book_to_slide_BY_sections_V3

July 1, 2025

1 Set up Paths

[11]:
```
# Cell 1: Setup and Configuration
import os
import re
import logging
import warnings
from docx import Document
import pdfplumber
import ollama
from tenacity import retry, stop_after_attempt, wait_exponential, RetryError
import json

# Setup Logger for this cell
logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s -␣
 ↪%(message)s')
logger = logging.getLogger(__name__)

# --- 1. CORE SETTINGS ---
# Set this to True for EPUB, False for PDF. This controls the entire notebook's␣
 ↪flow.
PROCESS_EPUB = True # for EPUB
# PROCESS_EPUB = False # for PDF

# --- 2. INPUT FILE NAMES ---
# The name of the Unit Outline file (e.g., DOCX, PDF)
UNIT_OUTLINE_FILENAME = "ICT312 Digital Forensic_Final.docx" # epub
# UNIT_OUTLINE_FILENAME = "ICT311 Applied Cryptography.docx" # pdf

EXTRACT_UO = False

# The names of the book files
EPUB_BOOK_FILENAME = "Bill Nelson, Amelia Phillips, Christopher Steuart - Guide␣
 ↪to Computer Forensics and Investigations_ Processing Digital␣
 ↪Evidence-Cengage Learning (2018).epub"
```

```
PDF_BOOK_FILENAME = "(Chapman & Hall_CRC Cryptography and Network Security␣
 ↪Series) Jonathan Katz, Yehuda Lindell - Introduction to Modern␣
 ↪Cryptography-CRC Press (2020).pdf"

# --- 3. DIRECTORY STRUCTURE ---
# Define the base path to your project to avoid hardcoding long paths everywhere
PROJECT_BASE_DIR = "/home/sebas_dev_linux/projects/course_generator"

# Define subdirectories relative to the base path
DATA_DIR = os.path.join(PROJECT_BASE_DIR, "data")
PARSE_DATA_DIR = os.path.join(PROJECT_BASE_DIR, "Parse_data")

# Construct full paths for clarity
INPUT_UO_DIR = os.path.join(DATA_DIR, "UO")
INPUT_BOOKS_DIR = os.path.join(DATA_DIR, "books")
OUTPUT_PARSED_UO_DIR = os.path.join(PARSE_DATA_DIR, "Parse_UO")
OUTPUT_PARSED_TOC_DIR = os.path.join(PARSE_DATA_DIR, "Parse_TOC_books")
OUTPUT_DB_DIR = os.path.join(DATA_DIR, "DataBase_Chroma")

# --- 4. LLM & EMBEDDING CONFIGURATION ---
LLM_PROVIDER = "ollama"  # Can be "ollama", "openai", "gemini"
OLLAMA_HOST = "http://localhost:11434"
OLLAMA_MODEL = "qwen3:8b" # "qwen3:8b", #"mistral:latest"
EMBEDDING_MODEL_OLLAMA = "nomic-embed-text"
CHUNK_SIZE = 800
CHUNK_OVERLAP = 100

# --- 5. DYNAMICALLY GENERATED PATHS & IDs (DO NOT EDIT THIS SECTION) ---
# This section uses the settings above to create all the necessary variables␣
 ↪for later cells.

# Extract Unit ID from the filename
def print_header(text: str, char: str = "="):
    """Prints a centered header to the console."""
    print("\n" + char * 80)
    print(text.center(80))
    print(char * 80)

def extract_uo_id_from_filename(filename: str) -> str:
    match = re.match(r'^[A-Z]+\d+', os.path.basename(filename))
    if match:
        return match.group(0)
    raise ValueError(f"Could not extract a valid Unit ID from filename:␣
 ↪'{filename}'")

try:
    UNIT_ID = extract_uo_id_from_filename(UNIT_OUTLINE_FILENAME)
```

```
except ValueError as e:
    print(f"Error: {e}")
    UNIT_ID = "UNKNOWN_ID"

# Full path to the unit outline file
FULL_PATH_UNIT_OUTLINE = os.path.join(INPUT_UO_DIR, UNIT_OUTLINE_FILENAME)

# Determine which book and output paths to use based on the PROCESS_EPUB flag
if PROCESS_EPUB:
    BOOK_PATH = os.path.join(INPUT_BOOKS_DIR, EPUB_BOOK_FILENAME)
    PRE_EXTRACTED_TOC_JSON_PATH = os.path.join(OUTPUT_PARSED_TOC_DIR,␣
 ↪f"{UNIT_ID}_epub_table_of_contents.json")
else:
    BOOK_PATH = os.path.join(INPUT_BOOKS_DIR, PDF_BOOK_FILENAME)
    PRE_EXTRACTED_TOC_JSON_PATH = os.path.join(OUTPUT_PARSED_TOC_DIR,␣
 ↪f"{UNIT_ID}_pdf_table_of_contents.json")

# Define paths for the vector database
file_type_suffix = 'epub' if PROCESS_EPUB else 'pdf'
CHROMA_PERSIST_DIR = os.path.join(OUTPUT_DB_DIR,␣
 ↪f"chroma_db_toc_guided_chunks_{file_type_suffix}")
CHROMA_COLLECTION_NAME = f"book_toc_guided_chunks_{file_type_suffix}_v2"

# Define path for the parsed unit outline
PARSED_UO_JSON_PATH = os.path.join(OUTPUT_PARSED_UO_DIR, f"{os.path.
 ↪splitext(UNIT_OUTLINE_FILENAME)[0]}_parsed.json")

# --- Sanity Check Printout ---
print("--- CONFIGURATION SUMMARY ---")
print(f"Processing Mode: {'EPUB' if PROCESS_EPUB else 'PDF'}")
print(f"Unit ID: {UNIT_ID}")
print(f"Unit Outline Path: {FULL_PATH_UNIT_OUTLINE}")
print(f"Book Path: {BOOK_PATH}")
print(f"Parsed UO Output Path: {PARSED_UO_JSON_PATH}")
print(f"Parsed ToC Output Path: {PRE_EXTRACTED_TOC_JSON_PATH}")
print(f"Vector DB Path: {CHROMA_PERSIST_DIR}")
print(f"Vector DB Collection: {CHROMA_COLLECTION_NAME}")
print("--- SETUP COMPLETE ---")
```

```
--- CONFIGURATION SUMMARY ---
Processing Mode: EPUB
Unit ID: ICT312
Unit Outline Path:
/home/sebas_dev_linux/projects/course_generator/data/UO/ICT312 Digital
Forensic_Final.docx
Book Path: /home/sebas_dev_linux/projects/course_generator/data/books/Bill
Nelson, Amelia Phillips, Christopher Steuart - Guide to Computer Forensics and
```

```
Investigations_ Processing Digital Evidence-Cengage Learning (2018).epub
Parsed UO Output Path:
/home/sebas_dev_linux/projects/course_generator/Parse_data/Parse_UO/ICT312
Digital Forensic_Final_parsed.json
Parsed ToC Output Path: /home/sebas_dev_linux/projects/course_generator/Parse_da
ta/Parse_TOC_books/ICT312_epub_table_of_contents.json
Vector DB Path: /home/sebas_dev_linux/projects/course_generator/data/DataBase_Ch
roma/chroma_db_toc_guided_chunks_epub
Vector DB Collection: book_toc_guided_chunks_epub_v2
--- SETUP COMPLETE ---
```

2 System Prompt

[12]:
```
UNIT_OUTLINE_SYSTEM_PROMPT_TEMPLATE = """
You are an expert academic assistant tasked with parsing a university unit␣
 ↪outline document and extracting key information into a structured JSON␣
 ↪format.

The input will be the raw text content of a unit outline. Your goal is to␣
 ↪identify and extract the following details and structure them precisely as␣
 ↪specified in the JSON schema below. Note: do not change any key name

**JSON Output Schema:**

```json
{{
  "unitInformation": {{
    "unitCode": "string | null",
    "unitName": "string | null",
    "creditPoints": "integer | null",
    "unitRationale": "string | null",
    "prerequisites": "string | null"
  }},
  "learningOutcomes": [
    "string"
  ],
  "assessments": [
    {{
      "taskName": "string",
      "description": "string",
      "dueWeek": "string | null",
      "weightingPercent": "integer | null",
      "learningOutcomesAssessed": "string | null"
    }}
  ],
  "weeklySchedule": [
```
```

```
    {{
      "week": "string",
      "contentTopic": "string",
      "requiredReading": "string | null"
    }}
  ],
  "requiredReadings": [
    "string"
  ],
  "recommendedReadings": [
    "string"
  ]
}}

Instructions for Extraction:
Unit Information: Locate Unit Code, Unit Name, Credit Points. Capture 'Unit␣
 ↪Overview / Rationale' as unitRationale. Identify prerequisites.
Learning Outcomes: Extract each learning outcome statement.
Assessments: Each task as an object. Capture full task name, description, Due␣
 ↪Week, Weighting % (number), and Learning Outcomes Assessed.
weeklySchedule: Each week as an object. Capture Week, contentTopic, and␣
 ↪requiredReading.
Required and Recommended Readings: List full text for each.
**Important Considerations for the LLM**:
Pay close attention to headings and table structures.
If information is missing, use null for string/integer fields, or an empty list␣
 ↪[] for array fields.
Do no change keys in the template given
Ensure the output is ONLY the JSON object, starting with {{{{ and ending with␣
 ↪}}}}. No explanations or conversational text before or after the JSON.
Now, parse the following unit outline text:
--- UNIT_OUTLINE_TEXT_START ---
{outline_text}
--- UNIT_OUTLINE_TEXT_END ---
"""
```

[13]:
```
# Place this in a new cell after your imports, or within Cell 3 before the␣
 ↪functions.
# This code is based on the schema from your screenshot on page 4.

from pydantic import BaseModel, Field, ValidationError
from typing import List, Optional
import time

# Define Pydantic models that match your JSON schema
class UnitInformation(BaseModel):
    unitCode: Optional[str] = None
```

```
    unitName: Optional[str] = None
    creditPoints: Optional[int] = None
    unitRationale: Optional[str] = None
    prerequisites: Optional[str] = None

class Assessment(BaseModel):
    taskName: str
    description: str
    dueWeek: Optional[str] = None
    weightingPercent: Optional[int] = None
    learningOutcomesAssessed: Optional[str] = None

class WeeklyScheduleItem(BaseModel):
    week: str
    contentTopic: str
    requiredReading: Optional[str] = None

class ParsedUnitOutline(BaseModel):
    unitInformation: UnitInformation
    learningOutcomes: List[str]
    assessments: List[Assessment]
    weeklySchedule: List[WeeklyScheduleItem]
    requiredReadings: List[str]
    recommendedReadings: List[str]
```

3 Extrac Unit outline details to process following steps - output raw json with UO details

[4]:
```
# Cell 3: Parse Unit Outline


# --- Helper Functions for Parsing ---
def extract_text_from_file(filepath: str) -> str:
    _, ext = os.path.splitext(filepath.lower())
    if ext == '.docx':
        doc = Document(filepath)
        full_text = [p.text for p in doc.paragraphs]
        for table in doc.tables:
            for row in table.rows:
                full_text.append(" | ".join(cell.text for cell in row.cells))
        return '\n'.join(full_text)
    elif ext == '.pdf':
        with pdfplumber.open(filepath) as pdf:
            return "\n".join(page.extract_text() for page in pdf.pages if page.
 ↪extract_text())
    else:
```

```
        raise TypeError(f"Unsupported file type: {ext}")

def parse_llm_json_output(content: str) -> dict:
    try:
        match = re.search(r'\{.*\}', content, re.DOTALL)
        if not match: return None
        return json.loads(match.group(0))
    except (json.JSONDecodeError, TypeError):
        return None

@retry(stop=stop_after_attempt(3), wait=wait_exponential(min=2, max=10))
def call_ollama_with_retry(client, prompt):
    logger.info(f"Calling Ollama model '{OLLAMA_MODEL}'...")
    response = client.chat(
        model=OLLAMA_MODEL,
        messages=[{"role": "user", "content": prompt}],
        format="json",
        options={"temperature": 0.0}
    )
    if not response or 'message' not in response or not response['message'].
 ↪get('content'):
        raise ValueError("Ollama returned an empty or invalid response.")
    return response['message']['content']

# --- Main Orchestration Function for this Cell ---
def parse_and_save_outline_robust(
    input_filepath: str,
    output_filepath: str,
    prompt_template: str,
    max_retries: int = 3
):
    logger.info(f"Starting to robustly process Unit Outline: {input_filepath}")

    if not os.path.exists(input_filepath):
        logger.error(f"Input file not found: {input_filepath}")
        return

    try:
        outline_text = extract_text_from_file(input_filepath)
        if not outline_text.strip():
            logger.error("Extracted text is empty. Aborting.")
            return
    except Exception as e:
        logger.error(f"Failed to extract text from file: {e}", exc_info=True)
        return

    client = ollama.Client(host=OLLAMA_HOST)
```

```
    current_prompt = prompt_template.format(outline_text=outline_text)

    for attempt in range(max_retries):
        logger.info(f"Attempt {attempt + 1}/{max_retries} to parse outline.")

        try:
            # Call the LLM
            llm_output_str = call_ollama_with_retry(client, current_prompt)

            # Find the JSON blob in the response
            json_blob = parse_llm_json_output(llm_output_str) # Your existing␣
↪helper
            if not json_blob:
                raise ValueError("LLM did not return a parsable JSON object.")

            # *** THE KEY VALIDATION STEP ***
            # Try to parse the dictionary into your Pydantic model.
            # This will raise a `ValidationError` if keys are wrong, types are␣
↪wrong, or fields are missing.
            parsed_data = ParsedUnitOutline.model_validate(json_blob)

            # If successful, save the validated data and exit the loop
            logger.info("Successfully validated JSON structure against Pydantic␣
↪model.")
            os.makedirs(os.path.dirname(output_filepath), exist_ok=True)
            with open(output_filepath, 'w', encoding='utf-8') as f:
                # Use .model_dump_json() for clean, validated output
                f.write(parsed_data.model_dump_json(indent=2))

            logger.info(f"Successfully parsed and saved Unit Outline to:␣
↪{output_filepath}")
            return # Exit function on success

        except ValidationError as e:
            logger.warning(f"Validation failed on attempt {attempt + 1}. Error:␣
↪{e}")
            # Formulate a new prompt with the error message for self-correction
            error_feedback = (
                f"\n\nYour previous attempt failed. You MUST correct the␣
↪following errors:\n"
                f"{e}\n\n"
                f"Please regenerate the entire JSON object, ensuring it␣
↪strictly adheres to the schema "
                f"and corrects these specific errors. Do not change any key␣
↪names."
            )
```

```
            current_prompt = current_prompt + error_feedback # Append the error␣
 ↪to the prompt

        except Exception as e:
            # Catch other errors like network issues from call_ollama_with_retry
            logger.error(f"An unexpected error occurred on attempt {attempt +␣
 ↪1}: {e}", exc_info=True)
            # You might want to wait before retrying for non-validation errors
            time.sleep(5)

    logger.error(f"Failed to get valid structured data from the LLM after␣
 ↪{max_retries} attempts.")


# --- In your execution block, call the new function ---
# parse_and_save_outline(...) becomes:

if EXTRACT_UO:
    parse_and_save_outline_robust(
        input_filepath=FULL_PATH_UNIT_OUTLINE,
        output_filepath=PARSED_UO_JSON_PATH,
        prompt_template=UNIT_OUTLINE_SYSTEM_PROMPT_TEMPLATE
    )
```

4 Extract TOC from epub or epub

[14]:
```
# Cell 4: Extract Book Table of Contents (ToC)
# This cell extracts the ToC from the specified book (EPUB or PDF)
# and saves it to the path defined in Cell 1.

from ebooklib import epub, ITEM_NAVIGATION
from bs4 import BeautifulSoup
import fitz  # PyMuPDF
import json

# --- EPUB Extraction Logic ---
def parse_navpoint(navpoint, level=0):
    # (Your existing parse_navpoint function)
    title = navpoint.navLabel.text.strip()
    # Add filtering logic here if needed
    node = {"level": level, "title": title, "children": []}
    for child_navpoint in navpoint.find_all('navPoint', recursive=False):
        child_node = parse_navpoint(child_navpoint, level + 1)
        if child_node: node["children"].append(child_node)
    return node
```

```python
def parse_li(li_element, level=0):
    # (Your existing parse_li function)
    a_tag = li_element.find('a')
    if a_tag:
        title = a_tag.get_text(strip=True)
        # Add filtering logic here if needed
        node = {"level": level, "title": title, "children": []}
        nested_ol = li_element.find('ol')
        if nested_ol:
            for sub_li in nested_ol.find_all('li', recursive=False):
                child_node = parse_li(sub_li, level + 1)
                if child_node: node["children"].append(child_node)
        return node
    return None

def extract_epub_toc(epub_path, output_json_path):
    print(f"Processing EPUB ToC for: {epub_path}")
    toc_data = []
    book = epub.read_epub(epub_path)
    for nav_item in book.get_items_of_type(ITEM_NAVIGATION):
        soup = BeautifulSoup(nav_item.get_content(), 'xml')
        if nav_item.get_name().endswith('.ncx'):
            print("INFO: Found EPUB 2 (NCX) Table of Contents.")
            navmap = soup.find('navMap')
            if navmap:
                for navpoint in navmap.find_all('navPoint', recursive=False):
                    node = parse_navpoint(navpoint, level=0)
                    if node: toc_data.append(node)
        else:
            print("INFO: Found EPUB 3 (XHTML) Table of Contents.")
            toc_nav = soup.select_one('nav[epub|type="toc"]')
            if toc_nav:
                top_ol = toc_nav.find('ol')
                if top_ol:
                    for li in top_ol.find_all('li', recursive=False):
                        node = parse_li(li, level=0)
                        if node: toc_data.append(node)
        if toc_data: break

    if toc_data:
        os.makedirs(os.path.dirname(output_json_path), exist_ok=True)
        with open(output_json_path, 'w', encoding='utf-8') as f:
            json.dump(toc_data, f, indent=2, ensure_ascii=False)
        print(f"  Successfully wrote EPUB ToC to: {output_json_path}")
    else:
        print("  WARNING: No ToC data extracted from EPUB.")
```

```python
# --- PDF Extraction Logic ---
def build_pdf_hierarchy(toc_list):
    """
    Builds a hierarchical structure from a flat ToC list from PyMuPDF.
    MODIFIED: Normalizes levels to start at 0 for consistency with EPUB.
    """
    root = []
    # The parent_stack keys are now level-based, starting from -1 for the root's parent.
    parent_stack = {-1: {"children": root}}

    for level, title, page in toc_list:
        # --- FIX: NORMALIZE LEVEL TO START AT 0 ---
        # fitz/PyMuPDF ToC levels start at 1, so we subtract 1.
        normalized_level = level - 1

        node = {
            "level": normalized_level,
            "title": title.strip(),
            "page": page,
            "children": []
        }

        # Find the correct parent in the stack. The parent's level is one less than the current node's.
        # This logic correctly places the node under its parent in the hierarchy.
        parent_node = parent_stack[normalized_level - 1]
        parent_node["children"].append(node)

        # Add the current node to the stack so it can be a parent for subsequent nodes.
        parent_stack[normalized_level] = node

    return root

def extract_pdf_toc(pdf_path, output_json_path):
    print(f"Processing PDF ToC for: {pdf_path}")
    try:
        doc = fitz.open(pdf_path)
        toc = doc.get_toc()
        if not toc:
            print("  WARNING: This PDF has no embedded bookmarks (ToC).")
            hierarchical_toc = []
        else:
            print(f"INFO: Found {len(toc)} bookmark entries.")
            hierarchical_toc = build_pdf_hierarchy(toc)
```

```
        os.makedirs(os.path.dirname(output_json_path), exist_ok=True)
        with open(output_json_path, 'w', encoding='utf-8') as f:
            json.dump(hierarchical_toc, f, indent=2, ensure_ascii=False)
        print(f"  Successfully wrote PDF ToC to: {output_json_path}")

    except Exception as e:
        print(f"An error occurred during PDF ToC extraction: {e}")

# --- Execute ToC Extraction ---
if PROCESS_EPUB:
    extract_epub_toc(BOOK_PATH, PRE_EXTRACTED_TOC_JSON_PATH)
else:
    extract_pdf_toc(BOOK_PATH, PRE_EXTRACTED_TOC_JSON_PATH)
```

```
Processing EPUB ToC for:
/home/sebas_dev_linux/projects/course_generator/data/books/Bill Nelson, Amelia
Phillips, Christopher Steuart - Guide to Computer Forensics and Investigations_
Processing Digital Evidence-Cengage Learning (2018).epub
INFO: Found EPUB 2 (NCX) Table of Contents.
  Successfully wrote EPUB ToC to: /home/sebas_dev_linux/projects/course_generato
r/Parse_data/Parse_TOC_books/ICT312_epub_table_of_contents.json
```

5 Hirachical DB base on TOC

5.1 Process Book

[15]:
```
# Cell 5: Create Hierarchical Vector Database (with Sequential ToC ID and Chunk␣
 ↪ID)
# This cell processes the book, enriches it with hierarchical and sequential␣
 ↪metadata,
# chunks it, and creates the final vector database.

import os
import json
import shutil
import logging
from typing import List, Dict, Any, Tuple
from langchain_core.documents import Document
from langchain_community.document_loaders import PyPDFLoader,␣
 ↪UnstructuredEPubLoader
from langchain_ollama.embeddings import OllamaEmbeddings
from langchain_chroma import Chroma
from langchain.text_splitter import RecursiveCharacterTextSplitter

# Setup Logger for this cell
```

```
logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s -␣
 ↪%(message)s')
logger = logging.getLogger(__name__)

# --- Helper: Clean metadata values for ChromaDB ---
def clean_metadata_for_chroma(value: Any) -> Any:
    """Sanitizes metadata values to be compatible with ChromaDB."""
    if isinstance(value, list): return ", ".join(map(str, value))
    if isinstance(value, dict): return json.dumps(value)
    if isinstance(value, (str, int, float, bool)) or value is None: return value
    return str(value)

# --- Core Function to Process Book with Pre-extracted ToC ---
def process_book_with_extracted_toc(
    book_path: str,
    extracted_toc_json_path: str,
    chunk_size: int,
    chunk_overlap: int
) -> Tuple[List[Document], List[Dict[str, Any]]]:

    logger.info(f"Processing book '{os.path.basename(book_path)}' using ToC␣
 ↪from '{os.path.basename(extracted_toc_json_path)}'.")

    # 1. Load the pre-extracted hierarchical ToC
    try:
        with open(extracted_toc_json_path, 'r', encoding='utf-8') as f:
            hierarchical_toc = json.load(f)
        if not hierarchical_toc:
            logger.error(f"Pre-extracted ToC at '{extracted_toc_json_path}' is␣
 ↪empty or invalid.")
            return [], []
        logger.info(f"Successfully loaded pre-extracted ToC with␣
 ↪{len(hierarchical_toc)} top-level entries.")
    except Exception as e:
        logger.error(f"Error loading pre-extracted ToC JSON: {e}",␣
 ↪exc_info=True)
        return [], []

    # 2. Load all text elements/pages from the book
    all_raw_book_docs: List[Document] = []
    _, file_extension = os.path.splitext(book_path.lower())

    if file_extension == ".epub":
        loader = UnstructuredEPubLoader(book_path, mode="elements",␣
 ↪strategy="fast")
        try:
```

```
            all_raw_book_docs = loader.load()
            logger.info(f"Loaded {len(all_raw_book_docs)} text elements from␣
↪EPUB.")
        except Exception as e:
            logger.error(f"Error loading EPUB content: {e}", exc_info=True)
            return [], hierarchical_toc
    elif file_extension == ".pdf":
        loader = PyPDFLoader(book_path)
        try:
            all_raw_book_docs = loader.load()
            logger.info(f"Loaded {len(all_raw_book_docs)} pages from PDF.")
        except Exception as e:
            logger.error(f"Error loading PDF content: {e}", exc_info=True)
            return [], hierarchical_toc
    else:
        logger.error(f"Unsupported book file format: {file_extension}")
        return [], hierarchical_toc

    if not all_raw_book_docs:
        logger.error("No text elements/pages loaded from the book.")
        return [], hierarchical_toc

    # 3. Create enriched LangChain Documents by matching ToC to content
    final_documents_with_metadata: List[Document] = []

    # Flatten the ToC, AND add a unique sequential ID for sorting and␣
↪validation.
    flat_toc_entries: List[Dict[str, Any]] = []

    def _add_ids_and_flatten_recursive(nodes: List[Dict[str, Any]],␣
↪current_titles_path: List[str], counter: List[int]):
        """
        Recursively traverses ToC nodes to flatten them and assign a unique,␣
↪sequential toc_id.
        """
        for node in nodes:
            toc_id = counter[0]
            counter[0] += 1
            title = node.get("title", "").strip()
            if not title: continue
            new_titles_path = current_titles_path + [title]
            entry = {
                "titles_path": new_titles_path,
                "level": node.get("level"),
                "full_title_for_matching": title,
                "toc_id": toc_id
            }
```

```
            if "page" in node: entry["page"] = node["page"]
            flat_toc_entries.append(entry)
            if node.get("children"):
                _add_ids_and_flatten_recursive(node.get("children", []),␣
↪new_titles_path, counter)

    toc_id_counter = [0]
    _add_ids_and_flatten_recursive(hierarchical_toc, [], toc_id_counter)
    logger.info(f"Flattened ToC and assigned sequential IDs to␣
↪{len(flat_toc_entries)} entries.")

    # Logic for PDF metadata assignment
    if file_extension == ".pdf" and any("page" in entry for entry in␣
↪flat_toc_entries):
        logger.info("Assigning metadata to PDF pages based on ToC page numbers..
↪.")
        flat_toc_entries.sort(key=lambda x: x.get("page", -1) if x.get("page")␣
↪is not None else -1)
        for page_doc in all_raw_book_docs:
            page_num_0_indexed = page_doc.metadata.get("page", -1)
            page_num_1_indexed = page_num_0_indexed + 1
            assigned_metadata = {"source": os.path.basename(book_path),␣
↪"page_number": page_num_1_indexed}
            best_match_toc_entry = None
            for toc_entry in flat_toc_entries:
                toc_page = toc_entry.get("page")
                if toc_page is not None and toc_page <= page_num_1_indexed:
                    if best_match_toc_entry is None or toc_page >␣
↪best_match_toc_entry.get("page", -1):
                        best_match_toc_entry = toc_entry
                elif toc_page is not None and toc_page > page_num_1_indexed:
                    break
            if best_match_toc_entry:
                for i, title_in_path in␣
↪enumerate(best_match_toc_entry["titles_path"]):
                    assigned_metadata[f"level_{i+1}_title"] = title_in_path
                assigned_metadata['toc_id'] = best_match_toc_entry.get('toc_id')
            else:
                assigned_metadata["level_1_title"] = "Uncategorized PDF Page"
            cleaned_meta = {k: clean_metadata_for_chroma(v) for k, v in␣
↪assigned_metadata.items()}
            final_documents_with_metadata.append(Document(page_content=page_doc.
↪page_content, metadata=cleaned_meta))

    # Logic for EPUB metadata assignment
    elif file_extension == ".epub":
```

```
        logger.info("Assigning metadata to EPUB elements by matching ToC titles␣
↪in text...")
        toc_titles_for_search = [entry for entry in flat_toc_entries if entry.
↪get("full_title_for_matching")]
        current_hierarchy_metadata = {}
        for element_doc in all_raw_book_docs:
            element_text = element_doc.page_content.strip() if element_doc.
↪page_content else ""
            if not element_text: continue
            for toc_entry in toc_titles_for_search:
                if element_text == toc_entry["full_title_for_matching"]:
                    current_hierarchy_metadata = {"source": os.path.
↪basename(book_path)}
                    for i, title_in_path in enumerate(toc_entry["titles_path"]):
                        current_hierarchy_metadata[f"level_{i+1}_title"] =␣
↪title_in_path
                    current_hierarchy_metadata['toc_id'] = toc_entry.
↪get('toc_id')
                    if "page" in toc_entry:␣
↪current_hierarchy_metadata["epub_toc_page"] = toc_entry["page"]
                    break
            if not current_hierarchy_metadata:
                doc_metadata_to_assign = {"source": os.path.
↪basename(book_path), "level_1_title": "EPUB Preamble", "toc_id": -1}
            else:
                doc_metadata_to_assign = current_hierarchy_metadata.copy()
            cleaned_meta = {k: clean_metadata_for_chroma(v) for k, v in␣
↪doc_metadata_to_assign.items()}
            final_documents_with_metadata.
↪append(Document(page_content=element_text, metadata=cleaned_meta))

    else: # Fallback
        final_documents_with_metadata = all_raw_book_docs

    if not final_documents_with_metadata:
        logger.error("No documents were processed or enriched with hierarchical␣
↪metadata.")
        return [], hierarchical_toc

    logger.info(f"Total documents prepared for chunking:␣
↪{len(final_documents_with_metadata)}")

    text_splitter = RecursiveCharacterTextSplitter(
        chunk_size=chunk_size,
        chunk_overlap=chunk_overlap,
        length_function=len
```

```
    )
    final_chunks = text_splitter.split_documents(final_documents_with_metadata)
    logger.info(f"Split into {len(final_chunks)} final chunks, inheriting␣
 ↪hierarchical metadata.")

    # --- MODIFICATION START: Add a unique, sequential chunk_id to each chunk␣
 ↪---
    logger.info("Assigning sequential chunk_id to all final chunks...")
    for i, chunk in enumerate(final_chunks):
        chunk.metadata['chunk_id'] = i
    logger.info(f"Assigned chunk_ids from 0 to {len(final_chunks) - 1}.")
    # --- MODIFICATION END ---

    return final_chunks, hierarchical_toc

# --- Main Execution Block for this Cell ---

if not os.path.exists(PRE_EXTRACTED_TOC_JSON_PATH):
    logger.error(f"CRITICAL: Pre-extracted ToC file not found at␣
 ↪'{PRE_EXTRACTED_TOC_JSON_PATH}'.")
    logger.error("Please run the 'Extract Book Table of Contents (ToC)' cell␣
 ↪(Cell 4) first.")
else:
    final_chunks_for_db, toc_reloaded = process_book_with_extracted_toc(
        book_path=BOOK_PATH,
        extracted_toc_json_path=PRE_EXTRACTED_TOC_JSON_PATH,
        chunk_size=CHUNK_SIZE,
        chunk_overlap=CHUNK_OVERLAP
    )

    if final_chunks_for_db:
        if os.path.exists(CHROMA_PERSIST_DIR):
            logger.warning(f"Deleting existing ChromaDB directory:␣
 ↪{CHROMA_PERSIST_DIR}")
            shutil.rmtree(CHROMA_PERSIST_DIR)

        logger.info(f"Initializing embedding model '{EMBEDDING_MODEL_OLLAMA}'␣
 ↪and creating new vector database...")
        embedding_model = OllamaEmbeddings(model=EMBEDDING_MODEL_OLLAMA)

        vector_db = Chroma.from_documents(
            documents=final_chunks_for_db,
            embedding=embedding_model,
            persist_directory=CHROMA_PERSIST_DIR,
            collection_name=CHROMA_COLLECTION_NAME
        )
```

```
        reloaded_db = Chroma(persist_directory=CHROMA_PERSIST_DIR,␣
↪embedding_function=embedding_model, collection_name=CHROMA_COLLECTION_NAME)
        count = reloaded_db._collection.count()

        print("-" * 50)
        logger.info(f" Vector DB created successfully at:␣
↪{CHROMA_PERSIST_DIR}")
        logger.info(f" Collection '{CHROMA_COLLECTION_NAME}' contains {count}␣
↪documents.")
        print("-" * 50)
    else:
        logger.error(" Failed to generate chunks. Vector DB not created.")
```

```
2025-07-01 20:57:57,274 - INFO - Processing book 'Bill Nelson, Amelia Phillips,
Christopher Steuart - Guide to Computer Forensics and Investigations_ Processing
Digital Evidence-Cengage Learning (2018).epub' using ToC from
'ICT312_epub_table_of_contents.json'.
2025-07-01 20:57:57,275 - INFO - Successfully loaded pre-extracted ToC with 28
top-level entries.
2025-07-01 20:57:59,343 - INFO - Note: NumExpr detected 32 cores but
"NUMEXPR_MAX_THREADS" not set, so enforcing safe limit of 16.
2025-07-01 20:57:59,344 - INFO - NumExpr defaulting to 16 threads.
[WARNING] Could not load translations for en-US
  data file translations/en.yaml not found
2025-07-01 20:58:06,901 - WARNING - Could not load translations for en-US
  data file translations/en.yaml not found
[WARNING] The term Abstract has no translation defined.

2025-07-01 20:58:06,905 - WARNING - The term Abstract has no translation
defined.

2025-07-01 20:58:10,045 - INFO - Loaded 11815 text elements from EPUB.
2025-07-01 20:58:10,046 - INFO - Flattened ToC and assigned sequential IDs to
877 entries.
2025-07-01 20:58:10,047 - INFO - Assigning metadata to EPUB elements by matching
ToC titles in text…
2025-07-01 20:58:10,366 - INFO - Total documents prepared for chunking: 11483
2025-07-01 20:58:10,530 - INFO - Split into 11774 final chunks, inheriting
hierarchical metadata.
2025-07-01 20:58:10,530 - INFO - Assigning sequential chunk_id to all final
chunks…
2025-07-01 20:58:10,532 - INFO - Assigned chunk_ids from 0 to 11773.
2025-07-01 20:58:10,537 - INFO - Initializing embedding model 'nomic-embed-text'
and creating new vector database…
2025-07-01 20:58:10,568 - INFO - Anonymized telemetry enabled. See
https://docs.trychroma.com/telemetry for more information.
```

```
2025-07-01 20:59:20,442 - INFO - HTTP Request: POST
http://127.0.0.1:11434/api/embed "HTTP/1.1 200 OK"
2025-07-01 21:00:32,720 - INFO - HTTP Request: POST
http://127.0.0.1:11434/api/embed "HTTP/1.1 200 OK"
2025-07-01 21:00:45,373 - INFO - HTTP Request: POST
http://127.0.0.1:11434/api/embed "HTTP/1.1 200 OK"
2025-07-01 21:00:45,884 - INFO -   Vector DB created successfully at: /home/seba
s_dev_linux/projects/course_generator/data/DataBase_Chroma/chroma_db_toc_guided_
chunks_epub
2025-07-01 21:00:45,885 - INFO -   Collection 'book_toc_guided_chunks_epub_v2'
contains 11774 documents.

--------------------------------------------------
--------------------------------------------------
```

5.1.1 Full Database Health & Hierarchy Diagnostic Report

[18]:
```python
# Cell 5.1: Full Database Health & Hierarchy Diagnostic Report (V5 - with␣
↪Content Preview)

import os
import json
import logging
import random
from typing import List, Dict, Any

# You might need to install pandas if you haven't already
try:
    import pandas as pd
    pandas_available = True
except ImportError:
    pandas_available = False

try:
    from langchain_chroma import Chroma
    from langchain_ollama.embeddings import OllamaEmbeddings
    from langchain_core.documents import Document
    langchain_available = True
except ImportError:
    langchain_available = False

# Setup Logger
logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s -␣
↪%(message)s')
logger = logging.getLogger(__name__)

# --- HELPER FUNCTIONS ---
def print_header(text: str, char: str = "="):
```

```
    """Prints a centered header to the console."""
    print("\n" + char * 80)
    print(text.center(80))
    print(char * 80)

def count_total_chunks(node: Dict) -> int:
    """Recursively counts all chunks in a node and its children."""
    total = node.get('_chunks', 0)
    for child_node in node.get('_children', {}).values():
        total += count_total_chunks(child_node)
    return total

def print_hierarchy_report(node: Dict, indent_level: int = 0):
    """
    Recursively prints the reconstructed hierarchy, sorting by sequential ToC␣
 ↪ID.
    """
    sorted_children = sorted(
        node.get('_children', {}).items(),
        key=lambda item: item[1].get('_toc_id', float('inf'))
    )

    for title, child_node in sorted_children:
        prefix = "  " * indent_level + "|-- "
        total_chunks_in_branch = count_total_chunks(child_node)
        direct_chunks = child_node.get('_chunks', 0)
        toc_id = child_node.get('_toc_id', 'N/A')
        print(f"{prefix}{title} [ID: {toc_id}] (Total:␣
 ↪{total_chunks_in_branch}, Direct: {direct_chunks})")
        print_hierarchy_report(child_node, indent_level + 1)

def find_testable_sections(node: Dict, path: str, testable_list: List):
    """
    Recursively find sections with a decent number of "direct" chunks to test␣
 ↪sequence on.
    """
    if node.get('_chunks', 0) > 10 and not node.get('_children'):
        testable_list.append({
            "path": path,
            "toc_id": node.get('_toc_id'),
            "chunk_count": node.get('_chunks')
        })

    for title, child_node in node.get('_children', {}).items():
        new_path = f"{path} -> {title}" if path else title
        find_testable_sections(child_node, new_path, testable_list)
```

```python
# --- MODIFIED TEST FUNCTION ---
def verify_chunk_sequence_and_content(vector_store: Chroma, hierarchy_tree:␣
↪Dict):
    """
    Selects a random ToC section, verifies chunk sequence, and displays the␣
↪reassembled content.
    """
    print_header("Chunk Sequence & Content Integrity Test", char="-")
    logger.info("Verifying chunk order and reassembling content for a random␣
↪ToC section.")

    # 1. Find a good section to test
    testable_sections = []
    find_testable_sections(hierarchy_tree, "", testable_sections)

    if not testable_sections:
        logger.warning("Could not find a suitable section with enough chunks to␣
↪test. Skipping content test.")
        return

    random_section = random.choice(testable_sections)
    test_toc_id = random_section['toc_id']
    section_title = random_section['path'].split(' -> ')[-1]

    logger.info(f"Selected random section for testing:␣
↪'{random_section['path']}' (toc_id: {test_toc_id})")

    # 2. Retrieve all documents (content + metadata) for that toc_id
    try:
        # Use .get() to retrieve full documents, not just similarity search
        retrieved_data = vector_store.get(
            where={"toc_id": test_toc_id},
            include=["metadatas", "documents"]
        )

        # Combine metadatas and documents into LangChain Document objects
        docs = [Document(page_content=doc, metadata=meta) for doc, meta in␣
↪zip(retrieved_data['documents'], retrieved_data['metadatas'])]

        logger.info(f"Retrieved {len(docs)} document chunks for toc_id␣
↪{test_toc_id}.")

        if len(docs) < 1:
            logger.warning("No chunks found in the selected section. Skipping.")
            return
```

```
        # 3. Sort the documents by chunk_id
        # Handle cases where chunk_id might be missing for robustness
        docs.sort(key=lambda d: d.metadata.get('chunk_id', -1))

        chunk_ids = [d.metadata.get('chunk_id') for d in docs]
        if None in chunk_ids:
            logger.error("TEST FAILED: Some retrieved chunks are missing a␣
↪'chunk_id'.")
            return

        # 4. Verify sequence
        is_sequential = all(chunk_ids[i] == chunk_ids[i-1] + 1 for i in␣
↪range(1, len(chunk_ids)))

        # 5. Reassemble and print content
        full_content = "\n".join([d.page_content for d in docs])

        print("\n" + "-"*25 + " CONTENT PREVIEW " + "-"*25)
        print(f"Title: {section_title} [toc_id: {test_toc_id}]")
        print(f"Chunk IDs: {chunk_ids}")
        print("-" * 70)
        print(full_content)
        print("-" * 23 + " END CONTENT PREVIEW " + "-"*23 + "\n")

        if is_sequential:
            logger.info("  TEST PASSED: Chunk IDs for the section are␣
↪sequential and content is reassembled.")
        else:
            logger.warning("TEST PASSED (with note): Chunk IDs are not␣
↪perfectly sequential but are in increasing order.")
            logger.warning("This is acceptable. Sorting by chunk_id␣
↪successfully restored narrative order.")

    except Exception as e:
        logger.error(f"TEST FAILED: An error occurred during chunk sequence␣
↪verification: {e}", exc_info=True)


# --- MAIN DIAGNOSTIC FUNCTION ---
def run_full_diagnostics():
    if not langchain_available:
        logger.error("LangChain components not installed. Skipping diagnostics.
↪")
        return
    if not pandas_available:
```

```
        logger.warning("Pandas not installed. Some reports may not be available.
↪")

    print_header("Full Database Health & Hierarchy Diagnostic Report")

    # 1. Connect to the Database
    logger.info("Connecting to the vector database...")
    if not os.path.exists(CHROMA_PERSIST_DIR):
        logger.error(f"FATAL: Chroma DB directory not found at␣
↪{CHROMA_PERSIST_DIR}.")
        return

    vector_store = Chroma(
        persist_directory=CHROMA_PERSIST_DIR,
        embedding_function=OllamaEmbeddings(model=EMBEDDING_MODEL_OLLAMA),
        collection_name=CHROMA_COLLECTION_NAME
    )
    logger.info("Successfully connected to the database.")

    # 2. Retrieve ALL Metadata
    total_docs = vector_store._collection.count()
    if total_docs == 0:
        logger.warning("Database is empty. No diagnostics to run.")
        return

    logger.info(f"Retrieving metadata for all {total_docs} chunks...")
    metadatas = vector_store.get(limit=total_docs,␣
↪include=["metadatas"])['metadatas']
    logger.info("Successfully retrieved all metadata.")

    # 3. Reconstruct the Hierarchy Tree
    logger.info("Reconstructing hierarchy from chunk metadata...")
    hierarchy_tree = {'_children': {}}
    chunks_without_id = 0

    for meta in metadatas:
        toc_id = meta.get('toc_id')
        if toc_id is None or toc_id == -1:
            chunks_without_id += 1
            node_title = meta.get('level_1_title', 'Orphaned Chunks')
            if node_title not in hierarchy_tree['_children']:
                hierarchy_tree['_children'][node_title] = {'_children': {},␣
↪'_chunks': 0, '_toc_id': float('inf')}
            hierarchy_tree['_children'][node_title]['_chunks'] += 1
            continue

        current_node = hierarchy_tree
```

```
        for level in range(1, 7):
            level_key = f'level_{level}_title'
            title = meta.get(level_key)
            if not title: break
            if title not in current_node['_children']:
                current_node['_children'][title] = {'_children': {}, '_chunks':␣
 ↪0, '_toc_id': float('inf')}
            current_node = current_node['_children'][title]

        current_node['_chunks'] += 1
        current_node['_toc_id'] = min(current_node['_toc_id'], toc_id)

    logger.info("Hierarchy reconstruction complete.")

    # 4. Print Hierarchy Report
    print_header("Reconstructed Hierarchy Report (Book Order)", char="-")
    print_hierarchy_report(hierarchy_tree)

    # 5. Run Chunk Sequence and Content Test
    verify_chunk_sequence_and_content(vector_store, hierarchy_tree)

    # 6. Final Summary
    print_header("Diagnostic Summary", char="-")
    print(f"Total Chunks in DB: {total_docs}")

    if chunks_without_id > 0:
        logger.warning(f"Found {chunks_without_id} chunks MISSING a valid␣
 ↪'toc_id'. Check 'Orphaned' sections.")
    else:
        logger.info("All chunks contain valid 'toc_id' metadata. Sequential␣
 ↪integrity is maintained.")

    print_header("Diagnostic Complete")

# --- Execute Diagnostics ---
if 'CHROMA_PERSIST_DIR' in locals() and langchain_available:
    run_full_diagnostics()
else:
    logger.error("Skipping diagnostics: Global variables not defined or␣
 ↪LangChain not available.")
```

```
2025-07-01 21:01:44,268 - INFO - Connecting to the vector database…
2025-07-01 21:01:44,283 - INFO - Successfully connected to the database.
2025-07-01 21:01:44,285 - INFO - Retrieving metadata for all 11774 chunks…

================================================================================
                Full Database Health & Hierarchy Diagnostic Report
```

```
================================================================================

2025-07-01 21:01:44,652 - INFO - Successfully retrieved all metadata.
2025-07-01 21:01:44,653 - INFO - Reconstructing hierarchy from chunk metadata…
2025-07-01 21:01:44,662 - INFO - Hierarchy reconstruction complete.
2025-07-01 21:01:44,664 - INFO - Verifying chunk order and reassembling content
for a random ToC section.
2025-07-01 21:01:44,665 - INFO - Selected random section for testing: 'Chapter
13. Cloud Forensics -> Conducting a Cloud Investigation -> Windows Prefetch
Artifacts' (toc_id: 456)
2025-07-01 21:01:44,668 - INFO - Retrieved 26 document chunks for toc_id 456.
2025-07-01 21:01:44,669 - INFO -  TEST PASSED: Chunk IDs for the section are
sequential and content is reassembled.
2025-07-01 21:01:44,669 - WARNING - Found 21 chunks MISSING a valid 'toc_id'.
Check 'Orphaned' sections.


--------------------------------------------------------------------------------
                  Reconstructed Hierarchy Report (Book Order)
--------------------------------------------------------------------------------
|-- Preface [ID: 3] (Total: 10, Direct: 10)
|-- Introduction [ID: 4] (Total: 73, Direct: 73)
|-- About the Authors [ID: 5] (Total: 5, Direct: 5)
|-- Acknowledgments [ID: 6] (Total: 20, Direct: 20)
|-- Chapter 1. Understanding the Digital Forensics Profession and Investigations
[ID: 7] (Total: 4566, Direct: 23)
  |-- An Overview of Digital Forensics [ID: 9] (Total: 60, Direct: 18)
    |-- Digital Forensics and Other Related Disciplines [ID: 10] (Total: 18,
Direct: 18)
    |-- A Brief History of Digital Forensics [ID: 11] (Total: 13, Direct: 13)
    |-- Understanding Case Law [ID: 12] (Total: 3, Direct: 3)
    |-- Developing Digital Forensics Resources [ID: 13] (Total: 8, Direct: 8)
  |-- Preparing for Digital Investigations [ID: 14] (Total: 84, Direct: 5)
    |-- Understanding Law Enforcement Agency Investigations [ID: 15] (Total: 10,
Direct: 10)
    |-- Following Legal Processes [ID: 16] (Total: 13, Direct: 13)
    |-- Understanding Private-Sector Investigations [ID: 17] (Total: 56, Direct:
3)
      |-- Establishing Company Policies [ID: 18] (Total: 5, Direct: 5)
      |-- Displaying Warning Banners [ID: 19] (Total: 19, Direct: 19)
      |-- Designating an Authorized Requester [ID: 20] (Total: 9, Direct: 9)
      |-- Conducting Security Investigations [ID: 21] (Total: 15, Direct: 15)
      |-- Distinguishing Personal and Company Property [ID: 22] (Total: 5,
Direct: 5)
  |-- Maintaining Professional Conduct [ID: 23] (Total: 10, Direct: 10)
  |-- Preparing a Digital Forensics Investigation [ID: 24] (Total: 97, Direct:
4)
    |-- An Overview of a Computer Crime [ID: 25] (Total: 12, Direct: 12)
    |-- An Overview of a Company Policy Violation [ID: 26] (Total: 4, Direct: 4)
```

```
    |-- Taking a Systematic Approach [ID: 27] (Total: 77, Direct: 16)
      |-- Assessing the Case [ID: 28] (Total: 11, Direct: 11)
      |-- Planning Your Investigation [ID: 29] (Total: 41, Direct: 41)
      |-- Securing Your Evidence [ID: 30] (Total: 9, Direct: 9)
  |-- Procedures for Private-Sector High-Tech Investigations [ID: 31] (Total:
124, Direct: 2)
    |-- Employee Termination Cases [ID: 32] (Total: 2, Direct: 2)
    |-- Internet Abuse Investigations [ID: 33] (Total: 19, Direct: 19)
    |-- E-mail Abuse Investigations [ID: 34] (Total: 16, Direct: 16)
    |-- Attorney-Client Privilege Investigations [ID: 35] (Total: 33, Direct:
33)
    |-- Industrial Espionage Investigations [ID: 36] (Total: 52, Direct: 41)
      |-- Interviews and Interrogations in High-Tech Investigations [ID: 37]
(Total: 11, Direct: 11)
  |-- Understanding Data Recovery Workstations and Software [ID: 38] (Total: 37,
Direct: 18)
    |-- Setting Up Your Workstation for Digital Forensics [ID: 39] (Total: 19,
Direct: 19)
  |-- Conducting an Investigation [ID: 40] (Total: 109, Direct: 8)
    |-- Gathering the Evidence [ID: 41] (Total: 14, Direct: 14)
    |-- Understanding Bit-stream Copies [ID: 42] (Total: 8, Direct: 6)
      |-- Acquiring an Image of Evidence Media [ID: 43] (Total: 2, Direct: 2)
    |-- Analyzing Your Digital Evidence [ID: 44] (Total: 48, Direct: 44)
      |-- Some Additional Features of Autopsy [ID: 45] (Total: 4, Direct: 4)
    |-- Completing the Case [ID: 46] (Total: 22, Direct: 12)
      |-- Autopsy's Report Generator [ID: 47] (Total: 10, Direct: 10)
    |-- Critiquing the Case [ID: 48] (Total: 9, Direct: 9)
  |-- Chapter Review [ID: inf] (Total: 4022, Direct: 0)
    |-- Chapter Summary [ID: 50] (Total: 211, Direct: 211)
    |-- Key Terms [ID: 51] (Total: 309, Direct: 309)
    |-- Review Questions [ID: 52] (Total: 1785, Direct: 1785)
    |-- Hands-On Projects [ID: 53] (Total: 1527, Direct: 1527)
    |-- Case Projects [ID: 54] (Total: 190, Direct: 190)
|-- Chapter 2. The Investigator's Office and Laboratory [ID: 55] (Total: 331,
Direct: 22)
  |-- Understanding Forensics Lab Accreditation Requirements [ID: 57] (Total:
86, Direct: 7)
    |-- Identifying Duties of the Lab Manager and Staff [ID: 58] (Total: 7,
Direct: 7)
    |-- Lab Budget Planning [ID: 59] (Total: 18, Direct: 18)
    |-- Acquiring Certification and Training [ID: 60] (Total: 54, Direct: 4)
      |-- International Association of Computer Investigative Specialists [ID:
61] (Total: 13, Direct: 13)
      |-- ISC2 Certified Cyber Forensics Professional [ID: 62] (Total: 2,
Direct: 2)
      |-- High Tech Crime Network [ID: 63] (Total: 19, Direct: 19)
      |-- EnCase Certified Examiner Certification [ID: 64] (Total: 2, Direct: 2)
      |-- AccessData Certified Examiner [ID: 65] (Total: 2, Direct: 2)
```

```
      |-- Preparing a Target Drive for Acquisition in Linux [ID: 113] (Total:
45, Direct: 45)
      |-- Acquiring Data with dd in Linux [ID: 114] (Total: 32, Direct: 32)
      |-- Acquiring Data with dcfldd in Linux [ID: 115] (Total: 14, Direct: 14)
    |-- Capturing an Image with AccessData FTK Imager Lite [ID: 116] (Total: 46,
Direct: 46)
  |-- Validating Data Acquisitions [ID: 117] (Total: 32, Direct: 5)
    |-- Linux Validation Methods [ID: 118] (Total: 21, Direct: 3)
      |-- Validating dd-Acquired Data [ID: 119] (Total: 12, Direct: 12)
      |-- Validating dcfldd-Acquired Data [ID: 120] (Total: 6, Direct: 6)
    |-- Windows Validation Methods [ID: 121] (Total: 6, Direct: 6)
  |-- Performing RAID Data Acquisitions [ID: 122] (Total: 30, Direct: 2)
    |-- Understanding RAID [ID: 123] (Total: 15, Direct: 15)
    |-- Acquiring RAID Disks [ID: 124] (Total: 13, Direct: 13)
  |-- Using Remote Network Acquisition Tools [ID: 125] (Total: 39, Direct: 5)
    |-- Remote Acquisition with ProDiscover [ID: 126] (Total: 20, Direct: 20)
    |-- Remote Acquisition with EnCase Enterprise [ID: 127] (Total: 7, Direct:
7)
    |-- Remote Acquisition with R-Tools R-Studio [ID: 128] (Total: 2, Direct: 2)
    |-- Remote Acquisition with WetStone US-LATT PRO [ID: 129] (Total: 2,
Direct: 2)
    |-- Remote Acquisition with F-Response [ID: 130] (Total: 3, Direct: 3)
  |-- Using Other Forensics Acquisition Tools [ID: 131] (Total: 27, Direct: 2)
    |-- PassMark Software ImageUSB [ID: 132] (Total: 2, Direct: 2)
    |-- ASR Data SMART [ID: 133] (Total: 7, Direct: 7)
    |-- Runtime Software [ID: 134] (Total: 12, Direct: 12)
    |-- ILookIX IXImager [ID: 135] (Total: 2, Direct: 2)
    |-- SourceForge [ID: 136] (Total: 2, Direct: 2)
|-- Chapter 4. Processing Crime and Incident Scenes [ID: 143] (Total: 413,
Direct: 29)
  |-- Identifying Digital Evidence [ID: 145] (Total: 76, Direct: 13)
    |-- Understanding Rules of Evidence [ID: 146] (Total: 63, Direct: 63)
  |-- Collecting Evidence in Private-Sector Incident Scenes [ID: 147] (Total:
24, Direct: 24)
  |-- Processing Law Enforcement Crime Scenes [ID: 148] (Total: 24, Direct: 6)
    |-- Understanding Concepts and Terms Used in Warrants [ID: 149] (Total: 18,
Direct: 18)
  |-- Preparing for a Search [ID: 150] (Total: 40, Direct: 2)
    |-- Identifying the Nature of the Case [ID: 151] (Total: 3, Direct: 3)
    |-- Identifying the Type of OS or Digital Device [ID: 152] (Total: 4,
Direct: 4)
    |-- Determining Whether You Can Seize Computers and Digital Devices [ID:
153] (Total: 4, Direct: 4)
    |-- Getting a Detailed Description of the Location [ID: 154] (Total: 7,
Direct: 7)
    |-- Determining Who Is in Charge [ID: 155] (Total: 2, Direct: 2)
    |-- Using Additional Technical Expertise [ID: 156] (Total: 4, Direct: 4)
    |-- Determining the Tools You Need [ID: 157] (Total: 11, Direct: 11)
```

```
    |-- Preparing the Investigation Team [ID: 158] (Total: 3, Direct: 3)
  |-- Securing a Digital Incident or Crime Scene [ID: 159] (Total: 9, Direct: 9)
  |-- Seizing Digital Evidence at the Scene [ID: 160] (Total: 72, Direct: 4)
    |-- Preparing to Acquire Digital Evidence [ID: 161] (Total: 8, Direct: 8)
    |-- Processing Incident or Crime Scenes [ID: 162] (Total: 34, Direct: 34)
    |-- Processing Data Centers with RAID Systems [ID: 163] (Total: 2, Direct:
2)
    |-- Using a Technical Advisor [ID: 164] (Total: 9, Direct: 9)
    |-- Documenting Evidence in the Lab [ID: 165] (Total: 4, Direct: 4)
    |-- Processing and Handling Digital Evidence [ID: 166] (Total: 11, Direct:
11)
  |-- Storing Digital Evidence [ID: 167] (Total: 18, Direct: 7)
    |-- Evidence Retention and Media Storage Needs [ID: 168] (Total: 4, Direct:
4)
    |-- Documenting Evidence [ID: 169] (Total: 7, Direct: 7)
  |-- Obtaining a Digital Hash [ID: 170] (Total: 42, Direct: 42)
  |-- Reviewing a Case [ID: 171] (Total: 79, Direct: 8)
    |-- Sample Civil Investigation [ID: 172] (Total: 23, Direct: 23)
    |-- An Example of a Criminal Investigation [ID: 173] (Total: 4, Direct: 4)
    |-- Reviewing Background Information for a Case [ID: 174] (Total: 4, Direct:
4)
    |-- Planning the Investigation [ID: 175] (Total: 7, Direct: 7)
    |-- Conducting the Investigation: Acquiring Evidence with OSForensics [ID:
176] (Total: 33, Direct: 33)
|-- Chapter 5. Working with Windows and CLI Systems [ID: 183] (Total: 471,
Direct: 22)
  |-- Understanding File Systems [ID: 185] (Total: 33, Direct: 3)
    |-- Understanding the Boot Sequence [ID: 186] (Total: 8, Direct: 8)
    |-- Understanding Disk Drives [ID: 187] (Total: 14, Direct: 14)
    |-- Solid-State Storage Devices [ID: 188] (Total: 8, Direct: 8)
  |-- Exploring Microsoft File Structures [ID: 189] (Total: 71, Direct: 5)
    |-- Disk Partitions [ID: 190] (Total: 39, Direct: 39)
    |-- Examining FAT Disks [ID: 191] (Total: 27, Direct: 24)
      |-- Deleting FAT Files [ID: 192] (Total: 3, Direct: 3)
  |-- Examining NTFS Disks [ID: 193] (Total: 168, Direct: 14)
    |-- NTFS System Files [ID: 194] (Total: 6, Direct: 6)
    |-- MFT and File Attributes [ID: 195] (Total: 20, Direct: 20)
    |-- MFT Structures for File Data [ID: 196] (Total: 69, Direct: 3)
      |-- MFT Header Fields [ID: 197] (Total: 7, Direct: 7)
      |-- Attribute 0x10: Standard Information [ID: 198] (Total: 8, Direct: 8)
      |-- Attribute 0x30: File Name [ID: 199] (Total: 18, Direct: 18)
      |-- Attribute 0x40: Object_ID [ID: 200] (Total: 10, Direct: 10)
      |-- Attribute 0x80: Data for a Resident File [ID: 201] (Total: 8, Direct:
8)
      |-- Attribute 0x80: Data for a Nonresident File [ID: 202] (Total: 6,
Direct: 6)
      |-- Interpreting a Data Run [ID: 203] (Total: 9, Direct: 9)
    |-- NTFS Alternate Data Streams [ID: 204] (Total: 14, Direct: 14)
```

```
    |-- NTFS Compressed Files [ID: 205] (Total: 3, Direct: 3)
    |-- NTFS Encrypting File System [ID: 206] (Total: 5, Direct: 5)
    |-- EFS Recovery Key Agent [ID: 207] (Total: 8, Direct: 8)
    |-- Deleting NTFS Files [ID: 208] (Total: 20, Direct: 20)
    |-- Resilient File System [ID: 209] (Total: 9, Direct: 9)
  |-- Understanding Whole Disk Encryption [ID: 210] (Total: 26, Direct: 11)
    |-- Examining Microsoft BitLocker [ID: 211] (Total: 9, Direct: 9)
    |-- Examining Third-Party Disk Encryption Tools [ID: 212] (Total: 6, Direct:
6)
  |-- Understanding the Windows Registry [ID: 213] (Total: 56, Direct: 9)
    |-- Exploring the Organization of the Windows Registry [ID: 214] (Total: 18,
Direct: 18)
    |-- Examining the Windows Registry [ID: 215] (Total: 29, Direct: 29)
  |-- Understanding Microsoft Startup Tasks [ID: 216] (Total: 47, Direct: 3)
    |-- Startup in Windows 7, Windows 8, and Windows 10 [ID: 217] (Total: 5,
Direct: 5)
    |-- Startup in Windows NT and Later [ID: 218] (Total: 39, Direct: 10)
      |-- Startup Files for Windows Vista [ID: 219] (Total: 6, Direct: 6)
      |-- Startup Files for Windows XP [ID: 220] (Total: 17, Direct: 17)
      |-- Windows XP System Files [ID: 221] (Total: 4, Direct: 4)
      |-- Contamination Concerns with Windows XP [ID: 222] (Total: 2, Direct: 2)
  |-- Understanding Virtual Machines [ID: 223] (Total: 48, Direct: 10)
    |-- Creating a Virtual Machine [ID: 224] (Total: 38, Direct: 38)
|-- Chapter 6. Current Digital Forensics Tools [ID: 231] (Total: 315, Direct:
22)
  |-- Evaluating Digital Forensics Tool Needs [ID: 233] (Total: 184, Direct: 10)
    |-- Types of Digital Forensics Tools [ID: 234] (Total: 11, Direct: 4)
      |-- Hardware Forensics Tools [ID: 235] (Total: 2, Direct: 2)
      |-- Software Forensics Tools [ID: 236] (Total: 5, Direct: 5)
    |-- Tasks Performed by Digital Forensics Tools [ID: 237] (Total: 153,
Direct: 3)
      |-- Acquisition [ID: 238] (Total: 22, Direct: 22)
      |-- Validation and Verification [ID: 239] (Total: 14, Direct: 14)
      |-- Extraction [ID: 240] (Total: 25, Direct: 25)
      |-- Reconstruction [ID: 241] (Total: 66, Direct: 66)
      |-- Reporting [ID: 242] (Total: 23, Direct: 23)
    |-- Tool Comparisons [ID: 243] (Total: 6, Direct: 6)
    |-- Other Considerations for Tools [ID: 244] (Total: 4, Direct: 4)
  |-- Digital Forensics Software Tools [ID: 245] (Total: 41, Direct: 4)
    |-- Command-Line Forensics Tools [ID: 246] (Total: 14, Direct: 14)
    |-- Linux Forensics Tools [ID: 247] (Total: 19, Direct: 4)
      |-- Smart [ID: 248] (Total: 4, Direct: 4)
      |-- Helix 3 [ID: 249] (Total: 3, Direct: 3)
      |-- Kali Linux [ID: 250] (Total: 2, Direct: 2)
      |-- Autopsy and Sleuth Kit [ID: 251] (Total: 4, Direct: 4)
      |-- Forcepoint Threat Protection [ID: 252] (Total: 2, Direct: 2)
    |-- Other GUI Forensics Tools [ID: 253] (Total: 4, Direct: 4)
  |-- Digital Forensics Hardware Tools [ID: 254] (Total: 38, Direct: 3)
```

```
    |-- Forensic Workstations [ID: 255] (Total: 13, Direct: 7)
      |-- Building Your Own Workstation [ID: 256] (Total: 6, Direct: 6)
    |-- Using a Write-Blocker [ID: 257] (Total: 17, Direct: 17)
    |-- Recommendations for a Forensic Workstation [ID: 258] (Total: 5, Direct:
5)
  |-- Validating and Testing Forensics Software [ID: 259] (Total: 30, Direct: 2)
    |-- Using National Institute of Standards and Technology Tools [ID: 260]
(Total: 13, Direct: 13)
    |-- Using Validation Protocols [ID: 261] (Total: 15, Direct: 6)
      |-- Digital Forensics Examination Protocol [ID: 262] (Total: 5, Direct: 5)
      |-- Digital Forensics Tool Upgrade Protocol [ID: 263] (Total: 4, Direct:
4)
|-- Chapter 7. Linux and Macintosh File Systems [ID: 270] (Total: 274, Direct:
17)
  |-- Examining Linux File Structures [ID: 272] (Total: 131, Direct: 77)
    |-- File Structures in Ext4 [ID: 273] (Total: 54, Direct: 8)
      |-- Inodes [ID: 274] (Total: 22, Direct: 22)
      |-- Hard Links and Symbolic Links [ID: 275] (Total: 24, Direct: 24)
  |-- Understanding Macintosh File Structures [ID: 276] (Total: 58, Direct: 6)
    |-- An Overview of Mac File Structures [ID: 277] (Total: 23, Direct: 23)
    |-- Forensics Procedures in Mac [ID: 278] (Total: 29, Direct: 18)
      |-- Acquisition Methods in macOS [ID: 279] (Total: 11, Direct: 11)
  |-- Using Linux Forensics Tools [ID: 280] (Total: 68, Direct: 5)
    |-- Installing Sleuth Kit and Autopsy [ID: 281] (Total: 21, Direct: 21)
    |-- Examining a Case with Sleuth Kit and Autopsy [ID: 282] (Total: 42,
Direct: 42)
|-- Chapter 8. Recovering Graphics Files [ID: 289] (Total: 240, Direct: 19)
  |-- Recognizing a Graphics File [ID: 291] (Total: 54, Direct: 4)
    |-- Understanding Bitmap and Raster Images [ID: 292] (Total: 13, Direct: 13)
    |-- Understanding Vector Graphics [ID: 293] (Total: 2, Direct: 2)
    |-- Understanding Metafile Graphics [ID: 294] (Total: 2, Direct: 2)
    |-- Understanding Graphics File Formats [ID: 295] (Total: 14, Direct: 14)
    |-- Understanding Digital Photograph File Formats [ID: 296] (Total: 19,
Direct: 2)
      |-- Examining the Raw File Format [ID: 297] (Total: 5, Direct: 5)
      |-- Examining the Exchangeable Image File Format [ID: 298] (Total: 12,
Direct: 12)
  |-- Understanding Data Compression [ID: 299] (Total: 101, Direct: 2)
    |-- Lossless and Lossy Compression [ID: 300] (Total: 8, Direct: 8)
    |-- Locating and Recovering Graphics Files [ID: 301] (Total: 6, Direct: 6)
    |-- Identifying Graphics File Fragments [ID: 302] (Total: 3, Direct: 3)
    |-- Repairing Damaged Headers [ID: 303] (Total: 6, Direct: 6)
    |-- Searching for and Carving Data from Unallocated Space [ID: 304] (Total:
39, Direct: 9)
      |-- Planning Your Examination [ID: 305] (Total: 4, Direct: 4)
      |-- Searching for and Recovering Digital Photograph Evidence [ID: 306]
(Total: 26, Direct: 26)
    |-- Rebuilding File Headers [ID: 307] (Total: 22, Direct: 22)
```

```
    |-- Reconstructing File Fragments [ID: 308] (Total: 15, Direct: 15)
  |-- Identifying Unknown File Formats [ID: 309] (Total: 47, Direct: 14)
    |-- Analyzing Graphics File Headers [ID: 310] (Total: 5, Direct: 5)
    |-- Tools for Viewing Images [ID: 311] (Total: 5, Direct: 5)
    |-- Understanding Steganography in Graphics Files [ID: 312] (Total: 16,
Direct: 16)
    |-- Using Steganalysis Tools [ID: 313] (Total: 7, Direct: 7)
  |-- Understanding Copyright Issues with Graphics [ID: 314] (Total: 19, Direct:
19)
|-- Chapter 9. Digital Forensics Analysis and Validation [ID: 321] (Total: 248,
Direct: 16)
  |-- Determining What Data to Collect and Analyze [ID: 323] (Total: 99, Direct:
6)
    |-- Approaching Digital Forensics Cases [ID: 324] (Total: 35, Direct: 30)
      |-- Refining and Modifying the Investigation Plan [ID: 325] (Total: 5,
Direct: 5)
    |-- Using Autopsy to Validate Data [ID: 326] (Total: 23, Direct: 8)
      |-- Installing NSRL Hashes in Autopsy [ID: 327] (Total: 15, Direct: 15)
    |-- Collecting Hash Values in Autopsy [ID: 328] (Total: 35, Direct: 35)
  |-- Validating Forensic Data [ID: 329] (Total: 51, Direct: 3)
    |-- Validating with Hexadecimal Editors [ID: 330] (Total: 31, Direct: 28)
      |-- Using Hash Values to Discriminate Data [ID: 331] (Total: 3, Direct: 3)
    |-- Validating with Digital Forensics Tools [ID: 332] (Total: 17, Direct:
17)
  |-- Addressing Data-Hiding Techniques [ID: 333] (Total: 82, Direct: 2)
    |-- Hiding Files by Using the OS [ID: 334] (Total: 3, Direct: 3)
    |-- Hiding Partitions [ID: 335] (Total: 8, Direct: 8)
    |-- Marking Bad Clusters [ID: 336] (Total: 6, Direct: 6)
    |-- Bit-Shifting [ID: 337] (Total: 34, Direct: 34)
    |-- Understanding Steganalysis Methods [ID: 338] (Total: 10, Direct: 10)
    |-- Examining Encrypted Files [ID: 339] (Total: 4, Direct: 4)
    |-- Recovering Passwords [ID: 340] (Total: 15, Direct: 15)
|-- Chapter 10. Virtual Machine Forensics, Live Acquisitions, and Network
Forensics [ID: 347] (Total: 287, Direct: 17)
  |-- An Overview of Virtual Machine Forensics [ID: 349] (Total: 176, Direct: 7)
    |-- Type 2 Hypervisors [ID: 350] (Total: 32, Direct: 6)
      |-- Parallels Desktop [ID: 351] (Total: 2, Direct: 2)
      |-- KVM [ID: 352] (Total: 2, Direct: 2)
      |-- Microsoft Hyper-V [ID: 353] (Total: 2, Direct: 2)
      |-- VMware Workstation and Workstation Player [ID: 354] (Total: 14,
Direct: 14)
      |-- VirtualBox [ID: 355] (Total: 6, Direct: 6)
    |-- Conducting an Investigation with Type 2 Hypervisors [ID: 356] (Total:
103, Direct: 70)
      |-- Other VM Examination Methods [ID: 357] (Total: 13, Direct: 13)
      |-- Using VMs as Forensics Tools [ID: 358] (Total: 20, Direct: 20)
    |-- Working with Type 1 Hypervisors [ID: 359] (Total: 34, Direct: 34)
  |-- Performing Live Acquisitions [ID: 360] (Total: 18, Direct: 15)
```

```
    |-- Performing a Live Acquisition in Windows [ID: 361] (Total: 3, Direct: 3)
  |-- Network Forensics Overview [ID: 362] (Total: 76, Direct: 4)
    |-- The Need for Established Procedures [ID: 363] (Total: 4, Direct: 4)
    |-- Securing a Network [ID: 364] (Total: 13, Direct: 13)
    |-- Developing Procedures for Network Forensics [ID: 365] (Total: 41,
Direct: 8)
      |-- Reviewing Network Logs [ID: 366] (Total: 11, Direct: 11)
      |-- Using Network Tools [ID: 367] (Total: 4, Direct: 4)
      |-- Using Packet Analyzers [ID: 368] (Total: 18, Direct: 18)
    |-- Investigating Virtual Networks [ID: 369] (Total: 7, Direct: 7)
    |-- Examining the Honeynet Project [ID: 370] (Total: 7, Direct: 7)
|-- Chapter 11. E-mail and Social Media Investigations [ID: 377] (Total: 302,
Direct: 20)
  |-- Exploring the Role of E-mail in Investigations [ID: 379] (Total: 11,
Direct: 11)
  |-- Exploring the Roles of the Client and Server in E-mail [ID: 380] (Total:
10, Direct: 10)
  |-- Investigating E-mail Crimes and Violations [ID: 381] (Total: 101, Direct:
4)
    |-- Understanding Forensic Linguistics [ID: 382] (Total: 6, Direct: 6)
    |-- Examining E-mail Messages [ID: 383] (Total: 28, Direct: 8)
      |-- Copying an E-mail Message [ID: 384] (Total: 20, Direct: 20)
    |-- Viewing E-mail Headers [ID: 385] (Total: 33, Direct: 33)
    |-- Examining E-mail Headers [ID: 386] (Total: 14, Direct: 14)
    |-- Examining Additional E-mail Files [ID: 387] (Total: 6, Direct: 6)
    |-- Tracing an E-mail Message [ID: 388] (Total: 7, Direct: 7)
    |-- Using Network E-mail Logs [ID: 389] (Total: 3, Direct: 3)
  |-- Understanding E-mail Servers [ID: 390] (Total: 33, Direct: 13)
    |-- Examining UNIX E-mail Server Logs [ID: 391] (Total: 12, Direct: 12)
    |-- Examining Microsoft E-mail Server Logs [ID: 392] (Total: 8, Direct: 8)
  |-- Using Specialized E-mail Forensics Tools [ID: 393] (Total: 91, Direct: 22)
    |-- Using Magnet AXIOM to Recover E-mail [ID: 394] (Total: 14, Direct: 14)
    |-- Using a Hex Editor to Carve E-mail Messages [ID: 395] (Total: 41,
Direct: 41)
    |-- Recovering Outlook Files [ID: 396] (Total: 7, Direct: 7)
    |-- E-mail Case Studies [ID: 397] (Total: 7, Direct: 7)
  |-- Applying Digital Forensics Methods to Social Media Communications [ID:
398] (Total: 36, Direct: 23)
    |-- Forensics Tools for Social Media Investigations [ID: 399] (Total: 13,
Direct: 13)
|-- Chapter 12. Mobile Device Forensics and the Internet of Anything [ID: 406]
(Total: 169, Direct: 8)
  |-- Understanding Mobile Device Forensics [ID: 408] (Total: 65, Direct: 19)
    |-- Mobile Phone Basics [ID: 409] (Total: 24, Direct: 24)
    |-- Inside Mobile Devices [ID: 410] (Total: 22, Direct: 11)
      |-- SIM Cards [ID: 411] (Total: 11, Direct: 11)
  |-- Understanding Acquisition Procedures for Mobile Devices [ID: 412] (Total:
75, Direct: 30)
```

```
    |-- Mobile Forensics Equipment [ID: 413] (Total: 30, Direct: 6)
      |-- SIM Card Readers [ID: 414] (Total: 7, Direct: 7)
      |-- Mobile Phone Forensics Tools and Methods [ID: 415] (Total: 17, Direct:
17)
    |-- Using Mobile Forensics Tools [ID: 416] (Total: 15, Direct: 15)
  |-- Understanding Forensics in the Internet of Anything [ID: 417] (Total: 21,
Direct: 21)
|-- Chapter 13. Cloud Forensics [ID: 424] (Total: 290, Direct: 20)
  |-- An Overview of Cloud Computing [ID: 426] (Total: 42, Direct: 2)
    |-- History of the Cloud [ID: 427] (Total: 4, Direct: 4)
    |-- Cloud Service Levels and Deployment Methods [ID: 428] (Total: 13,
Direct: 13)
    |-- Cloud Vendors [ID: 429] (Total: 15, Direct: 15)
    |-- Basic Concepts of Cloud Forensics [ID: 430] (Total: 8, Direct: 8)
  |-- Legal Challenges in Cloud Forensics [ID: 431] (Total: 56, Direct: 2)
    |-- Service Level Agreements [ID: 432] (Total: 25, Direct: 17)
      |-- Policies, Standards, and Guidelines for CSPs [ID: 433] (Total: 5,
Direct: 5)
      |-- CSP Processes and Procedures [ID: 434] (Total: 3, Direct: 3)
    |-- Jurisdiction Issues [ID: 435] (Total: 6, Direct: 6)
    |-- Accessing Evidence in the Cloud [ID: 436] (Total: 23, Direct: 3)
      |-- Search Warrants [ID: 437] (Total: 10, Direct: 10)
      |-- Subpoenas and Court Orders [ID: 438] (Total: 10, Direct: 10)
  |-- Technical Challenges in Cloud Forensics [ID: 439] (Total: 33, Direct: 5)
    |-- Architecture [ID: 440] (Total: 12, Direct: 12)
    |-- Analysis of Cloud Forensic Data [ID: 441] (Total: 2, Direct: 2)
    |-- Anti-Forensics [ID: 442] (Total: 3, Direct: 3)
    |-- Incident First Responders [ID: 443] (Total: 5, Direct: 5)
    |-- Role Management [ID: 444] (Total: 2, Direct: 2)
    |-- Standards and Training [ID: 445] (Total: 4, Direct: 4)
  |-- Acquisitions in the Cloud [ID: 446] (Total: 21, Direct: 8)
    |-- Encryption in the Cloud [ID: 447] (Total: 13, Direct: 13)
  |-- Conducting a Cloud Investigation [ID: 448] (Total: 105, Direct: 2)
    |-- Investigating CSPs [ID: 449] (Total: 8, Direct: 8)
    |-- Investigating Cloud Customers [ID: 450] (Total: 3, Direct: 3)
    |-- Understanding Prefetch Files [ID: 451] (Total: 3, Direct: 3)
    |-- Examining Stored Cloud Data on a PC [ID: 452] (Total: 63, Direct: 5)
      |-- Dropbox [ID: 453] (Total: 8, Direct: 8)
      |-- Google Drive [ID: 454] (Total: 21, Direct: 21)
      |-- OneDrive [ID: 455] (Total: 29, Direct: 29)
    |-- Windows Prefetch Artifacts [ID: 456] (Total: 26, Direct: 26)
  |-- Tools for Cloud Forensics [ID: 457] (Total: 13, Direct: 5)
    |-- Forensic Open-Stack Tools [ID: 458] (Total: 4, Direct: 4)
    |-- F-Response for the Cloud [ID: 459] (Total: 2, Direct: 2)
    |-- Magnet AXIOM Cloud [ID: 460] (Total: 2, Direct: 2)
|-- Chapter 14. Report Writing for High-Tech Investigations [ID: 467] (Total:
263, Direct: 16)
  |-- Understanding the Importance of Reports [ID: 469] (Total: 31, Direct: 19)
```

```
    |-- Limiting a Report to Specifics [ID: 470] (Total: 3, Direct: 3)
    |-- Types of Reports [ID: 471] (Total: 9, Direct: 9)
  |-- Guidelines for Writing Reports [ID: 472] (Total: 138, Direct: 19)
    |-- What to Include in Written Preliminary Reports [ID: 473] (Total: 9,
Direct: 9)
    |-- Report Structure [ID: 474] (Total: 8, Direct: 8)
    |-- Writing Reports Clearly [ID: 475] (Total: 17, Direct: 8)
      |-- Considering Writing Style [ID: 476] (Total: 6, Direct: 6)
      |-- Including Signposts [ID: 477] (Total: 3, Direct: 3)
    |-- Designing the Layout and Presentation of Reports [ID: 478] (Total: 85,
Direct: 44)
      |-- Providing Supporting Material [ID: 479] (Total: 3, Direct: 3)
      |-- Formatting Consistently [ID: 480] (Total: 2, Direct: 2)
      |-- Explaining Examination and Data Collection Methods [ID: 481] (Total:
3, Direct: 3)
      |-- Including Calculations [ID: 482] (Total: 2, Direct: 2)
      |-- Providing for Uncertainty and Error Analysis [ID: 483] (Total: 2,
Direct: 2)
      |-- Explaining Results and Conclusions [ID: 484] (Total: 3, Direct: 3)
      |-- Providing References [ID: 485] (Total: 20, Direct: 20)
      |-- Including Appendixes [ID: 486] (Total: 6, Direct: 6)
  |-- Generating Report Findings with Forensics Software Tools [ID: 487] (Total:
78, Direct: 2)
    |-- Using Autopsy to Generate Reports [ID: 488] (Total: 76, Direct: 76)
|-- Chapter 15. Expert Testimony in Digital Investigations [ID: 495] (Total:
329, Direct: 14)
  |-- Preparing for Testimony [ID: 497] (Total: 62, Direct: 15)
    |-- Documenting and Preparing Evidence [ID: 498] (Total: 12, Direct: 12)
    |-- Reviewing Your Role as a Consulting Expert or an Expert Witness [ID:
499] (Total: 9, Direct: 9)
    |-- Creating and Maintaining Your CV [ID: 500] (Total: 8, Direct: 8)
    |-- Preparing Technical Definitions [ID: 501] (Total: 12, Direct: 12)
    |-- Preparing to Deal with the News Media [ID: 502] (Total: 6, Direct: 6)
  |-- Testifying in Court [ID: 503] (Total: 155, Direct: 2)
    |-- Understanding the Trial Process [ID: 504] (Total: 10, Direct: 10)
    |-- Providing Qualifications for Your Testimony [ID: 505] (Total: 59,
Direct: 59)
    |-- General Guidelines on Testifying [ID: 506] (Total: 47, Direct: 27)
      |-- Using Graphics During Testimony [ID: 507] (Total: 10, Direct: 10)
      |-- Avoiding Testimony Problems [ID: 508] (Total: 7, Direct: 7)
      |-- Understanding Prosecutorial Misconduct [ID: 509] (Total: 3, Direct: 3)
    |-- Testifying During Direct Examination [ID: 510] (Total: 12, Direct: 12)
    |-- Testifying During Cross-Examination [ID: 511] (Total: 25, Direct: 25)
  |-- Preparing for a Deposition or Hearing [ID: 512] (Total: 33, Direct: 6)
    |-- Guidelines for Testifying at Depositions [ID: 513] (Total: 19, Direct:
10)
      |-- Recognizing Deposition Problems [ID: 514] (Total: 9, Direct: 9)
    |-- Guidelines for Testifying at Hearings [ID: 515] (Total: 8, Direct: 8)
```

```
  |-- Preparing Forensics Evidence for Testimony [ID: 516] (Total: 65, Direct:
34)
    |-- Preparing a Defense of Your Evidence-Collection Methods [ID: 517]
(Total: 31, Direct: 31)
|-- Chapter 16. Ethics for the Expert Witness [ID: 524] (Total: 310, Direct: 13)
  |-- Applying Ethics and Codes to Expert Witnesses [ID: 526] (Total: 58,
Direct: 11)
    |-- Forensics Examiners' Roles in Testifying [ID: 527] (Total: 5, Direct: 5)
    |-- Considerations in Disqualification [ID: 528] (Total: 22, Direct: 22)
    |-- Traps for Unwary Experts [ID: 529] (Total: 16, Direct: 16)
    |-- Determining Admissibility of Evidence [ID: 530] (Total: 4, Direct: 4)
  |-- Organizations with Codes of Ethics [ID: 531] (Total: 26, Direct: 2)
    |-- International Society of Forensic Computer Examiners [ID: 532] (Total:
11, Direct: 11)
    |-- International High Technology Crime Investigation Association [ID: 533]
(Total: 6, Direct: 6)
    |-- American Bar Association [ID: 535] (Total: 5, Direct: 5)
    |-- American Psychological Association [ID: 536] (Total: 2, Direct: 2)
  |-- Ethical Difficulties in Expert Testimony [ID: 537] (Total: 19, Direct: 6)
    |-- Ethical Responsibilities Owed to You [ID: 538] (Total: 9, Direct: 9)
    |-- Standard Forensics Tools and Tools You Create [ID: 539] (Total: 4,
Direct: 4)
  |-- An Ethics Exercise [ID: 540] (Total: 194, Direct: 4)
    |-- Performing a Cursory Exam of a Forensic Image [ID: 541] (Total: 18,
Direct: 18)
    |-- Performing a Detailed Exam of a Forensic Image [ID: 542] (Total: 33,
Direct: 33)
    |-- Performing the Exam [ID: 543] (Total: 76, Direct: 2)
      |-- Preparing for an Examination [ID: 544] (Total: 74, Direct: 74)
    |-- Interpreting Attribute 0x80 Data Runs [ID: 545] (Total: 44, Direct: 2)
      |-- Finding Attribute 0x80 an MFT Record [ID: 546] (Total: 21, Direct: 21)
      |-- Configuring Data Interpreter Options in WinHex [ID: 547] (Total: 6,
Direct: 6)
      |-- Calculating Data Runs [ID: 548] (Total: 15, Direct: 15)
    |-- Carving Data Run Clusters Manually [ID: 549] (Total: 19, Direct: 19)
|-- Lab Manual for Guide to Computer Forensics and Investigations [ID: 556]
(Total: 2165, Direct: 1)
  |-- Chapter 12. Mobile Device Forensics [ID: 792] (Total: 13, Direct: 10)
    |-- Lab 12.1. Examining Cell Phone Storage Devices [ID: 794] (Total: 1,
Direct: 1)
    |-- Lab 12.2. Using FTK Imager Lite to View Text Messages, Phone Numbers,
and Photos [ID: 799] (Total: 1, Direct: 1)
    |-- Lab 12.3. Using Autopsy to Search Cloud Backups of Mobile Devices [ID:
804] (Total: 1, Direct: 1)
  |-- Chapter 1. Understanding the Digital Forensics Profession and
Investigations [ID: inf] (Total: 1673, Direct: 0)
    |-- Lab 1.1. Installing Autopsy for Windows [ID: 560] (Total: 1670, Direct:
1)
```

```
      |-- Objectives [ID: 561] (Total: 702, Direct: 345)
        |-- Materials Required [ID: 562] (Total: 357, Direct: 357)
      |-- Activity [ID: 563] (Total: 967, Direct: 967)
    |-- Lab 1.2. Downloading FTK Imager Lite [ID: 565] (Total: 1, Direct: 1)
    |-- Lab 1.3. Downloading WinHex [ID: 570] (Total: 1, Direct: 1)
    |-- Lab 1.4. Using Autopsy for Windows [ID: 575] (Total: 1, Direct: 1)
  |-- Chapter 2. The Investigator's Office and Laboratory [ID: inf] (Total: 5,
Direct: 0)
    |-- Lab 2.1. Wiping a USB Drive Securely [ID: 582] (Total: 1, Direct: 1)
    |-- Lab 2.2. Using Directory Snoop to Image a USB Drive [ID: 587] (Total: 1,
Direct: 1)
    |-- Lab 2.3. Converting a Raw Image to an .E01 Image [ID: 592] (Total: 1,
Direct: 1)
    |-- Lab 2.4. Imaging Evidence with FTK Imager Lite [ID: 597] (Total: 1,
Direct: 1)
    |-- Lab 2.5. Viewing Images in FTK Imager Lite [ID: 602] (Total: 1, Direct:
1)
  |-- Chapter 3. Data Acquisition [ID: inf] (Total: 70, Direct: 0)
    |-- Lab 3.1. Creating a DEFT Zero Forensic Boot CD and USB Drive [ID: 609]
(Total: 67, Direct: 1)
      |-- Activity [ID: inf] (Total: 66, Direct: 0)
        |-- Creating a DEFT Zero Boot CD [ID: 613] (Total: 14, Direct: 14)
        |-- Creating a Bootable USB DEFT Zero Drive [ID: 614] (Total: 14,
Direct: 14)
        |-- Learning DEFT Zero Features [ID: 615] (Total: 38, Direct: 38)
    |-- Lab 3.2. Examining a FAT Image [ID: 617] (Total: 1, Direct: 1)
    |-- Lab 3.3. Examining an NTFS Image [ID: 622] (Total: 1, Direct: 1)
    |-- Lab 3.4. Examining an HFS+ Image [ID: 627] (Total: 1, Direct: 1)
  |-- Chapter 4. Processing Crime and Incident Scenes [ID: inf] (Total: 36,
Direct: 0)
    |-- Lab 4.1. Creating a Mini-WinFE Boot CD [ID: 634] (Total: 33, Direct: 1)
      |-- Activity [ID: inf] (Total: 32, Direct: 0)
        |-- Setting Up Mini-WinFE [ID: 638] (Total: 8, Direct: 8)
        |-- Creating a Mini-WinFE ISO Image [ID: 639] (Total: 24, Direct: 24)
    |-- Lab 4.2. Using Mini-WinFE to Boot and Image a Windows Computer [ID: 641]
(Total: 1, Direct: 1)
    |-- Lab 4.3. Testing the Mini-WinFE Write-Protection Feature [ID: 646]
(Total: 1, Direct: 1)
    |-- Lab 4.4. Creating an Image with Guymager [ID: 651] (Total: 1, Direct: 1)
  |-- Chapter 5. Working with Windows and CLI Systems [ID: inf] (Total: 4,
Direct: 0)
    |-- Lab 5.1. Using DART to Export Windows Registry Files [ID: 658] (Total:
1, Direct: 1)
    |-- Lab 5.2. Examining the SAM Hive [ID: 663] (Total: 1, Direct: 1)
    |-- Lab 5.3. Examining the SYSTEM Hive [ID: 668] (Total: 1, Direct: 1)
    |-- Lab 5.4. Examining the ntuser.dat Registry File [ID: 673] (Total: 1,
Direct: 1)
  |-- Chapter 6. Current Digital Forensics Tools [ID: inf] (Total: 79, Direct:
```

```
0)
    |-- Lab 6.1. Using Autopsy 4.7.0 to Search an Image File [ID: 680] (Total:
41, Direct: 1)
      |-- Activity [ID: inf] (Total: 40, Direct: 0)
        |-- Installing Autopsy 4.7.0 [ID: 684] (Total: 12, Direct: 12)
        |-- Searching E-mail in Autopsy 4.7.0 [ID: 685] (Total: 28, Direct: 28)
    |-- Lab 6.2. Using OSForensics to Search an Image of a Hard Drive [ID: 687]
(Total: 1, Direct: 1)
    |-- Lab 6.3. Examining a Corrupt Image File with FTK Imager Lite, Autopsy,
and WinHex [ID: 692] (Total: 37, Direct: 1)
      |-- Activity [ID: inf] (Total: 36, Direct: 0)
        |-- Testing an Image File in Autopsy 4.3.0 [ID: 696] (Total: 16, Direct:
16)
        |-- Examining Image Files in WinHex [ID: 697] (Total: 20, Direct: 20)
  |-- Chapter 7. Linux and Macintosh File Systems [ID: inf] (Total: 3, Direct:
0)
    |-- Lab 7.1. Using Autopsy to Process a Mac OS X Image [ID: 701] (Total: 1,
Direct: 1)
    |-- Lab 7.2. Using Autopsy to Process a Mac OS 9 Image [ID: 706] (Total: 1,
Direct: 1)
    |-- Lab 7.3. Using Autopsy to Process a Linux Image [ID: 711] (Total: 1,
Direct: 1)
  |-- Chapter 8. Recovering Graphics Files [ID: inf] (Total: 3, Direct: 0)
    |-- Lab 8.1. Using Autopsy to Analyze Multimedia Files [ID: 718] (Total: 1,
Direct: 1)
    |-- Lab 8.2. Using OSForensics to Analyze Multimedia Files [ID: 723] (Total:
1, Direct: 1)
    |-- Lab 8.3. Using WinHex to Analyze Multimedia Files [ID: 728] (Total: 1,
Direct: 1)
  |-- Chapter 9. Digital Forensics Analysis and Validation [ID: inf] (Total: 9,
Direct: 0)
    |-- Lab 9.1. Using Autopsy to Search for Keywords in an Image [ID: 735]
(Total: 1, Direct: 1)
    |-- Lab 9.2. Validating File Hash Values with FTK Imager Lite [ID: 740]
(Total: 1, Direct: 1)
    |-- Lab 9.3. Validating File Hash Values with WinHex [ID: 745] (Total: 7,
Direct: 1)
      |-- Objectives [ID: inf] (Total: 6, Direct: 0)
        |-- Materials Required: [ID: 747] (Total: 6, Direct: 6)
  |-- Chapter 10. Virtual Machine Forensics, Live Acquisitions, and Network
Forensics [ID: inf] (Total: 143, Direct: 0)
    |-- Lab 10.1. Analyzing a Forensic Image Hosting a Virtual Machine [ID: 752]
(Total: 41, Direct: 1)
      |-- Activity [ID: inf] (Total: 40, Direct: 0)
        |-- Installing MD5 Hashes in Autopsy [ID: 756] (Total: 8, Direct: 8)
        |-- Analyzing a Windows Image Containing a Virtual Machine [ID: 757]
(Total: 32, Direct: 32)
    |-- Lab 10.2. Conducting a Live Acquisition [ID: 759] (Total: 45, Direct: 1)
```

```
      |-- Activity [ID: inf] (Total: 44, Direct: 0)
        |-- Installing Tools for Live Acquisitions [ID: 763] (Total: 16, Direct:
16)
        |-- Exploring Tools for Live Acquisitions [ID: 764] (Total: 14, Direct:
14)
        |-- Capturing Data in a Live Acquisition [ID: 765] (Total: 14, Direct:
14)
    |-- Lab 10.3. Using Kali Linux for Network Forensics [ID: 767] (Total: 57,
Direct: 1)
      |-- Activity [ID: inf] (Total: 56, Direct: 0)
        |-- Installing Kali Linux [ID: 771] (Total: 26, Direct: 26)
        |-- Mounting Drives in Kali Linux [ID: 772] (Total: 12, Direct: 12)
        |-- Identifying Open Ports and Making a Screen Capture [ID: 773] (Total:
18, Direct: 18)
  |-- Chapter 11. E-mail and Social Media Investigations [ID: inf] (Total: 3,
Direct: 0)
    |-- Lab 11.1. Using OSForensics to Search for E-mails and Mailboxes [ID:
777] (Total: 1, Direct: 1)
    |-- Lab 11.2. Using Autopsy to Search for E-mails and Mailboxes [ID: 782]
(Total: 1, Direct: 1)
    |-- Lab 11.3. Finding Google Searches and Multiple E-mail Accounts [ID: 787]
(Total: 1, Direct: 1)
  |-- Chapter 13. Cloud Forensics [ID: inf] (Total: 3, Direct: 0)
    |-- Lab 13.1. Examining Dropbox Cloud Storage [ID: 811] (Total: 1, Direct:
1)
    |-- Lab 13.2. Examining Google Drive Cloud Storage [ID: 816] (Total: 1,
Direct: 1)
    |-- Lab 13.3. Examining OneDrive Cloud Storage [ID: 821] (Total: 1, Direct:
1)
  |-- Chapter 14. Report Writing for High-Tech Investigations [ID: inf] (Total:
3, Direct: 0)
    |-- Lab 14.1. Investigating Corporate Espionage [ID: 828] (Total: 1, Direct:
1)
    |-- Lab 14.2. Adding Evidence to a Case [ID: 833] (Total: 1, Direct: 1)
    |-- Lab 14.3. Preparing a Report [ID: 838] (Total: 1, Direct: 1)
  |-- Chapter 15. Expert Testimony in Digital Investigations [ID: inf] (Total:
44, Direct: 0)
    |-- Lab 15.1. Conducting a Preliminary Investigation [ID: 845] (Total: 1,
Direct: 1)
    |-- Lab 15.2. Investigating an Arsonist [ID: 850] (Total: 1, Direct: 1)
    |-- Lab 15.3. Recovering a Password from Password-Protected Files [ID: 855]
(Total: 42, Direct: 1)
      |-- Activity [ID: inf] (Total: 41, Direct: 0)
        |-- Verifying the Existence of a Warning Banner [ID: 859] (Total: 12,
Direct: 12)
        |-- Recovering a Password from Password-Protected Files [ID: 860]
(Total: 29, Direct: 29)
  |-- Chapter 16. Ethics for the Expert Witness [ID: inf] (Total: 73, Direct: 0)
```

```
    |-- Lab 16.1. Rebuilding an MFT Record from a Corrupt Image [ID: 864]
(Total: 73, Direct: 1)
      |-- Activity [ID: inf] (Total: 72, Direct: 0)
        |-- Creating a Duplicate Forensic Image [ID: 868] (Total: 14, Direct:
14)
        |-- Determining the Offset Byte Address of the Corrupt MFT Record [ID:
869] (Total: 12, Direct: 12)
        |-- Copying the Corrected MFT Record [ID: 870] (Total: 20, Direct: 20)
        |-- Extracting Additional Evidence [ID: 871] (Total: 26, Direct: 26)
|-- Appendix A. Certification Test References [ID: 873] (Total: 56, Direct: 56)
|-- Appendix B. Digital Forensics References [ID: 874] (Total: 109, Direct: 109)
|-- Appendix C. Digital Forensics Lab Considerations [ID: 875] (Total: 58,
Direct: 58)
|-- Appendix D. Legacy File System and Forensics Tools [ID: 876] (Total: 59,
Direct: 59)
|-- EPUB Preamble [ID: inf] (Total: 21, Direct: 21)

--------------------------------------------------------------------------------
                    Chunk Sequence & Content Integrity Test
--------------------------------------------------------------------------------

------------------------- CONTENT PREVIEW -------------------------
Title: Windows Prefetch Artifacts [toc_id: 456]
Chunk IDs: [6419, 6420, 6421, 6422, 6423, 6424, 6425, 6426, 6427, 6428, 6429,
6430, 6431, 6432, 6433, 6434, 6435, 6436, 6437, 6438, 6439, 6440, 6441, 6442,
6443, 6444]
-------------------------------------------------------------------
Windows Prefetch Artifacts
You can collect prefetch file artifacts with a disk editor or forensics tool. In
the following activity, you use WinHex's Data Interpreter to find an
application's MAC dates and times and the number of times Dropbox has run. You
need WinHex, OSForensics, and the image file InCh05.img from Chapter 5. Follow
these steps:
Note
Before beginning this activity, create a Work\Chap13\Chapter folder (referred to
as your "work folder" in the following steps).
1.
Copy InCh05.img from Chapter 5 to your work folder for this chapter. Start
OSForensics with the Run as administrator option, and click Continue Using Trial
Version. In the left pane, click Manage Case, if necessary. In the Manage Case
pane on the right, click the New Case button. In the New Case dialog box, type
InChap13 in the Case Name text box and your name in the Investigator text box.
For the Acquisition Type setting, click the Investigate Disk(s) from Another
Machine option button, and click Custom Location for the Case Folder setting.
Click the Browse button, navigate to and click your work folder, and then click
OK twice.
2.
To mount the disk image, scroll down the navigation pane on the left and click
```

