Day 2: Aptitude, Programming & Core Concepts

1. Aptitude: Ratios & Proportions

Ratios and proportions are the foundation of numerical reasoning, used to compare quantities and scale values.

Key Concepts

- **Ratio ($a : b$):** A comparison of two quantities of the same kind. It can be written as a/b.
- **Proportion ($a : b :: c : d$):** An equation that states that two ratios are equal ($a/b = c/d$).
- **Properties:**
  - **Invertendo:** If $a : b = c : d$, then $b : a = d : c$.
  - **Alternendo:** If $a : b = c : d$, then $a : c = b : d$.
  - **Componendo & Dividendo:** If $a/b = c/d$, then $(a + b)/(a - b) = (c + d)/(c - cold)$.

2. Programming: Sum of Digits

The "Sum of Digits" problem is a classic exercise for understanding number manipulation using the Modulo (`%`) and Floor Division (`//`) operators.

Logic

1. Initialize `sum = 0` .
2. Extract the last digit using `num % 10` .
3. Add the digit to `sum` .
4. Remove the last digit using `num // 10` (integer division).
5. Repeat until the number becomes 0.

Python Implementation

```
def sum_of_digits(n):
    total = 0
    while n > 0:
        total += n % 10
        n //= 10
    return total

print(sum_of_digits(1234)) # Output: 10
```

3. Python: Operators & Expressions

Operators are symbols that perform operations on variables and values.

Type	Operators	Description
Arithmetic	`+` , `-` , `*` , `/` , `//` , `%` , `**`	Math operations (includes floor div and power).
Relational	`==` , `!=` , `>` , `<` , `>=` , `<=`	Comparison; returns Boolean (`True` / `False`).
Logical	`and` , `or` , `not`	Used to combine conditional statements.
Assignment	`=` , `+=` , `-=` , `*=` , `/=`	Assigns or updates variable values.

4. C/C++: Operators & Precedence

In C/C++, understanding the "Order of Operations" is crucial for writing bug-free code.

Operator Precedence (Highest to Lowest)

1. **Postfix:** `()` , `[]` , `->` , `++` , `--`
2. **Unary:** `+` , `-` , `!` , `~` , `(type)` , `*` , `&` , `sizeof`
3. **Multiplicative:** `*` , `/` , `%`
4. **Additive:** `+` , `-`
5. **Relational:** `<` , `<=` , `>` , `>=`
6. **Equality:** `==` , `!=`
7. **Logical:** `&&` (AND) then `||` (OR)
8. **Assignment:** `=` , `+=` , `-=` , etc.

5. SQL: Data Types

Choosing the correct data type ensures database efficiency and data integrity.

Numeric Types

- `INT` : Whole numbers.
- `DECIMAL(p,s)` : Exact fixed-point numbers (e.g., money).
- `FLOAT` / `REAL` : Approximate floating-point numbers.

String/Text Types

- `CHAR(n)` : Fixed-length string (padded with spaces).
- `VARCHAR(n)` : Variable-length string (more space-efficient).
- `TEXT` : For long-form data (descriptions, comments).

Date & Time

- `DATE` : Format `YYYY-MM-DD` .
- `TIMESTAMP` : Records a specific point in time (often used for "Created At" fields).

Logical

- `BOOLEAN` : Stores `TRUE` or `FALSE` (often represented as `1` or `0`).

6. Practice Questions

1. **Aptitude:** If $x : y = 3 : 4$, what is the value of $(4x + 3y) : (3x + 4y)$?
2. **Programming:** In the Sum of Digits logic, why do we use floor division (`//`) instead of normal division (`/`) in Python?
3. **Operators:** Evaluate the Python expression: `10 + 5 * 2 ** 3` .
4. **Precedence:** In C++, which operator is evaluated first in the expression `!a && b || c` ?
5. **SQL:** Which SQL data type is most appropriate for storing a user's age?
6. **Aptitude (Mean Proportional):** Find the mean proportional between 9 and 16.
7. **Programming (Edge Case):** What will the "Sum of Digits" program return if the input is a negative number? How can you modify the code to handle this?
8. **Python (Short-Circuiting):** In the expression `True or (10 / 0 == 0)` , does Python raise a `ZeroDivisionError` ? Why or why not?
9. **C/C++ (Increment):** What is the output of the following C++ code snippet? `int x = 5; int y = x++; cout << y << " " << x;`
10. **SQL (Precision):** What is the difference between `DECIMAL(5,2)` and `DECIMAL(10,2)` in terms of the maximum value they can store?