📘 Section 2: Module 2 – Transforming Commands for Visualizations

🔹 Module Overview

This module focuses on **turning raw data into useful charts and tables** for reporting and dashboards using Splunk's transforming commands.

🔹 9. Visualization Data Structures

Transforming commands are used to reshape event data. The core structure includes:

- **Single-Series Tables**: One metric field (e.g., count by source)
- **Multi-Series Tables**: More than one metric across categories (e.g., average response time by host and status)

🔹 10. Types of Visualizations

- **Tables**: Basic output from stats, chart, timechart
- **Charts**:
  - **Column/Bar Charts** – for comparisons
  - **Line/Area Charts** – for trends over time
  - **Pie Charts** – for proportions
  - **Scatter/Bubble** – for relationships

🔹 11. Statistics Tables

Command:

spl

| stats count by status

This returns a simple key-value summary table.

🔹 12. The chart Command – Single Series

Syntax:

spl

| chart count by source

- Shows the count per source.
- Good for pie/column chart visualizations.

🔹 13. The chart Command – Multi-Series

Syntax:

spl

| chart avg(bytes) over status by host

This creates a **multi-series matrix** (hosts on x-axis, status on y-axis, avg(bytes) as values).

🔹 14. The timechart Command – Single Series

Syntax:

spl

| timechart count

- Automatically bins by time.
- Produces line charts (by default).

🔹 **15. The timechart Command – Multi-Series**

Syntax:

spl

```
| timechart avg(cpu_usage) by host
```

- Each host becomes a separate line in the chart.

🔹 **16. Scatter & Bubble Charts**

Use when comparing:

- **2 metrics:** Scatter
- **3 metrics:** Bubble (third metric as size)

Example:

spl

```
| stats avg(speed) as Speed, avg(size) as Size by host
```

Then use scatter/bubble chart in the Visualization tab.

🔹 **17. Formatting Statistics Tables**

Options include:

- **Decimal formatting**
- **Renaming fields**
- **Rounding**
- Use eval or fieldformat:

spl

```
| eval speed=round(speed,2)
```

🔹 **18. Formatting Visualizations**

You can configure:

- **Axis names**
- **Legend position**
- **Color schemes**
- **Threshold indicators** (on single value viz)

✅ **Quiz 1: Transforming Commands for Visualizations**

💡 **Exam Tips:**

- chart, timechart, stats, eventstats – all can be used with transaction.
- Use stats when grouping by fields, transaction when grouping by session boundaries.
- timechart automatically bins _time.

📘 **Section 3: Module 3 – Advanced Visualizations**

🔹 **19. Module Overview**

This module teaches **custom visualizations** to improve user dashboards.

🔹 20. Single Value Visualizations

Best for KPIs like:

spl

```
| stats count
```

- Can include trend indicators, colors.
- Use thresholds to indicate performance (green/yellow/red).

🔹 21. Maps

Use with fields like lat and lon.

Command:

spl

```
| inputlookup cities.csv | geostats latfield=latitude longfield=longitude count
```

Enables plotting data on geographic maps.

🔹 22. Creating a Trendline

Use trendline command:

spl

```
| trendline sma5(count) as TrendCount
```

- sma = simple moving average over 5 time bins.

✅ Quiz 2: Advanced Visualizations

💡 Exam Tips:

- Use geostats to visualize geographic data.
- sma = Simple Moving Average.
- Single value with sparkline helps show both metric and trend.

📘 Section 4: Module 4 – Filtering & Formatting Results

🔹 23. Module Overview

This module teaches how to **clean, transform, and control output** using commands like eval, if, case.

🔹 24. Using the eval Command

Syntax:

spl

```
| eval status_type=if(status==200, "Success", "Failure")
```

- Used for new fields and calculations.

