```
# Install Required Dependencies
!pip install flask
!pip install ftfy
!pip install gunicorn
!pip install nltk
!pip install numpy
!pip install pandas
!pip install regex
!pip install requests
!pip install requests-html
!pip install scikit-learn
!pip install scipy
!pip install beautifulsoup4
!pip install tika
!pip install spacy
!python -m spacy download en_core_web_sm
!pip install fake-useragent
!pip install openpyxl
!pip install zipfile36
!pip install xlsxwriter  # Add this line
```

```
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (1.0.12
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /usr/local/lib/python3.11/dist-packages (from spacy) (2.0.11)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /usr/local/lib/python3.11/dist-packages (from spacy) (3.0.9)
Requirement already satisfied: thinc<8.4.0,>=8.3.4 in /usr/local/lib/python3.11/dist-packages (from spacy) (8.3.6)
Requirement already satisfied: wasabi<1.2.0,>=0.9.1 in /usr/local/lib/python3.11/dist-packages (from spacy) (1.1.3)
Requirement already satisfied: srsly<3.0.0,>=2.4.3 in /usr/local/lib/python3.11/dist-packages (from spacy) (2.5.1)
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in /usr/local/lib/python3.11/dist-packages (from spacy) (2.0.10)
Requirement already satisfied: weasel<0.5.0,>=0.1.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (0.4.1)
Requirement already satisfied: typer<1.0.0,>=0.3.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (0.15.2)
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (4.67.1)
Requirement already satisfied: numpy>=1.19.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (2.0.2)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (2.32.3)
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4 in /usr/local/lib/python3.11/dist-packages (from spa
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-packages (from spacy) (3.1.6)
Requirement already satisfied: setuptools in /usr/local/lib/python3.11/dist-packages (from spacy) (75.2.0)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (24.2)
Requirement already satisfied: langcodes<4.0.0,>=3.2.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (3.5.0)
Requirement already satisfied: language-data>=1.2 in /usr/local/lib/python3.11/dist-packages (from langcodes<4.0.0,>=3.2
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.11/dist-packages (from pydantic!=1.8,!=1
Requirement already satisfied: pydantic-core==2.33.1 in /usr/local/lib/python3.11/dist-packages (from pydantic!=1.8,!=1.
Requirement already satisfied: typing-extensions>=4.12.2 in /usr/local/lib/python3.11/dist-packages (from pydantic!=1.8,
Requirement already satisfied: typing-inspection>=0.4.0 in /usr/local/lib/python3.11/dist-packages (from pydantic!=1.8,!
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests<3.0.0,
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests<3.0.0,>=2.13.0->sp
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests<3.0.0,>=2.13
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests<3.0.0,>=2.13
Requirement already satisfied: blis<1.4.0,>=1.3.0 in /usr/local/lib/python3.11/dist-packages (from thinc<8.4.0,>=8.3.4->
Requirement already satisfied: confection<1.0.0,>=0.0.1 in /usr/local/lib/python3.11/dist-packages (from thinc<8.4.0,>=8
Requirement already satisfied: click>=8.0.0 in /usr/local/lib/python3.11/dist-packages (from typer<1.0.0,>=0.3.0->spacy)
Requirement already satisfied: shellingham>=1.3.0 in /usr/local/lib/python3.11/dist-packages (from typer<1.0.0,>=0.3.0->
Requirement already satisfied: rich>=10.11.0 in /usr/local/lib/python3.11/dist-packages (from typer<1.0.0,>=0.3.0->spacy
Requirement already satisfied: cloudpathlib<1.0.0,>=0.7.0 in /usr/local/lib/python3.11/dist-packages (from weasel<0.5.0,
Requirement already satisfied: smart-open<8.0.0,>=5.2.1 in /usr/local/lib/python3.11/dist-packages (from weasel<0.5.0,>=
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.11/dist-packages (from jinja2->spacy) (3.0.2)
Requirement already satisfied: marisa-trie>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from language-data>=1.2->l
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.11/dist-packages (from rich>=10.11.0->typ
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.11/dist-packages (from rich>=10.11.0->t
Requirement already satisfied: wrapt in /usr/local/lib/python3.11/dist-packages (from smart-open<8.0.0,>=5.2.1->weasel<0
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.11/dist-packages (from markdown-it-py>=2.2.0->rich>=
Collecting en-core-web-sm==3.8.0
  Downloading https://github.com/explosion/spacy-models/releases/download/en_core_web_sm-3.8.0/en_core_web_sm-3.8.0-py3-
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.8/12.8 MB 52.9 MB/s eta 0:00:00
✔ Download and installation successful
You can now load the package via spacy.load('en_core_web_sm')
⚠ Restart to reload dependencies
If you are in a Jupyter or Colab notebook, you may need to restart Python in
order to load all the package's dependencies. You can do this by selecting the
'Restart kernel' or 'Restart runtime' option.
Requirement already satisfied: fake-useragent in /usr/local/lib/python3.11/dist-packages (2.2.0)
Requirement already satisfied: openpyxl in /usr/local/lib/python3.11/dist-packages (3.1.5)
Requirement already satisfied: et-xmlfile in /usr/local/lib/python3.11/dist-packages (from openpyxl) (2.0.0)
Requirement already satisfied: zipfile36 in /usr/local/lib/python3.11/dist-packages (0.1.3)
Collecting xlsxwriter
  Downloading XlsxWriter-3.2.3-py3-none-any.whl.metadata (2.7 kB)
Downloading XlsxWriter-3.2.3-py3-none-any.whl (169 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 169.4/169.4 kB 3.3 MB/s eta 0:00:00
Installing collected packages: xlsxwriter
Successfully installed xlsxwriter-3.2.3
```

```
# Import Required Libraries
import os
import re
import pandas as pd
import numpy as np
import spacy
```

```
from spacy.matcher import Matcher
from spacy.matcher import PhraseMatcher
from tika import parser
import nltk
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from bs4 import BeautifulSoup
import csv
import zipfile
from datetime import datetime

# Download required NLTK data
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('universal_tagset')
nltk.download('maxent_ne_chunker')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('brown')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package universal_tagset to /root/nltk_data...
[nltk_data]   Package universal_tagset is already up-to-date!
[nltk_data] Downloading package maxent_ne_chunker to
[nltk_data]     /root/nltk_data...
[nltk_data]   Package maxent_ne_chunker is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package brown to /root/nltk_data...
[nltk_data]   Package brown is already up-to-date!
True
```

```
def cleaningText(text):
    text = text.replace('\\n', '\n')
    text = text.replace('\\t', '\n')
    text = text.replace('\\r', '\n')
    text = text.replace('\n', ' ')
    text = re.sub('http\S+\s*', ' ', text)
    text = re.sub('RT|cc', ' ', text)
    text = re.sub('#\S+', ' ', text)
    text = re.sub('@\S+', ' ', text)
    text = re.sub('\s+', ' ', text)
    text = re.sub(r'[^\x00-\x7f]', "", text)
    return text.strip().lower()

def preprocessing(text):
    return cleaningText(text)

def vectorizing(skills, job):
    count_matrix = []
    for jobs in job:
        text = [skills, jobs]
        cv = TfidfVectorizer()
        count_matrix.append(cv.fit_transform(text))
    return count_matrix

def coSim(vector):
    matchPercentage = []
    for vec in vector:
        matchPercentage.append(cosine_similarity(vec)[0][1] * 100)
        matchPercentage = [round(percent, 2) for percent in matchPercentage]
    return matchPercentage


# Load spaCy model
nlp = spacy.load('en_core_web_sm')
matcher = Matcher(nlp.vocab)

# Create a list of common skills
skill = [
    # Programming Languages
    "python", "java", "javascript", "c++", "c#", "ruby", "php", "swift", "kotlin", "go",
    # Web Technologies
    "html", "css", "react", "angular", "vue.js", "node.js", "express.js", "django", "flask",
    "bootstrap", "jquery", "rest api", "graphql", "xml", "json",
```

```
        # Databases
        "sql", "mysql", "postgresql", "mongodb", "oracle", "redis", "elasticsearch",
        # Cloud & DevOps
        "aws", "azure", "google cloud", "docker", "kubernetes", "jenkins", "git", "github",
        "devops", "ci/cd", "linux", "unix", "bash",
        # Data Science & AI
        "machine learning", "deep learning", "artificial intelligence", "data analysis",
        "pandas", "numpy", "scipy", "scikit-learn", "tensorflow", "pytorch", "keras",
        "data visualization", "tableau", "power bi",
        # Mobile Development
        "android", "ios", "react native", "flutter", "xamarin",
        # Other Technical Skills
        "agile", "scrum", "jira", "testing", "debugging", "api development",
        "microservices", "rest", "soap", "version control",
        # Soft Skills
        "problem solving", "team work", "communication", "leadership", "project management",
        "time management", "analytical skills", "critical thinking",
        # Microsoft Office
        "microsoft office", "excel", "word", "powerpoint", "outlook",
        # Additional Tools
        "photoshop", "illustrator", "figma", "sketch", "adobe creative suite",
