Atomic Data Types (Foundations)

Q1. Logical

1. Create a logical vector of length 5 with at least one NA.
2. Convert it to numeric.
3. Explain the output.

Q2. Integer vs Numeric

1. Create one integer and one numeric variable with the same value.
2. Compare:
    - typeof()
    - object.size()
3. Why does R treat them differently?

Q3. Numeric Precision

1. Store 0.1 + 0.2 in a variable.
2. Compare it with 0.3 using ==.
3. Explain the result.

Q4. Complex Numbers

1. Create a complex number.
2. Extract real and imaginary parts.
3. Multiply two complex numbers and explain the result.

Q5. Character Data

1. Create a character vector of 4 elements.
2. Check its length and memory usage.
3. Convert it to factor and compare storage.

Q6. Raw Type

1. Convert your name into raw bytes.
2. Convert it back to character.
3. Where is raw data used in real applications?

Special Values (Critical for Real Data)

Q7. NA Handling

1. Create a numeric vector with multiple NA values.
2. Try to calculate mean without removing NA.
3. Fix the issue and explain.

Q8. Typed NA

1. Create:
    - NA_integer_
    - NA_character_
2. Combine each with a vector of another type.

3. Observe coercion.

Q9. NaN vs NA vs Inf

1. Generate one example of each.
2. Test them using:
    - is.na()
    - is.nan()
    - is.finite()
3. Explain the differences.

Q10. NULL

1. Create a list with one element as NULL.
2. Check its length.
3. Remove that element and observe behavior.

Data Structures

Q11. Vector Coercion

1. Combine logical, integer, numeric, and character into one vector.
2. Predict the output type before running.
3. Verify using typeof().

Q12. Matrix

1. Create a 3×3 matrix.
2. Assign row and column names.
3. Extract:
    - 2nd row
    - 3rd column
    - diagonal elements

Q13. Array

1. Create a 3D array (2×2×3).
2. Explain how data is filled internally.
3. Extract all values from 2nd layer.

Q14. List

1. Create a nested list containing:
    - numeric
    - character
    - another list
2. Access deeply nested elements using:
    - [[ ]]
    - $

Q15. Data Frame

1. Create a data frame with:
    - integer
    - character
    - numeric
2. Check structure.
3. Explain why data frame is actually a list.

Factor (High-Value Concept)

Q16. Factor Internals

1. Create a factor with 3 levels.
2. Convert it to numeric directly.
3. Explain why the output is misleading.

Q17. Ordered Factor

1. Create an ordered factor (Low, Medium, High).
2. Compare two values.
3. Explain how ordering works internally.

Date and Time

Q18. Date Type

1. Create today's date.
2. Subtract 10 days.
3. Find the difference in days.

Q19. POSIXct vs POSIXlt

1. Create current time in both formats.
2. Compare:
    - typeof()
    - class()
3. Explain when to use each.

Type Inspection & Coercion

Q20. typeof vs class

1. Create a factor and a Date.
2. Compare typeof() and class().
3. Explain why they differ.

Q21. Coercion Rules

1. Create vectors that trigger each coercion step.
2. Document the coercion hierarchy.

Memory & Performance (Advanced)

Q22. Memory Comparison

1. Create a vector of 1 million:

    - integers
    - numerics
2. Compare memory usage.
3. Draw conclusions.

Q23. Copy-on-Modify

1. Create a vector.
2. Assign it to another variable.
3. Modify one element.
4. Observe memory behavior.

Error & Debug Thinking

Q24. Identify the Error

Explain what is wrong and how to fix:

```
if (x == NA) {
  print("Missing")
}
```

Q25. Predict the Output (NO RUNNING)

```
x <- factor(c("10", "20"))
as.numeric(x)
```

Explain.

BONUS (Optional but Powerful)

Q26. Real-World Case

You receive a CSV where:

- IDs should be integer
- Gender should be factor
- Visit date is character

Explain how you would **clean and convert types safely**.

Q27. One-Line Summary

Write **one sentence per data type** explaining its purpose.

Q28.Data Type Selection (Thinking Test)

Problem

You are designing a dataset for an **online food delivery app**.

Choose the **most appropriate R data type** and explain *why* for:

1. Order ID
2. Customer name
3. Order amount
4. Payment completed (Yes/No)
5. Order priority (Low, Medium, High)
6. Delivery date

7. Exact delivery timestamp
8. Discount percentage
9. Item count
10. Customer rating (1–5)

Q29.Import & Fix Types

Problem

You imported CSV data where **everything came as character**.

```
data <- data.frame(
  id = c("101", "102", "103"),
  amount = c("250.50", "300.75", "NA"),
  paid = c("TRUE", "FALSE", "TRUE"),
  order_date = c("2026-01-20", "2026-01-21", "2026-01-22"),
  stringsAsFactors = FALSE
)
```

Tasks

1. Convert each column to the **correct type**
2. Handle missing values safely
3. Verify using str()

Q30: Predict Without Running (Interview Level)

Predict Output

```
x <- c(1L, TRUE, 2.5)
typeof(x)

y <- c("100", 200, 300)
typeof(y)

z <- factor(c("A", "B", "A"))
as.character(z)
as.numeric(z)
```

Explain each result.