🔹 25. Calculating Fields

Example:

spl

```
| eval total=price * quantity
```

🔹 **26. Rounding Field Values – round() Function**

Syntax:

spl

```
| eval value=round(value,2)
```

🔹 **27. Converting Fields – tostring() Function**

Used to convert numeric to string:

spl

```
| eval value=tostring(value, "commas")
```

🔹 **28. String Concatenation**

spl

```
| eval fullname=firstname . " " . lastname
```

🔹 **29. The eval Function (Detailed)**

Can be used with:

- Mathematical expressions
- String manipulation
- Conditional logic (if, case)

🔹 **30. The if() Function**

spl

```
| eval status_type=if(status==404, "Error", "OK")
```

🔹 **31. The case() Function**

spl

```
| eval severity=case(status==200,"OK",status==404,"Not Found",1==1,"Other")
```

🔹 **32. The fillnull Command**

Used to fill missing values:

spl

```
| fillnull value="N/A"
```

🔹 **33. Filtering Search Results – search Command**

Used after a pipeline:

spl

```
... | search status=200
```

🔹 **34. Filtering Search Results – where Command**

Used for conditional filters:
spl

... | where price > 100

✅ **Quiz 3: Filtering & Formatting Results**

💡 **Exam Tips:**

- Use if() for binary condition; case() for multiple.
- where works only on transformed results.
- Use fillnull for cleaner tables.

Section 5: Module 5 – Correlating Events

📘 **35. Module Overview**

Provides a conceptual foundation of correlating events across different logs using SPL (Search Processing Language).

📘 **36. Correlating Events – Overview**

- **Event Correlation**: Tying together events from multiple sources to tell a complete story (e.g., login and logout actions).
- Common need in **security**, **IT operations**, and **compliance audits**.

📘 **37. The transaction Command**

- Groups events that share the same field values and occurred within a specific timeframe.
- **Syntax**:

spl

transaction user maxspan=30m

  - user = grouping field
  - maxspan=30m = total time window of the transaction

- **Options**:
  - startswith= / endswith= – delimit the start and end conditions.
  - maxevents= – max events allowed per transaction (default is 1000).
  - maxpause= – time gap allowed between successive events.

Use Case: Detecting when a user logs in but doesn't log out.

📘 **38. Filtering Transactions**

- Add filters after transaction to refine:

spl

transaction user startswith="login" endswith="logout" | search duration > 300

📘 **39. Transaction Constraints**

- **Limitations**:
  - Slower in large environments.
  - Limited to 1000 events per transaction.
  - Memory-intensive.

📘 40. Report on Transactions

- Extract metadata:

spl

```
transaction user startswith="login" endswith="logout"
| stats avg(duration), count by user
```

📘 41. transaction vs stats

Feature	transaction	stats
Performance	Slower, memory-heavy	Fast and scalable
Use Case	Start-End correlation	Field-based aggregations
Syntax	transaction command	stats count by field
Example	login-logout session detection	total logins per user

✅ Exam Tips:

- Use transaction when **event start and end must be captured**, e.g., login to logout.
- Prefer stats in **large datasets** for efficiency.
- **Default limit = 1000 events** per transaction.

Section 6: Module 6 – Creating & Managing Fields

📘 42. Module Overview

Focus on how Splunk identifies, extracts, and manages fields.

📘 43. Overview of Knowledge Objects

- Reusable objects (e.g., tags, fields, macros) created during search and shared with users.
- Field extractions are a major type.

📘 44. Why Extract Fields from Data?

- Makes unstructured data **queryable**.
- Allows you to create dashboards and alerts based on new fields.

📘 45. Structured vs Unstructured Data

Data Type	Example	Field Extraction
Structured	JSON, XML	Automatic
Unstructured	Logs, free-text	Manual/Regex

📘 46. Field Discovery (Auto-Extraction)

- Splunk automatically extracts fields at **index time** and **search time**.
- Visible in the **Fields Sidebar** of the Search UI.

📘 47. Field Extractions with Knowledge Objects

- Created manually in Splunk Web:
  - Settings → Fields → Field Extractions
  - Use **regex** or **delimiters**.

📘 48. Delimiter Field Extractions

- Use when log data is CSV-like:

spl

```
| rex field=_raw "^(?<field1>\w+), (?<field2>\d+), (?<field3>\w+)"
```

📘 49. RegEx Field Extractions

- RegEx is flexible and powerful:

spl

```
| rex "user=(?<username>\w+)"
```

📘 50. Modify RegEx Expressions

- Test in Splunk field extractor GUI.
- Use named capture groups (?<fieldname>) to name fields.

✅ Exam Tips:

- Use **auto-extraction** when fields are predictable.
- For **complex logs**, use **rex** or field extractor tool.
- **Search-time field extraction** is lighter and flexible.

Section 7: Module 7 – Creating Field Aliases & Calculated Fields

📘 51. Module Overview

Focuses on **normalizing fields** without modifying raw data.

📘 52. What is a Field Alias?

- Alternate name for an existing field.
- Useful when different sourcetypes use different field names for the same thing.
- **Example**:

spl

```
host = server1 OR hostname = server1
```

📘 53. Creating Field Aliases

- Go to:
  Settings → Fields → Field Aliases
- Define:
  - **Original field**
  - **Alias name**
  - **Applicable sourcetype**

📘 54. What is a Calculated Field?

- Field created dynamically using an expression (eval).
- Does not exist in raw data.

📘 55. Creating Calculated Fields

- Go to:
  Settings → Fields → Calculated Fields
- **Syntax Example**:

spl

eval total_bytes = bytes_in + bytes_out

✅ **Exam Tips:**

- Use **field aliases** for consistent reporting across sourcetypes.
- Use **calculated fields** to derive new values like totals or status flags.
- Both are **search-time knowledge objects**.

Section 8: Module 8 – Creating Tags & Event Types

📘 **56. Module Overview**

Organize and categorize your data for better usability and faster searching.

📘 **57. What are Tags?**

- Labels that describe **fields or field values**.
- Help with grouping for **compliance**, **reporting**, or **visuals**.

📘 **58. Creating Tags**

- Go to:
  Settings → Fields → Tags
- Select:
  - **Field value**
  - Assign a tag name.

📘 **59. Managing Tags**

- View, edit, and delete via **Tag Manager**.
- Tags are stored as knowledge objects.

📘 **60. What are Event Types?**

- A **saved search** with a label.
- Group related events under a name.

📘 **61. Creating Event Types**

- Use:

spl

sourcetype=access_combined status=404
Then save as event type: error_404

📘 **62. Tagging Event Types**

- Tags can be applied **to event types**, making them easy to search.

📘 **63. Event Types vs Saved Reports**

Feature	Event Type	Saved Report
Purpose	Classification of events	Store and share results
Editable	Yes	Yes
Scheduling	No	Yes

✅ **Exam Tips:**

- Tags enhance **metadata**.
- Event types are often used in **dashboards** and **alerts**.
- Tag + Event Type is a common **SIEM practice**.

Section 9: Module 9 – Creating & Using Macros

📘 **64. Module Overview**

Allows you to **reuse SPL** using parameterized logic.

📘 **65. What is a Macro?**

- A **search fragment** stored as a reusable object.
- Stored in: macros.conf

📘 **66. Creating a Basic Macro**

- Create via:
  Settings → Advanced Search → Search Macros
- **Syntax Example**:

spl

`failed_logins`
Macro:
spl

sourcetype=secure action=failure

📘 **67. Creating a Macro with Arguments**

- Macros can accept parameters:

spl

`bytes_macro(host)`
Definition:
spl

host=$host$

📘 **68. Validating Macro Arguments**

- Use isnum(), isstr() to validate.
- Helps prevent injection/mistakes.

✅ **Exam Tips:**

- Macros **simplify long searches**.

- Use arguments for **flexibility**.
- Enclose macros in **backticks**: `macro_name`

Section 10: Module 10 – Creating & Using Workflow Actions

📘 **69. Module Overview**

Workflow actions allow users to interact with search results by drilling into external or internal Splunk resources (like lookups, dashboards, or websites).

📘 **70. What is a Workflow Action?**

- A workflow action is a **contextual action** linked to a field in your search result.
- Use it to:
  - Open related dashboards.
  - Trigger a secondary search.
  - Redirect to an external tool with the value.

📘 **71. Creating a GET Workflow Action**

- **GET** appends field values to a URL query string.
- Example: Link IPs in logs to VirusTotal.

Steps:

- Go to: Settings → Fields → Workflow Actions → Add new
- Action type: **link**
- Link method: **GET**
- Example:

ruby

https://www.virustotal.com/gui/ip-address/ip

📘 **72. Creating a POST Workflow Action**

- Sends field values to another server via **HTTP POST**.
- Used for:
  - Logging
  - Ticketing systems (like JIRA, ServiceNow)

Steps:

- Action type: **link**
- Link method: **POST**
- Provide key-value pairs of fields.

📘 **73. Creating a Search Workflow Action**

- Launches another search using a field.
- Example:

ini

index=firewall src_ip="src_ip"

Tip: Use quotes to prevent query issues with special characters.

✅ **Exam Tips:**

- **GET** = Used for **URL-based** lookups (ex: open VirusTotal).

