Study Guide: Data Collection and Governance Module 2

DSC 208R - Data Management for Analytics

Section 1: Overview

This study guide covers the first section of Module 2: Data Collection and Governance. It focuses on the data science lifecycle, with particular emphasis on the sourcing stage where raw data is transformed into analytics-ready datasets. Understanding this process is fundamental to effective data science practice, as research consistently shows that data scientists spend the majority of their time on data collection and preparation activities.

1 Section 1: Overview

1.1 Overview

By the end of this section, you should be able to:

- Describe the complete lifecycle of real-world data science projects
- Explain why data scientists spend most of their time on data preparation
- Define the sourcing stage and its importance in the data science pipeline
- Identify the key challenges that make data sourcing difficult
- Understand the data-centric AI approach to data science
- Recognize the four high-level activities in the sourcing stage

1.2 The Data Science Lifecycle

Data Science Lifecycle Steps

1. Data acquisition
2. Data preparation
3. Data cleaning
4. Feature Engineering
5. Model Selection
6. Training & Inference
7. Serving
8. Monitoring

1.3 Time Allocation in Data Science

Research Findings

Multiple industry surveys consistently show that data scientists spend the majority of their time on data-related tasks rather than on model building:
Key Sources:

- CrowdFlower Data Science Report 2016
- Kaggle State of ML and Data Science Survey 2018
- IDC-Alteryx State of Data Science and Analytics Report 2019

These reports highlight that activities like data collection, cleaning, and organization consume significantly more time than algorithm development or model training.

1.4 The Sourcing Stage

1.4.1 Definition and Context

Key Insight

Data science applications do not exist in a vacuum. They work with the data-generating process and prediction application. The sourcing stage is where you transform raw datasets into "analytics/ML-ready" datasets.
The rough end point of sourcing is when data is prepared for:

- SQL analytics for Business Intelligence
- Data/feature engineering for ML/AI analytics

1.5 Challenges in Data Sourcing

1. **Heterogeneity**: Diverse data modalities, file formats, and sources
2. **Access constraints**: Limited data availability or permissions
3. **Application diversity**: Various prediction applications with different requirements
4. **Data volatility**: Unpredictable and continual edits to datasets
5. **Data quality issues**: Messy, incomplete, ambiguous, or erroneous data
6. **Scale**: Managing large volumes of data
7. **Governance**: Poor data management practices in organizations

1.6 The Data-Centric AI Movement

Data-Centric AI

The Data-Centric AI movement (https://datacentricai.org/) represents a shift in focus from model architecture to data quality. This approach recognizes that improving data quality often yields better results than developing more sophisticated algorithms.
Key principles include:

- Systematic approaches to data improvement
- Consistent data labeling
- Comprehensive data documentation
- Iterative data refinement

1.7 Sourcing Stage Activities

Sourcing Process Flow

Raw data sources/repos → Acquiring → Reorganizing → Cleaning → Data/Feature Engineering for Analytics/ML → Analytics Results
Note: Labeling & Amplification may be required in some cases

1.8 Four High-Level Activities

- **Acquiring**: Obtaining data from various sources
- **Reorganizing**: Transforming data into usable formats
- **Cleaning**: Correcting errors and handling missing values
- **Data/Feature Engineering**: Creating features for analysis

1.9 Study Questions

1. Why do data scientists spend more time on data preparation than on model building?

Solution

Data scientists spend more time on data preparation because real-world data is typically messy, incomplete, and not immediately usable for analysis. Industry surveys consistently show that activities like data collection, cleaning, and organization consume the majority of a data scientist's time. This is necessary because high-quality data is fundamental to accurate models and insights. Even the most sophisticated algorithms will produce poor results with low-quality data, making thorough preparation an essential investment.

2. What is the sourcing stage in the data science lifecycle, and why is it important?

Solution

The sourcing stage is where raw datasets are transformed into "analytics/ML-ready" datasets. It's important because it creates the foundation for all subsequent analysis. This stage includes acquiring data from various sources, reorganizing it into usable formats, cleaning it to address quality issues, and performing initial feature engineering. The sourcing stage is critical because the quality of data preparation directly impacts the effectiveness of models and analyses. Without proper sourcing, even the most sophisticated algorithms will fail to produce valuable insights.

3. What are the main challenges in the sourcing stage of the data science lifecycle?

Solution

The main challenges in the sourcing stage include:

- Heterogeneity of data modalities, file formats, and sources
- Data access and availability constraints
- Diverse requirements for different prediction applications
- Unpredictable and continual edits to datasets
- Messy, incomplete, ambiguous, or erroneous data
- Managing large volumes of data efficiently
- Poor data governance practices in organizations

These challenges explain why the sourcing stage consumes so much time and requires careful attention to ensure quality results in subsequent analysis stages.

4. How does the data-centric AI approach differ from traditional AI development?

Solution

The data-centric AI approach shifts focus from model architecture to data quality. Traditional AI development often emphasizes creating more sophisticated algorithms and neural network architectures. In contrast, the data-centric approach recognizes that improving data quality (through better collection, cleaning, labeling, and documentation) often yields better results than algorithmic improvements. This approach promotes systematic data improvement, consistent labeling standards, comprehensive documentation, and iterative refinement of datasets rather than just iterative refinement of models.

1.10 Additional Resources

- CrowdFlower Data Science Report 2016

- Data-Centric AI Movement

Key Takeaway

The sourcing stage is the foundation of successful data science projects. Understanding the complete data science lifecycle and the challenges of data sourcing is essential for effective practice. Data scientists spend the majority of their time on data preparation activities, highlighting the critical importance of mastering the sourcing process. The data-centric AI movement further emphasizes that data quality often matters more than algorithm sophistication.

Section 2: Data Organization

This study guide covers the second section of Module 2: Data Collection and Governance, focusing on data organization and file formats. It explores the relationship between different data structures and their file representations, with a focus on structured, semi-structured, and unstructured data formats. Understanding these concepts is crucial for effective data acquisition, storage, and processing in the data science lifecycle.

