.apache.org/docs/latest/api/python/pyspark.sql.html#pyspark.sql.DataFrame"
    ]
   },
   {
    "cell_type": "code",
    "execution_count": null,
    "metadata": {},
    "outputs": [],
    "source": [
     "def getFieldNames():\n",
     "    #TODO Please enter your code here, you are not required to use the template code
below\n",
     "    #some reference:
https://spark.apache.org/docs/latest/api/python/pyspark.sql.html#pyspark.sql.DataFrame\n",
     "    return df.###"
    ]
   },
   {
    "cell_type": "markdown",
    "metadata": {},
    "source": [
     "Now it is time to grab a PARQUET file and create a dataframe out of it. Using SparkSQL you
can handle it like a database. "
    ]
   },
   {
    "cell_type": "code",
    "execution_count": null,
    "metadata": {},
    "outputs": [],
    "source": [
     "!wget https://github.com/IBM/coursera/blob/master/coursera_ds/washing.parquet?raw=true\n",
     "!mv washing.parquet?raw=true washing.parquet"
    ]
   },
   {
    "cell_type": "code",
    "execution_count": null,
    "metadata": {},
    "outputs": [],
    "source": [
     "df = spark.read.parquet('washing.parquet')\n",
     "df.createOrReplaceTempView('washing')\n",
     "df.show()"
    ]
   },
```

```
   {
    "cell_type": "markdown",
    "metadata": {},
    "source": [
     "The following cell can be used to test your count function"
    ]
   },
   {
    "cell_type": "code",
    "execution_count": null,
    "metadata": {},
    "outputs": [],
    "source": [
     "cnt = None\n",
     "nof = None\n",
     "fn = None\n",
     "\n",
     "cnt = count()\n",
     "print(cnt)"
    ]
   },
   {
    "cell_type": "markdown",
    "metadata": {},
    "source": [
     "The following cell can be used to test your getNumberOfFields function"
    ]
   },
   {
    "cell_type": "code",
    "execution_count": null,
    "metadata": {},
    "outputs": [],
    "source": [
     "nof = getNumberOfFields()\n",
     "print(nof)"
    ]
   },
   {
    "cell_type": "markdown",
    "metadata": {},
    "source": [
     "The following cell can be used to test your getFieldNames function"
    ]
   },
   {
    "cell_type": "code",
    "execution_count": null,
    "metadata": {},
    "outputs": [],
    "source": [
     "fn = getFieldNames()\n",
     "print(fn)"
    ]
   },
   {
    "cell_type": "markdown",
    "metadata": {},
    "source": [
     "Congratulations, you are done. So please submit your solutions to the grader now.\n",
     "\n",
     "# Start of Assignment-Submission\n",
     "\n",
     "The first thing we need to do is to install a little helper library for submitting the
solutions to the coursera grader:\n"
    ]
   },
   {
    "cell_type": "code",
```

```
    "execution_count": null,
    "metadata": {},
    "outputs": [],
    "source": [
     "!rm -f rklib.py\n",
     "!wget https://raw.githubusercontent.com/IBM/coursera/master/rklib.py"
    ]
   },
   {
    "cell_type": "markdown",
    "metadata": {},
    "source": [
     "Now it’s time to submit first solution. Please make sure that the token variable contains a
 valid submission token. You can obtain it from the coursera web page of the course using the
 grader section of this assignment.\n",
     "\n",
     "Please specify you email address you are using with cousera as well.\n"
    ]
   },
   {
    "cell_type": "code",
    "execution_count": null,
    "metadata": {},
    "outputs": [],
    "source": [
     "from rklib import submit, submitAll\n",
     "import json\n",
     "\n",
     "key = \"SVDiVSHNEeiDqw70MIp2vA\"\n",
     "\n",
     "if type(23) != type(cnt):\n",
     "    raise ValueError('Please make sure that \"cnt\" is a number')\n",
     "    \n",
     "if type(23) != type(nof):\n",
     "    raise ValueError('Please make sure that \"nof\" is a number')\n",
     "\n",
     "if type([]) != type(fn):\n",
     "    raise ValueError('Please make sure that \"fn\" is a list')\n",
     "\n",
     "email = #### your code here ###\n",
     "token = #### your code here ### (have a look here if you need more information on how to
 obtain the token https://youtu.be/GcDo0Rwe06U?t=276)\n",
     "\n",
     "parts_data = {}\n",
     "parts_data[\"2FjQw\"] = json.dumps(cnt)\n",
     "parts_data[\"j8gMs\"] = json.dumps(nof)\n",
     "parts_data[\"xaauC\"] = json.dumps(fn)\n",
     "\n",
     "\n",
     "submitAll(email, token, key, parts_data)"
    ]
   }
  ],
  "metadata": {
   "kernelspec": {
    "display_name": "Python 3.6",
    "language": "python",
    "name": "python3"
   },
   "language_info": {
    "codemirror_mode": {
     "name": "ipython",
     "version": 3
    },
    "file_extension": ".py",
    "mimetype": "text/x-python",
    "name": "python",
    "nbconvert_exporter": "python",
    "pygments_lexer": "ipython3",
```

```
    "version": "3.6.9"
   }
  },
  "nbformat": 4,
  "nbformat_minor": 1
 }
```