```
{
 "cells": [
  {
   "cell_type": "markdown",
   "metadata": {
    "application/vnd.databricks.v1+cell": {
     "cellMetadata": {},
     "inputWidgets": {},
     "nuid": "8e813f0a-a5df-4ec7-b0cd-8b55975b4ab0",
     "showTitle": false,
     "title": ""
    }
   },
   "source": [
    "## Common Error Types\n",
    "\n",
    "1. Syntax Errors\n",
    "2. Permission Errors\n",
    "3. Reference Errors\n",
    "4. Language Errors"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {
    "application/vnd.databricks.v1+cell": {
     "cellMetadata": {},
     "inputWidgets": {},
     "nuid": "70a93653-c7b9-4049-b26c-2a8a1d55934d",
     "showTitle": false,
     "title": ""
    }
   },
   "source": [
    "### Example: Syntax Errors"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 0,
   "metadata": {
    "application/vnd.databricks.v1+cell": {
     "cellMetadata": {
      "byteLimit": 2048000,
      "rowLimit": 10000
     },
     "inputWidgets": {},
     "nuid": "14e6241b-3376-4261-9f72-39839b741db1",
     "showTitle": false,
     "title": ""
    }
   },
```

```
   "outputs": [
    {
     "output_type": "display_data",
     "data": {
      "text/plain": [
       "\u001B[0;36m  File \u001B[0;32m<command-
2793793367620319>:2\u001B[0;36m\u001B[0m\n",
       "\u001B[0;31m    print('Syntax)\u001B[0m\n",
       "\u001B[0m          ^\u001B[0m\n",
       "\u001B[0;31mSyntaxError\u001B[0m\u001B[0;31m:\u001B[0m
unterminated string literal (detected at line 2)\n"
      ]
     },
     "metadata": {
      "application/vnd.databricks.v1+output": {
       "arguments": {},
       "data": "\u001B[0;36m  File \u001B[0;32m<command-
2793793367620319>:2\u001B[0;36m\u001B[0m\n\u001B[0;31m
print('Syntax)\u001B[0m\n\u001B[0m
^\u001B[0m\n\u001B[0;31mSyntaxError\u001B[0m\u001B[0;31m:\u001B[0m
unterminated string literal (detected at line 2)\n",
       "errorSummary": "<span class='ansi-red-fg'>SyntaxError</span>:
unterminated string literal (detected at line 2) (<command-
2793793367620319>, line 2)",
       "errorTraceType": "ansi",
       "metadata": {},
       "type": "ipynbError"
      }
     },
     "output_type": "display_data"
    }
   ],
   "source": [
    "print('No syntax error')\n",
    "print('Syntax)"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {
    "application/vnd.databricks.v1+cell": {
     "cellMetadata": {},
     "inputWidgets": {},
     "nuid": "e3dee7fb-fe26-40af-bb12-bf35cf6aa5eb",
     "showTitle": false,
     "title": ""
    }
   },
   "source": [
    "## Debugging Tools"
   ]
  },
  {
   "cell_type": "markdown",
```

```
   "metadata": {
    "application/vnd.databricks.v1+cell": {
     "cellMetadata": {},
     "inputWidgets": {},
     "nuid": "961ba492-7bf7-4643-9a13-3c866f5a0db9",
     "showTitle": false,
     "title": ""
    }
   },
   "source": [
    "### Debugging in Databricks notebooks\n",
    "\n",
    "Notebooks run on Databricks Runtime 11.2 and above support [The
Python Debugger](https://docs.python.org/3/library/pdb.html) (pdb).\n",
    "\n",
    "Some examples of using pdb in a notebook:\n",
    "- Use `%debug` to debug from the last exception. This is helpful
when you run into an unexpected error and are trying to debug the cause
(similar to `pdb.pm()`).\n",
    "- Use `%pdb on` to automatically start the interactive debugger
after exceptions (but before program terminates).\n",
    "\n",
    "Note that when you use these commands, you must finish using the
debugger before you can run any other cell. Here are a few ways to exit
the debugger:\n",
    "- `c` or `continue` to finish running the cell.\n",
    "- `exit` to throw an error and stop code execution.\n",
    "- Cancel the command by clicking `Cancel` next to the output box."