Q31:NA vs NULL (Trick Assignment)

Problem

Given:

```
a <- c(1, NA, 3)
b <- NULL
```

1. What is length(a)?
2. What is length(b)?
3. What happens when you do:

```
c(a, b)
```

4. Why is NULL dangerous in data pipelines?

Q32:Memory Awareness

Problem

You need to store **10 million user IDs**.

1. Should you use integer or numeric?
2. Write code to compare memory usage.
3. Explain business impact of wrong choice.

Q33:Factor Disaster Scenario

Problem

A junior analyst runs:

df$salary <- as.numeric(df$salary)

But salary was a factor.

Tasks

1. Explain what goes wrong
2. Show how to fix it safely
3. Describe a real-world consequence

Q34:Date vs Character Bug

Problem

order_date <- c("2026-01-10", "2026-01-02")

order_date > "2026-01-05"

1. Why is this dangerous?
2. Fix the code
3. Explain correct behavior

Q35:POSIXct vs POSIXlt Choice

Problem

You are building:

- A **login tracking system**
- A **time-of-day analytics dashboard**

Which type do you choose for each and why?

Q36:Vector Recycling Trap

Problem

price <- c(100, 200, 300)

discount <- c(10, 20)

final <- price - discount

1. Predict output
2. Explain recycling rule
3. Why is this dangerous?

Q37:Copy-on-Modify Scenario

Problem

```
data <- 1:5
backup <- data
data[1] <- 999
```

1. Does backup change?
2. Why?
3. How does R protect memory?

Q38:Real-World Debugging

Problem

This code does not work as expected:

```
if (total_amount > 1000) {
  print("High value")
}
```

total_amount sometimes contains NA.

Tasks

1. Explain the bug
2. Fix the condition
3. Explain production impact

Q39:Design a Validation Rule

Problem

Design R checks to ensure:

1. ID is integer
2. Amount is numeric and positive
3. Status is one of: "NEW", "PAID", "CANCELLED"
4. Date is valid

(No need to code fully; logic is enough)

Q40:Think Like a Reviewer

Problem

Which of the following choices would you **reject** in a code review and why?

1. IDs stored as numeric
2. Gender stored as character
3. Dates stored as character
4. Priority stored as ordered factor
5. Boolean flags stored as "Y" / "N"

Q41:Type Conversion Pipeline

Problem

You receive a messy dataset daily.

Design a **safe conversion pipeline**:

1. Inspect
2. Convert
3. Validate

4. Stop on failure

Describe steps in plain English.

Q42:Mini Project (Optional but Powerful)

Problem

Simulate a **small e-commerce dataset** (10 rows) with:

- integer ID
- factor status
- numeric amount
- Date
- logical flag

Then:

1. Print structure
2. Summarize by status
3. Handle missing values

Q43: Type Choice Under Pressure

Scenario

You are building a **bank transaction monitoring system**.

For each field, decide:

- correct R type
- what could go wrong if chosen incorrectly

Fields:

1. Transaction ID
2. Account number
3. Transaction amount
4. Transaction status (SUCCESS, FAILED, PENDING)
5. Transaction date
6. Transaction timestamp
7. Fraud flag
8. Retry count

Q44: Silent Bug Detection

Scenario

This code runs **without error**, but results are wrong:

```
total <- sum(c("100", "200", "300"))
```

Tasks:

1. Predict the output
2. Explain why it happens
3. Fix it safely
4. Explain real-world impact

Q45: Factor vs Character Decision

Scenario

You store **city names** in a dataset of 50 million rows.
Tasks:

1. Choose factor or character
2. Justify memory & performance
3. When would you change your decision?

Q46:Date Corruption Case

Scenario

Data comes as:
c("01-02-2026", "05-01-2026")
Tasks:

1. Explain ambiguity
2. Convert safely
3. Explain consequences of wrong interpretation

Q47: NA Explosion

Scenario

A KPI dashboard suddenly shows **all values as NA**.
mean(revenue)
Tasks:

1. List 3 reasons this could happen
2. Fix the calculation
3. Add a defensive check

Q48:Logical Trap

Scenario

This filter gives unexpected rows:
df[df$active == TRUE,]
Tasks:

1. Why could this fail?
2. Fix it
3. Explain best practice

Q49:Vector Recycling Disaster

Scenario

Salary calculation:

```
salary <- c(30000, 40000, 50000, 60000)
bonus  <- c(5000, 7000)

salary + bonus
```

Tasks:

1. Predict output
2. Explain why R allows this
3. Show how to prevent it

Q50: Memory Meltdown

Scenario

Your R script crashes on a machine with 8 GB RAM.

