Data Preparation - Exam Questions

Objective Questions (MCQ/MSQ) - 20 Questions

1. Which jq command extracts all “name” fields from a JSON array?

A) `jq '.name'`
B) `jq '.[] | .name'`
C) `jq 'select(.name)'`
D) `jq 'map(.name)'`

Answer: B

2. In DuckDB, which function is used to read CSV files with automatic schema detection?

A) `read_csv()`
B) `read_csv_auto()`
C) `auto_csv()`
D) `csv_read()`

Answer: B

3. Which PIL/Pillow method maintains aspect ratio when resizing?

A) `resize()`
B) `scale()`
C) `thumbnail()`
D) `transform()`

Answer: C

4. What does the `ijson` library primarily help with?

A) Creating JSON files
B) Validating JSON syntax
C) Streaming large JSON files
D) Converting JSON to XML

Answer: C

5. Which Excel function splits text into multiple columns?

A) `SPLIT()`
B) `TEXTSPLIT()`
C) `DIVIDE()`
D) `SEPARATE()`

Answer: B

6. In FFmpeg, which parameter extracts audio without video?

A) `-an`
B) `-vn`
C) `-audio-only`
D) `-extract-audio`

Answer: B

7. Which OpenRefine feature helps merge similar text values?

A) Faceting
B) Clustering
C) Filtering
D) Sorting

Answer: B

8. What is the correct JMESPath query to filter items where population > 700000?

A) `locations[?info.population > 700000]`

B) `locations[?info.population > `700000`]`

C) `locations.filter(population > 700000)`

D) `locations[info.population > 700000]`

Answer: B

9. Which yt-dlp option downloads only audio?

A) `--audio-only`

B) `--extract-audio`

C) `--no-video`

D) `--audio-format`

Answer: B

10. In pandas, which method converts JSON strings in a column to normalized DataFrame?

A) `json.loads()`

B) `pd.json_normalize()`

C) `df.json_expand()`

D) `pd.read_json()`

Answer: B

11. Which ImageMagick command resizes all JPG files in a directory?

A) `convert *.jpg -resize 800x600`

B) `mogrify -resize 800x600 *.jpg`

C) `magick -resize 800x600 *.jpg`

D) `resize *.jpg 800x600`

Answer: B

12. What does the `-strip` option do in ImageMagick?

A) Removes image borders
B) Removes metadata
C) Removes transparency
D) Removes color profiles

Answer: B

13. Which Whisper model provides the best accuracy?

A) `tiny`
B) `base`
C) `medium`
D) `large-v3`

Answer: D

14. In DuckDB, which SQL function extracts JSON string values?

A) `json_get()`
B) `json_extract_string()`
C) `json_value()`
D) `get_json()`

Answer: B

15. Which shell command counts unique values in the first column?

A) `cut -f1 | sort | uniq -c`
B) `awk '{print $1}' | sort | uniq -c`
C) `cut -d' ' -f1 | sort | uniq -c`
D) All of the above

Answer: D

16. What is the primary advantage of Parquet over CSV?

A) Human readable

B) Smaller size and faster queries

C) Better compression only

D) Easier to edit

Answer: B

17. Which Python library is best for streaming JSON parsing?

A) `json`

B) `pandas`

C) `ijson`

D) `jsonlines`

Answer: C

18. In Excel, which function extracts text after a delimiter?

A) `TEXTAFTER()`

B) `RIGHT()`

C) `MID()`

D) `EXTRACT()`

Answer: A

19. Which FFmpeg option sets audio sample rate to 16kHz?

A) `-ar 16000`

B) `-rate 16000`

C) `-sample 16000`

D) `-freq 16000`

Answer: A

20. What does `COALESCE` function do in SQL?

A) Combines columns
B) Returns first non-NULL value
C) Counts NULL values
D) Removes duplicates

Answer: B

Subjective/Scenario Questions - 20 Questions

1. Tool Selection Scenario

You have a 50GB JSON log file that needs to be processed for analysis. The file contains nested objects and you need to extract specific fields while filtering based on conditions. Which approach would you choose and why? Compare at least 3 different tools/methods.