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {
    "application/vnd.databricks.v1+cell": {
     "cellMetadata": {},
     "inputWidgets": {},
     "nuid": "fbc9e45a-e0dd-4a07-8305-308fb0d77c11",
     "showTitle": false,
     "title": ""
    }
   },
   "source": [
    "### `%debug` : Post-mortem debugging\n",
    "To use `%debug` in Databricks notebooks:\n",
    "1. Run commands in the notebook until an exception is raised.\n",
    "2. Run `%debug` in a new cell. The debugger starts running in the
output area of the cell.\n",
    "3. To inspect a variable, type the variable name in the input field
and press **Enter**.  \n",
    "4. You can change context and perform other debugger tasks, like
variable inspection, using these commands. For the complete list of
debugger commands, see the [pdb
documentation](https://docs.python.org/3/library/pdb.html). Type the
letter and then press **Enter**.  \n",
```

```
    "   - `n`: next line\n",
    "   - `u`: move up 1 level out of the current stack frame\n",
    "   - `d`: move down 1 level out of the current stack frame\n",
    "5. Exit the debugger using one of the methods described in the first
cell of this notebook.\n",
    "\n",
    "Below is an example following these steps using `%debug`."
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 0,
   "metadata": {
    "application/vnd.databricks.v1+cell": {
     "cellMetadata": {
      "byteLimit": 2048000,
      "rowLimit": 10000
     },
     "inputWidgets": {},
     "nuid": "8a630e6b-1bfa-4b23-abe5-9325e2c11dad",
     "showTitle": false,
     "title": ""
    }
   },
   "outputs": [
    {
     "output_type": "stream",
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "You're score is: 0.1\n"
     ]
    }
   ],
   "source": [
    "%python\n",
    "class ComplexSystem1:\n",
    "  def getAccuracy(self, correct, total):\n",
    "    # ...\n",
    "    accuracy = correct / total\n",
    "    # ...\n",
    "    return accuracy\n",
    "  \n",
    "class UserTest:\n",
    "  def __init__(self, system, correct, total):\n",
    "    self.system = system\n",
    "    self.correct = correct\n",
    "    self.total = total \n",
    "    \n",
    "  def printScore(self):\n",
    "    print(f\"You're score is: {self.system.getAccuracy(self.correct,
self.total)}\")\n",
    "  \n",
    "test = UserTest(\n",
```

```
    "  system = ComplexSystem1(),\n",
    "  correct = 10,\n",
    "  total = 100\n",
    ")\n",
    " \n",
    "test.printScore()"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 0,
   "metadata": {
    "application/vnd.databricks.v1+cell": {
     "cellMetadata": {},
     "inputWidgets": {},
     "nuid": "67714ae4-464a-4a52-8e68-14b593aa3854",
     "showTitle": false,
     "title": ""
    }
   },
   "outputs": [],
   "source": [
    "%debug"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {
    "application/vnd.databricks.v1+cell": {
     "cellMetadata": {},
     "inputWidgets": {},
     "nuid": "91ac604b-78f5-478c-b036-7fdfb69c6ee3",
     "showTitle": false,
     "title": ""
    }
   },
   "source": [
    "### `%pdb on` : Pre-mortem debugging\n",
    "To use `%pdb on` in Databricks notebooks:\n",
    "1. Turn auto pdb on by running `%pdb on` in the first cell of your notebook.\n",
    "2. Run commands in the notebook until an exception is raised. The interactive debugger starts.\n",
    "3. To inspect a variable, type the variable name in the input field and press **Enter**.  \n",
    "4. You can change context and perform other debugger tasks, like variable inspection, using these commands. For the complete list of debugger commands, see the [pdb documentation](https://docs.python.org/3/library/pdb.html). Type the letter and then press **Enter**.  \n",
    "  - `n`: next line\n",
    "  - `u`: move up 1 level out of the current stack frame\n",
    "  - `d`: move down 1 level out of the current stack frame\n",
```

```
    "5. Exit the debugger using one of the methods described in the first
cell of this notebook.\n",
    "\n",
    "Below is an example following these steps using `%pdb on`."
