```
#PySpark Task Set – Part 3
#HR & Workforce Analytics, DataFrame APIs, Joins, SQL, Date Logic,Aggregation, UDFs, Views


from pyspark.sql import SparkSession
import pandas as pd
from io import StringIO

spark=SparkSession.builder.appName("HR Workforce Analytics").getOrCreate()

print(spark)
```

```
<pyspark.sql.session.SparkSession object at 0x7d5a6e281c10>
```

```
#Data
#Sample CSV and JSON files
employees_csv = """EmpID,Name,Department,JoinDate,Salary,ManagerID
1,Anita,HR,2021-05-01,55000,
2,Raj,Engineering,2020-03-15,80000,1
3,Simran,Engineering,2022-07-10,75000,1
4,Aamir,Marketing,2019-11-20,60000,1
5,Nisha,HR,2023-01-05,50000,1
"""

attendance_csv = """EmpID,Date,Status
1,2024-04-01,Present
1,2024-04-02,Present
2,2024-04-01,Absent
2,2024-04-02,Present
3,2024-04-01,Present
3,2024-04-02,Present
4,2024-04-01,Absent
4,2024-04-02,Absent
5,2024-04-01,Present
5,2024-04-02,Present
"""

bonuses_json = """
[
{"EmpID": 1, "Year": 2023, "Bonus": 5000},
{"EmpID": 2, "Year": 2023, "Bonus": 7000},
{"EmpID": 3, "Year": 2023, "Bonus": 6500},
{"EmpID": 4, "Year": 2023, "Bonus": 6000},
{"EmpID": 5, "Year": 2023, "Bonus": 4000}
]
"""

# Saving
with open("employees.csv", "w") as f:
    f.write(employees_csv)

with open("attendance.csv", "w") as f:
    f.write(attendance_csv)

with open("bonuses.json", "w") as f:
    f.write(bonuses_json)


#Task 1- Ingestion & Exploration

from pyspark.sql.functions import *

employees = spark.read.csv("employees.csv", header=True, inferSchema=True)
attendance = spark.read.csv("attendance.csv", header=True, inferSchema=True)
bonuses = spark.read.json("bonuses.json")

employees.printSchema()
attendance.printSchema()
bonuses.printSchema()

employees.show()
attendance.show()
bonuses.show()

# Distinct departments
distinct_dept = employees.select("Department").distinct()
```

```
distinct_dept.count()
distinct_dept.show()
```

```
 |-- ManagerID: integer (nullable = true)

root
 |-- EmpID: integer (nullable = true)
 |-- Date: date (nullable = true)
 |-- Status: string (nullable = true)

root
 |-- Bonus: long (nullable = true)
 |-- EmpID: long (nullable = true)
 |-- Year: long (nullable = true)
 |-- _corrupt_record: string (nullable = true)

+-----+------+-----------+----------+------+---------+
|EmpID|  Name| Department|  JoinDate|Salary|ManagerID|
+-----+------+-----------+----------+------+---------+
|    1| Anita|         HR|2021-05-01| 55000|     NULL|
|    2|   Raj|Engineering|2020-03-15| 80000|        1|
|    3|Simran|Engineering|2022-07-10| 75000|        1|
|    4| Aamir|  Marketing|2019-11-20| 60000|        1|
|    5| Nisha|         HR|2023-01-05| 50000|        1|
+-----+------+-----------+----------+------+---------+

+-----+----------+-------+
|EmpID|      Date| Status|
+-----+----------+-------+
|    1|2024-04-01|Present|
|    1|2024-04-02|Present|
|    2|2024-04-01| Absent|
|    2|2024-04-02|Present|
|    3|2024-04-01|Present|
|    3|2024-04-02|Present|
|    4|2024-04-01| Absent|
|    4|2024-04-02| Absent|
|    5|2024-04-01|Present|
|    5|2024-04-02|Present|
+-----+----------+-------+

+-----+-----+----+---------------+
|Bonus|EmpID|Year|_corrupt_record|
+-----+-----+----+---------------+
| NULL| NULL|NULL|              [|
| 5000|    1|2023|           NULL|
| 7000|    2|2023|           NULL|
| 6500|    3|2023|           NULL|
| 6000|    4|2023|           NULL|
| 4000|    5|2023|           NULL|
| NULL| NULL|NULL|              ]|
+-----+-----+----+---------------+

+-----------+
| Department|
+-----------+
|Engineering|
|         HR|
|  Marketing|
+-----------+
```

```
#Task 2-Data Frame Operations

from pyspark.sql.functions import datediff, current_date, round

#Add a column TenureYears using datediff() and round()
emp_df = employees.withColumn("TenureYears", round(datediff(current_date(), col("JoinDate")) / 365, 2))
emp_df.show()

#Calculate TotalCompensation = Salary + Bonus .
