

Day 26 Comprehensive Study Notes

1. Aptitude: Caselets (Data Interpretation)

Caselets are paragraph-based data sets that require you to extract information and organize it (usually into a table or Venn diagram) before solving.

- **Step-by-Step Approach:**

1. **Skimming:** Read the whole paragraph to identify the main subjects (e.g., Number of students, types of courses, genders).
2. **Mapping:** Create a table or a Venn diagram immediately. Do not try to solve questions directly from the text.
3. **Variable Assignment:** Use x or $100x$ for unknown values that are linked by percentages.
4. **Filtering:** Ignore "filler" sentences that don't provide numerical relationships or constraints.

2. Programming: Longest Substring Without Repeating Characters

This problem is best solved using the **Sliding Window** technique with a hash map or a frequency array.

- **Concept:**

- Maintain two pointers: `left` and `right`.
- Expand `right` to include characters.
- If a duplicate character is encountered, shrink the window from the `left` until the duplicate is removed.

- **Efficiency:**

- **Time Complexity:** $O(n)$, where n is the length of the string (each character is visited at most twice).
- **Space Complexity:** $O(\min(m, n))$, where m is the size of the character set (e.g., 256 for ASCII).

- **Key Logic (Pseudo-code):**

```
while right < len(s):
    if s[right] in char_set:
        char_set.remove(s[left])
        left += 1
    else:
        char_set.add(s[right])
        max_len = max(max_len, right - left + 1)
```

```
right += 1
```

3. OOPS Concept: Basics

Object-Oriented Programming is a paradigm based on the concept of "objects," which can contain data and code.

- **Class:** A blueprint or template for creating objects (e.g., `Class: Car`).
- **Object:** A specific instance of a class (e.g., `MyTesla`).
- **Attributes (Properties):** The data associated with an object (e.g., `color` , `speed`).
- **Methods (Functions):** The behaviors or actions an object can perform (e.g., `drive()` , `brake()`).
- **Why use it? * Modularity:** Easier to troubleshoot.
 - **Reusability:** Code can be reused through inheritance.
 - **Flexibility:** Easier to maintain and update.

4. SQL: Mastering JOINS

JOINS allow you to query data from multiple tables by finding commonalities.

Join Type	Description
INNER JOIN	Returns rows only when there is a match in both tables.
LEFT JOIN	Returns all rows from the left table, and matched rows from the right. If no match, right side is NULL.
RIGHT JOIN	Returns all rows from the right table, and matched rows from the left. If no match, left side is NULL.
FULL JOIN	Returns rows when there is a match in either table. Combined view of everything.
CROSS JOIN	Returns the Cartesian product (every row from table A matched with every row from table B).

- **Syntax Tip:** Always use table aliases to keep queries clean:

```
SELECT a.name ,  
b.order_date FROM Users a JOIN Orders b ON a.id = b.user_id;
```