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 0,
   "metadata": {
    "application/vnd.databricks.v1+cell": {
     "cellMetadata": {},
     "inputWidgets": {},
     "nuid": "577027fe-ebce-4536-9aec-72fa30198257",
     "showTitle": false,
     "title": ""
    }
   },
   "outputs": [],
   "source": [
    "%pdb on"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 0,
   "metadata": {
    "application/vnd.databricks.v1+cell": {
     "cellMetadata": {
      "byteLimit": 2048000,
      "rowLimit": 10000
     },
     "inputWidgets": {},
     "nuid": "42576489-0250-42cb-92eb-7f96a38fe2ea",
     "showTitle": false,
     "title": ""
    }
   },
   "outputs": [
    {
     "output_type": "stream",
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "> \u001B[0;32m/databricks/python/lib/python3.10/site-
packages/ipykernel/kernelbase.py\u001B[0m(1219)\u001B[0;36m_input_request
\u001B[0;34m()\u001B[0m\n\u001B[0;32m   1217 \u001B[0;31m
\u001B[0;32mexcept\u001B[0m
\u001B[0mKeyboardInterrupt\u001B[0m\u001B[0;34m:\u001B[0m\u001B[0;34m\u00
1B[0m\u001B[0;34m\u001B[0m\u001B[0m\n\u001B[0m\u001B[0;32m   1218
\u001B[0;31m                \u001B[0;31m# re-raise KeyboardInterrupt, to
truncate
traceback\u001B[0m\u001B[0;34m\u001B[0m\u001B[0;34m\u001B[0m\u001B[0m\n\u
001B[0m\u001B[0;32m-> 1219 \u001B[0;31m
```

```
\u001B[0;32mraise\u001B[0m
\u001B[0mKeyboardInterrupt\u001B[0m\u001B[0;34m(\u001B[0m\u001B[0;34m\"In
terrupted by user\"\u001B[0m\u001B[0;34m)\u001B[0m
\u001B[0;32mfrom\u001B[0m
\u001B[0;32mNone\u001B[0m\u001B[0;34m\u001B[0m\u001B[0;34m\u001B[0m\u001B
[0m\n\u001B[0m\u001B[0;32m   1220 \u001B[0;31m
\u001B[0;32mexcept\u001B[0m
\u001B[0mException\u001B[0m\u001B[0;34m:\u001B[0m\u001B[0;34m\u001B[0m\u0
01B[0;34m\u001B[0m\u001B[0m\n\u001B[0m\u001B[0;32m   1221 \u001B[0;31m
\u001B[0mself\u001B[0m\u001B[0;34m.\u001B[0m\u001B[0mlog\u001B[0m\u001B[0
;34m.\u001B[0m\u001B[0mwarning\u001B[0m\u001B[0;34m(\u001B[0m\u001B[0;34m
\"Invalid Message:\"\u001B[0m\u001B[0;34m,\u001B[0m
\u001B[0mexc_info\u001B[0m\u001B[0;34m=\u001B[0m\u001B[0;32mTrue\u001B[0m
\u001B[0;34m)\u001B[0m\u001B[0;34m\u001B[0m\u001B[0;34m\u001B[0m\u001B[0m
\n\u001B[0m\n"
     ]
    },
    {
     "output_type": "display_data",
     "data": {
      "text/plain": [
       "ipdb>  "
      ]
     },
     "metadata": {},
     "output_type": "display_data"
    },
    {
     "output_type": "stream",
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "*** NameError: name 'test' is not defined\n"
     ]
    },
    {
     "output_type": "display_data",
     "data": {
      "text/plain": [
       "ipdb>  "
      ]
     },
     "metadata": {},
     "output_type": "display_data"
    },
    {
     "output_type": "display_data",
     "data": {
      "text/html": [
       "<style scoped>\n",
       "  .ansiout {\n",
       "    display: block;\n",
       "    unicode-bidi: embed;\n",
       "    white-space: pre-wrap;\n",
```

```
       "    word-wrap: break-word;\n",
       "    word-break: break-all;\n",
       "    font-family: \"Source Code Pro\", \"Menlo\", monospace;;\n",
       "    font-size: 13px;\n",
       "    color: #555;\n",
       "    margin-left: 4px;\n",
       "    line-height: 19px;\n",
       "  }\n",
       "</style>"
      ]
     },
     "metadata": {
      "application/vnd.databricks.v1+output": {
       "arguments": {},
       "data": "",
