Development Tools - Exam Questions

Objective Questions (MCQ/MSQ) - 20 Questions

1. Which VS Code shortcut opens the command palette?

A) `Ctrl+Shift+P`
B) `Ctrl+P`
C) `Ctrl+Shift+O`
D) `Ctrl+K`

Answer: A

2. What does the `uv` tool primarily manage in Python projects?

A) Virtual environments only
B) Package dependencies and virtual environments
C) Code formatting
D) Testing frameworks

Answer: B

3. Which `npx` command creates a new React application?

A) `npx react-app create`
B) `npx create-react-app myapp`
C) `npx new-react-app`
D) `npx react create myapp`

Answer: B

4. In Git, which command shows the commit history?

A) `git history`

B) `git log`

C) `git show`

D) `git commits`

Answer: B

5. Which CSS selector targets elements with a specific class?

A) `#classname`

B) `.classname`

C) `@classname`

D) `*classname`

Answer: B

6. What does `JSON.parse()` do in JavaScript?

A) Converts object to JSON string

B) Converts JSON string to object

C) Validates JSON syntax

D) Formats JSON output

Answer: B

7. Which bash command displays the current working directory?

A) `cwd`

B) `pwd`

C) `dir`

D) `path`

Answer: B

8. In Excel, which function looks up values in a table?

A) `VLOOKUP()`

B) `XLOOKUP()`

C) `LOOKUP()`
D) All of the above

Answer: D

9. Which SQLite command shows all tables in a database?

A) `.tables`
B) `.show tables`
C) `SHOW TABLES`
D) `LIST TABLES`

Answer: A

10. What does GitHub Copilot primarily provide?

A) Version control
B) AI-powered code suggestions
C) Project management
D) Code deployment

Answer: B

11. Which Unicode encoding is most commonly used for web content?

A) UTF-16
B) UTF-32
C) UTF-8
D) ASCII

Answer: C

12. In browser DevTools, which panel shows network requests?

A) Console
B) Elements

C) Network
D) Sources

Answer: C

13. Which Git command creates a new branch?

A) `git branch newbranch`
B) `git checkout -b newbranch`
C) `git create newbranch`
D) Both A and B

Answer: D

14. What does the `llm` CLI tool primarily interface with?

A) Local databases
B) Large Language Models
C) Linux systems
D) Log files

Answer: B

15. Which Excel function splits text into columns?

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

Answer: B

16. In JSON, which data types are supported?

A) String, Number, Boolean
B) Object, Array, null
C) Both A and B
D) Only strings and numbers

Answer: C

17. Which bash operator redirects output to a file?

A) `|`
B) `>`
C) `<`
D) `&`

Answer: B

18. What does `git clone` do?

A) Creates a copy of a local repository
B) Downloads a remote repository
C) Duplicates a branch
D) Copies files between directories

Answer: B

19. Which VS Code extension is essential for Python development?

A) Python
B) Pylance
C) Python Debugger
D) All of the above

Answer: D

20. In SQLite, which command imports CSV data?

A) `.import file.csv table`
B) `IMPORT CSV file.csv`
C) `LOAD DATA file.csv`
D) `.load file.csv`

Answer: A

Subjective/Scenario Questions - 20 Questions

1. Development Environment Setup

Design a complete development environment for a data science team working on Python projects with version control, dependency management, and code quality tools. What tools would you choose and how would you configure them?

Answer: Use **VS Code** with Python extensions, **uv** for dependency management and virtual environments, **Git** with conventional commits, **pre-commit hooks** for code quality (black, flake8, mypy), **GitHub** for collaboration, and **Docker** for environment consistency. Configure workspace settings, establish coding standards, and implement CI/CD pipelines.

2. Version Control Strategy

Your team of 10 developers needs a Git workflow for a data science project with notebooks, datasets, and code. Design a branching strategy and collaboration workflow that handles both code and data versioning.

Answer: Implement **Git Flow** with feature branches, use **DVC** for data versioning, **nbstripout** for notebook cleaning, and **Git LFS** for large files. Establish branch protection rules, require pull request reviews, use semantic versioning, and implement automated testing. Create separate workflows for experimental notebooks vs. production code.

3. Code Quality Framework

Establish a comprehensive code quality framework for a Python data science project. What tools, standards, and processes would you implement?

Answer: Implement **black** for formatting, **flake8** for linting, **mypy** for type checking, **pytest** for testing, **pre-commit** for automation. Establish coding standards, documentation requirements (docstrings), test coverage thresholds, and code review processes. Use **GitHub Actions** for CI/CD and quality gates.

4. Cross-Platform Development

Design a development setup that works consistently across Windows, macOS, and Linux for a distributed team. What challenges would you address and how?

Answer: Use **Docker** for environment consistency, **VS Code** with Remote Development extensions, **uv** for cross-platform Python management, and **Git** with proper line ending configuration. Address path differences with **pathlib**, use environment variables for configuration, and provide setup scripts for each platform.

5. Database Integration Strategy

Design a workflow for data scientists to work with multiple databases (SQLite, PostgreSQL, MongoDB) while maintaining code portability and version control. What tools and patterns would you use?

Answer: Use **SQLAlchemy** for database abstraction, **environment variables** for connection strings, **database migrations** with Alembic, and **Docker Compose** for local development. Implement connection pooling, query optimization, and data access patterns. Use **DBeaver** or similar tools for database exploration.

6. Automated Testing for Data Science

Create a testing strategy for data science projects that includes data validation, model testing, and pipeline testing. What types of tests would you implement?

