Converting snake_case (backend) → camelCase (frontend)

Scenario:
Backend database uses `snake_case` → `user_id`, `first_name`
Frontend JavaScript expects `camelCase` → `userId`, `firstName`

Goal: Automatically convert keys in a JSON/dictionary from `snake_case` to `camelCase`.

Example Python code:

python Copy

```python
def snake_to_camel(snake_str):
    parts = snake_str.split('_')
    return parts[0] + ''.join(word.capitalize() for word in parts[1:])

data = {"user_id": 101, "first_name": "Deeptha", "last_name": "Prabhu"}

converted = {snake_to_camel(k): v for k, v in data.items()}
print(converted)
```

Output:

python Copy

```python
{'userId': 101, 'firstName': 'Deeptha', 'lastName': 'Prabhu'}
```

Use: Keeps data consistent between backend and frontend.

Cleaning and Processing Customer Feedback (String Operations)

Scenario:
You receive raw customer messages with punctuation, inconsistent cases, etc.

Goal:

- Convert all text to lowercase
- Remove punctuation
- Find keywords like "delay", "quality", or "refund"

Example Python code:

python Copy

```python
import re

feedback = "The product QUALITY was bad! I want a REFUND... Delivery Delay again!"

# 1. Convert to lowercase
feedback = feedback.lower()

# 2. Remove unwanted characters
feedback = re.sub(r'[^a-z\s]', '', feedback)

# 3. Check for keywords
keywords = ["delay", "quality", "refund"]
found = [word for word in keywords if word in feedback]

print("Cleaned Text:", feedback)
print("Keywords Found:", found)
```

✅ **Output:**

Copy

```
Cleaned Text: the product quality was bad i want a refund delivery delay again
Keywords Found: ['delay', 'quality', 'refund']
```

Use: Prepares clean text for sentiment or keyword analysis.

🔲 Aadhaar Number Validation (Pre-check)

Scenario:
Applicants must submit a **12-digit Aadhaar number.**
Invalid if it has less or more than 12 digits.

Goal: Check the length before saving or processing.

Example Python code:

python Copy

```python
aadhaar = input("Enter Aadhaar Number: ")

if aadhaar.isdigit() and len(aadhaar) == 12:
    print("✅ Valid Aadhaar Number")
else:
    print("❌ Invalid Aadhaar Number. Must be exactly 12 digits.")
```

Use: Prevents invalid data from entering the system.

🔲 Ticket Price Calculation (Based on Age)

Scenario:
Cinema "The Grand Picture House" uses age-based pricing:

- Child (<13): ₹50
- Adult (≥13): ₹150

Example Python code:

python Copy

```python
age = int(input("Enter age of customer: "))

if age < 13:
    price = 50
else:
    price = 150

print(f"Ticket Price: ₹{price}")
```

✅ **Output Example:**

Copy

```
Enter age of customer: 10
Ticket Price: ₹50
```

Use: Automates billing logic for customers.

mtcars dataset – Bar Plot for Gear Categories

Goal: Show how many car models fall into each gear category (3, 4, 5).

R Code:

r Copy

```r
# Load dataset
data(mtcars)

# Create a frequency table for gears
gear_counts <- table(mtcars$gear)

# Bar plot
barplot(gear_counts,
        main = "Distribution of Car Models by Gear Type",
        xlab = "Number of Gears",
        ylab = "Number of Car Models",
        col = c("lightblue", "lightgreen", "lightpink"))
```

Explanation:

- Uses `table()` to count how many cars have 3, 4, or 5 gears.
- `barplot()` visually shows the distribution.

Feed Efficiency Project – Bar Plot for Feed Types

Dataset: Built-in R dataset `chickwts` (chicken weights by feed type).
Goal: Show how many experimental subjects are in each feed group.

R Code:

r Copy

```r
# Load dataset
data(chickwts)

# Create frequency table
feed_counts <- table(chickwts$feed)

# Bar plot
barplot(feed_counts,
        main = "Distribution of Experimental Subjects by Feed Type",
        xlab = "Feed Type",
        ylab = "Number of Chickens",
        col = rainbow(length(feed_counts)))
```

Explanation:

- Displays how many chickens were tested under each feed type.

mtcars dataset – Clustered Bar Plot (Gear vs Cylinders)

Goal: Compare the distribution of car models by `gear` and `cyl` using gradient colors.

R Code:

```r
# Load dataset
data(mtcars)

# Create a contingency table
gear_cyl_table <- table(mtcars$gear, mtcars$cyl)

# Clustered bar plot with gradient colors
barplot(gear_cyl_table,
        beside = TRUE,
        col = colorRampPalette(c("lightblue", "blue"))(3),
        main = "Car Models by Gear and Cylinder Count",
        xlab = "Number of Gears",
        ylab = "Number of Car Models")
legend("topright",
       legend = colnames(gear_cyl_table),
       fill = colorRampPalette(c("lightblue", "blue"))(3),
       title = "Cylinders")
```

✅ Explanation:

- `beside = TRUE` makes clustered bars.
- Gradient colors distinguish cylinder groups.

Simple Linear Relationship – Scatter Plot using ggplot2

Goal: Visualize a small dataset before modeling.