       "errorSummary": "Cancelled",
       "errorTraceType": "html",
       "metadata": {},
       "type": "ipynbError"
      }
     },
     "output_type": "display_data"
    }
   ],
   "source": [
    "%python\n",
    "class ComplexSystem1:\n",
    "  def getAccuracy(self, correct, total):\n",
    "    # ...\n",
    "    accuracy = correct / total\n",
    "    # ...\n",
    "    return accuracy\n",
    "  \n",
    "class UserTest:\n",
    "  def __init__(self, system, correct, total):\n",
    "    self.system = system\n",
    "    self.correct = correct\n",
    "    self.total = 0 \n",
    "    \n",
    "  def printScore(self):\n",
    "    print(f\"You're score is: {self.system.getAccuracy(self.correct,
self.total)}\")\n",
    "  \n",
    "test = UserTest(\n",
    "  system = ComplexSystem1(),\n",
    "  correct = 10,\n",
    "  total = 100\n",
    ")\n",
    " \n",
    "test.printScore()"
   ]
  },
  {
   "cell_type": "markdown",
```

```
   "metadata": {
    "application/vnd.databricks.v1+cell": {
     "cellMetadata": {},
     "inputWidgets": {},
     "nuid": "3c704a8e-d5e2-4143-9529-5a307f686535",
     "showTitle": false,
     "title": ""
    }
   },
   "source": [
    "### Variable Explorer"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 0,
   "metadata": {
    "application/vnd.databricks.v1+cell": {
     "cellMetadata": {
      "byteLimit": 2048000,
      "rowLimit": 10000
     },
     "inputWidgets": {},
     "nuid": "53d10aae-a12a-4089-b2e2-4e3b80a85eed",
     "showTitle": false,
     "title": ""
    }
   },
   "outputs": [],
   "source": [
    "from pyspark.sql.types import StructType,StructField, StringType,
IntegerType\n",
    "\n",
    "var_1 = \"string\"\n",
    "var_2 = {\n",
    "  \"key\": \"value\"\n",
    "}\n",
    "var_3 = 10\n",
    "columns = [\"language\",\"users_count\"]\n",
    "data = [(\"Java\", \"20000\"), (\"Python\", \"100000\"), (\"Scala\",
\"3000\")]\n",
    "\n",
    "data2 = [(\"James\",\"\",\"Smith\",\"36636\",\"M\",3000),\n",
    "    (\"Michael\",\"Rose\",\"\",\"40288\",\"M\",4000),\n",
    "    (\"Robert\",\"\",\"Williams\",\"42114\",\"M\",4000),\n",
    "    (\"Maria\",\"Anne\",\"Jones\",\"39192\",\"F\",4000),\n",
    "    (\"Jen\",\"Mary\",\"Brown\",\"\",\"F\",-1)\n",
    "  ]\n",
    "\n",
    "schema = StructType([ \\\n",
    "    StructField(\"firstname\",StringType(),True), \\\n",
    "    StructField(\"middlename\",StringType(),True), \\\n",
    "    StructField(\"lastname\",StringType(),True), \\\n",
    "    StructField(\"id\", StringType(), True), \\\n",
```

```
    "    StructField(\"gender\", StringType(), True), \\\n",
    "    StructField(\"salary\", IntegerType(), True) \\\n",
    "  ])\n",
    " \n",
    "df = spark.createDataFrame(data=data2,schema=schema)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 0,
   "metadata": {
    "application/vnd.databricks.v1+cell": {
     "cellMetadata": {
      "byteLimit": 2048000,
      "rowLimit": 10000
     },
     "inputWidgets": {},
     "nuid": "f9f55ab9-01c4-4b99-a5c9-335988c39cc5",
     "showTitle": false,
     "title": ""
    }
   },
   "outputs": [
    {
     "output_type": "stream",
     "name": "stdout",
     "output_type": "stream",
     "text": [
      ">
\u001B[0;32m/databricks/python_shell/dbruntime/display.py\u001B[0m(43)\u0
01B[0;36madd_custom_display_data\u001B[0;34m()\u001B[0m\n\u001B[0;32m
