Amazon Review Scraper & Sentiment Analyzer

Q1: Walk me through how your script collects and processes Amazon product reviews.
A: The script prompts for a product code (ASIN), then iteratively fetches review pages using requests, custom headers, cookies, and proxies to simulate a real browser session. It parses each page with BeautifulSoup, extracting review text and date, and filters for reviews from the last five days. After scraping, it saves the data to CSV, then loads it for sentiment analysis using TextBlob, appending sentiment and confidence scores for each review[2].

Q2: How do you manage cookies, headers, and proxies to avoid detection and ensure reliable scraping?
A: I set realistic headers (including user-agent and referer), manage session cookies, and use a proxy service to rotate IPs. This helps bypass rate limits and geo-restrictions. For production, I would automate cookie refresh and proxy rotation to further reduce detection risk[2].

Q3: Describe your error handling and logging strategy during the scraping process.
A: Each extraction block is wrapped in try/except, logging full tracebacks for failed reviews without stopping the script. This ensures partial data is still collected, and errors can be analyzed and fixed later. I also print API endpoints and response codes for debugging[2].

Q4: How does your sentiment analysis pipeline work?
A: After scraping, I load the reviews into pandas and apply a function that uses TextBlob to compute sentiment polarity. Reviews are labeled as positive, negative, or neutral, with confidence based on polarity magnitude. TextBlob is lightweight and effective for short English texts, but for production, I'd consider transformer-based models for better accuracy on nuanced or multilingual data[2].

Q5: How do you handle non-string or malformed review entries before sentiment analysis?
A: The analysis function checks if the review is a string; if not, it defaults to 'neutral' with zero confidence. This prevents runtime errors and ensures the pipeline is robust to missing or malformed data[2].

Q6: Explain how you determine the confidence score for each sentiment label.
A: The confidence score is the absolute value of TextBlob's polarity (range 0 to 1). For more nuanced confidence, I could calibrate scores using a labeled dataset or

switch to models that provide probability estimates. I also review edge cases to refine the scoring approach[2].

Q7: What are the ethical and legal considerations when scraping and analyzing user-generated content from Amazon?
A: I ensure compliance with Amazon's terms of service, avoid scraping personal data, and respect robots.txt. For sentiment analysis, I anonymize data and use it only for aggregate insights, not for targeting individuals. I'm mindful of data privacy laws like GDPR and always disclose data sources in any reporting.

Q8: If you wanted to support sentiment analysis in multiple languages, how would you adapt your pipeline?
A: I would integrate language detection (e.g., langdetect) and use multilingual sentiment models from HuggingFace or Google Cloud NLP, which support a wide range of languages and provide more accurate sentiment classification for non-English reviews.

Q9: How would you automate and schedule large-scale review scraping and analysis for ongoing monitoring?
A: I'd containerize the script with Docker, schedule it with cron or a workflow orchestrator like Airflow, and store results in a database or cloud storage. For scalability, I'd parallelize requests and use cloud VMs or serverless functions, monitoring for failures and alerting on anomalies.

Q10: If you observe a high error rate or data gaps in your output, how do you diagnose and resolve the issue?
A: I review logs for failed requests or parsing errors, inspect the raw HTML for structural changes, and test selectors interactively. I also monitor for proxy/cookie failures and adjust scraping intervals or headers as needed. Regular code reviews and automated tests help maintain robustness[2].