Answer: Use `ijson` for streaming processing to handle the large file size without memory issues. Alternative approaches: (1) DuckDB with `read_json_auto()` for SQL-based filtering and extraction, (2) `jq` for command-line processing with streaming, (3) Pandas with chunking for smaller subsets. `ijson` is preferred for memory efficiency with large files, while DuckDB offers better performance for complex queries.

2. Data Pipeline Design

Design a data preparation pipeline for processing daily CSV exports from multiple sources (varying schemas, encoding issues, missing values). Describe your approach including tools, error handling, and validation steps.

Answer: Pipeline: (1) Use DuckDB's `read_csv_auto()` with `ignore_errors=true` for initial ingestion, (2) OpenRefine for schema standardization and entity resolution, (3) Python/pandas for programmatic cleaning and validation, (4) Export to Parquet for downstream processing. Include data quality checks, logging, and fallback mechanisms for corrupted files.

3. Image Processing Strategy

You need to process 10,000 product images for an e-commerce site: resize to multiple dimensions, add watermarks, optimize for web, and extract metadata. Design an efficient processing workflow.

Answer: Use ImageMagick for batch processing with `mogrify` for resizing and `composite` for watermarks. Implement parallel processing with shell scripts or Python multiprocessing. Use `-strip` to remove metadata, optimize with appropriate quality settings, and organize output by size categories. Consider using PIL/Pillow for programmatic control and error handling.

4. Audio Transcription Workflow

Design a system to transcribe and analyze customer service calls (various languages, background noise, different speakers). Compare different transcription approaches and recommend the best strategy.

Answer: Multi-tier approach: (1) Use Whisper `medium` model for general transcription with language auto-detection, (2) Gemini API for intelligent summarization and key point extraction, (3) Preprocessing with FFmpeg for noise reduction and audio normalization. Consider speaker diarization and confidence scoring for quality control.

5. JSON Data Integration

You receive JSON data from 5 different APIs with varying structures. Some are nested, some are flat, and they need to be combined into a single analytical dataset. Describe your integration strategy.

Answer: Use JMESPath for standardized field extraction from each API, `pd.json_normalize()` for flattening nested structures, and DuckDB for joining and harmonizing the datasets. Implement schema mapping and validation rules. Consider using `ijson` for large API responses and maintain data lineage documentation.

6. Text Processing Challenge

Clean and standardize a dataset containing addresses, names, and phone numbers from multiple countries with inconsistent formatting. What tools and techniques would you use?

Answer: Use OpenRefine for interactive clustering and entity resolution, regex patterns for standardization, and pandas for programmatic cleaning. Implement country-specific validation rules, use fuzzy matching for address normalization, and apply consistent formatting standards. Document all transformation rules for reproducibility.

7. Performance Optimization

Your current data preparation pipeline takes 6 hours to process daily data. The bottleneck appears to be in CSV parsing and transformation. How would you optimize this?

Answer: Replace CSV with Parquet for intermediate storage, use DuckDB for columnar processing instead of pandas, implement parallel processing for independent transformations, and optimize I/O with appropriate chunk sizes. Consider incremental processing for unchanged data and caching intermediate results.

8. Error Handling Strategy

Design a robust error handling system for a data preparation pipeline that processes files from external vendors with unpredictable quality issues.

Answer: Implement multi-level validation: (1) File format validation, (2) Schema compliance checking, (3) Data quality rules, (4) Business logic validation. Use `try-except` blocks with specific error types, logging with structured formats, and fallback mechanisms. Maintain error catalogs and automated alerting for critical failures.

9. Cross-Platform Compatibility

Your data preparation scripts need to work across Windows, macOS, and Linux environments. What considerations and tools would you use to ensure compatibility?