2 Section 2: Data Organization

2.1 Learning Objectives

By the end of this section, you should be able to:

- Identify different data modalities in modern data-intensive applications
- Understand the relationship between files, file formats, and databases
- Compare and contrast different structured data models (Relations, Matrices, DataFrames)
- Explain the characteristics of semi-structured data formats
- Evaluate tradeoffs between different file formats (e.g., Parquet vs. CSV/JSON)
- Recognize how data lakes organize and store different types of data

2.2 Data Modalities in Modern Applications

Data Modalities

Modern data-intensive applications typically work with multiple data modalities:

- Structured data (e.g., relational data)
- Sequence data (e.g., IoT time series)
- Semi-structured data (e.g., JSON logs)
- Graph-structured data (e.g., social media)
- Text files, documents (e.g., reviews)
- Multimedia data (images, audio, video, etc.)
- Multimodal files (e.g., PDFs, notebooks)

2.3 Fundamental Concepts

2.3.1 Files and File Formats

- **File**: A persistent sequence of bytes that stores a logically coherent digital object for an application
- **File Format**: An application-specific standard that dictates how to interpret and process a file's bytes
  - Hundreds of file formats exist (e.g., TXT, DOC, GIF, MPEG)
  - Vary by data models/types, domain-specific requirements, etc.
- **Metadata**: Summary or organizing information about file content (payload) stored with the file itself; format dependent
- **Directory**: A cataloging structure with references to files and/or other directories
  - Treated as a special kind of file

2.4 Databases and Data Models

Key Insight

- **Database**: An organized collection of interrelated data
- **Data Model**: An abstract model that defines the organization of data in a formal, mathematically precise way
  - Examples: Relations, XMLs, Matrices, DataFrames
- **Every database is just an abstraction on top of data files!**
  - Logical level: Data model for higher-level reasoning
  - Physical level: How bytes are layered on top of files
- All data systems (RDBMSs, Spark, TensorFlow, etc.) are software systems that manipulate data files under the hood

2.5 Structured Data Models

2.6 Relations

- Form of data with regular structure
- Implemented in Relational Databases
- Most RDBMSs and Spark serialize a relation as binary file(s), often compressed

2.7 Relational File Formats

- Different RDBMSs and Spark/HDFS-based tools serialize relation/tabular data in different binary formats, often compressed
- One file per relation
- Data layout can be row-oriented vs. columnar (e.g., ORC, Parquet) vs. hybrid formats
- RDBMS vendor-specific vs. open Apache formats

- Parquet becoming especially popular

2.8 Other Structured Data Models

- **DataFrame**: Form of data with regular substructure
- **Matrix**: Mathematical structure of rows and columns
- **Tensor**: Multi-dimensional array
  - Typically serialized as restricted ASCII text file (TSV, CSV, etc.)
  - Matrix/tensor can also use binary formats
  - Can layer on Relations too
- **Sequence** (Includes Time-series):
  - Can layer on Relations, Matrices, or DataFrames, or be treated as first-class data model
  - Inherits flexibility in file formats (text, binary, etc.)

Comparing Structured Data Models

What is the difference between Relation, Matrix, and DataFrame?

- **Ordering**: Matrix and DataFrame have row/col numbers; Relation is orderless on both axes
- **Schema Flexibility**: Matrix cells are numbers. Relation tuples conform to pre-defined schema. DataFrame has no pre-defined schema but all rows/cols can have names; col cells can be mixed types
- **Transpose**: Supported by Matrix & DataFrame, not by Relations

2.9 Semi-structured Data Models

- Form of data with less regular/more flexible substructure than structured data

- **Tree-Structured**:
  - Typically serialized as restricted ASCII text file with formatting as JSON, YML, XML, etc.
  - Some data systems also offer binary file formats
  - Can layer on Relations too
- **Graph-Structured**:
  - Typically serialized with JSON or similar textual formats
  - Some data systems also offer binary file formats
  - Again, can layer on Relations too

2.10 Data Lakes and File Format Tradeoffs

2.11 Data Lake Concept

- **Data "Lake"**: Loose coupling of data file format for storage and data/query processing stack (vs. RDBMS's tight coupling)
- Common pattern: JSON for raw data; Parquet for processed data

Parquet vs. Text-Based Files Tradeoffs

Pros of Parquet:

- **Less storage**: Stores in compressed form; can be much smaller (even 10x); lowers read latency
- **Column pruning**: Enables app to read only columns needed to DRAM; even lower query latency
- **Schema on file**: Rich metadata, stats inside format itself
- **Complex types**: Can store them in a column

Cons of Parquet:

- **Human-readability**: Cannot open with text apps directly
- **Mutability**: Parquet is immutable/read-only; no in-place edits
- **Decompression/Deserialization overhead**: Depends on application tool; can go either way

Adoption in practice: CSV/JSON support more pervasive but Parquet is catching up

2.12 Other Common File Formats

- **Text File** (aka plaintext): Human-readable ASCII characters
- **Multimedia files**: Encode tensors and/or time-series of signals
  - Myriad binary formats, typically with (lossy) compression
  - Examples: WAV for audio, MP4 for video, etc.
- **Docs/Multimodal File**: Myriad app-specific rich binary formats

2.13 Study Questions

1. What is the relationship between databases and files?

Solution

Databases are abstractions built on top of files. At the physical level, all database systems (whether relational, NoSQL, or specialized systems) ultimately store and manipulate data as files on disk. Databases provide logical data models and operations that hide the complexity of file management, offering features like indexing, transactions, and query optimization. The database management system handles the translation between the logical data model that users interact with and the physical storage of bytes in files. This relationship is fundamental to understanding how data systems work under the hood.

2. Compare and contrast Relations, Matrices, and DataFrames.

Solution

Relations, Matrices, and DataFrames differ in several key aspects:

Ordering:

- Relations: No inherent ordering of rows or columns
- Matrices and DataFrames: Have explicit row/column numbers and ordering

Schema Flexibility:

- Relations: Tuples conform to a pre-defined schema
- Matrices: Cells are typically numbers of the same type
- DataFrames: No pre-defined schema; all rows/columns can have names; column cells can be of mixed types

Operations:

- Relations: Support relational algebra operations
- Matrices: Support mathematical operations like transpose
- DataFrames: Support both data manipulation and transpose operations

These differences make each model suitable for different types of data and analytical tasks.

3. Why might you choose Parquet over CSV for storing large datasets?

Solution

Parquet offers several advantages over CSV for large datasets:

- **Storage efficiency:** Parquet uses compression that can reduce file size by up to 10x compared to CSV, saving storage costs and reducing I/O time.
- **Column pruning:** Parquet's columnar format allows applications to read only the specific columns needed for a query, rather than loading the entire dataset, significantly improving performance for queries that access only a subset of columns.
- **Schema enforcement:** Parquet stores schema information with the data, ensuring consistency and reducing errors when reading the data.
- **Rich metadata:** Parquet files contain statistics and other metadata that query engines can use to optimize execution plans.
- **Support for complex types:** Parquet can efficiently store nested structures, arrays, and other complex data types that would be difficult to represent in CSV.