Mount Drive Image.
3.
In the "Mounted virtual disks" window, click the Mount new button. In the OSFMount - Mount drive dialog box that opens, click the … button next to the Image file text box, navigate to your work folder, click InCh05.img , click Open, and then click OK.
4.
Click File Name Search in the left pane, click the … button next to the Config button, navigate to and click the drive letter where you mounted InCh05.img, and then click OK. In the Search String text box, type DROPBOX, and then click Search.
5.
When the search is finished, scroll through the list of results and find the DROPBOX.EXE-AA3E8112.pf file in the path C:\Windows\Prefetch (although your drive letter might differ). Right-click this file and click Open With.
6.
In the How do you want to open this file? dialog box, click Look for another app on this PC. In the "Open with" window, navigate to C:\Program Files \WinHex(or the location where you installed WinHex), and click WinHex.exe .
7.
In WinHex, click View from the menu, point to Show, and click Data Interpreter. Click Options, Data Interpreter from the menu to open the Data Interpreter Options dialog box. Click the Win32 FILETIME (64 bit) check box, if necessary, and then click OK.
8.
Click offset 0x80, and notice the date and time displayed in the Data Interpreter window. Repeat by clicking at offsets 0x88, 0x90, and 0x98 to view the other dates and times.
9.
Click Options, Data Interpreter from the menu again, click to clear the Win32 FILETIME (64 bit) check box, click the 8 bit, unsigned check box, and then click OK.
10.
Move the cursor and click at offset 0xD4 to see how many times this application has been run. When you’re finished, exit OSForensics and WinHex.
Tip
For more information on recovering evidence from computers with Dropbox, Google Drive, and OneDrive, review "Cloud Forensics" (Maegan Katz and Ryan Montelbano, www.champlain.edu/Documents/LCDI/CloudForensics.pdf, 2013) and "Cloud Storage Forensics" (Mattia Epifani, https://digital-forensics.sans.org/summit-archives/Prague_Summit/Cloud_Storage_Forensics_Mattia_Eppifani.pdf, 2013).
---------- END CONTENT PREVIEW ----------