Answer: Implement **unit tests** for functions, **integration tests** for pipelines, **data validation tests** with Great Expectations, **model performance tests**, and **regression tests** for model outputs. Use **pytest** with fixtures, **property-based testing** with Hypothesis, and **continuous testing** in CI/CD pipelines.

7. Documentation and Knowledge Management

Design a documentation system for a data science team that captures code documentation, data schemas, model specifications, and project knowledge. What tools and processes would you use?

Answer: Use **Sphinx** for code documentation, **MkDocs** for project documentation, **Jupyter notebooks** for analysis documentation, **data catalogs** for schema documentation, and **wikis** for knowledge sharing. Implement automated documentation generation, version control for docs, and regular review processes.

8. Performance Monitoring and Debugging

Establish monitoring and debugging practices for data science applications. What tools would you use to identify performance bottlenecks and debug issues?

Answer: Use **profiling tools** (cProfile, line_profiler), **memory monitoring** (memory_profiler), **logging frameworks** (loguru, structlog), **APM tools** (New Relic, DataDog), and **debugging tools** (pdb, VS Code debugger). Implement performance benchmarking, error tracking, and alerting systems.

9. Dependency Management Strategy

Design a dependency management strategy for a large data science project with multiple teams and environments. How would you handle version conflicts and ensure reproducibility?

Answer: Use **uv** for Python dependencies with lock files, **Docker** for system dependencies, **conda** for scientific packages when needed, and **pip-tools** for dependency resolution. Implement dependency scanning for security, regular updates with testing, and environment isolation strategies.

10. Code Review Process

Establish a code review process specifically tailored for data science projects. What guidelines and tools would you implement?

Answer: Create **review checklists** for data science code, use **GitHub/GitLab** for pull requests, implement **automated checks** (linting, testing), establish **domain expertise requirements** for reviews, and create **templates** for different types of changes (models, data processing, analysis). Include data validation and reproducibility checks.

11. Security and Compliance

Design security practices for a data science development environment handling sensitive data. What measures would you implement?

Answer: Implement **secrets management** (HashiCorp Vault, AWS Secrets Manager), **access controls** (RBAC, MFA), **data encryption** at rest and in transit, **audit logging**, **secure coding practices**, and **compliance frameworks** (SOC 2, GDPR). Use **security scanning** tools and regular security assessments.

12. Collaboration Tools Integration

Design an integrated toolchain that connects development tools with communication and project management platforms. What integrations would you implement?

Answer: Integrate **Git** with **Slack/Teams** for notifications, **Jira/Asana** for issue tracking, **GitHub Actions** with project management tools, **VS Code** with collaboration extensions, and **Jupyter** with sharing platforms. Implement automated status updates and progress tracking.

13. Local Development Optimization

Optimize local development environments for data scientists working with large datasets and computationally intensive tasks. What strategies would you implement?

Answer: Use **SSD storage** for fast I/O, **sufficient RAM** for in-memory processing, **GPU acceleration** for ML tasks, **local caching** strategies, **data sampling** for development, **incremental processing**, and **efficient development workflows**. Implement resource monitoring and optimization guidelines.

14. Remote Development Setup

Design a remote development solution for data scientists who need access to powerful computing resources and large datasets. What architecture would you implement?

Answer: Use **cloud development environments** (GitHub Codespaces, AWS Cloud9), **remote VS Code** connections, **JupyterHub** deployments, **VPN access** to resources, **shared storage** solutions, and **container-based** development. Implement security controls and resource management.

15. Continuous Integration for Data Science

Create a CI/CD pipeline specifically designed for data science projects. What stages and checks would you include?

Answer: Include **code quality checks**, **data validation**, **model testing**, **performance benchmarking**, **security scanning**, **documentation generation**, and **deployment automation**. Use **GitHub Actions/GitLab CI** with custom runners, implement **parallel execution**, and create **environment-specific** deployments.

16. Tool Standardization Strategy

Establish tool standardization across a data science organization while allowing flexibility for specific use cases. How would you balance standardization with innovation?

Answer: Create **approved tool lists** with justification processes for exceptions, **standard configurations** with customization options, **training programs** for standard tools, **evaluation frameworks** for new tools, and **migration strategies** for tool changes. Implement governance processes and regular reviews.

17. Backup and Recovery

Design backup and recovery strategies for development environments and code repositories. What would you include in your disaster recovery plan?

Answer: Implement **automated backups** of repositories, **distributed version control**, **cloud storage** for critical data, **environment recreation** scripts, **documentation backup**, and **recovery testing**. Use **multiple backup locations**, **versioned backups**, and **recovery time objectives**.

18. Onboarding Process

Create an onboarding process for new data scientists joining your development environment. What training and setup procedures would you establish?

Answer: Develop **setup automation** scripts, **training materials** for tools and processes, **mentorship programs**, **hands-on exercises**, **documentation walkthroughs**, and **gradual responsibility increase**. Create **checklists**, **video tutorials**, and **feedback mechanisms** for continuous improvement.

19. Performance Benchmarking

Establish performance benchmarking for development tools and processes. How would you measure and improve developer productivity?

Answer: Track **build times**, **test execution times**, **deployment frequency**, **code review turnaround**, **bug resolution time**, and **developer satisfaction**. Use **metrics dashboards**, **regular surveys**, **performance profiling**, and **optimization initiatives**. Implement **continuous improvement** processes.

20. Legacy System Integration

Design strategies for integrating modern development tools with legacy systems and databases. What approaches would you use to modernize gradually?

Answer: Implement **API wrappers** for legacy systems, **data extraction** pipelines, **gradual migration** strategies, **compatibility layers**, **documentation** of legacy systems, and **risk assessment** for changes. Use **strangler fig pattern**, **feature toggles**, and **parallel running** during transitions.