▯ Low-Level Design (LLD) – HR Analytics System

Difficulty Level: Medium | Total Marks: 20
Standards Followed: 4 Functions | 4 Visible Test Cases

▯ Summary of Corrections (Based on SME Feedback)

- ▯ Combined data loading into a single function for better cohesion
- ▯ Used datetime parsing with `pd.to_datetime()`
- ▯ Grouping and filtering logic follows best practices
- ▯ Output structures match evaluator expectations

▯ Concepts Tested

▯ Reading CSVs into DataFrames using `pd.read_csv()`
▯ Filtering with datetime conditions
▯ Grouping and aggregation using `groupby()`
▯ Set operations for identifying absentees

▯ Problem Statement

You are provided with two datasets from a company's HR system:

- **`employees.csv`** containing the list of registered employees
- **`attendance.csv`** containing daily login/logout timestamps

Your task is to implement a mini HR analytics system using Pandas that performs essential insights:
loading data, computing working hours, detecting late logins, and identifying absentees.

▯ Operations

▯ 1. Load Data

▯ Load employee and attendance files in a single function.
▯ **Function Prototype:**

```
python
CopyEdit
def load_data(emp_path: str, att_path: str) -> tuple:
```

 Input:

- `"employees.csv"`
- `"attendance.csv"`

 Output:

- Tuple → (`employees_df`, `attendance_df`)

 Implementation Flow:

- Use `pd.read_csv()` to read both files
- Parse `login_time` and `logout_time` as datetime using `pd.to_datetime()`
- Return both DataFrames as a tuple

---

2. Total Working Hours

 Compute total working hours for each employee.
 Function Prototype:

```
python
CopyEdit
def total_working_hours(attendance_df: pd.DataFrame) -> dict:
```

 Input: attendance DataFrame
 Output: Dictionary → `{emp_id: total_hours}`

 Implementation Flow:

- Subtract `login_time` from `logout_time` to compute duration
- Use `groupby('emp_id')` and `sum()`
- Convert result to dictionary rounded to 2 decimals

---

3. Late Joiners

 Return list of employees who logged in after 10:00 AM.
 Function Prototype:

```
python
```

```
CopyEdit
def late_joiners(attendance_df: pd.DataFrame) -> list:
```

□ Input: attendance DataFrame
□ Output: List of emp_ids

□ **Implementation Flow:**

- Use `.dt.time` on `login_time` and compare with 10:00:00
- Return list of unique emp_ids sorted

□ 4. Absentees

□ Return list of employees who never logged in.
□ **Function Prototype:**

```
python
CopyEdit
def absentees(employees_df: pd.DataFrame, attendance_df: pd.DataFrame) ->
list:
```

□ Input: employees_df and attendance_df
□ Output: List of emp_ids

□ **Implementation Flow:**

- Get all emp_ids from employees_df
- Get emp_ids from attendance_df
- Use set difference to find absentees
- Return sorted list

□ Implementation Hints

```
python
CopyEdit
# Starter template
import pandas as pd

class HRAnalytics:

    def load_data(self, emp_path: str, att_path: str) -> tuple:
        pass  # TODO

    def total_working_hours(self, attendance_df: pd.DataFrame) -> dict:
        pass  # TODO
    def late_joiners(self, attendance_df: pd.DataFrame) -> list:
        pass  # TODO

    def absentees(self, employees_df: pd.DataFrame, attendance_df:
pd.DataFrame) -> list:
        pass  # TODO
```

□ Test Cases & Marks Allocation

Test Case ID	Description	Associated Function	Marks
TC1	Load both CSVs	`load_data()`	□ 5
TC2	Total working hours calculation	`total_working_hours()`	□ 5
TC3	Detect late joiners	`late_joiners()`	□ 5
TC4	Identify absent employees	`absentees()`	□ 5
Total	–		□ 20

□ Visible Test Cases

□ TC1: Load Data

□ Input:

- `"employees.csv"`
- `"attendance.csv"`

□ Output:

- Tuple of DataFrames with appropriate columns

□ TC2: Total Working Hours

□ Input: attendance_df
□ Output:

```
python
CopyEdit
{'E101': 16.0, 'E102': 7.75, 'E103': 8.0}
```

□ TC3: Late Joiners

□ Input: attendance_df
□ Output:

```
python
CopyEdit
["E102"]
```

□ TC4: Absentees

□ Input: employees_df, attendance_df
□ Output:

```
python
CopyEdit
["E104", "E105"]
```