In [1]:
```
#Aufgabe von Manpreet Misson; Timothy Gregorian und Omar Sidi Mammar - Gruppe 4
```

Spark Exercise

Apache Spark is an excellent tool for data engineering projects due to its robust ability to process large-scale data efficiently through distributed computing. Spark's in-memory processing capabilities significantly enhance the speed of data operations, making it ideal for handling big data workloads. It supports various data sources and formats, offering versatility in data ingestion and transformation. Additionally, Spark's rich API supports multiple programming languages such as Python, Java, and Scala, catering to diverse developer preferences. Its ecosystem, which includes libraries for SQL, machine learning, and graph processing, provides a comprehensive suite for building complex data pipelines and analytics, making it a powerful and flexible choice for data engineering tasks.

Use Python, `pyspark` and `pandas` to explore Apache Spark RDD and DataFrame:

Spark RDD

Spark RDD (Resilient Distributed Dataset) is a fundamental data structure in Apache Spark that enables fault-tolerant, distributed processing of large datasets across multiple nodes in a cluster. Spark RDDs provide a higher-level abstraction for performing distributed data processing tasks, including both map (transformations) and reduce (aggregations) operations.

Import Necessary Libraries

In [2]:
```
# TODO
from pyspark.sql import SparkSession
from pyspark.sql.functions import explode, split, avg
import shutil
import ast
```

Spark Context and Session

Initialize Spark Context and Spark Session

In [16]:
```
# TODO
spark = SparkSession.builder.appName("StackOverflow_SparExercise").getOrCreate()
spark.sparkContext.setLogLevel("ERROR")
```

Load Data into RDD

In [17]:
```
# TODO
df = spark.read.csv("DataSpark_Exercise3/srp_2022_cleaned.csv",
                    header=True, inferSchema=True)
rdd = df.rdd
print(rdd.first())
```

```
Row(ConvertedCompYearly=215232.0, LanguageHaveWorkedWith='C#;JavaScript;SQL;TypeScript', MainBranch='I am a developer by profession', Country='Israel', YearsCode=20, LanguageList="['C#', 'JavaScript', 'SQL', 'TypeScript']")
```

Map Operation

Split data into individual parts and create key-value pairs

In [5]:
```
# TODO
lang_salary_rdd = rdd.flatMap(
    lambda row: [
        (lang.strip(), float(row.ConvertedCompYearly))
        for lang in ast.literal_eval(row.LanguageList)
        if row.LanguageList is not None
    ]
)
```

Reduce Operation

Reduce your key-value pairs

In [6]:
```
# TODO
lang_avg_rdd = (
    lang_salary_rdd
    .mapValues(lambda salary: (salary, 1))
    .reduceByKey(lambda a, b: (a[0] + b[0], a[1] + b[1]))
    .mapValues(lambda x: x[0] / x[1])
)
```

Collect Results

Because of lazy evaluation, the map-reduce operation is performed only now. Show what you calculated.

In [18]:
```
# TODO
top_10 = lang_avg_rdd.sortBy(lambda x: x[1], ascending=False).take(10)

for lang, avg_salary in top_10:
    print(f"{lang}: ${avg_salary:,.2f}")
```

```
Clojure: $104,573.23
Erlang: $101,957.38
F#: $95,571.05
Scala: $95,196.36
Elixir: $94,000.62
Ruby: $93,030.30
Perl: $92,804.84
LISP: $92,711.36
OCaml: $91,666.16
Go: $90,028.04
```

Save Results

In [19]:
```
# TODO

shutil.rmtree("DataSpark_Exercise3/lang_avg_salary_rdd", ignore_errors=True)
lang_salary_rdd.saveAsTextFile("DataSpark_Exercise3/lang_avg_salary_rdd")
print("Ergebnis erfolgreich gespeichert unter: DataSpark_Exercise3/lang_avg_sala
```

```
[Stage 31:============================>                            (1 + 1) / 2]
Ergebnis erfolgreich gespeichert unter: DataSpark_Exercise3/lang_avg_salary_rdd
```

Spark DataFrame

Spark DataFrame is a distributed collection of data organized into named columns, designed for efficient data manipulation and analysis in Apache Spark. It is used for various data processing tasks such as data ingestion, transformation, querying, and analysis in Apache Spark, providing a high-level abstraction that simplifies working with structured data.

Load Data into DataFrame

In [9]:
```
# TODO
df = spark.read.csv("DataSpark_Exercise3/srp_2022_cleaned.csv",
                    header=True, inferSchema=True)
```

View DataFrame Schema

In [10]:
```
# TODO
df.printSchema()
```

```
root
 |-- ConvertedCompYearly: double (nullable = true)
 |-- LanguageHaveWorkedWith: string (nullable = true)
 |-- MainBranch: string (nullable = true)
 |-- Country: string (nullable = true)
 |-- YearsCode: integer (nullable = true)
 |-- LanguageList: string (nullable = true)
```

View DataFrame Data

In [11]:
```
# TODO
df.show(5, truncate=False)
```

```
+------------------+--------------------------------------------------+----------
------------------+--------------------------------------------------------+--------
-+-----------------------------------------------------------------+
|ConvertedCompYearly|LanguageHaveWorkedWith                            |MainBranch
|Country                                                 |YearsCode|LanguageList
|
+------------------+--------------------------------------------------+----------
------------------+--------------------------------------------------------+--------
-+-----------------------------------------------------------------+
|215232.0           |C#;JavaScript;SQL;TypeScript                      |I am a deve
loper by profession|Israel                                               |20
|['C#', 'JavaScript', 'SQL', 'TypeScript']                           |
|60307.0            |Bash/Shell;C#;HTML/CSS;JavaScript;PowerShell;SQL|I am a deve
loper by profession|United Kingdom of Great Britain and Northern Ireland|5
|['Bash/Shell', 'C#', 'HTML/CSS', 'JavaScript', 'PowerShell', 'SQL']|
|65000.0            |C;HTML/CSS;Rust;SQL;Swift;TypeScript              |I am a deve
loper by profession|United States of America                             |12
|['C', 'HTML/CSS', 'Rust', 'SQL', 'Swift', 'TypeScript']             |
|110000.0           |HTML/CSS;JavaScript;PHP;Python;R;Ruby;Scala       |I am a deve
loper by profession|United States of America                             |11
|['HTML/CSS', 'JavaScript', 'PHP', 'Python', 'R', 'Ruby', 'Scala']   |
|19224.0            |C#;Java;PHP;Python;R                              |I am a deve
loper by profession|Czech Republic                                       |7
|['C#', 'Java', 'PHP', 'Python', 'R']                                |
+------------------+--------------------------------------------------+----------
------------------+--------------------------------------------------------+--------
-+-----------------------------------------------------------------+
only showing top 5 rows
```

Filter Data

Performe a filter operation on a column

In [12]:
```
# TODO
df_filtered = df.filter(df["YearsCode"] > 10)
df_filtered.show(5)
```

```
+------------------+--------------------+--------------------+--------------
----+---------+--------------------+
|ConvertedCompYearly|LanguageHaveWorkedWith|          MainBranch|             Cou
ntry|YearsCode|        LanguageList|
+------------------+--------------------+--------------------+--------------
----+---------+--------------------+
|          215232.0|  C#;JavaScript;SQL...|I am a developer ...|              Is
rael|       20|['C#', 'JavaScrip...|
|           65000.0|  C;HTML/CSS;Rust;S...|I am a developer ...|United States of
...|       12|['C', 'HTML/CSS',...|
|          110000.0|  HTML/CSS;JavaScri...|I am a developer ...|United States of
...|       11|['HTML/CSS', 'Jav...|
|          202623.0|            Python;SQL|I am a developer ...|             Aus
tria|       13|   ['Python', 'SQL']|
|           51192.0|     C#;PowerShell;SQL|I am a developer ...|             Aus
tria|       36|['C#', 'PowerShel...|
+------------------+--------------------+--------------------+--------------
----+---------+--------------------+
only showing top 5 rows
```

Group By and Aggregate

Performe a group by and aggregat operation

In [13]:
```
# TODO
df.groupBy("Country").avg("ConvertedCompYearly") \
  .orderBy("avg(ConvertedCompYearly)", ascending=False) \
  .show(10)
```

```
+-------------------+------------------------+
|            Country|avg(ConvertedCompYearly)|
+-------------------+------------------------+
|United States of ...|       134451.2444578495|
|             Israel|      121977.64265927978|
|Lao People's Demo...|                112091.0|
|        Switzerland|      110161.94103773584|
|            Andorra|                108347.0|
|            Iceland|      106058.66666666667|
|             Canada|       97041.57719714964|
|          Australia|       96005.21911764707|
|            Denmark|       94006.86956521739|
|            Ireland|       90257.05517241379|
+-------------------+------------------------+
only showing top 10 rows
```

Save DataFrame to Parquet

In [14]:
```
# TODO
df.write.mode("overwrite").parquet
("DataSpark_Exercise3/lang_avg_salary_parquet")
print("Parquet-Datei erfolgreich gespeichert unter: DataSpark_Exercise3/lang_avg
```

```
Parquet-Datei erfolgreich gespeichert unter: DataSpark_Exercise3/lang_avg_salary_
parquet
```

Spark SQL-Query Abfrage

In [15]:
```
# Wir haben das hier am Ende hinzugefügt, da beim Abgabefenster steht: "performe

# Register the DataFrame as a temporary SQL view
df.createOrReplaceTempView("developer_data")

# SQL-Abfrage: Welche Programmiersprachen korrelieren mit höherem Gehalt?
spark.sql("""
    SELECT
        language,
        ROUND(AVG(ConvertedCompYearly), 2) AS avg_salary
    FROM (
        SELECT
            EXPLODE(SPLIT(LanguageHaveWorkedWith, ';')) AS language,
            ConvertedCompYearly
        FROM developer_data
        WHERE ConvertedCompYearly IS NOT NULL
    ) tmp
    GROUP BY language
    ORDER BY avg_salary DESC
    LIMIT 10
""").show()
```

```
+--------+----------+
|language|avg_salary|
+--------+----------+
| Clojure| 104573.23|
|  Erlang| 101957.38|
|      F#|  95571.05|
|   Scala|  95196.36|
|  Elixir|  94000.62|
|    Ruby|   93030.3|
|    Perl|  92804.84|
|    LISP|  92711.36|
|   OCaml|  91666.16|
|      Go|  90028.04|
+--------+----------+
```