These advantages make Parquet particularly suitable for analytical workloads on large datasets, especially in data lake environments.

4. What are the key differences between structured and semi-structured data?

Solution

Structured and semi-structured data differ in several important ways:

Structured Data:

- Has a rigid, predefined schema
- Organized in a regular, predictable format (e.g., tables with rows and columns)
- All records follow the same format and contain the same fields
- Examples: relational databases, CSV files, matrices
- Easier to query using languages like SQL

Semi-structured Data:

- Has a flexible, self-describing schema
- Contains tags or markers to separate elements and enforce hierarchies
- Records may have different fields or structures
- Examples: JSON, XML, YAML, graph data
- Requires specialized query languages or parsers
- Better for representing complex, nested, or variable data

The key distinction is that structured data follows a strict schema where every record has the same structure, while semi-structured data allows for flexibility in the structure of individual records while still maintaining some organizational principles.

5. Explain the concept of a data lake and how it differs from traditional databases.

Solution

A data lake is a storage repository that holds a vast amount of raw data in its native format until needed. It differs from traditional databases in several key ways:

Data Lakes:

- Store data in its raw, original format (schema-on-read)
- Loosely couple the storage format from the processing system
- Support multiple data types (structured, semi-structured, unstructured)
- Typically use object storage (e.g., S3, HDFS)
- Often use file formats like Parquet for processed data and JSON/CSV for raw data
- Optimized for flexibility and analytical processing

Traditional Databases:

- Require data to conform to a predefined schema (schema-on-write)
- Tightly couple storage and processing
- Primarily support structured data
- Use specialized storage engines
- Typically use proprietary file formats
- Optimized for transactional processing and data integrity

The fundamental difference is that data lakes prioritize flexibility and scalability for diverse data types, while traditional databases prioritize structure, consistency, and transaction support.

2.14 Additional Resources

- What is Parquet? - Databricks
- Preventing the Death of the DataFrame
- Future of Data Lakes - CIDR 2021 Paper

Key Takeaway

Understanding data organization and file formats is fundamental to effective data management in data science. Different data modalities require different storage approaches, and the choice of data model and file format has significant implications for storage efficiency, query performance, and analytical capabilities. As data scientists spend the majority of their time on data preparation activities, mastering these concepts can substantially improve workflow efficiency and analytical outcomes.

Section 3: Data Acquisition

This study guide covers the third section of Module 2: Data Collection and Governance, focusing on data acquisition. This section explores how to obtain data from various sources, the challenges involved in data acquisition, and the concept of dataset discovery. Understanding effective data acquisition strategies is crucial as it forms the first step in the sourcing stage of the data science lifecycle.

3 Section 3: Data Acquisition

3.1 Learning Objectives

By the end of this section, you should be able to:

- Identify different types of data sources and their access methods
- Understand the challenges in acquiring data from diverse sources
- Apply strategies to mitigate common data acquisition challenges
- Explain the concept of dataset discovery and its importance
- Recognize how data acquisition fits into the broader data science workflow

3.2 Data Acquisition in the Sourcing Stage

Sourcing Process Flow

Raw data sources/repos → Acquiring → Reorganizing → Cleaning → Data/Feature Engineering for Analytics/ML → Analytics Results
Note: Labeling & Amplification may be required in some cases

3.3 Types of Data Sources

Data Sources and Access Methods

Different sources have different "query languages" and/or APIs to acquire data:

- **Structured data**: Typically managed by RDBMSs; queried using SQL
- **Semistructured data**: Exported from key-value stores (e.g., MongoDB)
- **Graph data**: Typically managed by graph DBMSs such as Neo4j
- **JSON logs, text files, multimedia, etc.**: Typically just files on S3, HDFS, etc.

3.4 Real-World Examples of Data Acquisition

3.5 Recommendation Systems

Example: Recommendation System (e.g., Netflix)

Prediction Application: Identify top movies to display for user
Data Sources Required:

- User data and past click logs
- Movie data
- Movie images

3.6 Social Media Analytics

Example: Social Media Analytics

Prediction Application: Predicts which tweets will go viral
Data Sources Required:

- Tweets as JSON
- Structured metadata
- Graph data
- Entity Dictionaries

3.7 Challenges in Data Acquisition

Acquisition Challenges and Mitigation Strategies

Challenges:

- Different sources have different "query languages" and/or APIs to acquire data
- Heterogeneity of data sources and modalities
- Access control and authentication issues
- Volume management
- Scale issues
- Manual errors in data collection

Mitigation Strategies:

- **For heterogeneity**: Critically assess if you really need all data sources/modalities
- **For access control**: Learn organization's data security and authentication policies
- **For volume**: Evaluate if you really need all data
- **For scale**: Avoid copying files one by one
- **For manual errors**: Use automated workflow tools such as AirFlow

3.8 Dataset Discovery

3.9 Concept and Importance

Dataset Discovery

Some organizations have built "data discovery" tools to help ML users find relevant datasets. These tools aim to make it easier to locate and access the most appropriate data for specific analytical tasks.
Key Aspects:

- Goal: Make it easier to find relevant datasets
- Approach: Relevance ranking over schemas/metadata
- Metadata standards: schema.org/Dataset
- Augmentation potential: Tabular datasets especially amenable for augmentation
- Join suggestions: Foreign keys (FK) implicitly suggest possible joins

3.10 Example: Google GOODS

Example: Google GOODS

GOODS (Google Dataset Search) is an internal system at Google that:

- Catalogs billions of tables within Google
- Extracts schema from files
- Assigns versions and owners
- Provides search functionality and dashboards

This system demonstrates how large organizations can address dataset discovery challenges at scale.

3.11 Study Questions

1. Why is data acquisition considered a critical first step in the data science lifecycle?

   Solution

   Data acquisition is the critical first step in the data science lifecycle because it determines what raw material is available for all subsequent steps. Without proper data acquisition, the entire data science process is compromised. Effective acquisition ensures that the right data, from the right sources, in the right formats is available for processing. Since different data sources require different query languages and APIs, this step requires careful planning and understanding of the various data repositories and their access methods.

2. How do the data acquisition requirements differ for a recommendation system versus social media analytics?

Solution

The data acquisition requirements differ significantly between these applications:

For recommendation systems (e.g., Netflix):

- Requires user data and past click logs (likely from relational databases)
- Needs movie metadata (structured data)
- Includes movie images (multimedia files)
- Focus is on user-item interactions and item properties

For social media analytics:

- Requires tweets as JSON (semi-structured data)
- Needs structured metadata
- Includes graph data (relationships between users)
- Uses entity dictionaries
- Focus is on content, network structure, and temporal patterns

These differences highlight why data acquisition strategies must be tailored to the specific application and data sources involved.

3. What are the main challenges in acquiring data from heterogeneous sources, and how can they be addressed?

Solution

Main challenges in acquiring data from heterogeneous sources include:

- **Different query languages and APIs**: Each source may require unique access methods
- **Heterogeneity of data formats**: Structured, semi-structured, and unstructured data require different handling
- **Access control issues**: Navigating various authentication systems
- **Volume management**: Determining what subset of data is actually needed
- **Scale challenges**: Inefficient file-by-file copying
- **Manual errors**: Risk of human error in data collection processes

These challenges can be addressed through:

- Critical assessment of which data sources are truly necessary
- Learning organization's data security policies in advance
- Evaluating if all data is needed or if samples would suffice
- Using automated workflow tools like Apache Airflow
- Implementing systematic approaches rather than ad-hoc solutions

4. Explain the concept of dataset discovery and why it's important in large organizations.

Solution

Dataset discovery refers to the process of finding and identifying relevant datasets within an organization's data ecosystem. It's particularly important in large organizations because:

- Large organizations often have thousands or even millions of datasets distributed across various systems
- Without discovery tools, valuable datasets may remain unknown to data scientists
- Manual searching for relevant data is time-consuming and inefficient
- Potential connections between datasets (like foreign key relationships) may not be obvious

Systems like Google's GOODS address these issues by:

- Cataloging billions of tables across the organization
- Automatically extracting schema information
- Assigning versions and identifying owners
- Providing search functionality and dashboards
- Suggesting potential dataset relationships

Effective dataset discovery significantly reduces the time data scientists spend searching for relevant data and helps them identify valuable connections between datasets.

5. How can automated workflow tools like Airflow help in the data acquisition process?

Solution

Automated workflow tools like Airflow help in the data acquisition process by:

- **Reducing manual errors**: Automating repetitive tasks eliminates human mistakes in data collection
- **Improving scalability**: Handling large-scale data acquisition that would be impractical manually
- **Ensuring consistency**: Applying the same acquisition logic consistently across runs
- **Scheduling**: Automating regular data refreshes on predetermined schedules
- **Dependency management**: Handling complex dependencies between different data acquisition tasks
- **Error handling**: Providing robust error detection and recovery mechanisms
- **Monitoring**: Offering visibility into the acquisition process with logs and alerts

By addressing these aspects, workflow tools help create more reliable, efficient, and maintainable data acquisition pipelines.

3.12 Additional Resources

- Google's Data Discovery Paper
- Metadata for Dataset Search
- GOODS: Organizing Google's Datasets

Key Takeaway

Data acquisition is the critical first step in the data science lifecycle that determines what raw material is available for all subsequent analysis. Effective acquisition requires understanding diverse data sources, their access methods, and the specific needs of the analytical task. As organizations accumulate more data, dataset discovery becomes increasingly important for finding relevant data efficiently. By implementing systematic approaches to data acquisition and leveraging tools for dataset discovery, data scientists can establish a solid foundation for successful analytics and machine learning projects.

Section 4: Data Reorganization

This study guide covers the fourth section of Module 2: Data Collection and Governance, focusing on data reorganization and preparation. This section explores how to transform acquired data into formats suitable for analytics and machine learning tasks. Understanding these processes is essential as they form critical steps in the sourcing stage of the data science lifecycle, where raw data is transformed into analytics-ready datasets.

4 Section 4: Data Reorganization

4.1 Learning Objectives

By the end of this section, you should be able to:

- Understand the role of data reorganization and preparation in the data science workflow
- Identify common steps in the reorganization process for different data types
- Apply appropriate techniques to transform data for specific analytics and ML tasks
- Recognize the importance of automation, documentation, and versioning in data preparation
- Explain how feature stores and ML platforms support data reorganization workflows

4.2 Data Reorganization in the Sourcing Stage

Sourcing Process Flow

Raw data sources/repos → Acquiring → Reorganizing → Cleaning → Data/Feature Engineering for Analytics/ML → Analytics Results
Note: Labeling & Amplification may be required in some cases

4.3 Approaches to Data Reorganization

Key Insight

How to reorganize data depends on:

- The types of data being processed
- The specific analytics or ML task at hand

Common tools and technologies used include:

- SQL for relational data transformation
- MapReduce for distributed processing
- File I/O APIs for direct file manipulation

4.4 Common Reorganization Steps

Common Steps in Data Reorganization

- **Format conversion**: Changing file formats (e.g., export table → CSV → TFRecords)
- **Decompression**: Extracting compressed data (e.g., multimedia)
- **Key-key joins**: Combining multimodal data using common identifiers
- **Key-FK joins**: Joining relational data using foreign key relationships

Fundamental Challenge

Raw datasets sit in source systems in their own formats. The goal of reorganization is to unify and restructure them for analytics and machine learning tasks.

4.5 Real-World Examples of Data Reorganization

4.5.1 Fraud Detection in Banking

Example: Fraud Detection in Banking

Prediction Application: Detect fraudulent transactions in banking data
Reorganization Steps:

- Start with large single-table CSV file on HDFS
- Perform joins to denormalize related data
- Flatten JSON records containing transaction details

4.6 Image Captioning on Social Media

Example: Image Captioning on Social Media

Prediction Application: Generate descriptive captions for social media images
Reorganization Steps:

- Process large binary file with 1 image tensor and 1 string per line
- Fuse JSON records containing metadata
- Extract image tensors for computer vision processing

4.7 Best Practices for Data Reorganization

Best Practices

- **Automation**: Use scripts for reorganization workflows; Apache Airflow
- **Documentation**: Maintain notes/READMEs for code
- **Provenance**: Manage metadata on source/rationale for each data source and feature
- **Versioning**: Reorganization is never "one-and-done"! Maintain logs of what version has what and when
- **Tools**: Typically need both code (SQL, Python) and scripts (bash)

4.8 Advanced Concepts in Data Reorganization

4.8.1 Feature Stores

Feature Stores

"Feature stores" in industry help catalogue ML data. They serve as a central repository for features used in machine learning models.
Example: Uber's Michelangelo platform includes a feature store component.
Reference: Uber Engineering Blog: Michelangelo

4.9 ML Platforms

ML Platforms

"ML platforms" help streamline reorganization and preparation workflows. Key capabilities include:

- Lightweight and flexible schemas
- Automated data validation
- Integration with feature stores and model training

Example: TensorFlow Extended (TFX) provides components for data validation, transformation, and feature engineering.
Reference: TensorFlow Extended (TFX) Guide

4.10 Study Questions

1. Why is data reorganization a critical step in the data science lifecycle?

   Solution

   Data reorganization is critical because raw datasets typically sit in source systems in their own formats, which are often not directly suitable for analytics or machine learning. Reorganization transforms these diverse formats into unified, consistent structures that can be effectively processed by analytical tools and algorithms. Without proper reorganization, subsequent steps like cleaning and feature engineering would be significantly more difficult or impossible. The reorganization step bridges the gap between how data is stored for operational purposes and how it needs to be structured for analytical purposes.

2. What are the common steps involved in data reorganization, and why are they necessary?

Solution

Common steps in data reorganization include:

- **Format conversion**: Changing file formats (e.g., export table → CSV → TFRecords) is necessary because different analytical tools require specific input formats.
- **Decompression**: Extracting compressed data (e.g., multimedia) is needed to access the actual content for analysis.
- **Key-key joins**: Combining multimodal data using common identifiers allows integration of different data types (e.g., text and images) that belong together.
- **Key-FK joins**: Joining relational data using foreign key relationships helps create a more complete view by combining information from multiple tables.

These steps are necessary because they transform data from its original storage format into structures that are optimized for the specific analytical or machine learning task at hand. They help create a unified view of data that may originally be scattered across different systems and formats.

3. How do the reorganization requirements differ for fraud detection versus image captioning applications?

Solution

The reorganization requirements differ significantly between these applications:

For fraud detection in banking:

- Starts with structured data (large single-table CSV file on HDFS)
- Requires joins to denormalize related data (e.g., account information, customer profiles)
- Needs flattening of JSON records containing transaction details
- Focus is on creating a comprehensive tabular dataset with all relevant features
- Primarily deals with structured and semi-structured data

For image captioning on social media:

- Processes binary files containing image tensors and text
- Requires fusion of JSON records containing metadata
- Needs extraction of image tensors for computer vision processing
- Focus is on creating paired data of images and text
- Deals with multimodal data (images and text)

These differences highlight why reorganization approaches must be tailored to the specific data types and analytical goals of each application.

4. Why is versioning important in data reorganization, and how should it be implemented?

Solution

Versioning is important in data reorganization because:

- Reorganization is never a "one-and-done" process; it evolves as data sources, requirements, and analytical techniques change
- Different versions of reorganized data may produce different analytical results
- Reproducibility of analyses requires knowing exactly which version of the data was used
- Debugging and troubleshooting are much easier when changes between versions are tracked
- Collaboration among team members requires clear communication about which data version is being used

Effective versioning should be implemented by:

- Maintaining logs that document what each version contains and when it was created
- Recording the exact transformation steps applied to create each version
- Using version control systems for reorganization code and scripts
- Implementing naming conventions that clearly identify different versions
- Documenting the rationale for changes between versions

This approach ensures transparency, reproducibility, and effective collaboration in the data reorganization process.

5. How do feature stores and ML platforms support the data reorganization process?

Solution

Feature stores and ML platforms support data reorganization in several ways:

Feature Stores:

- Catalog ML data, making it easier to discover and reuse features
- Provide a central repository for features, reducing redundant reorganization efforts
- Maintain metadata about features, including their sources and transformations
- Enable consistent feature definitions across different models and applications
- Support both batch and real-time feature serving

ML Platforms:

- Streamline reorganization workflows with predefined components and pipelines
- Support lightweight and flexible schemas that adapt to changing data
- Automate data validation to ensure quality throughout the reorganization process
- Provide tools for monitoring and debugging reorganization steps
- Integrate with feature stores and model training systems

Examples like Uber's Michelangelo platform and TensorFlow Extended (TFX) demonstrate how these systems can significantly improve the efficiency and reliability of data reorganization processes in production environments.

4.11 Additional Resources

- Uber Engineering Blog: Michelangelo ML Platform
- TensorFlow Extended (TFX) Guide

Key Takeaway

Data reorganization and preparation are essential steps that transform raw data from diverse sources into formats suitable for analytics and machine learning. The approach to reorganization depends on the specific data types and analytical tasks involved. Best practices include automation, documentation, provenance tracking, and versioning. Modern tools like feature stores and ML platforms can significantly streamline these processes. Effective reorganization creates a solid foundation for subsequent data cleaning and feature engineering steps.

Section 5: Data Labeling and Amplification

This study guide covers the fifth section of Module 2: Data Collection and Governance, focusing on data labeling and amplification. These processes are critical for supervised machine learning applications, where models learn from labeled examples. While not required for all data science projects, labeling and amplification are essential when working with applications that need ground truth labels for training and evaluation.

5 Section 5: Data Labeling and Amplification

5.1 Learning Objectives

By the end of this section, you should be able to:

- Understand the role of data labeling in supervised machine learning
- Recognize how labeled data contributes to model performance
- Define what constitutes a "label" for different prediction tasks and data types
- Categorize applications based on their labeling requirements
- Compare manual and programmatic labeling approaches
- Explain data amplification techniques and their applications

5.2 Data Labeling in the Sourcing Stage

Sourcing Process Flow

Raw data sources/repos → Acquiring → Reorganizing → Cleaning → Data/Feature Engineering for Analytics/ML → Analytics Results
Note: Labeling & Amplification may be required in some cases

5.3 The Importance of Labeled Data

Key Insight

- Most recent AI successes are due to supervised machine learning
- Large datasets alone are not enough—need labeled datasets, i.e., pairs of (input, output) examples
- Research shows that model performance continues to improve with larger labeled datasets

Research Example

Google's research on object detection performance shows that:

- Performance (measured by mAP@[.5,.95] on COCO-minival) increases with dataset size
- Pre-training on larger subsets of JFT-300M consistently improves detection performance

Reference: Google AI Blog: The Unreasonable Effectiveness of Data

5.4 Understanding Data Labeling

5.4.1 Definition and Concepts

Definition

Data Labeling: Process of annotating an example (raw or featurized) with ground truth label for a given prediction task.
The notion of "label" is prediction task-specific and data type-specific; can be almost any data structure!

5.4.2 Example: Image Labeling

Example: Multiple Labels for the Same Image

Q: What is a label for an image of a dog on a couch?
It depends on the prediction task:

- "Dog" (object recognition)
- "Couch" (object recognition with different focus)
- "Shiba Inu" (dog breed classifier)
- "Yes" (meme classifier)
- Dog with bounding box coordinates (object detection)
- Highlighted pixels showing the dog (image segmentation)

5.5 Categorizing Applications by Labeling Needs

Three Types of Applications Based on Label Sources

With respect to sources of labels, there are three kinds of prediction applications:

1. **Data-generating process offers labels naturally**
   - Examples: Customer churn prediction, forecasting
2. **Product/service users offer labels (in)directly**
   - Examples: Email spam filters, online advertising, product recommendations, photo tagging, web search
3. **Need application-specific extra effort for labels**
   - Examples: Radiology, self-driving cars, species classification, video surveillance, machine translation, knowledge base construction, document summarization

5.6 Programmatic Labeling

Programmatic Labeling Approach

Basic Idea: Instead of manually labeling each example, write programs/rules/heuristics that encode some domain intuition to label examples en masse.
Pros:

- Improved labeling productivity
- Likely lower costs

Cons:

- Need to write code
- Less reliable accuracy
- Unclear if complex prediction outputs are supportable

Reference: Snorkel: Rapid Training Data Creation with Weak Supervision

5.7 Data Amplification

Data Amplification Techniques

Methods to expand labeled datasets without additional manual labeling:

- **Label-preserving transforms**: Common in computer vision (e.g., rotation, flipping, cropping)
- **Synthesis**: Sometimes possible in robotics/scientific/engineering applications
  - Physical laws-based
  - Simulation-based
- **Challenges**: Tricky; needs knowledge of underlying data distribution

5.8 Study Questions

1. Why is data labeling necessary for supervised machine learning, and how does it fit into the data science lifecycle?

Solution

Data labeling is necessary for supervised machine learning because these algorithms learn by example, requiring input-output pairs for training. The algorithm learns to map inputs to outputs by observing these labeled examples.
In the data science lifecycle, labeling typically occurs after data acquisition, reorganization, and cleaning, but before feature engineering and model training. It's a critical step that:

- Transforms raw data into training examples with ground truth outputs
- Defines the prediction task by specifying what the model should learn to predict
- Provides the foundation for model evaluation by establishing ground truth

Unlike other steps in the sourcing stage (acquisition, reorganization, cleaning), labeling is not always required. It's specifically needed for supervised learning applications where the raw data doesn't naturally include the target outputs. For unsupervised learning or when labels are inherent in the data-generating process, this step may be skipped.
Research from Google and others has consistently shown that the quality and quantity of labeled data directly impact model performance, often more significantly than algorithm selection or hyperparameter tuning.

2. What makes a "label" different for various prediction tasks, even when using the same raw data?

Solution

A "label" differs across prediction tasks because it represents the specific output the model should learn to predict, which varies based on the task's objective. Using the example from the lecture of an image showing a dog on a couch:

- For general object recognition, the label might be "dog" or "couch" depending on what's considered the primary subject
- For breed classification, the label would be the specific breed (e.g., "Shiba Inu")
- For a meme classifier, the label might be a simple "yes" or "no"
- For object detection, the label includes both the class ("dog") and the bounding box coordinates
- For image segmentation, the label is a pixel-level mask highlighting the dog

This demonstrates that the notion of a "label" is both prediction task-specific and data type-specific. The same raw data can have completely different labels depending on what we're trying to predict. This is why clearly defining the prediction task is a crucial first step before beginning any labeling effort.
The format and complexity of labels also vary widely - from simple class names to complex structures like bounding boxes, segmentation masks, or even sequences (for tasks like machine translation).

3. How do the labeling requirements differ across the three categories of applications mentioned in the lecture?

Solution

The three categories of applications have significantly different labeling requirements:

1. **Data-generating process offers labels naturally:**
   - Labels are inherent in the historical data
   - Minimal additional labeling effort required
   - Examples: Customer churn prediction (label = whether customer churned), forecasting (label = actual future values)
   - Advantage: Labeling can be largely automated
2. **Product/service users offer labels (in)directly:**
   - Labels come from user interactions with the product/service
   - Moderate effort to capture and structure this feedback
   - Examples: Email spam filters (users mark emails as spam), product recommendations (users click or purchase)
   - Advantage: Labels continuously accumulate through normal product usage
3. **Need application-specific extra effort for labels:**
   - Requires dedicated labeling projects
   - High effort, often involving domain experts
   - Examples: Medical imaging, self-driving cars, document summarization
   - Challenge: Most resource-intensive and potentially expensive

These differences significantly impact project planning, resource allocation, and timelines. Category 3 applications typically require the most upfront investment in labeling infrastructure and processes.

4. Compare and contrast manual and programmatic labeling approaches.

Solution

Manual and programmatic labeling represent two fundamentally different approaches:

Manual Labeling:

- Process: Human annotators review each example and assign labels
- Accuracy: Generally high, especially with domain experts
- Scalability: Limited by human resources and time
- Cost: Typically high, especially for specialized domains
- Flexibility: Can handle complex, nuanced labeling tasks

Programmatic Labeling:

- Process: Uses code, rules, or heuristics to automatically label data
- Accuracy: Generally lower than expert manual labeling
- Scalability: Highly scalable to millions of examples
- Cost: Higher upfront development cost, lower per-label cost
- Flexibility: Limited to patterns that can be codified
- Implementation: Frameworks like Snorkel help create labeling functions

The optimal approach depends on the specific task, domain, budget, and quality requirements. Many modern labeling systems use hybrid approaches that combine the strengths of both methods, such as using programmatic labeling for initial passes, with human verification of uncertain cases.

5. What is data amplification, and what techniques are commonly used

for it?

Solution

Data amplification refers to techniques that expand a labeled dataset by creating new labeled examples without requiring additional manual labeling. Based on the lecture, common techniques include:

1. **Label-preserving transformations:**
   - Most common in computer vision
   - Examples: rotation, flipping, cropping, color adjustments
   - Preserves the semantic meaning of the image while creating variation
   - Helps models become invariant to these transformations
2. **Synthetic data generation:**
   - Based on physical laws, simulations, or generative models
   - Common in robotics, scientific, and engineering applications
   - Can create examples for rare or dangerous scenarios
   - Requires knowledge of the underlying data distribution

The key challenge with data amplification is ensuring that the generated examples are realistic and representative of the true data distribution. Poor amplification can introduce biases or unrealistic examples that harm model performance. This is why amplification techniques need to be carefully designed with domain knowledge about the data and the prediction task.