emp_bonus_df = emp_df.join(bonuses, "EmpID")
emp_bonus_df = emp_bonus_df.withColumn("TotalCompensation", col("Salary") + col("Bonus"))
emp_bonus_df.show()

#Filter employees with more than 2 years in the company.
filtered_emp = emp_bonus_df.filter(col("TenureYears") > 2)
filtered_emp.show()

#Show employees who report to a manager ( ManagerID is not null ).
emp_bonus_df.filter(col("ManagerID").isNotNull()).show()
```

```
+-----+------+-----------+----------+------+---------+-----------+
|EmpID|  Name| Department|  JoinDate|Salary|ManagerID|TenureYears|
+-----+------+-----------+----------+------+---------+-----------+
|    1| Anita|         HR|2021-05-01| 55000|     NULL|       4.11|
|    2|   Raj|Engineering|2020-03-15| 80000|        1|       5.24|
|    3|Simran|Engineering|2022-07-10| 75000|        1|       2.92|
|    4| Aamir|  Marketing|2019-11-20| 60000|        1|       5.56|
|    5| Nisha|         HR|2023-01-05| 50000|        1|       2.43|
+-----+------+-----------+----------+------+---------+-----------+

+-----+------+-----------+----------+------+---------+-----------+-----+----+---------------+-----------------+
|EmpID|  Name| Department|  JoinDate|Salary|ManagerID|TenureYears|Bonus|Year|_corrupt_record|TotalCompensation|
+-----+------+-----------+----------+------+---------+-----------+-----+----+---------------+-----------------+
|    1| Anita|         HR|2021-05-01| 55000|     NULL|       4.11| 5000|2023|           NULL|            60000|
|    2|   Raj|Engineering|2020-03-15| 80000|        1|       5.24| 7000|2023|           NULL|            87000|
|    3|Simran|Engineering|2022-07-10| 75000|        1|       2.92| 6500|2023|           NULL|            81500|
|    4| Aamir|  Marketing|2019-11-20| 60000|        1|       5.56| 6000|2023|           NULL|            66000|
|    5| Nisha|         HR|2023-01-05| 50000|        1|       2.43| 4000|2023|           NULL|            54000|
+-----+------+-----------+----------+------+---------+-----------+-----+----+---------------+-----------------+

+-----+------+-----------+----------+------+---------+-----------+-----+----+---------------+-----------------+
|EmpID|  Name| Department|  JoinDate|Salary|ManagerID|TenureYears|Bonus|Year|_corrupt_record|TotalCompensation|
+-----+------+-----------+----------+------+---------+-----------+-----+----+---------------+-----------------+
|    1| Anita|         HR|2021-05-01| 55000|     NULL|       4.11| 5000|2023|           NULL|            60000|
|    2|   Raj|Engineering|2020-03-15| 80000|        1|       5.24| 7000|2023|           NULL|            87000|
|    3|Simran|Engineering|2022-07-10| 75000|        1|       2.92| 6500|2023|           NULL|            81500|
|    4| Aamir|  Marketing|2019-11-20| 60000|        1|       5.56| 6000|2023|           NULL|            66000|
|    5| Nisha|         HR|2023-01-05| 50000|        1|       2.43| 4000|2023|           NULL|            54000|