- **POST** = Used for **form submissions** or **ticketing**.
- Use **workflow actions** for analyst interactivity within Splunk.

Section 11: Module 11 – Creating Data Models

📘 74. Module Overview
Data Models are structured representations of your raw data, optimized for Pivot use and acceleration.

📘 75. What is a Data Model?
- A **hierarchical mapping** of data for reporting and visualization.
- Backbone of **Pivot** and **Data Model Acceleration** (DMA).
- Stored in datamodels.conf.

📘 76. Creating a Data Model
- Go to: Settings → Data Models → New
- Define:
  - Title
  - App
  - Root dataset type: **event**, **search**, or **transaction**

📘 77. Adding Fields to a Data Model – Auto-Extracted
- Fields that Splunk identifies during search time are added directly.
- Appears automatically in Pivot.

📘 78. Adding Fields via Eval Expressions
- Use **eval expressions** to define fields.
- Example:

spl

eval severity=if(status>=400, "High", "Low")

📘 79. Adding Fields via Lookup
- Leverages lookup table to enrich dataset.
- Example: Add geo info via ip_location.csv.

📘 80. Adding Fields via Regular Expression
- Add field extraction via regex.
- Pattern Example:

ini

user=(?<username>\w+)

📘 81. Adding Fields via GeoIP
- Auto-enrich data using geolocation fields (like country, city, longitude).

📘 82. Adding a Root Transaction Dataset
- Base dataset built with a **transaction** definition.

- Useful when events must be grouped logically (login → logout).

📘 **83. Adding Child Datasets**

- Add datasets **under** the root (to organize by action, category).
- Inherit root filters and structure.

📘 **84. Using Pivot Tool**

- Pivot lets **non-technical users** generate reports using drag-drop.
- Works only with **accelerated** or **fully defined** data models.

📘 **85. Data Model Permissions & Acceleration**

- Enable DMA for faster queries.
- **Permissions** define who can edit or view data models.

📘 **86. Download & Upload Data Models**

- Export to JSON.
- Use to migrate between environments.

✅ **Exam Tips:**

- Data models are required for **Pivot**.
- **Root dataset** can be **event**, **search**, or **transaction**.
- Use **DMA** (Data Model Acceleration) for fast reports.

Section 12: Module 12 – Using the Common Information Model (CIM) Add-On

---

📘 **87. Module Overview**

CIM provides a **standardized naming scheme** across all data types in Splunk (e.g., authentication, network, email).

📘 **88. What is Splunk CIM?**

- A **data normalization framework**.
- Required for:
  - Splunk Enterprise Security
  - Unified dashboards
  - CIM-compliant apps

📘 **89. Splunk CIM Data Models**

- Bundled with predefined data models like:
  - Authentication
  - Network Traffic
  - Endpoint

📘 **90. Installing Splunk CIM Add-On**

- App name: **Splunk Common Information Model (CIM)**
- Install via Splunkbase.
- Available for free.

📘 91. Using CIM Add-On – Create Tags

- Tags link raw events to **CIM-compliant fields**.
- Example:
  - Tag action=login to tag=authentication

📘 92. Using CIM Add-On – Create Aliases

- Use **field aliases** to map raw field names to CIM ones.
- Example:
  - src_ip → src

📘 93. Using CIM Add-On – Create Missing Fields

- Create eval or calculated fields to match CIM needs.
- Example:

spl

```
eval src=coalesce(src_ip, ip)
```

📘 94. Validating Against Data Model

- Use datamodel command:

spl

```
| datamodel Authentication Authentication search
```

- Ensures your events match required structure.

✅ Exam Tips:

- CIM is **mandatory** for **Splunk Enterprise Security**.
- Use **tags**, **aliases**, and **calculated fields** to normalize data.
- Validate using **Pivot** or | datamodel SPL.

Section 13: Practice Tests

📘 95. Practice Test 1: Splunk Core Certified Power User Practice Test I

- 50 scenario-based MCQs
- Includes:
  - SPL command usage
  - Field extractions
  - Search optimizations

📘 96. Practice Test 2: Splunk Core Certified Power User Practice Test II

- Additional 50 questions
- Focus on:
  - Workflow Actions
  - Knowledge objects
  - CIM-related applications

✅ Final Exam Tips:

- Focus on **command behavior** and when to use stats, transaction, eval.

- Be fluent in:
    - Search vs. index time
    - Permissions and knowledge object scope
    - Naming conventions in CIM