        # Frameworks
        "spring boot", "laravel", "asp.net", "ruby on rails", "symfony",
        # Testing
        "junit", "selenium", "pytest", "jest", "mocha",
        # Project Management
        "agile methodology", "scrum", "kanban", "waterfall", "prince2",
        # Security
        "cybersecurity", "encryption", "security", "penetration testing", "ethical hacking",
        # Analytics
        "google analytics", "seo", "data mining", "statistical analysis", "a/b testing",
        # Architecture
        "system design", "software architecture", "design patterns", "mvc", "rest"
    ]

    # Create skills matcher
    skillsmatcher = PhraseMatcher(nlp.vocab)
    patterns = [nlp.make_doc(text) for text in skill if len(nlp.make_doc(text)) < 10]
    skillsmatcher.add("Job title", None, *patterns)

    def extract_skills(text):
        skills = []
        __nlp = nlp(text.lower())
        matches = skillsmatcher(__nlp)
        for match_id, start, end in matches:
            span = __nlp[start:end]
            skills.append(span.text)
        skills = list(set(skills))
        return skills

    def convert_pdf_to_txt(pdf_file):
        raw_text = parser.from_file(pdf_file, service='text')['content']
        full_string = re.sub(r'\n+', '\n', raw_text)
        full_string = full_string.replace("\r", "\n")
        full_string = full_string.replace("\t", " ")
        full_string = re.sub(r"\uf0b7", " ", full_string)
        full_string = re.sub(r"\(cid:\d{0,2}\)", " ", full_string)
        full_string = re.sub(r'• ', " ", full_string)
        resume_lines = full_string.splitlines(True)
        resume_lines = [re.sub('\s+', ' ', line.strip()) for line in resume_lines if line.strip()]
        return resume_lines


    def get_sample_jobs():
        return [
            {
                'position': 'Software Engineer',
                'company': 'Tech Corp',
                'salary': '$80,000 - $120,000 a year',
                'description': '''
                Looking for a skilled Software Engineer with:
                - Strong experience in Python, JavaScript, and web development
                - Proficiency in React.js, Node.js, and modern web frameworks
                - Experience with SQL databases and RESTful APIs
                - Knowledge of AWS cloud services
                - Familiarity with Git and CI/CD pipelines
                - Good problem-solving and analytical skills
                - Excellent communication and teamwork abilities
                ''',
            },
            {
                'position': 'Data Scientist',
                'company': 'Data Analytics Inc',
```

```
            'salary': '$90,000 - $130,000 a year',
            'description': '''
            Seeking an experienced Data Scientist with:
            - Advanced Python programming skills
            - Expertise in Machine Learning and Statistical Analysis
            - Proficiency with Pandas, NumPy, Scikit-learn
            - Experience with Big Data technologies
            - Knowledge of SQL and data visualization
            - Strong mathematical and analytical skills
            - Excellent research and documentation abilities
            ''',
        },
        {
            'position': 'Full Stack Developer',
            'company': 'Web Solutions Ltd',
            'salary': '$85,000 - $125,000 a year',
            'description': '''
            Full Stack Developer position requiring:
            - Expertise in JavaScript/TypeScript
            - React.js and Node.js experience
            - MongoDB and Express.js knowledge
            - Strong HTML5 and CSS3 skills
            - Experience with RESTful APIs
            - Version control with Git
            - Agile development methodology
            ''',
        },
        {
            'position': 'DevOps Engineer',
            'company': 'Cloud Systems Inc',
            'salary': '$95,000 - $140,000 a year',
            'description': '''
            Seeking DevOps Engineer with:
            - Strong Linux/Unix administration skills
            - Experience with Docker and Kubernetes
            - AWS/Azure cloud platform expertise
            - CI/CD pipeline implementation
            - Infrastructure as Code (Terraform)
            - Python/Shell scripting abilities
            - Security best practices knowledge
            ''',
        },
        {
            'position': 'Frontend Developer',
            'company': 'Creative Web Agency',
            'salary': '$75,000 - $110,000 a year',
            'description': '''
            Frontend Developer position requiring:
            - Advanced JavaScript and TypeScript
            - React.js or Vue.js expertise
            - Modern CSS and Sass
            - Responsive design experience
            - Web performance optimization
            - Cross-browser compatibility
            - UI/UX best practices
            ''',
        }
    ]


def process_resume(resume_file):
    try:
        # Use sample data directly
        results = get_sample_jobs()

        # Create DataFrame directly from results
        job_df = pd.DataFrame(results)
        stopw = set(stopwords.words('english'))

        # Process jobs
        job_df['test'] = job_df['description'].apply(lambda x: ' '.join([word for word in str(x).split() if word not in (stc
        df = job_df.drop_duplicates(subset='test').reset_index(drop=True)
        df['clean'] = df['test'].apply(preprocessing)
        jobdesc = (df['clean'].values.astype('U'))

        # Process resume
        resume_text = " ".join(convert_pdf_to_txt(resume_file))
        skills = extract_skills(resume_text)
        skills_text = ' '.join(skills)
        skills_processed = preprocessing(skills_text)

        # Calculate similarity
        count_matrix = vectorizing(skills_processed, jobdesc)
```

```
            matchPercentage = coSim(count_matrix)
            matchPercentage = pd.DataFrame(matchPercentage, columns=['Skills Match'])

            # Prepare results
            result_cosine = df[['position', 'company']]
            result_cosine = result_cosine.join(matchPercentage)
            result_cosine = result_cosine[['position', 'company', 'Skills Match']]
            result_cosine.columns = ['Job Title', 'Company', 'Skills Match']
            result_cosine = result_cosine.sort_values('Skills Match', ascending=False).reset_index(drop=True)

            # Add resume name and skills
            result_cosine['Resume Name'] = os.path.basename(resume_file)
            result_cosine['Skills'] = ', '.join(skills)

            return result_cosine

        except Exception as e:
            print(f"Error processing {resume_file}: {str(e)}")
            return pd.DataFrame(columns=['Job Title', 'Company', 'Skills Match', 'Resume Name', 'Skills'])


    def process_zip_file(zip_path):
        # Create a temporary directory for extracted files
        temp_dir = "temp_resumes"
        os.makedirs(temp_dir, exist_ok=True)

        # Extract ZIP file
        with zipfile.ZipFile(zip_path, 'r') as zip_ref:
            zip_ref.extractall(temp_dir)

        # Process all PDF files
        all_results = []
        for filename in os.listdir(temp_dir):
            if filename.lower().endswith('.pdf'):
                file_path = os.path.join(temp_dir, filename)
                results = process_resume(file_path)
                if not results.empty:
                    all_results.append(results)

        # Combine all results
        if all_results:
            final_results = pd.concat(all_results, ignore_index=True)

            # Remove duplicates and sort by Skills Match
            final_results = final_results.drop_duplicates(subset=['Job Title', 'Company', 'Resume Name'])
            final_results = final_results.sort_values('Skills Match', ascending=False)

            # Format the output
            pd.set_option('display.max_columns', None)
            pd.set_option('display.width', None)
            pd.set_option('display.max_colwidth', None)

            # Save to Excel with better formatting
            timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
            excel_filename = f"resume_analysis_results_{timestamp}.xlsx"

            try:
                # Try using xlsxwriter first
                with pd.ExcelWriter(excel_filename, engine='xlsxwriter') as writer:
                    final_results.to_excel(writer, sheet_name='Results', index=False)

                    # Get workbook and worksheet objects
                    workbook = writer.book
                    worksheet = writer.sheets['Results']

                    # Add formats
                    header_format = workbook.add_format({
                        'bold': True,
                        'text_wrap': True,
                        'valign': 'top',
                        'fg_color': '#D7E4BC',
                        'border': 1
                    })

                    # Format the header
                    for col_num, value in enumerate(final_results.columns.values):
                        worksheet.write(0, col_num, value, header_format)
                        worksheet.set_column(col_num, col_num, 20)  # Set column width
            except:
                # Fallback to openpyxl if xlsxwriter fails
                final_results.to_excel(excel_filename, index=False)

            print(f"\nResults saved to {excel_filename}")
```

```
            # Clean up
            import shutil
            shutil.rmtree(temp_dir)

            # Display results in a cleaner format
            print("\nAnalysis Results:")
            print("=" * 80)
            for _, row in final_results.iterrows():
                print(f"\nResume: {row['Resume Name']}")
                print(f"Job Title: {row['Job Title']}")
                print(f"Company: {row['Company']}")
                print(f"Skills Match: {row['Skills Match']}%")
                print(f"Skills: {row['Skills']}")
                print("-" * 80)

            return final_results
        else:
            print("No valid PDF files found in the ZIP file")
            return None

    # Example usage
    from google.colab import files
    uploaded = files.upload()  # This will prompt you to upload your ZIP file

    # Get the filename of the uploaded ZIP file
    zip_filename = list(uploaded.keys())[0]

    # Process the ZIP file
    results = process_zip_file(zip_filename)
```

Choose files Espin Shalo…ume.pdf.zip

- **Espin Shalo Resume.pdf.zip**(application/zip) - 101086 bytes, last modified: 28/04/2025 - 100% done

```
Saving Espin Shalo Resume.pdf.zip to Espin Shalo Resume.pdf (2).zip

Results saved to resume_analysis_results_20250428_081326.xlsx

Analysis Results:
================================================================================

Resume: Espin Shalo Resume.pdf
Job Title: Data Scientist
Company: Data Analytics Inc
Skills Match: 5.2%
Skills: leadership, project management, python, flutter, time management, sql
--------------------------------------------------------------------------------

Resume: Espin Shalo Resume.pdf
Job Title: Software Engineer
Company: Tech Corp
Skills Match: 4.92%
Skills: leadership, project management, python, flutter, time management, sql
--------------------------------------------------------------------------------

Resume: Espin Shalo Resume.pdf
Job Title: DevOps Engineer
Company: Cloud Systems Inc
Skills Match: 2.93%
Skills: leadership, project management, python, flutter, time management, sql
--------------------------------------------------------------------------------

Resume: Espin Shalo Resume.pdf
Job Title: Full Stack Developer
Company: Web Solutions Ltd
Skills Match: 0.0%
Skills: leadership, project management, python, flutter, time management, sql
--------------------------------------------------------------------------------

Resume: Espin Shalo Resume.pdf
Job Title: Frontend Developer
Company: Creative Web Agency
Skills Match: 0.0%
Skills: leadership, project management, python, flutter, time management, sql
--------------------------------------------------------------------------------
```

```
def process_multiple_resumes(zip_path):
    # Create a temporary directory for extracted files
    temp_dir = "temp_resumes"
    os.makedirs(temp_dir, exist_ok=True)

    # Extract ZIP file
    with zipfile.ZipFile(zip_path, 'r') as zip_ref:
        zip_ref.extractall(temp_dir)

    # Initialize lists to store all data
```

```
        # Initialize lists to store all data
        all_resume_data = []
        all_job_matches = []

        # Process each PDF file
        for filename in os.listdir(temp_dir):
            if filename.lower().endswith('.pdf'):
                file_path = os.path.join(temp_dir, filename)
                try:
                    # Extract resume text
                    resume_text = " ".join(convert_pdf_to_txt(file_path))
                    skills = extract_skills(resume_text)

                    # Store resume data
                    resume_data = {
                        'Resume Name': filename,
                        'Skills Found': ', '.join(skills),
                        'Total Skills': len(skills),
                        'Processing Date': datetime.now().strftime("%Y-%m-%d %H:%M:%S")
                    }
                    all_resume_data.append(resume_data)

                    # Process against sample jobs
                    results = process_resume(file_path)
                    if not results.empty:
                        # Add resume name to results
                        results['Resume Name'] = filename
                        all_job_matches.append(results)

                except Exception as e:
                    print(f"Error processing {filename}: {str(e)}")

        if all_resume_data and all_job_matches:
            # Create DataFrames
            resume_df = pd.DataFrame(all_resume_data)
            matches_df = pd.concat(all_job_matches, ignore_index=True)

            # Remove duplicates and sort matches by Skills Match
            matches_df = matches_df.drop_duplicates(subset=['Job Title', 'Company', 'Resume Name'])
            matches_df = matches_df.sort_values('Skills Match', ascending=False)

            # Create Excel writer
            timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
            excel_filename = f"comprehensive_resume_analysis_{timestamp}.xlsx"

            try:
                with pd.ExcelWriter(excel_filename, engine='xlsxwriter') as writer:
                    # Write Resume Summary sheet
                    resume_df.to_excel(writer, sheet_name='Resume Summary', index=False)

                    # Write Job Matches sheet
                    matches_df.to_excel(writer, sheet_name='Job Matches', index=False)

                    # Get workbook and worksheet objects
                    workbook = writer.book

                    # Add formats
                    header_format = workbook.add_format({
                        'bold': True,
                        'text_wrap': True,
                        'valign': 'top',
                        'fg_color': '#D7E4BC',
                        'border': 1
                    })

                    # Format Resume Summary sheet
                    worksheet1 = writer.sheets['Resume Summary']
                    for col_num, value in enumerate(resume_df.columns.values):
                        worksheet1.write(0, col_num, value, header_format)
                        worksheet1.set_column(col_num, col_num, 20)

                    # Format Job Matches sheet
                    worksheet2 = writer.sheets['Job Matches']
                    for col_num, value in enumerate(matches_df.columns.values):
                        worksheet2.write(0, col_num, value, header_format)
                        worksheet2.set_column(col_num, col_num, 20)

                    # Add a Summary sheet
                    summary_data = {
                        'Metric': [
                            'Total Resumes Processed',
                            'Total Jobs Matched',
                            'Average Skills per Resume',
                            'Highest Skills Match %',
```

```
                        'Highest Skills Match %',
                        'Lowest Skills Match %'
                    ],
                    'Value': [
                        len(all_resume_data),
                        len(matches_df),
                        resume_df['Total Skills'].mean(),
                        matches_df['Skills Match'].max(),
                        matches_df['Skills Match'].min()
                    ]
                }
                summary_df = pd.DataFrame(summary_data)
                summary_df.to_excel(writer, sheet_name='Summary', index=False)

                # Format Summary sheet
                worksheet3 = writer.sheets['Summary']
                for col_num, value in enumerate(summary_df.columns.values):
                    worksheet3.write(0, col_num, value, header_format)