+-----+------+-----------+----------+------+---------+-----------+-----+----+---------------+-----------------+

+-----+------+-----------+----------+------+---------+-----------+-----+----+---------------+-----------------+
|EmpID|  Name| Department|  JoinDate|Salary|ManagerID|TenureYears|Bonus|Year|_corrupt_record|TotalCompensation|
+-----+------+-----------+----------+------+---------+-----------+-----+----+---------------+-----------------+
|    2|   Raj|Engineering|2020-03-15| 80000|        1|       5.24| 7000|2023|           NULL|            87000|
|    3|Simran|Engineering|2022-07-10| 75000|        1|       2.92| 6500|2023|           NULL|            81500|
|    4| Aamir|  Marketing|2019-11-20| 60000|        1|       5.56| 6000|2023|           NULL|            66000|
|    5| Nisha|         HR|2023-01-05| 50000|        1|       2.43| 4000|2023|           NULL|            54000|
+-----+------+-----------+----------+------+---------+-----------+-----+----+---------------+-----------------+
```

```
#Task 3- Aggregation

#Avg salary per department
avg_salary_dept = employees.groupBy("Department").agg(avg("Salary"))
avg_salary_dept.show()

#No. of Employees under each manager
no_of_emp_manager = employees.groupBy("ManagerID").agg(count("*").alias("TeamSize"))
no_of_emp_manager.show()

#Count of absences per employee
count_of_absence = attendance.filter(col("Status") == "Absent") \
.groupBy("EmpID").count().withColumnRenamed("count", "AbsenceCount")
count_of_absence.show()
```

```
+-----------+-----------+
| Department|avg(Salary)|
+-----------+-----------+
|Engineering|    77500.0|
|         HR|    52500.0|
|  Marketing|    60000.0|
+-----------+-----------+

+---------+--------+
|ManagerID|TeamSize|
+---------+--------+
|     NULL|       1|
|        1|       4|
+---------+--------+

+-----+------------+
|EmpID|AbsenceCount|
+-----+------------+
|    4|           2|
|    2|           1|
+-----+------------+
```

```
#4. Joins
total_days = attendance.groupBy("EmpID").count().withColumnRenamed("count", "TotalDays")
present_days = attendance.filter(col("Status") == "Present") \
    .groupBy("EmpID").count().withColumnRenamed("count", "PresentDays")

attendance_rate = total_days.join(present_days, "EmpID") \
    .withColumn("AttendancePercent", round(col("PresentDays") / col("TotalDays") * 100, 2))

#Join employees and attendance → Get attendance % (Present days / Total days).
emp_attendance = employees.join(attendance_rate, "EmpID", "left")
emp_attendance.select("EmpID", "Name", "AttendancePercent").show()

#Join employees and bonuses → Show top 3 employees by TotalCompensation.
emp_bonus_totalcomp = emp_bonus_df.orderBy(col("TotalCompensation").desc()).select("EmpID", "Name", "TotalCompensation")
emp_bonus_totalcomp.show(3)

#Multi-level join: employees + bonuses + attendance .