R Code:

```r
# Install and load ggplot2 if not installed
# install.packages("ggplot2")
library(ggplot2)

# Sample dataset
data <- data.frame(
  X = c(1, 2, 3, 4, 5, 6, 7),
  Y = c(2, 3, 5, 7, 10, 13, 17)
)

# Scatter plot
ggplot(data, aes(x = X, y = Y)) +
  geom_point(color = "blue", size = 3) +
  ggtitle("Preliminary Scatter Plot of X vs Y") +
  xlab("X Variable") +
  ylab("Y Variable") +
  theme_minimal()
```

✅ Explanation:

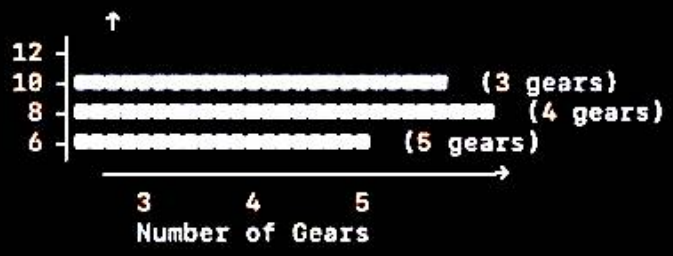
- Uses `ggplot2` for clean visualization.
- Helps spot any trend or linear relationship between X and Y.

🧩 Bar Plot – mtcars (Gears)

R Code Output:
A **simple vertical bar plot** showing the number of cars in each gear category (3, 4, 5). Each bar is colored differently.

Expected Output Visualization:

Copy



Interpretation:
Most cars in the dataset have **3 or 4 gears**, and fewer have **5 gears**.

🐔 Feed Types – chickwts Dataset

R Code Output:
A colorful **bar plot** with 6 bars, each representing one feed type (`casein`, `horsebean`, `linseed`, `meatmeal`, `soybean`, `sunflower`).

Expected Output Visualization:

Copy

Interpretation:
The number of experimental subjects (chickens) varies across different feed types.

🚗 Clustered Bar Plot – mtcars (Gear vs Cylinders)

R Code Output:
A **clustered bar chart** showing:

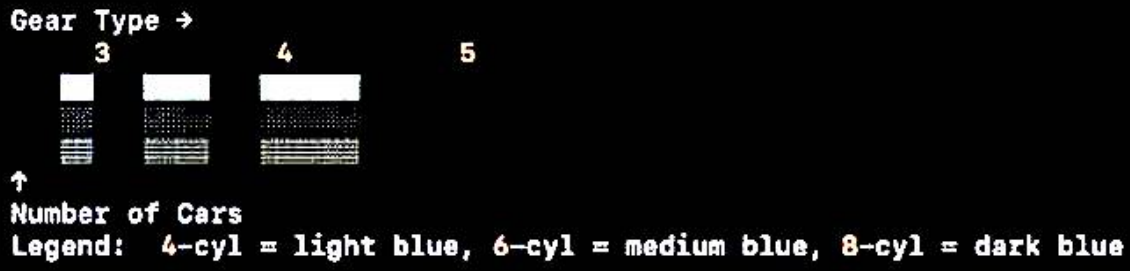
- X-axis → Gear types (3, 4, 5)
- Bars grouped by cylinder count (4, 6, 8)
- Bars use gradient shades of blue

R Code Output:
A **clustered bar chart** showing:

- X-axis → Gear types (3, 4, 5)
- Bars grouped by cylinder count (4, 6, 8)
- Bars use gradient shades of blue

Expected Output Visualization:

Copy



Interpretation:
3-gear cars are mostly **8-cylinder**, while 4-gear cars are often **4-cylinder**.

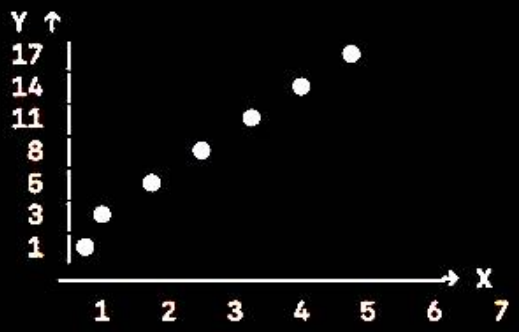
📈 Scatter Plot – Simple Linear Relationship

R Code Output:
A **scatter plot** (using `ggplot2`) with:

- X values on the horizontal axis
- Y values on the vertical axis
- Blue circular points forming an upward trend

Expected Output Visualization:

Copy



Interpretation:
Shows a **positive linear relationship** — as X increases, Y also increases.

[7]:
```
def snake_to_camel(snake_str):
    parts = snake_str.split('_')
    return parts[0] + ''.join(word.capitalize() for word in parts[1:])

data = {"user_id": 101, "first_name": "Deeptha", "last_name": "Prabhu"}

converted = {snake_to_camel(k): v for k, v in data.items()}
print(converted)
```

```
{'userId': 101, 'firstName': 'Deeptha', 'lastName': 'Prabhu'}
```

[8]:
```
import re

feedback = "The product QUALITY was bad! I want a REFUND... Delivery Delay again!"

# 1. Convert to lowercase
feedback = feedback.lower()

# 2. Remove unwanted characters
feedback = re.sub(r'[^a-z\s]', '', feedback)

# 3. Check for keywords
keywords = ["delay", "quality", "refund"]
found = [word for word in keywords if word in feedback]

print("Cleaned Text:", feedback)
print("Keywords Found:", found)
```

```
Cleaned Text: the product quality was bad i want a refund delivery delay again
Keywords Found: ['delay', 'quality', 'refund']
```

[9]:
```
aadhaar = input("Enter Aadhaar Number: ")

if aadhaar.isdigit() and len(aadhaar) == 12:
    print("✅ Valid Aadhaar Number")
else:
    print("❌ Invalid Aadhaar Number. Must be exactly 12 digits.")
```

```
Enter Aadhaar Number:  3588532468963
❌ Invalid Aadhaar Number. Must be exactly 12 digits.
```

[10]:
```
age = int(input("Enter age of customer: "))

if age < 13:
    price = 50
else:
    price = 150

print(f"Ticket Price: ₹{price}")
```

```
Enter age of customer:  21
Ticket Price: ₹150
```

[ ]: