🏀 Week 3 Assignment Key & Walkthrough: Basketball Data Manipulation

This key provides the solutions and explanations for the data analysis tasks using the synthetic Chicago Bulls dataset.

Setup: Importing Data and Libraries

The initial code block generates a synthetic DataFrame (`df`) containing performance metrics like workload, heart rate, jump count, and RPE for five players across multiple dates and session types (Practice, Game, Recovery).

Python
~~~
import pandas as pd
import numpy as np

# (Data generation code block)

df.head() # Displays the first 5 rows to confirm data structure ```

---

## 🧩 Task 1: Inspect the Dataset

**Goal:** Understand the structure, data types, and check for missing values.

| Method | Purpose |
| :--- | :--- |
| `df.info()` | Shows data types (`Dtype`) and non-null counts for each
column. |
| `df.describe()` | Provides descriptive statistics (mean, std dev, min, max,
quartiles) for all numerical columns. |
| `df.isnull().sum()` | Checks the total count of missing (`NaN`) values for
every column. |

**Solution Code:**

```python
# Display dataset info
print(df.info())

# Display summary statistics
print("\nSummary Statistics:")
print(df.describe())

# Check for missing values
print("\nMissing Values Check:")
print(df.isnull().sum())
~~~

🧩 Task 2: Data Filtering and Player Averages

Goal: Isolate sessions where the team played a **Game**, and then find the average workload for each player from only those games.

Key Concept: Boolean Filtering We use `df["Session_Type"] == "Game"` to create a True/False mask, which then filters the DataFrame to keep only the 'Game' rows.

Key Concept: Grouping (`groupby()`) After filtering, we use `groupby("Player")` to separate the data by player, and then apply the `.mean()` aggregation function to the `Workload_AU` column.

Solution Code:

Python
```
# 1. Filter only Game sessions
games_df = df[df["Session_Type"] == "Game"]

# 2. Show the average workload per player for Game sessions
avg_workload_games = games_df.groupby("Player")["Workload_AU"].mean()

print("Average Game Workload (AU) per Player:")
print(avg_workload_games.sort_values(ascending=False))
```

🧩 Task 3: Summary Statistics by Session Type

Goal: Group the entire dataset by `Session_Type` and find the average of **Workload_AU** and **RPE** across the whole team for each type (Practice, Game, Recovery).

Solution Code:

Python
```
# Group by Session_Type and calculate the mean for Workload_AU and RPE
avg_by_session = df.groupby("Session_Type")[["Workload_AU",
"RPE"]].mean().round(1)

print("Mean Workload and RPE by Session Type:")
print(avg_by_session)
```

🧩 Task 4: Visualization

Goal: Create a simple chart to compare the average workloads for all players.

*We'll reuse the `avg_workload_games` data from **Task 2** to make a comparison based on Game performance, similar to the tutorial.*

Solution Code:

Python
```
import matplotlib.pyplot as plt # Needed for plotting

plt.figure(figsize=(10, 6))

# Plot the average workload data from Task 2 as a bar chart
avg_workload_games.plot(kind='bar', color='darkblue')

# Add appropriate labels and titles
plt.title("Average Game Workload (AU) by Player", fontsize=16)
plt.ylabel("Workload (Arbitrary Units)", fontsize=12)
plt.xlabel("Player", fontsize=12)
plt.xticks(rotation=45, ha='right') # Rotate player names for readability
plt.grid(axis='y', linestyle='--')
plt.tight_layout()
plt.show()
```

🧩 Task 5: Interpretation

Goal: Summarize what the analysis reveals.

Based on the outputs from Task 3 (Mean Workload and RPE by Session Type), the following interpretation can be made:

Interpretation:

The data clearly shows that **Game** sessions result in the highest average **Workload_AU** and the highest average **RPE** (Rating of Perceived Exertion), which is expected as Games are the most intense activity. **Practice** sessions have a moderate Workload and RPE, serving as a high-intensity training stimulus. **Recovery** sessions have the lowest Workload and RPE, indicating a successful reduction in physiological stress to allow for physical recuperation.