Tasks:

1. List 4 datatype-related reasons
2. Suggest fixes
3. Explain trade-offs

Q51: List vs Data Frame Confusion

Scenario

A function returns:

result <- list(id = 1:5, score = c(80, 90, 85))

Tasks:

1. Why is this not a data frame?
2. Convert it correctly
3. Explain real-world consequences

Q52:Debugging typeof vs class

Scenario

A function behaves differently for two objects:

typeof(x)

class(x)

Tasks:

1. Explain difference in simple terms
2. Give a real bug caused by ignoring this
3. Show how you would debug it

Q52: Character Numbers Trap

Scenario

You read Excel data:

c("1000", "2,000", "3000")

Tasks:

1. Why as.numeric() fails
2. Clean data properly
3. Explain business risk

Q52: Ordered Factor Misuse

Scenario

Customer feedback stored as:

c("Good", "Excellent", "Poor")

Tasks:

1. Create correct ordered factor
2. Explain level order

3. Show one valid comparison

Q53:Copy-on-Modify Surprise

Scenario

A teammate says:

"R copied my 2GB object without warning."

Tasks:

1. Explain when copying happens
2. How to reduce memory impact
3. Best practices

Q54:Design a Safe Import Strategy

Scenario

Daily CSV import from third-party vendor.

Tasks:

1. Steps before conversion
2. Steps after conversion
3. Validation checks
4. Failure handling

(No code needed — design thinking)

Q54: End-to-End Thinking

Scenario

You receive user activity logs:

- user_id (text)
- event_time (text)
- event_type (text)
- duration (text)

Tasks:

1. Convert to correct types
2. Handle missing values
3. Prepare for analysis
4. Explain choices

Q55:Interview Brain-Twister

Predict Output (No Running)

```
x <- c(1, NA, 3)
y <- x[x > 2]
length(y)
```

Explain **why**.

Q56: Reject or Accept (Code Review)

Which would you reject and why?

1. Dates stored as character

2. Boolean flags stored as 0/1
3. IDs stored as factor
4. Amount stored as numeric
5. Status stored as character

Q56: Performance Design

Scenario

You must process **100 million rows daily**.

Tasks:

1. List datatype decisions that matter
2. Explain why
3. Give optimization priorities

Q57:Design a Type Audit Function

Scenario

You want to automatically detect datatype issues.

Tasks:

1. What should it check?
2. What should it warn?
3. What should stop execution?

Q58: Explain Like a Senior

Task

Explain to a junior developer:

- why NA ≠ NULL
- why factor ≠ character
- why numeric ≠ integer

In **simple language**.

Q1: The Smart Home Sensor

The Setup: You are programming a logic controller for a smart home. The system receives data from various sensors and needs to store them in the correct format to trigger actions.

Your Tasks:

1. **Numeric Types:** Store the current room temperature as a `float` (e.g., `22.5`) and the number of people in the room as an `int`.
2. **Boolean:** Create a variable `is_motion_detected`. If `people_count` is greater than 0, set this to `True`; otherwise, `False`.
3. **String Manipulation:** The system generates a status message. Create a string `status = "System Active"`. Use a string method to convert it to all uppercase.
4. **Casting:** You receive a sensor reading as a string `"19"`. Convert it to an `int` so you can add it to the `current_temperature`.

Q2: The E-Commerce Cart

The Setup: A customer is shopping on your website. You need to manage their "collection" of items. This scenario practices **Lists**, **Tuples**, and **Dictionaries**.

Your Tasks:

1. **List:** Create a list called `cart_items` containing three strings: `"Laptop"`, `"Mouse"`, and `"USB Cable"`. Add `"Keyboard"` to the end of the list.
2. **Tuple:** Store the dimensions of the laptop box (length, width, height) in a `tuple` called `box_dimensions`. *Challenge: Try to change one value in the tuple and observe the error.*
3. **Dictionary:** Create a dictionary `product_info` for the "Laptop". It should have keys for `"brand"`, `"price"`, and `"in_stock"` (a boolean).
4. **Set:** You have a list of categories: `["Electronics", "Office", "Electronics", "Gadgets"]`. Convert this list into a `set` to remove the duplicates.