full_df = employees.join(bonuses, "EmpID").join(attendance_rate, "EmpID", "left")
full_df.show()
```

```
+-----+------+-----------------+
|EmpID|  Name|AttendancePercent|
+-----+------+-----------------+
|    1| Anita|            100.0|
|    2|   Raj|             50.0|
|    3|Simran|            100.0|
|    4| Aamir|             NULL|
|    5| Nisha|            100.0|
+-----+------+-----------------+

+-----+------+-----------------+
|EmpID|  Name|TotalCompensation|
+-----+------+-----------------+
|    2|   Raj|            87000|
|    3|Simran|            81500|
|    4| Aamir|            66000|
+-----+------+-----------------+
only showing top 3 rows

+-----+------+-----------+----------+------+---------+-----+----+---------------+---------+-----------+-----------------+
|EmpID|  Name| Department|  JoinDate|Salary|ManagerID|Bonus|Year|_corrupt_record|TotalDays|PresentDays|AttendancePercent|
+-----+------+-----------+----------+------+---------+-----+----+---------------+---------+-----------+-----------------+
|    1| Anita|         HR|2021-05-01| 55000|     NULL| 5000|2023|           NULL|        2|          2|            100.0|
|    2|   Raj|Engineering|2020-03-15| 80000|        1| 7000|2023|           NULL|        2|          1|             50.0|
|    3|Simran|Engineering|2022-07-10| 75000|        1| 6500|2023|           NULL|        2|          2|            100.0|
|    4| Aamir|  Marketing|2019-11-20| 60000|        1| 6000|2023|           NULL|     NULL|       NULL|             NULL|
|    5| Nisha|         HR|2023-01-05| 50000|        1| 4000|2023|           NULL|        2|          2|            100.0|
+-----+------+-----------+----------+------+---------+-----+----+---------------+---------+-----------+-----------------+
```

```
#5. String & Date Functions
#Extract year and month from JoinDate .
employees = employees.withColumn("JoinYear", year("JoinDate")) \
                     .withColumn("JoinMonth", month("JoinDate"))
employees.show()

#Mask employee names using regex.
employees = employees.withColumn("MaskedName", regexp_replace("Name", "[aeiouAEIOU]", "*"))
employees.show()

#Use substring() to create EmpCode like "EMP001".
employees = employees.withColumn("EmpCode", format_string("EMP%03d", col("EmpID")))
employees.select("EmpID", "Name", "EmpCode", "MaskedName").show()
```

```
+-----+------+-----------+----------+------+---------+--------+---------+
|EmpID|  Name| Department|  JoinDate|Salary|ManagerID|JoinYear|JoinMonth|
+-----+------+-----------+----------+------+---------+--------+---------+
|    1| Anita|         HR|2021-05-01| 55000|     NULL|    2021|        5|
|    2|   Raj|Engineering|2020-03-15| 80000|        1|    2020|        3|
|    3|Simran|Engineering|2022-07-10| 75000|        1|    2022|        7|
|    4| Aamir|  Marketing|2019-11-20| 60000|        1|    2019|       11|
|    5| Nisha|         HR|2023-01-05| 50000|        1|    2023|        1|
+-----+------+-----------+----------+------+---------+--------+---------+

+-----+------+-----------+----------+------+---------+--------+---------+----------+
|EmpID|  Name| Department|  JoinDate|Salary|ManagerID|JoinYear|JoinMonth|MaskedName|
+-----+------+-----------+----------+------+---------+--------+---------+----------+
|    1| Anita|         HR|2021-05-01| 55000|     NULL|    2021|        5|     *n*t*|
|    2|   Raj|Engineering|2020-03-15| 80000|        1|    2020|        3|       R*j|
|    3|Simran|Engineering|2022-07-10| 75000|        1|    2022|        7|    S*mr*n|
|    4| Aamir|  Marketing|2019-11-20| 60000|        1|    2019|       11|     **m*r|
|    5| Nisha|         HR|2023-01-05| 50000|        1|    2023|        1|     N*sh*|
+-----+------+-----------+----------+------+---------+--------+---------+----------+
+-----+------+-------+----------+
|EmpID|  Name|EmpCode|MaskedName|
+-----+------+-------+----------+
|    1| Anita| EMP001|     *n*t*|
|    2|   Raj| EMP002|       R*j|
|    3|Simran| EMP003|    S*mr*n|
|    4| Aamir| EMP004|     **m*r|
|    5| Nisha| EMP005|     N*sh*|
+-----+------+-------+----------+
```

```
#6. Conditional & Null Handling
#Use when/otherwise to label performance:
    # “High” if Bonus > 6000
    # “Medium” if 4000–6000
    # “Low” otherwise
emp_bonus_df = emp_bonus_df.withColumn(
    "Performance",
    when(col("Bonus") > 6000, "High")
    .when((col("Bonus") > 4000) & (col("Bonus") <= 6000), "Medium")
    .otherwise("Low")
)
emp_bonus_df.show()

#Handle missing ManagerID using fillna("No Manager") .
