Low-Level Design (LLD) – Gym Membership Usage Analytics using Pandas

Domain: Health & Fitness
Difficulty: Medium | **Total Marks**: 20
Functions: 6 | **Visible Tests**: 5 | **Hidden Tests**: 3
Focus: DataFrame creation, groupBy, filtering, formatting, and derived columns

---

Concepts Tested

Pandas DataFrame creation
GroupBy + Aggregations (sum, count, mean)
Derived column logic
Filtering and sorting
Formatting and uniqueness logic
Function-level test independence

Problem Statement

You are analyzing gym membership usage logs to help improve services across multiple branches. Each record logs the member's name, branch location, workout type, and the number of minutes spent during a session. Your task is to extract insights using Pandas.

Operations

---

1. Create Usage DataFrame

- **Function Prototype**:

def create_usage_df(logs: list) -> pd.DataFrame:

- **Example Input**:

[["Alice", "Downtown", "Cardio", 45],

["Bob", "Uptown", "Weights", 60],

["Alice", "Downtown", "Yoga", 30]]

- **Expected Output**:

Member	Branch	Workout	Duration
Alice	Downtown	Cardio	45
Bob	Uptown	Weights	60
Alice	Downtown	Yoga	30

◇ **Flow**:

- Create DataFrame with columns: Member, Branch, Workout, Duration
- Ensure Duration is int type.

---

2. Compute Total Duration Per Member

◇ **Function Prototype**:

def compute_total_duration(usage_df: pd.DataFrame) -> pd.DataFrame:

◇ **Expected Output**:

Member	Total Duration
Alice	75
Bob	60

◇ **Flow**:

- Group by Member, sum Duration, reset index
- Rename Duration to Total Duration

---

3. Add Calorie Burn Column

◇ **Function Prototype**:

def add_calorie_column(usage_df: pd.DataFrame) -> pd.DataFrame:

◇ **Expected Output**:

Workout	Duration	Calories Burned
Cardio	45	360
Weights	60	420

◇ **Flow**:

- Define calorie burn rate per workout type (e.g. Cardio=8, Weights=7, Yoga=5)
- Multiply with duration
- Add column Calories Burned
- Round to nearest int

---

4. Filter Long Sessions

- **Function Prototype**:

def filter_long_sessions(usage_df: pd.DataFrame, min_minutes: int) -> pd.DataFrame:

- **Expected Output** (min_minutes = 45):

Member	Workout	Duration
Bob	Weights	60
Alice	Cardio	45

- **Flow**:
  - Filter where Duration ≥ threshold

---

5. Get Top N Calorie Sessions

- **Function Prototype**:

def get_top_sessions(usage_df: pd.DataFrame, n: int) -> pd.DataFrame:

- **Expected Output**:

Top N rows sorted by Calories Burned

- **Flow**:
  - Sort by Calories Burned descending
  - Return top N using .head(n)

---

6. Remove Duplicate Logs

- **Function Prototype**:

def remove_duplicates(usage_df: pd.DataFrame) -> pd.DataFrame:

- **Flow**:
  - Drop exact duplicate rows using drop_duplicates()

Test Case Matrix

Test ID	Description	Function	Marks
TC1	Create DataFrame from logs	create_usage_df()	2.5
TC2	Group by member and sum durations	compute_total_duration()	2.5
TC3	Add calories based on workout type & time	add_calorie_column()	2.5
TC4	Filter sessions longer than X minutes	filter_long_sessions()	2.5
TC5	Top calorie-burning sessions	get_top_sessions()	2.5
HTC1	Remove duplicate logs	remove_duplicates()	2.5
HTC2	Handle edge case with zero duration	compute_total_duration()	2.5
HTC3	Handle tie in top calorie-burning sessions	get_top_sessions()	2.5

Sample Visible Test Cases

TC1 – Create DataFrame

Input:

create_usage_df([["Alice", "Downtown", "Cardio", 45]])

Expected: 1-row DataFrame with correct columns

TC2 – Total Duration

Input:

["Alice", "Yoga", 30], ["Alice", "Cardio", 45]

Output: Alice → 75

TC3 – Add Calories

Input:

Cardio → 8/min, 45 mins → 360 kcal

TC4 – Filter Sessions

Input:
Threshold = 45 → Return sessions of 45 mins and above

TC5 – Top Calorie Sessions

Input: Top 2 → Sorted by calories burned

Hidden Tests

HTC#	Check
HTC1	Duplicate session removal
HTC2	Zero duration session
HTC3	Tie in calories, return based on row order