Generate a complete Python program named **email\_analyzer.py** that implements a comprehensive email analysis platform with the following functionality:

1. **Domain Analysis:**
   * Analyze email sender domains (extract from sender\_email) to compute frequency and domain entropy.
   * Retrieve domain age using WHOIS.
   * Perform VirusTotal lookups to compute a threat score for domains.
2. **Email Content Analysis:**
   * Analyze the subject and body for scam indicators such as monetary phrases, urgency, suspicious links, and credential requests.
   * Calculate a comprehensive threat score by combining components from domain age, VirusTotal, content analysis, URL analysis, and subject action phrases.
   * Cluster emails using TF-IDF vectorization and KMeans; visualize clusters using 2D PCA.
3. **Pre-trained Classifier Integration:**
   * Load a pre-trained classifier and TF-IDF vectorizer (from files “pretrained\_classifier.joblib” and “pretrained\_vectorizer.joblib”) using joblib.
   * Use the classifier’s real-time predictions on incoming email text (subject and body) to adjust the threat score (e.g., if classified as “bad” or “spam,” increase threat score).
4. **Flask Web Interface & SQLite Backend:**
   * Provide a Flask-based web interface that supports user registration/login and CSV file upload for batch email analysis.
   * Generate detailed analysis reports and store them (with timestamps, filename, average threat score, etc.) in a SQLite database.
   * Include routes for viewing past reports.
5. **Adversarial Attack Stub:**
   * Include a stub function for future adversarial attack integration.
6. **Logging and Reporting:**
   * Log detailed messages throughout the analysis pipeline.
   * Generate and save a final report (as a text file and in the database).

Include necessary imports (e.g., flask, joblib, scikit-learn, nltk, aiosmtpd, pandas, matplotlib, seaborn, tldextract, whois) and inline comments for clarity. Ensure the code is self-contained and ready to run.