Diagnostic Summary

Total Chunks in DB: 11774

```
================================================================================
                                Diagnostic Complete
================================================================================
```

5.2 Test Data Base for content development

Require Description

[19]:
```
# Cell 6: Verify Vector Database (Final Version with Rich Diagnostic Output)

import os
import json
import re
import random
import logging
from typing import List, Dict, Any, Tuple, Optional

# Third-party imports
try:
    from langchain_chroma import Chroma
    from langchain_ollama.embeddings import OllamaEmbeddings
    from langchain_core.documents import Document
    langchain_available = True
except ImportError:
    langchain_available = False

# Setup Logger for this cell
logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')
logger = logging.getLogger(__name__)

# --- HELPER FUNCTIONS ---


def print_results(query_text: str, results: list, where_filter: Optional[Dict] = None):
    """
    Richly prints query results, showing the query, filter, and retrieved documents.
    """
    print("\n" + "-"*10 + " DIAGNOSTIC: RETRIEVAL RESULTS " + "-"*10)
    print(f"QUERY: '{query_text}'")
    if where_filter:
        print(f"FILTER: {json.dumps(where_filter, indent=2)}")

    if not results:
        print("--> No documents were retrieved for this query and filter.")
```

```
        print("-" * 55)
        return

    print(f"--> Found {len(results)} results. Displaying top {min(len(results),␣
 ↪3)}:")
    for i, doc in enumerate(results[:3]):
        print(f"\n[ RESULT {i+1} ]")
        content_preview = doc.page_content.replace('\n', ' ').strip()
        print(f"  Content : '{content_preview[:200]}...'")
        print(f"  Metadata: {json.dumps(doc.metadata, indent=2)}")
    print("-" * 55)


# --- HELPER FUNCTIONS FOR FINDING DATA (UNCHANGED) ---
def find_deep_entry(nodes: List[Dict], current_path: List[str] = []) ->␣
 ↪Optional[Tuple[Dict, List[str]]]:
    shuffled_nodes = random.sample(nodes, len(nodes))
    for node in shuffled_nodes:
        if node.get('level', 0) >= 2 and node.get('children'): return node,␣
 ↪current_path + [node['title']]
        if node.get('children'):
            path = current_path + [node['title']]
            deep_entry = find_deep_entry(node['children'], path)
            if deep_entry: return deep_entry
    return None

def find_chapter_title_by_number(toc_data: List[Dict], chap_num: int) ->␣
 ↪Optional[List[str]]:
    def search_nodes(nodes, num, current_path):
        for node in nodes:
            path = current_path + [node['title']]
            if re.match(rf"(Chapter\s)?{num}[.:\s]", node.get('title', ''), re.
 ↪IGNORECASE): return path
            if node.get('children'):
                found_path = search_nodes(node['children'], num, path)
                if found_path: return found_path
        return None
    return search_nodes(toc_data, chap_num, [])


# --- ENHANCED TEST CASES with DIAGNOSTIC OUTPUT ---

def basic_retrieval_test(db, outline):
    print_header("Test 1: Basic Retrieval", char="-")
    try:
        logger.info("Goal: Confirm the database is live and contains␣
 ↪thematically relevant content.")
```

```
        logger.info("Strategy: Perform a simple similarity search using the␣
↪course's 'unitName'.")
        query_text = outline.get("unitInformation", {}).get("unitName",␣
↪"introduction")

        logger.info(f"Action: Searching for query: '{query_text}'...")
        results = db.similarity_search(query_text, k=1)

        print_results(query_text, results) # <--- SHOW THE EVIDENCE

        logger.info("Verification: Check if at least one document was returned.
↪")
        assert len(results) > 0, "Basic retrieval query returned no results."

        logger.info("  Result: TEST 1 PASSED. The database is online and␣
↪responsive.")
        return True
    except Exception as e:
        logger.error(f"  Result: TEST 1 FAILED. Reason: {e}")
        return False

def deep_hierarchy_test(db, toc):
    print_header("Test 2: Deep Hierarchy Retrieval", char="-")
    try:
        logger.info("Goal: Verify that the multi-level hierarchical metadata␣
↪was ingested correctly.")
        logger.info("Strategy: Find a random, deeply nested sub-section and use␣
↪a precise filter to retrieve it.")
        deep_entry_result = find_deep_entry(toc)
        assert deep_entry_result, "Could not find a suitable deep entry (level␣
↪>= 2) to test."
        node, path = deep_entry_result
        query = node['title']

        logger.info(f"  - Selected random deep section: {' -> '.join(path)}")
        conditions = [{f"level_{i+1}_title": {"$eq": title}} for i, title in␣
↪enumerate(path)]
        w_filter = {"$and": conditions}

        logger.info("Action: Performing a similarity search with a highly␣
↪specific '$and' filter.")
        results = db.similarity_search(query, k=1, filter=w_filter)

        print_results(query, results, w_filter) # <--- SHOW THE EVIDENCE
```

```
        logger.info("Verification: Check if the precisely filtered query␣
↪returned any documents.")
        assert len(results) > 0, "Deeply filtered query returned no results."

        logger.info("  Result: TEST 2 PASSED. Hierarchical metadata is␣
↪structured correctly.")
        return True
    except Exception as e:
        logger.error(f"  Result: TEST 2 FAILED. Reason: {e}")
        return False

def advanced_alignment_test(db, outline, toc):
    print_header("Test 3: Advanced Unit Outline Alignment", char="-")
    try:
        logger.info("Goal: Ensure a weekly topic from the syllabus can be␣
↪mapped to the correct textbook chapter(s).")
        logger.info("Strategy: Pick a random week, find its chapter, and query␣
↪for the topic filtered by that chapter.")
        week_to_test = random.choice(outline['weeklySchedule'])
        logger.info(f"  - Selected random week: Week {week_to_test['week']} -␣
↪'{week_to_test['contentTopic']}'")

        reading = week_to_test.get('requiredReading', '')
        chap_nums_str = re.findall(r'\d+', reading)
        assert chap_nums_str, f"Could not find chapter numbers in required␣
↪reading: '{reading}'"
        logger.info(f"  - Extracted required chapter number(s):␣
↪{chap_nums_str}")

        chapter_paths = [find_chapter_title_by_number(toc, int(n)) for n in␣
↪chap_nums_str]
        chapter_paths = [path for path in chapter_paths if path is not None]
        assert chapter_paths, f"Could not map chapter numbers {chap_nums_str}␣
↪to a valid ToC path."

        level_1_titles = list(set([path[0] for path in chapter_paths]))
        logger.info(f"  - Mapped to top-level ToC entries: {level_1_titles}")

        or_filter = [{"level_1_title": {"$eq": title}} for title in␣
↪level_1_titles]
        w_filter = {"$or": or_filter} if len(or_filter) > 1 else or_filter[0]
        query = week_to_test['contentTopic']

        logger.info("Action: Searching for the weekly topic, filtered by the␣
↪mapped chapter(s).")
        results = db.similarity_search(query, k=5, filter=w_filter)
```

```
        print_results(query, results, w_filter) # <--- SHOW THE EVIDENCE

        logger.info("Verification: Check if at least one returned document is␣
↪from the correct chapter.")
        assert len(results) > 0, "Alignment query returned no results for the␣
↪correct section/chapter."

        logger.info("  Result: TEST 3 PASSED. The syllabus can be reliably␣
↪aligned with the textbook content.")
        return True
    except Exception as e:
        logger.error(f"  Result: TEST 3 FAILED. Reason: {e}")
        return False

def content_sequence_test(db, outline):
    print_header("Test 4: Content Sequence Verification", char="-")
    try:
        logger.info("Goal: Confirm that chunks for a topic can be re-ordered to␣
↪form a coherent narrative.")
        logger.info("Strategy: Retrieve several chunks for a random topic and␣
↪verify their 'chunk_id' is sequential.")
        topic_query = random.choice(outline['weeklySchedule'])['contentTopic']

        logger.info(f"Action: Performing similarity search for topic:␣
↪'{topic_query}' to get a set of chunks.")
        results = db.similarity_search(topic_query, k=10)

        print_results(topic_query, results) # <--- SHOW THE EVIDENCE

        docs_with_id = [doc for doc in results if 'chunk_id' in doc.metadata]
        assert len(docs_with_id) > 3, "Fewer than 4 retrieved chunks have a␣
↪'chunk_id' to test."

        chunk_ids = [doc.metadata['chunk_id'] for doc in docs_with_id]
        sorted_ids = sorted(chunk_ids)

        logger.info(f"  - Retrieved and sorted chunk IDs: {sorted_ids}")
        logger.info("Verification: Check if the sorted list of chunk_ids is␣
↪strictly increasing.")
        is_ordered = all(sorted_ids[i] >= sorted_ids[i-1] for i in range(1,␣
↪len(sorted_ids)))
        assert is_ordered, "The retrieved chunks' chunk_ids are not in␣
↪ascending order when sorted."
```

```
        logger.info("  Result: TEST 4 PASSED. Narrative order can be␣
 ↪reconstructed using 'chunk_id'.")
        return True
    except Exception as e:
        logger.error(f"  Result: TEST 4 FAILED. Reason: {e}")
        return False

# --- MAIN VERIFICATION EXECUTION ---
def run_verification():
    print_header("Database Verification Process")

    if not langchain_available:
        logger.error("LangChain libraries not found. Aborting tests.")
        return

    required_files = {
        "Chroma DB": CHROMA_PERSIST_DIR,
        "ToC JSON": PRE_EXTRACTED_TOC_JSON_PATH,
        "Parsed Outline": PARSED_UO_JSON_PATH
    }
    for name, path in required_files.items():
        if not os.path.exists(path):
            logger.error(f"Required '{name}' not found at '{path}'. Please run␣
 ↪previous cells.")
            return

    with open(PRE_EXTRACTED_TOC_JSON_PATH, 'r', encoding='utf-8') as f:
        toc_data = json.load(f)
    with open(PARSED_UO_JSON_PATH, 'r', encoding='utf-8') as f:
        unit_outline_data = json.load(f)

    logger.info("Connecting to DB and initializing components...")
    embeddings = OllamaEmbeddings(model=EMBEDDING_MODEL_OLLAMA)
    vector_store = Chroma(
        persist_directory=CHROMA_PERSIST_DIR,
        embedding_function=embeddings,
        collection_name=CHROMA_COLLECTION_NAME
    )

    results_summary = [
        basic_retrieval_test(vector_store, unit_outline_data),
        deep_hierarchy_test(vector_store, toc_data),
        advanced_alignment_test(vector_store, unit_outline_data, toc_data),
        content_sequence_test(vector_store, unit_outline_data)
    ]

    passed_count = sum(filter(None, results_summary))
```

```
    failed_count = len(results_summary) - passed_count

    print_header("Verification Summary")
    print(f"Total Tests Run: {len(results_summary)}")
    print(f"  Passed: {passed_count}")
    print(f"  Failed: {failed_count}")
    print_header("Verification Complete", char="=")

# --- Execute Verification ---
# Assumes global variables from Cell 1 are available in the notebook's scope
run_verification()
```

```
2025-07-01 21:02:48,736 - INFO - Connecting to DB and initializing components…
2025-07-01 21:02:48,746 - INFO - Goal: Confirm the database is live and contains
thematically relevant content.
2025-07-01 21:02:48,746 - INFO - Strategy: Perform a simple similarity search
using the course's 'unitName'.
2025-07-01 21:02:48,747 - INFO - Action: Searching for query: 'Digital
Forensic'…
2025-07-01 21:02:48,814 - INFO - HTTP Request: POST
http://127.0.0.1:11434/api/embed "HTTP/1.1 200 OK"
2025-07-01 21:02:48,818 - INFO - Verification: Check if at least one document
was returned.
2025-07-01 21:02:48,818 - INFO -   Result: TEST 1 PASSED. The database is online
and responsive.
2025-07-01 21:02:48,819 - INFO - Goal: Verify that the multi-level hierarchical
metadata was ingested correctly.
2025-07-01 21:02:48,819 - INFO - Strategy: Find a random, deeply nested sub-
section and use a precise filter to retrieve it.
2025-07-01 21:02:48,820 - INFO -    - Selected random deep section: Chapter 6.
Current Digital Forensics Tools -> Digital Forensics Hardware Tools -> Forensic
Workstations
2025-07-01 21:02:48,820 - INFO - Action: Performing a similarity search with a
highly specific '$and' filter.


================================================================================
                          Database Verification Process
================================================================================

--------------------------------------------------------------------------------
                             Test 1: Basic Retrieval
--------------------------------------------------------------------------------

---------- DIAGNOSTIC: RETRIEVAL RESULTS ----------
QUERY: 'Digital Forensic'
--> Found 1 results. Displaying top 1:
```

```
[ RESULT 1 ]
  Content : 'An Overview of Digital Forensics…'
  Metadata: {
  "chunk_id": 156,
  "source": "Bill Nelson, Amelia Phillips, Christopher Steuart - Guide to
Computer Forensics and Investigations_ Processing Digital Evidence-Cengage
Learning (2018).epub",
  "level_1_title": "Chapter 1. Understanding the Digital Forensics Profession
and Investigations",
  "toc_id": 9,
  "level_2_title": "An Overview of Digital Forensics"
}
----------------------------------------------------------

--------------------------------------------------------------------------------
                        Test 2: Deep Hierarchy Retrieval
--------------------------------------------------------------------------------

2025-07-01 21:02:48,953 - INFO - HTTP Request: POST
http://127.0.0.1:11434/api/embed "HTTP/1.1 200 OK"
2025-07-01 21:02:48,965 - INFO - Verification: Check if the precisely filtered
query returned any documents.
2025-07-01 21:02:48,965 - INFO -  Result: TEST 2 PASSED. Hierarchical metadata
is structured correctly.
2025-07-01 21:02:48,966 - INFO - Goal: Ensure a weekly topic from the syllabus
can be mapped to the correct textbook chapter(s).
2025-07-01 21:02:48,966 - INFO - Strategy: Pick a random week, find its chapter,
and query for the topic filtered by that chapter.
2025-07-01 21:02:48,967 - INFO -   - Selected random week: Week Week 9 - 'Email
and Social Media.'
2025-07-01 21:02:48,967 - INFO -   - Extracted required chapter number(s):
['2019', '978', '1', '337', '56894', '4', '11']
2025-07-01 21:02:48,970 - INFO -   - Mapped to top-level ToC entries: ['Chapter
11. E-mail and Social Media Investigations', 'Chapter 4. Processing Crime and
Incident Scenes', 'Chapter 1. Understanding the Digital Forensics Profession and
Investigations']
2025-07-01 21:02:48,970 - INFO - Action: Searching for the weekly topic,
filtered by the mapped chapter(s).
2025-07-01 21:02:49,080 - INFO - HTTP Request: POST
http://127.0.0.1:11434/api/embed "HTTP/1.1 200 OK"
2025-07-01 21:02:49,099 - INFO - Verification: Check if at least one returned
document is from the correct chapter.
2025-07-01 21:02:49,100 - INFO -  Result: TEST 3 PASSED. The syllabus can be
reliably aligned with the textbook content.
2025-07-01 21:02:49,101 - INFO - Goal: Confirm that chunks for a topic can be
re-ordered to form a coherent narrative.
2025-07-01 21:02:49,101 - INFO - Strategy: Retrieve several chunks for a random
topic and verify their 'chunk_id' is sequential.
```

```
2025-07-01 21:02:49,102 - INFO - Action: Performing similarity search for topic:
'Current Computer Forensics Tools.' to get a set of chunks.


---------- DIAGNOSTIC: RETRIEVAL RESULTS ----------
QUERY: 'Forensic Workstations'
FILTER: {
  "$and": [
    {
      "level_1_title": {
        "$eq": "Chapter 6. Current Digital Forensics Tools"
      }
    },
    {
      "level_2_title": {
        "$eq": "Digital Forensics Hardware Tools"
      }
    },
    {
      "level_3_title": {
        "$eq": "Forensic Workstations"
      }
    }
  ]
}
--> Found 1 results. Displaying top 1:

[ RESULT 1 ]
  Content : 'Forensic Workstations…'
  Metadata: {
  "level_3_title": "Forensic Workstations",
  "level_2_title": "Digital Forensics Hardware Tools",
  "toc_id": 255,
  "level_1_title": "Chapter 6. Current Digital Forensics Tools",
  "source": "Bill Nelson, Amelia Phillips, Christopher Steuart - Guide to
Computer Forensics and Investigations_ Processing Digital Evidence-Cengage
Learning (2018).epub",
  "chunk_id": 3311
}
----------------------------------------------------------


--------------------------------------------------------------------------------
                    Test 3: Advanced Unit Outline Alignment
--------------------------------------------------------------------------------

---------- DIAGNOSTIC: RETRIEVAL RESULTS ----------
QUERY: 'Email and Social Media.'
FILTER: {
```

```
  "$or": [
    {
      "level_1_title": {
        "$eq": "Chapter 11. E-mail and Social Media Investigations"
      }
    },
    {
      "level_1_title": {
        "$eq": "Chapter 4. Processing Crime and Incident Scenes"
      }
    },
    {
      "level_1_title": {
        "$eq": "Chapter 1. Understanding the Digital Forensics Profession and
Investigations"
      }
    }
  ]
}
--> Found 5 results. Displaying top 3:

[ RESULT 1 ]
  Content : 'Chapter 11. E-mail and Social Media Investigations…'
  Metadata: {
  "source": "Bill Nelson, Amelia Phillips, Christopher Steuart - Guide to
Computer Forensics and Investigations_ Processing Digital Evidence-Cengage
Learning (2018).epub",
  "chunk_id": 5378,
  "toc_id": 377,
  "level_1_title": "Chapter 11. E-mail and Social Media Investigations"
}

[ RESULT 2 ]
  Content : 'Chapter 11. E-mail and Social Media Investigations…'
  Metadata: {
  "chunk_id": 10484,
  "level_1_title": "Chapter 11. E-mail and Social Media Investigations",
  "source": "Bill Nelson, Amelia Phillips, Christopher Steuart - Guide to
Computer Forensics and Investigations_ Processing Digital Evidence-Cengage
Learning (2018).epub",
  "toc_id": 377
}

[ RESULT 3 ]
  Content : 'Social media can contain a lot of information, including the
following:…'
  Metadata: {
  "level_2_title": "Applying Digital Forensics Methods to Social Media
```

```
Communications",
  "chunk_id": 5636,
  "toc_id": 398,
  "source": "Bill Nelson, Amelia Phillips, Christopher Steuart - Guide to
Computer Forensics and Investigations_ Processing Digital Evidence-Cengage
Learning (2018).epub",
  "level_1_title": "Chapter 11. E-mail and Social Media Investigations"
}
------------------------------------------------------

--------------------------------------------------------------------------------
                    Test 4: Content Sequence Verification
--------------------------------------------------------------------------------

2025-07-01 21:02:49,217 - INFO - HTTP Request: POST
http://127.0.0.1:11434/api/embed "HTTP/1.1 200 OK"
2025-07-01 21:02:49,220 - INFO -   - Retrieved and sorted chunk IDs: [49, 3138,
3141, 3160, 3164, 3166, 3267, 3271, 3308, 9541]
2025-07-01 21:02:49,221 - INFO - Verification: Check if the sorted list of
chunk_ids is strictly increasing.
2025-07-01 21:02:49,221 - INFO -  Result: TEST 4 PASSED. Narrative order can be
reconstructed using 'chunk_id'.


---------- DIAGNOSTIC: RETRIEVAL RESULTS ----------
QUERY: 'Current Computer Forensics Tools.'
--> Found 10 results. Displaying top 3:

[ RESULT 1 ]
  Content : 'Chapter 6. Current Digital Forensics Tools…'
  Metadata: {
  "toc_id": 231,
  "level_1_title": "Chapter 6. Current Digital Forensics Tools",
  "source": "Bill Nelson, Amelia Phillips, Christopher Steuart - Guide to
Computer Forensics and Investigations_ Processing Digital Evidence-Cengage
Learning (2018).epub",
  "chunk_id": 3138
}

[ RESULT 2 ]
  Content : 'Chapter 6. Current Digital Forensics Tools…'
  Metadata: {
  "level_1_title": "Chapter 6. Current Digital Forensics Tools",
  "chunk_id": 9541,
  "source": "Bill Nelson, Amelia Phillips, Christopher Steuart - Guide to
Computer Forensics and Investigations_ Processing Digital Evidence-Cengage
Learning (2018).epub",
  "toc_id": 231
}
```

```
[ RESULT 3 ]
  Content : 'Software Forensics Tools…'
  Metadata: {
  "level_3_title": "Types of Digital Forensics Tools",
  "level_4_title": "Software Forensics Tools",
  "level_1_title": "Chapter 6. Current Digital Forensics Tools",
  "source": "Bill Nelson, Amelia Phillips, Christopher Steuart - Guide to
Computer Forensics and Investigations_ Processing Digital Evidence-Cengage
Learning (2018).epub",
  "level_2_title": "Evaluating Digital Forensics Tool Needs",
  "toc_id": 236,
  "chunk_id": 3166
}
----------------------------------------------------

================================================================================
                              Verification Summary
================================================================================
Total Tests Run: 4
  Passed: 4
  Failed: 0

================================================================================
                             Verification Complete
================================================================================
```