Q3: The Library Management System

The Setup: You are organizing a small digital library. This helps you practice more complex data structures and type-specific operations.

Your Tasks:

1. **Nested Data:** Create a list of dictionaries called `library`. Each dictionary should represent a book with keys: `"title"`, `"author"`, and `"year"`.
2. **Logic & Types:** The library adds a "unique ID" to every book. Use the `range()` function to generate IDs from 100 to 105 and store them in a list.
3. **String Formatting:** Print the first book's details using an f-string: *"The book {title} was written by {author}."*
4. **Type Checking:** Use the `type()` function to verify that your `library` variable is indeed a `list`.

Q4: The RPG Character Creator

The Setup: You are designing the backend for a fantasy Role Playing Game (RPG). You need to store complex player stats that change as the game progresses.

Your Tasks:

1. **Dictionary:** Create a dictionary named `player_stats`. Include:
    - `"name"` (string)
    - `"level"` (int)

    - `"inventory"` (a list of strings: `"sword"`, `"shield"`, `"potion"`)
2. **Updating Nesting:** The player finds a `"dragon scale"`. Append this new item to the list *inside* the dictionary.
3. **The "Level Up" Math:** The player leveled up! Increase the `level` value in the dictionary by 1.
4. **Tuple Protection:** Store the player's starting coordinates (X, Y) in a tuple called `spawn_point`. Explain why a tuple is better than a list for a spawn point.

Q5: The Weather Station Logger

The Setup: You have a device that records temperatures every hour. You need to handle data cleanup and summary.

Your Tasks:

1. **List to Set:** You have a list of hourly readings: `[22, 24, 22, 21, 24, 25, 22]`. Convert this to a `set` to find all the **unique** temperatures recorded that day.
2. **Boolean Logic:** Create a variable `is_freezing`. Use a comparison operator to set it to `True` if the minimum value in your list is below 0.
3. **Complex String:** Create a variable `report`. Use `.join()` to turn your list of unique temperatures (converted to strings) into a single string separated by commas.

Q6: The Restaurant Menu (Nested Logic)

The Setup: A digital menu needs to categorize items by "Appetizers," "Mains," and "Desserts."

Your Tasks:

1. **Nested Dictionary:** Create a dictionary called `menu`.
    - The keys should be categories (e.g., `"Mains"`).
    - The values should be another dictionary where the key is the dish name and the value is the price.
2. **Accessing Data:** Write a print statement that pulls the price of a specific dish from deep inside the `menu` dictionary.
3. **Type Discovery:** Use `isinstance()` to check if the value of the `"Mains"` key is a dictionary.

Let's push into the "Professional Grade" scenarios. These assignments focus on **Data Integrity**, **State Management**, and handling **Real-World API-style data structures**.

Q7: The Banking Transaction Log

The Setup: You are processing a stream of financial data. Precision and history are key here. You need to distinguish between the *action* and the *amount*.

Your Tasks:

1. **List of Tuples:** Create a variable `transactions`. Store at least four transactions as tuples, like: `("deposit", 500.25), ("withdrawal", 100.00)`.
2. **Immutability Logic:** Why did we use a **tuple** for each transaction instead of a **list**? (Hint: Can a transaction amount be changed after it happens?)
3. **Float Precision:** Create a `current_balance`. Calculate the sum of your transactions.
4. **String Formatting:** Use a "f-string" to display the balance formatted to 2 decimal places (e.g., `$400.25`).

Q8: The Social Media "Like" Tracker

The Setup: You're building the backend for a post's engagement. You need to ensure that one user cannot "Like" a post multiple times.

Your Tasks:

1. **Set:** Create a set called `user_ids_who_liked`. Add three unique strings (usernames).
2. **Duplicate Handling:** Try to `.add()` a username that is already in the set. Check the length of the set—did it change?
3. **Type Conversion:** You need to display these users alphabetically. Convert the `set` into a `list` and use the `.sort()` method.
4. **Boolean Check:** Create a variable `has_liked`. Use the `in` keyword to check if `"alpha_user"` is in your set.

Q9: The Global Shipping Manifest

The Setup: You are managing a shipping container. You have a mix of data: the destination (constant), the item list (changeable), and the weight (numeric).

Your Tasks:

1. **The Master Object:** Create a dictionary called `container`.
    - `id`: An integer.
    - `destination`: A string.
    - `contents`: A list of dictionaries (each dictionary should have `"item_name"` and `"weight"`).

2. **Deep Access:** Access the weight of the *second* item in the `contents` list.
3. **NoneType:** Create a variable `customs_clearance_date` and set it to `None`. This represents that the event hasn't happened yet.
4. **Type Validation:** Write a script that checks if `container["id"]` is an integer using `type()`.