The Workforce Intelligence Engine: Driving Performance Through Scalable People Analytics

Company Background

Headquartered in a major global technology hub, **Innovate Solutions Inc.** is a rapidly scaling technology consulting firm with operations across several continents. A significant portion of its explosive growth has been driven by its international delivery centers, which have collectively tripled their workforce in the last two years. This hyper-growth has been a business success, but it has created significant internal challenges across its global operations.

The Human Resources and Finance departments, which once managed operations with spreadsheets, are now overwhelmed. They lack a unified, data-driven view of the company's most valuable asset: its people.

The Current Situation

The company's core employee and department data exists but is fragmented across different systems and is not centrally analyzed. Basic questions from leadership about departmental costs, hiring trends, and employee tenure take days of manual effort to answer and are often prone to errors. This lack of insight presents several critical business problems:

- **Budget Uncertainty:** The finance team cannot get a clear, real-time picture of the salary budget distribution across departments, making financial planning difficult.
- **Talent Management Gaps:** HR has no efficient way to analyze employee tenure, identify hiring patterns, or compare performance metrics across different parts of the organization.
- **Operational Inefficiency:** Without a centralized analytics engine, answering even simple questions requires pulling data manually, hindering the ability to make fast, informed decisions.

Your Role & Objective

As a **Senior Data Analyst** on the newly formed "People Analytics" team, your primary objective is to build a foundational analytics engine using PySpark. You have been given a backlog of critical questions from HR, Finance, and executive leadership. Your mission is to ingest the core company data and write the production-level PySpark code necessary to answer this backlog, transforming raw data into actionable intelligence.

Available Data (CSV Files)

You have been provided with two data extracts representing the company's core HR data:

1. departments.csv

Column Name	Data Type	Description
department_id	Integer	Unique identifier for each department.
department_name	String	Name of the department.
location	String	City where the department is based.

2. employees.csv

Column Name	Data Type	Description
employee_id	Integer	Unique identifier for each employee.
employee_name	String	Full name of the employee.
hire_date	Date	Date when the employee was hired.
salary	Integer	Annual salary of the employee.
department_id	Integer	Foreign key linking to the departments table.

Your Deliverables: The Analytics Backlog

Your primary objective is to develop the PySpark code necessary to answer the following list of critical business questions.

Part 1: Guided Analytics Session

This initial set of questions will form the basis of a live, hands-on session:

1. Find all employees who earn more than $140,000.
2. Select only the employee's name, salary, and hire date, and rename 'employee_name' to 'name'.
3. Find employees who were hired in the year 2022.
4. List employees ordered by their hire date, with the most recent hires first.
5. How many employees are in each department?
6. What is the average, maximum, and minimum salary for each department?
7. What is the total salary budget for the entire company?
8. Find all employees who work in a specific city.
9. Add a new column to show each employee's salary as a percentage of their department's average salary.
10. Calculate the number of years each employee has been with the company.
11. Find the first and last hire date for each department.
12. Rank employees within each department based on their salary (highest first).
13. Calculate a running total of salaries within each department, ordered by hire date.
14. Create a new column to categorize employees as 'High', 'Medium', or 'Low' earners based on salary.
15. Find all employees whose name contains the letter 'a' (case-insensitive).

Part 2: Comprehensive Assignment

This comprehensive list of questions must be answered to complete the full analytics engine.

A. Basic DataFrame Operations

1. Display the first 10 rows of the employees DataFrame.
2. Show the schema of both the employees and departments DataFrames.
3. Create a new DataFrame that contains only the employee_name and salary columns.
4. Find all employees who work in the department with department_id 105.

5. List all employees who have a salary between $70,000 and $80,000.
6. Find all employees who were hired before January 1, 2018.
7. Display the entire employees DataFrame, sorted by salary in ascending order.

B. Aggregations & Grouping

8. Calculate the total number of employees in the company.
9. Count the number of unique departments.
10. Find the average salary of all employees.
11. Determine the oldest and most recent hire dates in the entire company.
12. For each department, calculate the total number of employees.
13. For each department, find the average employee salary.
14. Find which department has the highest maximum salary.
15. List all departments that have more than 50 employees.

C. Joins & Combined Data Analysis

16. Create a new DataFrame that includes all employee details along with their corresponding department name and location.
17. Find the names of all employees who work in the 'Sales' department.
18. List all departments and the number of employees in each, sorted by the employee count in descending order.
19. Which city (location) has the most employees?
20. Use a left join to find if there are any departments that have no employees.

D. Date/Time & String Functions

21. Create a new column called hire_year in the employees DataFrame that contains the year each employee was hired.
22. How many employees were hired in each year?
23. Create a new column employee_initials that contains the first letter of each employee's name.
24. Find all employees whose names start with the letter 'M'.
25. Calculate the tenure of each employee in months.

E. Advanced Scenarios & Window Functions

26. For each department, find the employee with the highest salary.
27. Assign a dense rank to employees in each department based on their hire date (oldest hire gets rank 1).
28. Calculate the difference between each employee's salary and the average salary of their department.
29. Write a User-Defined Function (UDF) to create a new column salary_grade. The grade should be 'A' for salary > 120k, 'B' for 80k-120k, and 'C' for salary < 80k.
30. Find the department with the highest average employee tenure.

F. Bonus Question

31. Identify the first employee hired in each department who had a salary greater than that department's final overall average salary.

Assignment Guidelines & Evaluation Criteria

This section outlines the technical requirements, learning goals, and success metrics for the assignment.

1. Technical Setup & Assumptions

Before beginning the assignment, please ensure your environment meets the following expectations:

- **Environment:** All solutions should be developed using **PySpark** running in a local Jupyter Notebook.
- **Data Source:** The employees.csv and departments.csv files are expected to be in the same working directory as your notebook.
- **Data Integrity:** It is assumed that all source **CSV files include headers** and that the data types follow the specifications provided in the case study.
- **Core Imports:** Your notebook should start by importing the necessary libraries: from pyspark.sql import SparkSession, functions as F, Window.

2. Learning Outcomes

By successfully completing this analytics backlog, you will demonstrate the ability to:

- Master **DataFrame basics**, including reading data, selecting and renaming columns, and applying complex filters.
- Implement a wide range of **aggregations, groupings, and joins** to synthesize data from multiple sources.
- Effectively utilize built-in **date/time and string functions** to manipulate and enrich data for analysis.
- Apply advanced **window functions** to perform sophisticated calculations like ranking and running totals.
- Construct and apply **User-Defined Functions (UDFs)** to encapsulate custom business logic.
- Translate high-level **business questions** into precise and efficient PySpark code.

3. Evaluation Criteria

A successful submission will be evaluated based on the following criteria:

- **Correctness:** The code must be executable and produce the correct, expected output for each question.
- **PySpark Best Practices:** The solution must adhere to best practices for distributed computing. Unnecessary calls to .collect() or .toPandas(), which can cause memory errors by moving all data to the driver node, should be avoided.
- **Readability:** The final notebook must be well-documented with clear **Markdown explanations** for the approach to complex problems and concise, commented code.
- **Robustness:** The code should be written to gracefully handle potential **edge cases**, such as null values in key columns or departments with no employees.

Expected Final Deliverable

The final deliverable is a single, well-commented Jupyter Notebook (.ipynb) containing the complete PySpark code and the corresponding outputs for all assigned questions.