6 Content Generation

6.1 Configuration and Scope

[20]:
```python
# Cell 7: Configuration and Scoping for Content Generation

import os
import json
import logging

# Setup Logger for this cell
logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s -␣
 ↪%(message)s')
logger = logging.getLogger(__name__)

# --- 1. DEFINE FILE PATHS AND GLOBAL TEST SETTINGS ---
# Assumes these variables are loaded from a previous setup cell (like Cell 1)
# For demonstration, we define them here. Replace with your actual global vars.
# PROJECT_BASE_DIR = "/path/to/your/project"
```

```
# PARSED_UO_JSON_PATH = os.path.join(PROJECT_BASE_DIR, "Parse_data/Parse_UO/
 ↪ICT312_Digital_Forensic_Final_parsed.json")
# PRE_EXTRACTED_TOC_JSON_PATH = os.path.join(PROJECT_BASE_DIR, "Parse_data/
 ↪Parse_TOC_books/ICT312_epub_table_of_contents.json")

# New configuration file paths
CONFIG_DIR = os.path.join(PROJECT_BASE_DIR, "configs")
SETTINGS_DECK_PATH = os.path.join(CONFIG_DIR, "settings_deck.json")
TEACHING_FLOWS_PATH = os.path.join(CONFIG_DIR, "teaching_flows.json")

# New output path for the processed settings
PROCESSED_SETTINGS_PATH = os.path.join(CONFIG_DIR, "processed_settings.json")

# --- Global Test Overrides (for easy testing) ---
# To test a specific week, change this from "all" to a list, e.g., [7]
# To test a different flow, change the teaching_flow_id.

TEST_OVERRIDE_WEEKS = [7] # e.g., [7] or [1, 2, 3] or "all"
TEST_OVERRIDE_FLOW_ID = "apply_topic_interactive" # or "standard_lecture"
TEST_OVERRIDE_SESSIONS_PER_WEEK = 1 # e.g., 1 or 2
TEST_OVERRIDE_DISTRIBUTION_STRATEGY = "even" # 'even', 'front_load', 'end_load'

def print_header(text: str, char: str = "="):
    """Prints a centered header to the console."""
    print("\n" + char * 80)
    print(text.center(80))
    print(char * 80)


def process_and_load_configurations():
    """
    Loads all configuration files, processes them to create a definitive plan,
    and returns a master configuration object.
    """
    print_header("Configuration and Scoping Process", char="-")

    # --- 2. LOAD ALL INPUT FILES ---
    logger.info("Loading all necessary configuration and data files...")
    try:
        with open(PARSED_UO_JSON_PATH, 'r', encoding='utf-8') as f:
            unit_outline = json.load(f)
        with open(PRE_EXTRACTED_TOC_JSON_PATH, 'r', encoding='utf-8') as f:
            book_toc = json.load(f)
        with open(SETTINGS_DECK_PATH, 'r', encoding='utf-8') as f:
            settings_deck = json.load(f)
        with open(TEACHING_FLOWS_PATH, 'r', encoding='utf-8') as f:
            teaching_flows = json.load(f)
```

```
        logger.info("All files loaded successfully.")
    except FileNotFoundError as e:
        logger.error(f"FATAL: A required configuration file was not found: {e}")
        return None

    # --- 3. PRE-PROCESS AND REFINE SETTINGS ---
    logger.info("Pre-processing settings_deck for definitive plan...")

    # Create a deep copy to avoid modifying the original object
    processed_settings = json.loads(json.dumps(settings_deck))

    # a. Smartly set Course ID and Unit Name from the Unit Outline
    unit_info = unit_outline.get("unitInformation", {})
    course_id = unit_info.get("unitCode", "UNKNOWN_COURSE")
    unit_name = unit_info.get("unitName", "Unknown Unit Name")

    processed_settings['course_id'] = course_id
    processed_settings['unit_name'] = unit_name # <-- NEW: Add unit_name

    logger.info(f"  - Set 'course_id' from Unit Outline: {course_id}")
    logger.info(f"  - Set 'unit_name' from Unit Outline: {unit_name}")

    # b. Apply test overrides for easier development
    processed_settings['teaching_flow_id'] = TEST_OVERRIDE_FLOW_ID
    logger.info(f"  - Using 'teaching_flow_id' from test override:␣
↪'{TEST_OVERRIDE_FLOW_ID}'")

    processed_settings['week_session_setup']['sessions_per_week'] =␣
↪TEST_OVERRIDE_SESSIONS_PER_WEEK
    logger.info(f"  - Using 'sessions_per_week' from test override:␣
↪{TEST_OVERRIDE_SESSIONS_PER_WEEK}")

    processed_settings['week_session_setup']['distribution_strategy'] =␣
↪TEST_OVERRIDE_DISTRIBUTION_STRATEGY
    logger.info(f"  - Using 'distribution_strategy' from test override:␣
↪'{TEST_OVERRIDE_DISTRIBUTION_STRATEGY}'")

    # c. Resolve the generation scope (which weeks to generate)
    scope = TEST_OVERRIDE_WEEKS
    if scope == "all":
        num_weeks = len(unit_outline.get('weeklySchedule', []))
        if num_weeks == 0:
            logger.error("Unit Outline 'weeklySchedule' is empty. Cannot␣
↪determine number of weeks.")
            return None
        final_scope = list(range(1, num_weeks + 1))
```

```
        logger.info(f"  - 'generation_scope' is 'all'. Resolved to {num_weeks}␣
↪weeks: {final_scope}")
    elif isinstance(scope, list) and all(isinstance(i, int) for i in scope):
        final_scope = scope
        logger.info(f"  - 'generation_scope' is a specific list from test␣
↪override: {final_scope}")
    else:
        logger.error(f"Invalid 'generation_scope' setting: {scope}. Must be␣
↪'all' or a list of integers.")
        return None

    processed_settings['generation_scope']['weeks'] = final_scope

    # --- 4. ASSEMBLE & SAVE FINAL CONFIGURATION ---
    master_config = {
        "processed_settings": processed_settings,
        "unit_outline": unit_outline,
        "book_toc": book_toc,
        "teaching_flows": teaching_flows
    }

    logger.info(f"Saving the final processed configuration to:␣
↪{PROCESSED_SETTINGS_PATH}")
    os.makedirs(os.path.dirname(PROCESSED_SETTINGS_PATH), exist_ok=True)
    with open(PROCESSED_SETTINGS_PATH, 'w', encoding='utf-8') as f:
        json.dump(processed_settings, f, indent=2)

    print_header("Configuration Complete", char="-")
    logger.info("Master configuration object is ready for the generation␣
↪pipeline.")

    return master_config

# --- EXECUTE THE CONFIGURATION PROCESS ---
master_config = process_and_load_configurations()

# Optional: Print a preview to verify the output
if master_config:
    print("\n--- Preview of Processed Settings ---")
    print(json.dumps(master_config['processed_settings'], indent=2))
    print(f"\nNumber of weeks to generate:␣
↪{len(master_config['processed_settings']['generation_scope']['weeks'])}")
    print("-------------------------------------")
```

```
2025-07-01 21:37:34,335 - INFO - Loading all necessary configuration and data
files…
2025-07-01 21:37:34,342 - INFO - All files loaded successfully.
```

```
2025-07-01 21:37:34,342 - INFO - Pre-processing settings_deck for definitive
plan…
2025-07-01 21:37:34,343 - INFO -   - Set 'course_id' from Unit Outline: ICT312
2025-07-01 21:37:34,343 - INFO -   - Set 'unit_name' from Unit Outline: Digital
Forensic
2025-07-01 21:37:34,343 - INFO -   - Using 'teaching_flow_id' from test
override: 'apply_topic_interactive'
2025-07-01 21:37:34,344 - INFO -   - Using 'sessions_per_week' from test
override: 1
2025-07-01 21:37:34,344 - INFO -   - Using 'distribution_strategy' from test
override: 'even'
2025-07-01 21:37:34,344 - INFO -   - 'generation_scope' is a specific list from
test override: [7]
2025-07-01 21:37:34,345 - INFO - Saving the final processed configuration to:
/home/sebas_dev_linux/projects/course_generator/configs/processed_settings.json
2025-07-01 21:37:34,346 - INFO - Master configuration object is ready for the
generation pipeline.

--------------------------------------------------------------------------------
                        Configuration and Scoping Process
--------------------------------------------------------------------------------

--------------------------------------------------------------------------------
                               Configuration Complete
--------------------------------------------------------------------------------

--- Preview of Processed Settings ---
{
  "course_id": "ICT312",
  "unit_name": "Digital Forensic",
  "teaching_flow_id": "apply_topic_interactive",
  "slide_count_strategy": {
    "method": "per_week",
    "target": 25
  },
  "week_session_setup": {
    "sessions_per_week": 1,
    "distribution_strategy": "even"
  },
  "generation_scope": {
    "weeks": [
      7
    ]
  }
}

Number of weeks to generate: 1
```

6.2 Planning Agent

[21]:
```
# Cell 8: The Data-Driven Planning Agent (Chunk-Count Allocation)

import os
import json
import re
import math
import logging
from typing import List, Dict, Any, Optional

# Setup Logger and LangChain components
logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')
logger = logging.getLogger(__name__)
try:
    from langchain_chroma import Chroma
    from langchain_ollama.embeddings import OllamaEmbeddings
    langchain_available = True
except ImportError:
    langchain_available = False

class PlanningAgent:
    """
    An agent that creates a content plan driven by the syllabus and uses
    data-driven metrics (chunk counts) to allocate presentation time.
    """
    def __init__(self, master_config: Dict, vector_store: Optional[Any] = None):
        self.config = master_config['processed_settings']
        self.unit_outline = master_config['unit_outline']
        self.book_toc = master_config['book_toc']
        self.flat_toc_with_ids = self._create_flat_toc_with_ids()
        # The agent now requires access to the vector store to get chunk counts
        self.vector_store = vector_store
        logger.info("Data-Driven PlanningAgent initialized successfully.")

    def _create_flat_toc_with_ids(self) -> List[Dict]:
        """Creates a flattened list of the ToC with a unique sequential ID for each node."""
        flat_list = []
        def flatten_recursive(nodes, counter):
            for node in nodes:
                node_id = counter[0]; counter[0] += 1
                flat_list.append({'toc_id': node_id, 'title': node['title'], 'node': node})
```

```
                if node.get('children'):
                    flatten_recursive(node.get('children'), counter)
        flatten_recursive(self.book_toc, [0])
        return flat_list

    def _identify_relevant_chapters(self, weekly_schedule_item: Dict) ->␣
↪List[int]:
        """Extracts chapter numbers precisely from the 'requiredReading' string.
↪"""
        reading_str = weekly_schedule_item.get('requiredReading', '')
        match = re.search(r'Chapter(s)?', reading_str, re.IGNORECASE)
        if not match: return []
        search_area = reading_str[match.start():]
        chap_nums_str = re.findall(r'\d+', search_area)
        if chap_nums_str:
            return sorted(list(set(int(n) for n in chap_nums_str)))
        return []

    def _find_chapter_node(self, chapter_number: int) -> Optional[Dict]:
        """Accurately finds the ToC node for a specific chapter number."""
        for item in self.flat_toc_with_ids:
            title = item['title']
            if re.match(rf"Chapter\s{chapter_number}(?:\D|$)", title):
                return item['node']
        return None

    def _get_chapter_sub_topics(self, chapter_node: Dict) -> List[Dict]:
        """Extracts direct, teachable sub-topics from a chapter node."""
        sub_topics = []
        if not chapter_node.get('children'): return []
        for child_node in chapter_node['children']:
            title_lower = child_node['title'].lower()
            if "review" in title_lower or "introduction" in title_lower:␣
↪continue
            matched_item = next((item for item in self.flat_toc_with_ids if␣
↪item['node'] is child_node), None)
            if matched_item:
                sub_topics.append({"title": matched_item['title'], "toc_id":␣
↪matched_item['toc_id']})
        return sub_topics

    # --- NEW DATA-DRIVEN ALLOCATION METHOD ---
    def _allocate_slides_by_chunk_count(self, topics: List[Dict],␣
↪total_slides_for_session: int) -> List[Dict]:
        """
```

```
        Allocates slides proportionally based on the number of chunks␣
↪associated with each topic.
        """
        if not self.vector_store:
            logger.error("Vector store not available. Cannot perform chunk␣
↪count allocation.")
            return topics # Fallback or handle error appropriately

        logger.info("Getting chunk counts for each topic to determine slide␣
↪allocation...")

        # Step 1: Get chunk count for each topic
        total_chunks_in_session = 0
        for topic in topics:
            toc_id = topic['toc_id']
            # .count() is not a standard API, use .get() and count the results
            count = len(self.vector_store._collection.get(where={"toc_id":␣
↪toc_id})['ids'])
            topic['chunk_count'] = count
            total_chunks_in_session += count

        logger.info(f"Total chunks for this session: {total_chunks_in_session}")

        # Step 2: Allocate slides proportionally
        content_slides = max(1, total_slides_for_session - 4)
        slides_allocated_so_far = 0

        for topic in topics:
            if total_chunks_in_session == 0:
                proportion = 1 / len(topics) # Fallback to even distribution
            else:
                proportion = topic['chunk_count'] / total_chunks_in_session

            # Use floor to avoid over-allocating, will distribute remainder␣
↪later
            topic['slides_allocated'] = math.floor(proportion * content_slides)
            slides_allocated_so_far += topic['slides_allocated']

        # Step 3: Distribute remainder slides to topics with the most chunks
        remainder = content_slides - slides_allocated_so_far
        if remainder > 0:
            # Sort topics by chunk count, descending, to give remainder slides␣
↪to largest topics
            sorted_topics = sorted(topics, key=lambda x: x.get('chunk_count',␣
↪0), reverse=True)
            for i in range(remainder):
```

```
                sorted_topics[i]['slides_allocated'] += 1

        return topics

    def create_content_plan_for_week(self, week_number: int) -> Optional[Dict]:
        """Orchestrates the planning process for a single week."""
        print_header(f"Planning Week {week_number}", char="*")
        weekly_schedule_item = self.unit_outline['weeklySchedule'][week_number␣
↪- 1]

        chapter_numbers = self._identify_relevant_chapters(weekly_schedule_item)
        if not chapter_numbers:
            logger.error(f"No valid chapter numbers found. Aborting.")
            return None

        total_slides_per_week = self.config['slide_count_strategy'].
↪get('target', 25)
        slides_per_session = total_slides_per_week // len(chapter_numbers) if␣
↪chapter_numbers else total_slides_per_week

        final_sessions_plan = []
        for i, chap_num in enumerate(chapter_numbers):
            logger.info(f"--- Planning Session {i+1} (Chapter {chap_num}) ---")
            chapter_node = self._find_chapter_node(chap_num)
            if not chapter_node: continue

            sub_topics = self._get_chapter_sub_topics(chapter_node)
            if not sub_topics: continue

            # Use the new data-driven allocation method
            topics_with_slides = self.
↪_allocate_slides_by_chunk_count(sub_topics, slides_per_session)

            final_sessions_plan.append({
                "session_number": i + 1,
                "session_topic": chapter_node['title'],
                "topics_to_cover": topics_with_slides
            })

        if not final_sessions_plan: return None

        week_plan = {"week": week_number, "overall_topic": weekly_schedule_item.
↪get('contentTopic'), "sessions": final_sessions_plan}
        print_header(f"Plan for Week {week_number} Generated Successfully",␣
↪char="*")
        return week_plan
```

```
# --- EXECUTION BLOCK ---
logger.info("--- Initializing Data-Driven Planning Agent Test ---")
if langchain_available:
    # Connect to the DB first, as the agent now requires it
    vector_store = Chroma(
        persist_directory=CHROMA_PERSIST_DIR,
        embedding_function=OllamaEmbeddings(model=EMBEDDING_MODEL_OLLAMA)
    )

    with open(os.path.join(CONFIG_DIR, "processed_settings.json"), 'r') as f:
        processed_settings = json.load(f)
    with open(PRE_EXTRACTED_TOC_JSON_PATH, 'r') as f:
        book_toc = json.load(f)
    with open(PARSED_UO_JSON_PATH, 'r') as f:
        unit_outline = json.load(f)

    master_config_from_file = {"processed_settings": processed_settings,␣
 ↪"unit_outline": unit_outline, "book_toc": book_toc}

    # Initialize the agent with the vector store connection
    planning_agent = PlanningAgent(master_config_from_file, vector_store)

    WEEK_TO_TEST = 7
    logger.info(f"--> Explicitly testing planning logic for Week␣
 ↪{WEEK_TO_TEST}")
    content_plan = planning_agent.create_content_plan_for_week(WEEK_TO_TEST)

    if content_plan:
        print("\n--- Generated Content Plan (Data-Driven Allocation) ---")
        print(json.dumps(content_plan, indent=2))

        PLAN_OUTPUT_DIR = os.path.join(PROJECT_BASE_DIR, "generated_plans")
        os.makedirs(PLAN_OUTPUT_DIR, exist_ok=True)
        plan_filename =␣
 ↪f"{processed_settings['course_id']}_Week{WEEK_TO_TEST}_plan.json"
        plan_filepath = os.path.join(PLAN_OUTPUT_DIR, plan_filename)
        with open(plan_filepath, 'w') as f: json.dump(content_plan, f, indent=2)
        logger.info(f"\nSuccessfully saved content plan for Week {WEEK_TO_TEST}␣
 ↪to: {plan_filepath}")
else:
    logger.error("LangChain/Chroma libraries not found. Cannot run the Planning␣
 ↪Agent.")
```

```
2025-07-01 21:38:11,053 - INFO - --- Initializing Data-Driven Planning Agent
Test ---
2025-07-01 21:38:11,084 - INFO - Data-Driven PlanningAgent initialized
```

```
successfully.
2025-07-01 21:38:11,088 - INFO - --> Explicitly testing planning logic for Week
7
2025-07-01 21:38:11,089 - INFO - --- Planning Session 1 (Chapter 7) ---
2025-07-01 21:38:11,089 - INFO - Getting chunk counts for each topic to
determine slide allocation…
2025-07-01 21:38:11,095 - INFO - Total chunks for this session: 0
2025-07-01 21:38:11,096 - INFO - --- Planning Session 2 (Chapter 8) ---
2025-07-01 21:38:11,096 - INFO - Getting chunk counts for each topic to
determine slide allocation…
2025-07-01 21:38:11,098 - INFO - Total chunks for this session: 0
2025-07-01 21:38:11,099 - INFO -
Successfully saved content plan for Week 7 to: /home/sebas_dev_linux/projects/co
urse_generator/generated_plans/ICT312_Week7_plan.json


********************************************************************************
                                Planning Week 7
********************************************************************************

********************************************************************************
                     Plan for Week 7 Generated Successfully
********************************************************************************

--- Generated Content Plan (Data-Driven Allocation) ---
{
  "week": 7,
  "overall_topic": "Linux Boot Processes and File Systems. Recovering Graphics
Files.",
  "sessions": [
    {
      "session_number": 1,
      "session_topic": "Chapter 7. Linux and Macintosh File Systems",
      "topics_to_cover": [
        {
          "title": "Examining Linux File Structures",
          "toc_id": 272,
          "chunk_count": 0,
          "slides_allocated": 3
        },
        {
          "title": "Understanding Macintosh File Structures",
          "toc_id": 276,
          "chunk_count": 0,
          "slides_allocated": 3
        },
        {
          "title": "Using Linux Forensics Tools",
```

```
            "toc_id": 280,
            "chunk_count": 0,
            "slides_allocated": 2
          }
        ]
      },
      {
        "session_number": 2,
        "session_topic": "Chapter 8. Recovering Graphics Files",
        "topics_to_cover": [
          {
            "title": "Recognizing a Graphics File",
            "toc_id": 291,
            "chunk_count": 0,
            "slides_allocated": 2
          },
          {
            "title": "Understanding Data Compression",
            "toc_id": 299,
            "chunk_count": 0,
            "slides_allocated": 2
          },
          {
            "title": "Identifying Unknown File Formats",
            "toc_id": 309,
            "chunk_count": 0,
            "slides_allocated": 2
          },
          {
            "title": "Understanding Copyright Issues with Graphics",
            "toc_id": 314,
            "chunk_count": 0,
            "slides_allocated": 2
          }
        ]
      }
    ]
}
```