41 \u001B[0;31m                    raw=True)\n\u001B[0m\u001B[0;32m     42
\u001B[0;31m        \u001B[0;32mif\u001B[0m \u001B[0mreturn_code\u001B[0m
\u001B[0;34m==\u001B[0m
\u001B[0;36m1\u001B[0m\u001B[0;34m:\u001B[0m\u001B[0;34m\u001B[0m\u001B[0
;34m\u001B[0m\u001B[0m\n\u001B[0m\u001B[0;32m---> 43 \u001B[0;31m
\u001B[0;32mraise\u001B[0m
\u001B[0mRuntimeError\u001B[0m\u001B[0;34m(\u001B[0m\u001B[0;34m\"Interru
pt user code on Spark
exceptions\"\u001B[0m\u001B[0;34m)\u001B[0m\u001B[0;34m\u001B[0m\u001B[0;
34m\u001B[0m\u001B[0m\n\u001B[0m\u001B[0;32m     44
\u001B[0;31m\u001B[0;34m\u001B[0m\u001B[0m\n\u001B[0m\u001B[0;32m     45
\u001B[0;31m    \u001B[0;32mdef\u001B[0m
\u001B[0mdisplay\u001B[0m\u001B[0;34m(\u001B[0m\u001B[0mself\u001B[0m\u00
1B[0;34m,\u001B[0m
\u001B[0minput\u001B[0m\u001B[0;34m=\u001B[0m\u001B[0;32mNone\u001B[0m\u0
01B[0;34m,\u001B[0m
\u001B[0;34m*\u001B[0m\u001B[0margs\u001B[0m\u001B[0;34m,\u001B[0m
\u001B[0;34m**\u001B[0m\u001B[0mkwargs\u001B[0m\u001B[0;34m)\u001B[0m\u00
1B[0;34m:\u001B[0m\u001B[0;34m\u001B[0m\u001B[0;34m\u001B[0m\n\u
001B[0m\n"
     ]
    },
    {
```

```
   "output_type": "display_data",
   "data": {
    "text/plain": [
     "ipdb>  "
    ]
   },
   "metadata": {},
   "output_type": "display_data"
  },
  {
   "output_type": "display_data",
   "data": {
    "text/html": [
     "<style scoped>\n",
     "  .ansiout {\n",
     "    display: block;\n",
     "    unicode-bidi: embed;\n",
     "    white-space: pre-wrap;\n",
     "    word-wrap: break-word;\n",
     "    word-break: break-all;\n",
     "    font-family: \"Source Code Pro\", \"Menlo\", monospace;;\n",
     "    font-size: 13px;\n",
     "    color: #555;\n",
     "    margin-left: 4px;\n",
     "    line-height: 19px;\n",
     "  }\n",
     "</style>"
    ]
   },
   "metadata": {
    "application/vnd.databricks.v1+output": {
     "arguments": {},
     "data": "",
     "errorSummary": "Cancelled",
     "errorTraceType": "html",
     "metadata": {},
     "type": "ipynbError"
    }
   },
   "output_type": "display_data"
  }
 ],
 "source": [
  "display(df)"
 ]
},
{
 "cell_type": "markdown",
 "metadata": {
  "application/vnd.databricks.v1+cell": {
   "cellMetadata": {},
   "inputWidgets": {},
   "nuid": "10fc623f-d0b0-4ba6-8a33-66bf4876a666",
   "showTitle": false,
```

```
     "title": ""
    }
   },
   "source": [
    "## Other Useful Features\n",
    "\n",
    "1. Highlighted Code Execution\n",
    "2. Formatting and Error  Highlighting \n",
    "3. Notebook Versioning"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {
    "application/vnd.databricks.v1+cell": {
     "cellMetadata": {},
     "inputWidgets": {},
     "nuid": "d456bcd2-70fd-4cbf-9236-3a88e669da27",
     "showTitle": false,
     "title": ""
    }
   },
   "source": [
    "### Example: Higlighted Code Execution \n",
    "\n",
    "Docs: https://docs.databricks.com/notebooks/notebooks-code.html#run-
selected-text"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 0,
   "metadata": {
    "application/vnd.databricks.v1+cell": {
     "cellMetadata": {
      "byteLimit": 2048000,
      "rowLimit": 10000
     },
     "inputWidgets": {},
     "nuid": "dd277539-9793-4599-9719-7b3d3deb22cd",
     "showTitle": false,
     "title": ""
    }
   },
   "outputs": [
    {
     "output_type": "stream",
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "line 1\n"