5.9 Additional Resources

- Google AI Blog: The Unreasonable Effectiveness of Data
- Snorkel: Rapid Training Data Creation with Weak Supervision

Key Takeaway

Data labeling and amplification are critical steps for supervised machine learning applications. The approach to labeling depends on the prediction task, data type, and application category. While some applications naturally provide labels or collect them through user interactions, others require dedicated labeling efforts. Programmatic labeling offers scalability advantages but may sacrifice some accuracy. Data amplification techniques can expand labeled datasets through transformations or synthesis, but require careful implementation to maintain data quality. Understanding these concepts is essential for effectively preparing data for supervised learning tasks.

Section 6: Data Governance and Privacy

This study guide covers the sixth section of Module 2: Data Collection and Governance, focusing on data governance and privacy. This section explores the principles and practices for managing data as valuable assets, the legal regulations that govern data handling, and the challenges of provenance management. Understanding these concepts is essential for responsible and compliant data management in any data science project.

6 Section 6: Data Governance and Privacy

6.1 Learning Objectives

By the end of this section, you should be able to:

- Understand the concept of data governance and its key aspects
- Identify major legal regulations governing data handling
- Explain the implications of privacy laws for data science projects
- Define data provenance and its importance in the data lifecycle
- Recognize the challenges in implementing provenance management
- Identify tools that support data governance and provenance tracking

6.2 Data Governance Fundamentals

Data as Valuable Entities

- Data are "entities" with "value"—not unlike people!
- Data have lifecycles:
  - Born: created
  - Live: used
  - Die: deleted
- Just as people need to be governed, so must data

6.3 Key Aspects of Data Governance

Six Pillars of Data Governance

- **Privacy & Security**: Who sees what, why? No breaches!
- **Stewardship**: Who owns what, when? Access control.
- **Cataloging**: What is it, where, how to access?
- **Defining**: Data dictionaries, business knowledge.
- **Quality**: Follow conventions, reduce errors.
- **Provenance**: Track usage, changes, versions. Auditing.

6.4 Legal Regulations on Data Handling

Key Insight

Just as laws exist to govern people, laws exist to govern data. There is a long history of laws surrounding data, though notably there are no laws (yet) specifically governing ML "algorithms"—the focus is on the data used in ML.

6.5 Major Privacy Regulations

FERPA 1974

- Family Educational Rights and Privacy Act
- Broadly applies to all "education records" of students
- Gives parents and eligible students rights to access and review educational records
- Requires written permission to release information from a student's record

Reference: FERPA: How to Manage Student Records

HIPAA 1996

- Health Insurance Portability and Accountability Act
- Broadly applies to all healthcare data, especially Personally Identifiable Information (PII)
- Establishes standards for the protection of sensitive patient health information
- Requires appropriate safeguards to protect the privacy of personal health information

GDPR 2018

- General Data Protection Regulation
- Applies to data collected from individuals in EU and EEA
- Establishes new rights on "personal data":
  - Right to access
  - Right to forget/erasure
  - Right to object
  - And more
- Many web companies scrambled to comply; some "exited" EU area
- Creates new technical challenges for data/ML infrastructure:
  - Metadata handling
  - Efficiency concerns
- Raises open legal and technical questions for ML applications:
  - Are ML models under purview?
  - What about derived/aggregated data?

CCPA (California Consumer Privacy Act)

- California's privacy law that went into effect in 2020
- Similar to GDPR but with some key differences
- Gives California residents rights over their personal information
- Applies to businesses that meet specific criteria

Reference: CCPA and GDPR: How the Privacy Laws Stack Up

6.6 Data Provenance Management

6.6.1 Definition and Importance

What is Provenance?

Provenance: "Chronology of the ownership, custody or location of a historical object"
In data science, provenance means tracking:

- All data objects throughout their lifecycle
- The origin, transformations, and usage of data

Why it matters:

- Ensures compliance with data regulations
- Supports auditing requirements
- Makes data easier to find and consume
- Enables reproducibility of analyses

6.6.2 Key Aspects of Provenance

- **Context**: Data creation, deletion, access/use, etc.
- **Metadata Evolution**: How metadata changes over time
- **Versioning**: Tracking versions of data and derived objects
- **ML-specific Tracking**:
  - Derived data (e.g., feature extraction)
  - ML artifacts (models, code/scripts, etc.)
  - Configuration settings

6.6.3 Provenance Management Challenges

Challenges

- Heterogeneity of data/ML platforms makes tracking notoriously messy/tedious
- Difficult questions about what to track:
  - Metadata?
  - Usage logs?
  - Versioning?
- Current state: Ad hoc, organization-specific practices and tools
- Need to learn organization-specific practices and APIs

6.6.4 Tools for Provenance Management

Emerging Open Source Tools

Several emerging open source tools can help with provenance management:

For ML artifacts:

- Weights & Biases
- MLFlow
- TensorFlow Extended
- TensorBoard

For SQL transformation code:

- dbt (data build tool)

For derived data:

- Feature stores (e.g., Feast, Tecton)

Example: MLFlow Experiment Tracking provides capabilities for tracking experiments, packaging code, and sharing models.

Tool Adoption in Practice

The adoption of data governance and provenance tools varies widely across organizations and industries. The Kaggle Survey 2021 provides insights into which tools are most commonly used by data professionals.
Reference: Kaggle Survey 2021

6.7 Study Questions

1. Why is data governance important in the data science lifecycle?

Solution

Data governance is important in the data science lifecycle because:

- Data are valuable entities with lifecycles that need to be managed responsibly
- Proper governance ensures compliance with legal regulations like FERPA, HIPAA, GDPR, and CCPA
- It establishes clear protocols for privacy, security, and access control
- It enables effective cataloging and discovery of data assets
- It promotes data quality and reduces errors
- It supports provenance tracking for reproducibility and auditing
- It clarifies ownership and stewardship responsibilities

Without proper governance, organizations risk legal penalties, security breaches, poor data quality, and inefficient use of data assets. In the context of data science, governance provides the framework that allows data scientists to work with data responsibly and effectively.

2. How do privacy regulations like GDPR impact machine learning projects?

Solution

Privacy regulations like GDPR impact machine learning projects in several significant ways:

- **Data collection limitations**: Requires explicit consent for collecting personal data, limiting what data can be used for training
- **Purpose limitation**: Data can only be used for the specific purposes for which it was collected
- **Right to erasure ("right to be forgotten")**: May require removing specific individuals' data from training sets and potentially retraining models
- **Right to explanation**: May require making ML models more interpretable to explain decisions
- **Data minimization**: Encourages using only the minimum necessary data for a specific purpose
- **Technical challenges**: Creates new requirements for metadata handling and efficiency in ML infrastructure
- **Open questions**: Raises unresolved issues about whether ML models themselves fall under regulatory purview and how to handle derived or aggregated data

These impacts require ML practitioners to incorporate privacy considerations throughout the ML lifecycle, from data collection and preparation to model deployment and monitoring.

