Atlassian – Senior Data Engineer Interview Master Cheat Sheet

Python String Manipulation, JSON Parsing, Scalability Reasoning, and One Unified Interview Code Example

1. Senior Data Engineer – What Interviewers Look For

• Clean, defensive Python code
• Handling messy real-world input
• Strong JSON manipulation skills
• Performance awareness (O(n), memory)
• Ability to explain scaling to Spark/Databricks

2. One End-to-End Python Example (Covers 90% Interview Needs)

Scenario:
You receive raw string input representing event logs (API / Kafka style).
Tasks:
• Clean strings
• Parse JSON
• Handle missing/malformed fields
• Aggregate metrics
• Produce final JSON output

```
import json
from collections import defaultdict

def process_events(raw_lines):
    results = defaultdict(int)

    for line in raw_lines:
        # Step 1: Clean string input
        line = line.strip()
        if not line:
            continue

        # Step 2: Safe JSON parsing
        try:
            event = json.loads(line)
        except json.JSONDecodeError:
            continue  # skip malformed records

        # Step 3: Defensive access
        event_type = event.get("type", "").lower()
        status = event.get("status", "").lower()

        # Step 4: Business logic
        if event_type == "issue_created" and status == "success":
            project = event.get("project", "unknown")
            results[project] += 1
    # Step 5: Serialize output
    return json.dumps(results, indent=2)

# Example usage
raw_input = [
    '{"type":"Issue_Created","status":"SUCCESS","project":"JIRA"}',
    '{"type":"issue_created","status":"success","project":"CONFLUENCE"}',
    ' malformed json ',
    ''
]

print(process_events(raw_input))
```

3. Why This Code Impresses at Senior Level

• Single-pass O(n) processing
• Defensive against bad input
• Clear variable naming
• Separation of concerns
• Easy to port to Spark (map → filter → groupBy)

4. How You Explain Scaling This to Spark/Databricks

• raw_lines → Spark DataFrame / Dataset
• json.loads → from_json() with schema
• for-loop → map / select
• defaultdict → groupBy + count
• Python logic becomes distributed transformations

5. Typical Atlassian Follow-up Questions

• What happens with billions of rows?
• How do you handle schema evolution?
• How do you monitor bad records?
• How would you test this logic?
• How do you make this idempotent?

6. Final Senior-Level Interview Tips

• State assumptions before coding
• Mention trade-offs explicitly
• Talk about observability (metrics, logs)
• Tie logic back to business outcomes