[22]:
```
# Cell 8.1: Diagnostic Comparison Dashboard for Planning Agent (Final Version)

import os
import json
import re
import logging
from typing import List, Dict, Any, Optional
```

```
# Ensure the PlanningAgent class from Cell 8 is available in memory
try:
    from langchain_chroma import Chroma
    from langchain_ollama.embeddings import OllamaEmbeddings
    langchain_available = True
except ImportError:
    langchain_available = False

def print_diag_header(text: str):
    """Prints a formatted header for a diagnostic section."""
    print("\n" + "="*80)
    print(f"DIAGNOSTIC DASHBOARD: {text}")
    print("="*80)

# This helper is not strictly needed for the diagnostic but kept for␣
 ↪completeness
def find_node_by_title(nodes: List[Dict], title: str) -> Optional[Dict]:
    """Recursively finds a node in the ToC by its exact title."""
    for node in nodes:
        if node.get('title') == title:
            return node
        if node.get('children'):
            found = find_node_by_title(node.get('children', []), title)
            if found:
                return found
    return None

# --- Main Diagnostic Function ---
def run_comparison_diagnostic(week_to_test: int):
    logger.info(f"--- Starting Diagnostic Comparison Dashboard for Week␣
 ↪{week_to_test} ---")

    # 1. --- Load All necessary data ---
    try:
        with open(PRE_EXTRACTED_TOC_JSON_PATH, 'r') as f:
            book_toc_data = json.load(f)
        with open(PARSED_UO_JSON_PATH, 'r') as f:
            unit_outline_data = json.load(f)
        with open(os.path.join(CONFIG_DIR, "processed_settings.json"), 'r') as␣
 ↪f:
            processed_settings = json.load(f)

        master_config = {
            "processed_settings": processed_settings,
            "unit_outline": unit_outline_data,
            "book_toc": book_toc_data
        }
```

```
        logger.info("Successfully loaded all raw data files.")
    except Exception as e:
        logger.error(f"Failed to load initial files. Error: {e}")
        return

    # --- MODIFICATION: Show the initial input from the Unit Outline ---
    print_diag_header("Ground Truth: Weekly Schedule from Unit Outline")
    try:
        weekly_item = unit_outline_data['weeklySchedule'][week_to_test - 1]
        print(f"--- Input for Week {week_to_test} ---")
        print(json.dumps(weekly_item, indent=2))
    except IndexError:
        logger.error(f"Week {week_to_test} not found in Unit Outline. Aborting.
↪")
        return

    # 2. --- Generate the Plan using the Agent ---
    print_diag_header("Generated Plan")
    agent = PlanningAgent(master_config)
    generated_plan = agent.create_content_plan_for_week(week_to_test)

    if not generated_plan:
        logger.error("Planning Agent failed to generate a plan. Aborting␣
↪diagnostic.")
        return

    print(json.dumps(generated_plan, indent=2))

    # 3. --- Extract the relevant slice from the ground-truth ToC ---
    print_diag_header("Ground Truth: Relevant Section(s) from book_toc.json")
    chapter_numbers = agent._identify_relevant_chapters(weekly_item)

    if not chapter_numbers:
        logger.error("Could not identify chapters to create ToC slice.")
    else:
        for chap_num in chapter_numbers:
            chapter_node = agent._find_chapter_node(chap_num)
            if chapter_node:
                print(f"\n--- ToC for '{chapter_node['title']}' ---")
                print(json.dumps(chapter_node, indent=2))
            else:
                print(f"\nCould not find Chapter {chap_num} in ToC.")


    # 4. --- Verify Plan against Vector Database ---
    print_diag_header("Vector Database Verification")
    if not langchain_available:
```

```
        logger.warning("LangChain/Chroma libraries not available. Skipping␣
↪Vector DB verification.")
        return

    try:
        logger.info("Connecting to ChromaDB to verify chunk counts...")
        embeddings = OllamaEmbeddings(model=EMBEDDING_MODEL_OLLAMA)
        vector_store = Chroma(
            persist_directory=CHROMA_PERSIST_DIR,
            embedding_function=embeddings,
            collection_name=CHROMA_COLLECTION_NAME
        )
        logger.info("Successfully connected to ChromaDB.")

        # This handles both single and multi-session plans
        for session in generated_plan.get('sessions', []):
            topics_to_verify = session.get('topics_to_cover', [])
            print(f"\nVerifying chunk counts for Session␣
↪{session['session_number']}: '{session['session_topic']}'")
            print("-" * 70)
            print(f"{'Topic Title':<55} | {'ToC ID'} | {'Chunk Count'}")
            print("-" * 70)

            for topic in topics_to_verify:
                toc_id = topic['toc_id']
                title = topic['title']
                retrieved = vector_store._collection.get(where={"toc_id":␣
↪toc_id})
                count = len(retrieved['ids'])

                title_short = (title[:50] + '...') if len(title) > 53 else title
                print(f"{title_short:<55} | {toc_id:<6} | {count}")

            print("-" * 70)

    except Exception as e:
        logger.error(f"An error occurred during Vector DB verification: {e}",␣
↪exc_info=True)


# --- Execute the Diagnostic Dashboard ---
WEEK_TO_TEST = 7 # Set the week you want to diagnose here
run_comparison_diagnostic(WEEK_TO_TEST)
print("\n" + "="*80)
print("DIAGNOSTIC DASHBOARD COMPLETE")
print("="*80)
```

```
2025-07-01 21:38:19,164 - INFO - --- Starting Diagnostic Comparison Dashboard
for Week 7 ---
2025-07-01 21:38:19,165 - INFO - Successfully loaded all raw data files.
2025-07-01 21:38:19,166 - INFO - Data-Driven PlanningAgent initialized
successfully.
2025-07-01 21:38:19,166 - INFO - --- Planning Session 1 (Chapter 7) ---
2025-07-01 21:38:19,167 - ERROR - Vector store not available. Cannot perform
chunk count allocation.
2025-07-01 21:38:19,167 - INFO - --- Planning Session 2 (Chapter 8) ---
2025-07-01 21:38:19,167 - ERROR - Vector store not available. Cannot perform
chunk count allocation.
2025-07-01 21:38:19,168 - INFO - Connecting to ChromaDB to verify chunk
counts…
2025-07-01 21:38:19,177 - INFO - Successfully connected to ChromaDB.


================================================================================
DIAGNOSTIC DASHBOARD: Ground Truth: Weekly Schedule from Unit Outline
================================================================================
--- Input for Week 7 ---
{
  "week": "Week 7",
  "contentTopic": "Linux Boot Processes and File Systems. Recovering Graphics
Files.",
  "requiredReading": "Nelson, Phillips, Steuart,\u00a0Guide to Computer
Forensics and Investigations, Sixth Edition, Cengage Learning 2019,
ISBN:978-1-337-56894-4 Chapters 7 &8"
}

================================================================================
DIAGNOSTIC DASHBOARD: Generated Plan
================================================================================

********************************************************************************
                                Planning Week 7
********************************************************************************

********************************************************************************
                     Plan for Week 7 Generated Successfully
********************************************************************************
{
  "week": 7,
  "overall_topic": "Linux Boot Processes and File Systems. Recovering Graphics
Files.",
  "sessions": [
    {
      "session_number": 1,
      "session_topic": "Chapter 7. Linux and Macintosh File Systems",
```

```
      "topics_to_cover": [
        {
          "title": "Examining Linux File Structures",
          "toc_id": 272
        },
        {
          "title": "Understanding Macintosh File Structures",
          "toc_id": 276
        },
        {
          "title": "Using Linux Forensics Tools",
          "toc_id": 280
        }
      ]
    },
    {
      "session_number": 2,
      "session_topic": "Chapter 8. Recovering Graphics Files",
      "topics_to_cover": [
        {
          "title": "Recognizing a Graphics File",
          "toc_id": 291
        },
        {
          "title": "Understanding Data Compression",
          "toc_id": 299
        },
        {
          "title": "Identifying Unknown File Formats",
          "toc_id": 309
        },
        {
          "title": "Understanding Copyright Issues with Graphics",
          "toc_id": 314
        }
      ]
    }
  ]
}

================================================================================
DIAGNOSTIC DASHBOARD: Ground Truth: Relevant Section(s) from book_toc.json
================================================================================

--- ToC for 'Chapter 7. Linux and Macintosh File Systems' ---
{
  "level": 0,
  "title": "Chapter 7. Linux and Macintosh File Systems",
```

```
"children": [
  {
    "level": 1,
    "title": "Chapter Introduction",
    "children": []
  },
  {
    "level": 1,
    "title": "Examining Linux File Structures",
    "children": [
      {
        "level": 2,
        "title": "File Structures in Ext4",
        "children": [
          {
            "level": 3,
            "title": "Inodes",
            "children": []
          },
          {
            "level": 3,
            "title": "Hard Links and Symbolic Links",
            "children": []
          }
        ]
      }
    ]
  },
  {
    "level": 1,
    "title": "Understanding Macintosh File Structures",
    "children": [
      {
        "level": 2,
        "title": "An Overview of Mac File Structures",
        "children": []
      },
      {
        "level": 2,
        "title": "Forensics Procedures in Mac",
        "children": [
          {
            "level": 3,
            "title": "Acquisition Methods in macOS",
            "children": []
          }
        ]
      }
```

```
  ]
},
{
  "level": 1,
  "title": "Using Linux Forensics Tools",
  "children": [
    {
      "level": 2,
      "title": "Installing Sleuth Kit and Autopsy",
      "children": []
    },
    {
      "level": 2,
      "title": "Examining a Case with Sleuth Kit and Autopsy",
      "children": []
    }
  ]
},
{
  "level": 1,
  "title": "Chapter Review",
  "children": [
    {
      "level": 2,
      "title": "Chapter Summary",
      "children": []
    },
    {
      "level": 2,
      "title": "Key Terms",
      "children": []
    },
    {
      "level": 2,
      "title": "Review Questions",
      "children": []
    },
    {
      "level": 2,
      "title": "Hands-On Projects",
      "children": []
    },
    {
      "level": 2,
      "title": "Case Projects",
      "children": []
    }
  ]
```

```
    }
  ]
}

--- ToC for 'Chapter 8. Recovering Graphics Files' ---
{
  "level": 0,
  "title": "Chapter 8. Recovering Graphics Files",
  "children": [
    {
      "level": 1,
      "title": "Chapter Introduction",
      "children": []
    },
    {
      "level": 1,
      "title": "Recognizing a Graphics File",
      "children": [
        {
          "level": 2,
          "title": "Understanding Bitmap and Raster Images",
          "children": []
        },
        {
          "level": 2,
          "title": "Understanding Vector Graphics",
          "children": []
        },
        {
          "level": 2,
          "title": "Understanding Metafile Graphics",
          "children": []
        },
        {
          "level": 2,
          "title": "Understanding Graphics File Formats",
          "children": []
        },
        {
          "level": 2,
          "title": "Understanding Digital Photograph File Formats",
          "children": [
            {
              "level": 3,
              "title": "Examining the Raw File Format",
              "children": []
            },
            {
```

```
            "level": 3,
            "title": "Examining the Exchangeable Image File Format",
            "children": []
          }
        ]
      }
    ]
  },
  {
    "level": 1,
    "title": "Understanding Data Compression",
    "children": [
      {
        "level": 2,
        "title": "Lossless and Lossy Compression",
        "children": []
      },
      {
        "level": 2,
        "title": "Locating and Recovering Graphics Files",
        "children": []
      },
      {
        "level": 2,
        "title": "Identifying Graphics File Fragments",
        "children": []
      },
      {
        "level": 2,
        "title": "Repairing Damaged Headers",
        "children": []
      },
      {
        "level": 2,
        "title": "Searching for and Carving Data from Unallocated Space",
        "children": [
          {
            "level": 3,
            "title": "Planning Your Examination",
            "children": []
          },
          {
            "level": 3,
            "title": "Searching for and Recovering Digital Photograph
Evidence",
            "children": []
          }
        ]
```

```
    },
    {
      "level": 2,
      "title": "Rebuilding File Headers",
      "children": []
    },
    {
      "level": 2,
      "title": "Reconstructing File Fragments",
      "children": []
    }
  ]
},
{
  "level": 1,
  "title": "Identifying Unknown File Formats",
  "children": [
    {
      "level": 2,
      "title": "Analyzing Graphics File Headers",
      "children": []
    },
    {
      "level": 2,
      "title": "Tools for Viewing Images",
      "children": []
    },
    {
      "level": 2,
      "title": "Understanding Steganography in Graphics Files",
      "children": []
    },
    {
      "level": 2,
      "title": "Using Steganalysis Tools",
      "children": []
    }
  ]
},
{
  "level": 1,
  "title": "Understanding Copyright Issues with Graphics",
  "children": []
},
{
  "level": 1,
  "title": "Chapter Review",
  "children": [
```

```
        {
          "level": 2,
          "title": "Chapter Summary",
          "children": []
        },
        {
          "level": 2,
          "title": "Key Terms",
          "children": []
        },
        {
          "level": 2,
          "title": "Review Questions",
          "children": []
        },
        {
          "level": 2,
          "title": "Hands-On Projects",
          "children": []
        },
        {
          "level": 2,
          "title": "Case Projects",
          "children": []
        }
      ]
    }
  ]
}
```


```
================================================================================
DIAGNOSTIC DASHBOARD: Vector Database Verification
================================================================================

Verifying chunk counts for Session 1: 'Chapter 7. Linux and Macintosh File
Systems'
------------------------------------------------------------------------
Topic Title                                              | ToC ID | Chunk Count
------------------------------------------------------------------------
Examining Linux File Structures                          | 272    | 77
Understanding Macintosh File Structures                  | 276    | 6
Using Linux Forensics Tools                              | 280    | 5
------------------------------------------------------------------------

Verifying chunk counts for Session 2: 'Chapter 8. Recovering Graphics Files'
------------------------------------------------------------------------
Topic Title                                              | ToC ID | Chunk Count
------------------------------------------------------------------------
```

```
Recognizing a Graphics File                              | 291    | 4
Understanding Data Compression                           | 299    | 2
Identifying Unknown File Formats                         | 309    | 14
Understanding Copyright Issues with Graphics             | 314    | 19
-------------------------------------------------------------------

================================================================================
DIAGNOSTIC DASHBOARD COMPLETE
================================================================================
```

Next steps

Chunnk relation wwith the weights of the number of the slides per subtopic, haave in mind that 1 hour of delivery is like 20-25 slides

to ensure to move to the case to handle i wourl like to ensure the concepts are clear when we discussde about sessions and week, sessions in this context is number of classes that we have for week, if we say week , 3 sessions in one week or sessions_per_week = 3 is 3 classes per week that require 3 different set of