3. What is data provenance, and why is it critical for data science projects?

Solution

Data provenance refers to the chronology of ownership, custody, or location of data throughout its lifecycle. It tracks the origin of data and any transformations applied to it.
Data provenance is critical for data science projects because it:

- **Ensures compliance**: Helps meet regulatory requirements by documenting data handling practices
- **Enables reproducibility**: Allows others to reproduce analyses by following the same data lineage
- **Supports debugging**: Makes it easier to identify where errors or issues may have been introduced
- **Facilitates collaboration**: Helps team members understand data sources and transformations
- **Improves data discovery**: Makes it easier to find and understand available data assets
- **Builds trust**: Increases confidence in results by documenting the complete data journey
- **Enables auditing**: Provides a trail that can be examined for compliance or quality assurance

For ML projects specifically, provenance must track not only the raw data but also derived features, model artifacts, and configuration settings to ensure full reproducibility and compliance.

4. What are the main challenges in implementing effective provenance management?

Solution

Implementing effective provenance management faces several significant challenges:

- **Heterogeneity**: The diverse ecosystem of data and ML platforms makes consistent tracking difficult
- **Scope determination**: Deciding what to track (metadata, usage logs, versions) is not straightforward
- **Organizational variation**: Current practices are largely ad hoc and organization-specific
- **Technical complexity**: Integrating provenance tracking across different systems requires significant engineering effort
- **Learning curve**: Team members need to learn organization-specific practices and APIs
- **Overhead**: Comprehensive tracking can add computational and storage overhead
- **Balancing detail**: Too little tracking is insufficient, but too much creates noise
- **Tool immaturity**: Many provenance management tools are still evolving

These challenges explain why provenance management often remains incomplete in practice, despite its recognized importance. Organizations typically need to develop custom solutions that balance comprehensiveness with practicality.

5. How do tools like MLFlow help with provenance management for machine learning projects?

Solution

Tools like MLFlow help with provenance management for machine learning projects by:

- **Experiment tracking**: Recording parameters, metrics, and artifacts for each experimental run
- **Model versioning**: Maintaining a history of model versions with their associated metadata
- **Code packaging**: Capturing the exact code used to create models for reproducibility
- **Model registry**: Providing a centralized repository for managing the full lifecycle of ML models
- **Lineage tracking**: Recording the relationships between data, features, and models
- **Collaboration support**: Enabling team members to share and build upon each other's work
- **Integration capabilities**: Connecting with other tools in the ML ecosystem

These capabilities address many of the core requirements for ML provenance, though they typically need to be complemented with other tools for complete coverage of the data science lifecycle. For example, feature stores like Feast or Tecton help track derived data, while tools like dbt help manage SQL transformation code.

6.8 Additional Resources

- FERPA: How to Manage Student Records
- CCPA and GDPR: How the Privacy Laws Stack Up
- Kaggle Survey 2021

Key Takeaway

Data governance and privacy are essential considerations in the data science lifecycle. Treating data as valuable entities requires implementing proper governance across six key pillars: privacy & security, stewardship, cataloging, defining, quality, and provenance. Legal regulations like FERPA, HIPAA, GDPR, and CCPA impose specific requirements on data handling that must be incorporated into data science workflows. Provenance management—tracking the origin, transformations, and usage of data—is critical for compliance, reproducibility, and effective collaboration, though it presents significant implementation challenges. Emerging tools are helping address these challenges, but organizations still need to develop practices tailored to their specific needs and contexts.

Module Overview

This guide covers fundamental data models with focus on relational and DataFrame paradigms. Key topics include structural components, constraints, and SQL operations.

7 DataFrame Model

Core Concepts

Historical Development:

- 1992: Originated in S language (Bell Labs)
- 2000: Adopted by R
- 2009: Pandas implementation in Python

Key Features:

- Hybrid operations: Relational + Linear Algebra + Spreadsheet
- Labeled axes (rows & columns)
- Heterogeneous data types per column

Comparative Analysis

Aspect	vs. Relational	vs. Matrices
Schema	Lazily-induced	N/A
Structure	Named/ordered rows & columns	Numeric indices
Data Types	Heterogeneous columns	Homogeneous elements
Operations	Filter/Join + Transpose + Pivot	Pure linear algebra

8 Relational Model

Structural Components

```
CREATE TABLE Students (
    sid      CHAR(20) PRIMARY KEY,
    name     CHAR(30),
    age      INTEGER,
    gpa      REAL
);
```

Core Elements:

- **Relation**: Table with attributes (columns) and tuples (rows)
- **Schema**: Structural metadata (name:type pairs)
- **Instance**: Current dataset conforming to schema

Constraints

Domain Constraints:

- Enforce data types (INT, CHAR, DATE)

Key Constraints:

- Candidate Key: Minimal unique identifier
- Primary Key: Chosen main identifier
- Super Key: Superset containing candidate key

Referential Integrity:

```
CREATE TABLE Enrolled (
    sid CHAR(20) REFERENCES Students(sid),
    ...
);
```

9 SQL Fundamentals

Essential Operations

Operation	SQL Example
Create Table	`CREATE TABLE Students (...);`
Insert Data	`INSERT INTO Students VALUES (...);`
Delete	`DELETE FROM Students WHERE age > 30;`
Update	`UPDATE Students SET gpa = 3.5 WHERE ...;`
Query	`SELECT name, gpa FROM Students WHERE ...;`
Alter	`ALTER TABLE Students ADD email VARCHAR;`

First Normal Form (1NF)

Requirement: Atomic values, no nested/repeating groups
Violation Example:

```
CREATE TABLE BadDesign (
    sid INT,
    courses_enrolled ARRAY   -- Invalid
);
```

1NF Solution:

```
CREATE TABLE Enrolled (
    sid INT REFERENCES Students,
    cid CHAR(10),
    grade REAL
);
```

10 Key Takeaways

Essential Concepts

1. **DataFrame Model** bridges relational and numerical computing
2. **Relational Model** requires explicit schema + constraints
3. **1NF** ensures atomic values through flat table structures
4. **SQL** enables declarative data definition and manipulation