Answer: Use Python with `pathlib` for cross-platform paths, Docker containers for environment consistency, and shell scripts with POSIX compliance. Avoid platform-specific tools, use UTF-8 encoding consistently, and test on all target platforms. Document environment requirements and provide setup scripts.

10. Real-time Processing

Adapt your batch data preparation pipeline to handle real-time streaming data. What changes would you make to your tools and architecture?

Answer: Replace batch tools with streaming alternatives: use Apache Kafka for data ingestion, streaming JSON parsers, and micro-batch processing with tools like Apache Spark Streaming. Implement stateful processing for aggregations and maintain data consistency with event sourcing patterns.

11. Data Quality Assessment

Create a comprehensive data quality framework for incoming datasets. What metrics would you track and how would you implement automated quality checks?

Answer: Track completeness (null rates), validity (format compliance), consistency (cross-field validation), accuracy (business rule compliance), and timeliness (freshness checks). Implement using Great Expectations or custom pandas functions with automated reporting and threshold-based alerting.

12. Legacy System Integration

You need to extract and prepare data from legacy systems that export in proprietary formats. Design an integration strategy.

Answer: Develop format-specific parsers, use intermediate standardized formats (JSON/Parquet), implement robust error handling for format variations, and maintain backward compatibility. Consider using specialized libraries or reverse-engineering tools for proprietary formats.

13. Scalability Planning

Your data preparation needs are growing from GB to TB scale. How would you redesign your current pandas-based pipeline?

Answer: Migrate to distributed processing with Dask or PySpark, use columnar storage (Parquet), implement data partitioning strategies, and leverage cloud-based processing services. Consider using DuckDB for single-machine scaling before moving to distributed systems.

14. Compliance and Security

Design data preparation workflows that comply with GDPR and handle PII data securely. What measures would you implement?

Answer: Implement data classification, PII detection and masking, audit logging, access controls, and data retention policies. Use encryption for data at rest and in transit, anonymization techniques, and maintain data lineage for compliance reporting.

15. Multi-format Integration

Process and combine data from CSV, JSON, XML, and database sources into a unified format. Design your integration approach.

Answer: Use DuckDB as a universal query engine for different formats, implement ETL pipelines with format-specific parsers, standardize on common schema, and use Parquet as the unified output format. Maintain metadata about source formats and transformation rules.

16. Automated Pipeline Monitoring

Design a monitoring system for your data preparation pipelines that can detect anomalies and performance issues automatically.

Answer: Implement metrics collection (processing time, error rates, data volume), statistical anomaly detection, automated alerting with escalation rules, and dashboard visualization. Use tools like Prometheus/Grafana or cloud monitoring services with custom metrics.

17. Version Control Strategy

Implement version control for both your data preparation code and the processed datasets. How would you handle schema evolution?

Answer: Use Git for code versioning, data versioning tools like DVC or Delta Lake for datasets, schema registries for evolution tracking, and semantic versioning for breaking changes. Implement backward compatibility checks and migration scripts.

18. Testing Framework

Design a comprehensive testing strategy for data preparation pipelines including unit tests, integration tests, and data validation tests.

Answer: Implement unit tests for individual functions, integration tests with sample datasets, property-based testing for edge cases, and data contract testing for schema validation. Use pytest with data-specific assertions and maintain test data fixtures.

19. Disaster Recovery

Design a disaster recovery plan for your data preparation infrastructure. What backup and recovery strategies would you implement?

Answer: Implement automated backups of code and data, maintain infrastructure as code for quick recovery, use geographically distributed storage, and test recovery procedures regularly. Include rollback mechanisms and point-in-time recovery capabilities.

20. Cost Optimization

Your cloud-based data preparation costs are increasing rapidly. Analyze potential cost optimization strategies while maintaining performance and reliability.

Answer: Implement data lifecycle management, use spot instances for non-critical processing, optimize storage tiers, implement caching strategies, and right-size compute resources. Monitor usage patterns and implement auto-scaling with cost controls.