                    worksheet3.set_column(col_num, col_num, 25)

        except Exception as e:
            print(f"Error creating Excel file: {str(e)}")
            # Fallback to simple Excel writing
            with pd.ExcelWriter(excel_filename, engine='openpyxl') as writer:
                resume_df.to_excel(writer, sheet_name='Resume Summary', index=False)
                matches_df.to_excel(writer, sheet_name='Job Matches', index=False)
                summary_df.to_excel(writer, sheet_name='Summary', index=False)

        print(f"\nComprehensive analysis saved to {excel_filename}")

        # Clean up
        import shutil
        shutil.rmtree(temp_dir)

        # Display summary in a more readable format
        print("\nAnalysis Summary:")
        print("=" * 80)
        print(f"Total Resumes Processed: {len(all_resume_data)}")
        print(f"Total Jobs Matched: {len(matches_df)}")
        print(f"Average Skills per Resume: {resume_df['Total Skills'].mean():.2f}")
        print(f"Highest Skills Match: {matches_df['Skills Match'].max():.2f}%")
        print(f"Lowest Skills Match: {matches_df['Skills Match'].min():.2f}%")
        print("=" * 80)

        # Display resume details in a cleaner format
        print("\nResume Details:")
        print("=" * 80)
        for _, row in resume_df.iterrows():
            print(f"\nResume: {row['Resume Name']}")
            print(f"Skills Found: {row['Skills Found']}")
            print(f"Total Skills: {row['Total Skills']}")
            print(f"Processing Date: {row['Processing Date']}")
            print("-" * 80)

        # Display top job matches in a cleaner format
        print("\nTop Job Matches:")
        print("=" * 80)
        for _, row in matches_df.iterrows():
            print(f"\nJob Title: {row['Job Title']}")
            print(f"Company: {row['Company']}")
            print(f"Skills Match: {row['Skills Match']}%")
            print("-" * 80)

        return resume_df, matches_df, summary_df
    else:
        print("No valid PDF files found in the ZIP file")
        return None, None, None

# Example usage
from google.colab import files
uploaded = files.upload()  # This will prompt you to upload your ZIP file

# Get the filename of the uploaded ZIP file
zip_filename = list(uploaded.keys())[0]

# Process multiple resumes
resume_summary, job_matches, summary_stats = process_multiple_resumes(zip_filename)
```

Choose files Espin Shalo…ume.pdf.zip

- **Espin Shalo Resume.pdf.zip**(application/zip) - 101086 bytes, last modified: 28/04/2025 - 100% done

```
Saving Espin Shalo Resume.pdf.zip to Espin Shalo Resume.pdf (8).zip

Comprehensive analysis saved to comprehensive_resume_analysis_20250428_082418.xlsx

Analysis Summary:
================================================================================
Total Resumes Processed: 1
Total Jobs Matched: 5
Average Skills per Resume: 6.00
Highest Skills Match: 5.20%
Lowest Skills Match: 0.00%
================================================================================

Resume Details:
================================================================================

Resume: Espin Shalo Resume.pdf
Skills Found: leadership, project management, python, flutter, time management, sql
Total Skills: 6
Processing Date: 2025-04-28 08:24:18
--------------------------------------------------------------------------------

Top Job Matches:
================================================================================

Job Title: Data Scientist
Company: Data Analytics Inc
Skills Match: 5.2%
--------------------------------------------------------------------------------

Job Title: Software Engineer
Company: Tech Corp
Skills Match: 4.92%
--------------------------------------------------------------------------------

Job Title: DevOps Engineer
Company: Cloud Systems Inc
Skills Match: 2.93%
--------------------------------------------------------------------------------

Job Title: Full Stack Developer
Company: Web Solutions Ltd
Skills Match: 0.0%
--------------------------------------------------------------------------------

Job Title: Frontend Developer
Company: Creative Web Agency
Skills Match: 0.0%
--------------------------------------------------------------------------------
```