employees_filled = employees.fillna({"ManagerID": "No Manager"})
employees_filled.show()
```

```
+-----+------+-----------+----------+------+---------+-----------+-----+----+---------------+-----------------+-----------+
|EmpID|  Name| Department|  JoinDate|Salary|ManagerID|TenureYears|Bonus|Year|_corrupt_record|TotalCompensation|Performance|
+-----+------+-----------+----------+------+---------+-----------+-----+----+---------------+-----------------+-----------+
|    1| Anita|         HR|2021-05-01| 55000|     NULL|       4.11| 5000|2023|           NULL|            60000|     Medium|
|    2|   Raj|Engineering|2020-03-15| 80000|        1|       5.24| 7000|2023|           NULL|            87000|       High|
|    3|Simran|Engineering|2022-07-10| 75000|        1|       2.92| 6500|2023|           NULL|            81500|       High|
|    4| Aamir|  Marketing|2019-11-20| 60000|        1|       5.56| 6000|2023|           NULL|            66000|     Medium|
|    5| Nisha|         HR|2023-01-05| 50000|        1|       2.43| 4000|2023|           NULL|            54000|        Low|
+-----+------+-----------+----------+------+---------+-----------+-----+----+---------------+-----------------+-----------+

+-----+------+-----------+----------+------+---------+--------+---------+----------+-------+
|EmpID|  Name| Department|  JoinDate|Salary|ManagerID|JoinYear|JoinMonth|MaskedName|EmpCode|
+-----+------+-----------+----------+------+---------+--------+---------+----------+-------+
|    1| Anita|         HR|2021-05-01| 55000|     NULL|    2021|        5|     *n*t*| EMP001|
|    2|   Raj|Engineering|2020-03-15| 80000|        1|    2020|        3|       R*j| EMP002|
|    3|Simran|Engineering|2022-07-10| 75000|        1|    2022|        7|    S*mr*n| EMP003|
|    4| Aamir|  Marketing|2019-11-20| 60000|        1|    2019|       11|     **m*r| EMP004|
|    5| Nisha|         HR|2023-01-05| 50000|        1|    2023|        1|     N*sh*| EMP005|
+-----+------+-----------+----------+------+---------+--------+---------+----------+-------+
```

```
#7. Spark SQL
# Create Database hr
spark.sql("CREATE DATABASE IF NOT EXISTS hr")
spark.catalog.setCurrentDatabase("hr")

# Save dataframes as tables
employees.write.mode("overwrite").saveAsTable("employees")
attendance.write.mode("overwrite").saveAsTable("attendance")
bonuses.write.mode("overwrite").saveAsTable("bonuses")

# SQL Queries
#Top paid employee in each department.
spark.sql("""
    SELECT Department, Name, Salary
    FROM employees
    WHERE (Department, Salary) IN (
        SELECT Department, MAX(Salary)
        FROM employees
        GROUP BY Department
    )
""").show()

#Attendance rate by department.
spark.sql("""
    SELECT
      e.Department,
      ROUND(AVG(CASE WHEN a.Status = 'Present' THEN 1 ELSE 0 END) * 100, 2) AS AttendanceRate
    FROM employees e
    JOIN attendance a ON e.EmpID = a.EmpID
    GROUP BY e.Department
""").show()

#Employees joined after 2021 with salary >
70,000.