     ]
    }
   ],
```

```
   "source": [
    "print(\"line 1\")\n",
    "print(\"line 2\")"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {
    "application/vnd.databricks.v1+cell": {
     "cellMetadata": {},
     "inputWidgets": {},
     "nuid": "99eba537-1f15-45ab-a70a-8ead3c568614",
     "showTitle": false,
     "title": ""
    }
   },
   "source": [
    "### Example: Formatting and Error Highlighting\n",
    "\n",
    "Docs: https://docs.databricks.com/notebooks/notebooks-
code.html#format-code-cells"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 0,
   "metadata": {
    "application/vnd.databricks.v1+cell": {
     "cellMetadata": {},
     "inputWidgets": {},
     "nuid": "a182ab85-eeda-4fb3-96e4-3c4e5cdd9925",
     "showTitle": false,
     "title": ""
    }
   },
   "outputs": [],
   "source": [
    "%sql\n",
    "Select * from demos.table "
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {
    "application/vnd.databricks.v1+cell": {
     "cellMetadata": {},
     "inputWidgets": {},
     "nuid": "f83bf007-b696-4049-bf0f-d693fce77940",
     "showTitle": false,
     "title": ""
    }
   },
   "source": [
    "## Spark UI - Spark Troubleshooting and Performance Tuning"
```

```
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 0,
   "metadata": {
    "application/vnd.databricks.v1+cell": {
     "cellMetadata": {
      "byteLimit": 2048000,
      "rowLimit": 10000
     },
     "inputWidgets": {},
     "nuid": "e3d6cf14-9c05-4ad0-9d0f-72b994bba4a1",
     "showTitle": false,
     "title": ""
    }
   },
   "outputs": [
    {
     "output_type": "display_data",
     "data": {
      "text/html": [
       "\n",
       "  <div style=\"width:1150px; margin:auto\">\n",
       "  <iframe\n",
       "
src=\"https://docs.google.com/presentation/d/1YJiMywiKhfWc3kkLiM83cjIyaHX
3J17uuQKUO3d5Dvk/embed?slide=8\"\n",
       "    frameborder=\"0\"\n",
       "    width=\"1150\"\n",
       "    height=\"683\"\n",
       "  ></iframe></div>\n",
       "  "
      ]
     },
     "metadata": {},
     "output_type": "display_data"
    }
   ],
   "source": [
    "def display_slide(slide_id, slide_number):\n",
    "  displayHTML(f'''\n",
    "  <div style=\"width:1150px; margin:auto\">\n",
    "  <iframe\n",
    "
src=\"https://docs.google.com/presentation/d/{slide_id}/embed?slide={slid
e_number}\"\n",
    "    frameborder=\"0\"\n",
    "    width=\"1150\"\n",
    "    height=\"683\"\n",
    "  ></iframe></div>\n",
    "  ''')\n",
    "display_slide('1YJiMywiKhfWc3kkLiM83cjIyaHX3J17uuQKUO3d5Dvk', '8')"
   ]
```

```
  },
  {
   "cell_type": "markdown",
   "metadata": {
    "application/vnd.databricks.v1+cell": {
     "cellMetadata": {},
     "inputWidgets": {},
     "nuid": "0e119766-6c2d-43ef-8d07-8e478c81250f",
     "showTitle": false,
     "title": ""
    }
   },
   "source": [
    "Spark Simulator: https://www.databricks.training/spark-ui-
simulator/index.html"
   ]
  }
 ],
 "metadata": {
  "application/vnd.databricks.v1+notebook": {
   "dashboards": [],
   "language": "python",
   "notebookMetadata": {
    "mostRecentlyExecutedCommandWithImplicitDF": {
     "commandId": 2793793367647935,
     "dataframes": [
      "_sqldf"
     ]
    },
    "pythonIndentUnit": 2
   },
   "notebookName": "Code Debugging in Databricks",
   "widgets": {}
  }
 },
 "nbformat": 4,
 "nbformat_minor": 0
}
```