spark.sql("""
    SELECT * FROM employees
    WHERE JoinDate > '2021-01-01' AND Salary > 70000
""").show()
```

```
+-----------+-----+------+
| Department| Name|Salary|
+-----------+-----+------+
|         HR|Anita| 55000|
|Engineering|  Raj| 80000|
|  Marketing|Aamir| 60000|
+-----------+-----+------+

+-----------+--------------+
| Department|AttendanceRate|
+-----------+--------------+
|Engineering|          75.0|
|         HR|         100.0|
|  Marketing|           0.0|
+-----------+--------------+

+-----+------+-----------+----------+------+---------+--------+---------+----------+-------+
|EmpID|  Name| Department|  JoinDate|Salary|ManagerID|JoinYear|JoinMonth|MaskedName|EmpCode|
+-----+------+-----------+----------+------+---------+--------+---------+----------+-------+
|    3|Simran|Engineering|2022-07-10| 75000|        1|    2022|        7|    S*mr*n| EMP003|
+-----+------+-----------+----------+------+---------+--------+---------+----------+-------+
```

```
#Task 8- Advanced(Optional)
from pyspark.sql.functions import udf
from pyspark.sql.types import StringType

# UDF for Department classification
def classify_dept(dept):
    return "Tech" if dept in ["Engineering"] else "Non-Tech"

classify_udf = udf(classify_dept, StringType())

classified_df = employees.withColumn("DeptType", classify_udf("Department"))
classified_df.select("EmpID", "Department", "DeptType").show()

# Assuming emp_attendance DataFrame is already created
emp_attendance.createOrReplaceTempView("emp_attendance_summary")

summary_df = spark.sql("SELECT * FROM emp_attendance_summary")
summary_df.show(truncate=False)

# Save as parquet partitioned by Department
summary_df.write.mode("overwrite").partitionBy("Department").parquet("emp_attendance_summary.parquet")

# Read back parquet to confirm and show
parquet_df = spark.read.parquet("emp_attendance_summary.parquet")
parquet_df.show(truncate=False)
```

```
+-----+-----------+--------+
|EmpID| Department|DeptType|
+-----+-----------+--------+
|    1|         HR|Non-Tech|
|    2|Engineering|    Tech|
|    3|Engineering|    Tech|
|    4|  Marketing|Non-Tech|
|    5|         HR|Non-Tech|
+-----+-----------+--------+

+-----+------+-----------+----------+------+---------+---------+-----------+-----------------+
|EmpID|Name  |Department |JoinDate  |Salary|ManagerID|TotalDays|PresentDays|AttendancePercent|
+-----+------+-----------+----------+------+---------+---------+-----------+-----------------+
|1    |Anita |HR         |2021-05-01|55000 |NULL     |2        |2          |100.0            |
|2    |Raj   |Engineering|2020-03-15|80000 |1        |2        |1          |50.0             |
|3    |Simran|Engineering|2022-07-10|75000 |1        |2        |2          |100.0            |
|4    |Aamir |Marketing  |2019-11-20|60000 |1        |NULL     |NULL       |NULL             |
|5    |Nisha |HR         |2023-01-05|50000 |1        |2        |2          |100.0            |
+-----+------+-----------+----------+------+---------+---------+-----------+-----------------+
+-----+------+----------+------+---------+---------+-----------+-----------------+-----------+
|EmpID|Name  |JoinDate  |Salary|ManagerID|TotalDays|PresentDays|AttendancePercent|Department |
+-----+------+----------+------+---------+---------+-----------+-----------------+-----------+
|1    |Anita |2021-05-01|55000 |NULL     |2        |2          |100.0            |HR         |
|5    |Nisha |2023-01-05|50000 |1        |2        |2          |100.0            |HR         |
|2    |Raj   |2020-03-15|80000 |1        |2        |1          |50.0             |Engineering|
|3    |Simran|2022-07-10|75000 |1        |2        |2          |100.0            |Engineering|
|4    |Aamir |2019-11-20|60000 |1        |NULL     |NULL       |NULL             |Marketing  |
+-----+------+----------+------+---------+---------+-----------+-----------------+-----------+
```