DSA 610 Redesign, Lecture 7 Outline

Lecture Outline: Data Maintenance, Consistency, Accuracy, Integrity, ACID, and Data Validation/Verification
Duration: 50 minutes

---

1. Introduction to Data Maintenance (5 minutes)

- **Objective:** Understand the importance of maintaining data quality in data-driven environments.
- **Content:**
  - **Definition:**
    - **Data Maintenance:** The process of ensuring data remains accurate, consistent, and reliable over time.
  - **Importance of Data Maintenance:**
    - **Decision-Making:** Accurate and consistent data is essential for making informed decisions.
    - **Regulatory Compliance:** Ensuring data meets legal and regulatory standards.
    - **Operational Efficiency:** Reliable data supports smooth business operations and reduces errors.

---

2. Data Consistency, Accuracy, and Integrity (10 minutes)

- **Objective:** Explore the key concepts of data consistency, accuracy, and integrity, and how they impact data quality.
- **Content:**
  - **Data Consistency:**
    - **Definition:** Ensuring that data is uniform and reliable across different systems and databases.
    - **Examples:**
      - **Cross-System Consistency:** Ensuring that customer information is the same across sales, CRM, and support databases.
      - **Time Consistency:** Maintaining the same data across different time periods (e.g., daily reports should match monthly summaries).
  - **Data Accuracy:**
    - **Definition:** Ensuring that data correctly reflects the real-world entities or events it represents.
    - **Examples:**
      - **Accurate Records:** Customer addresses must be accurate to ensure successful deliveries.
      - **Measurement Accuracy:** Sensor data must be accurate to support proper analysis in IoT applications.
  - **Data Integrity:**
    - **Definition:** Maintaining and ensuring the accuracy and consistency of data over its lifecycle.
    - **Components:**
      - **Entity Integrity:** Ensuring that each entity (e.g., record) is unique and identifiable.
      - **Referential Integrity:** Ensuring that relationships between entities are consistent (e.g., foreign keys in databases).
    - **Example:**

        - **Database Relationships:** A sales order in a database should not reference a non-existent customer.

---

3. ACID Properties in Databases (10 minutes)

- **Objective:** Learn about the ACID properties that ensure reliable transaction processing in databases.
- **Content:**
  - **Definition of ACID:**
    - **Atomicity:** Ensures that a transaction is all-or-nothing. If one part of the transaction fails, the entire transaction fails.
      - **Example:** Transferring money between bank accounts—both the debit and credit must occur, or neither should.
    - **Consistency:** Ensures that a transaction brings the database from one valid state to another, preserving database rules.
      - **Example:** Ensuring that data constraints (e.g., balance must be non-negative) are maintained after transactions.
    - **Isolation:** Ensures that transactions occurring concurrently do not affect each other's outcomes.
      - **Example:** Two users updating the same record should not see each other's intermediate states.
    - **Durability:** Ensures that once a transaction is committed, it remains so, even in the case of a system failure.
      - **Example:** After a transaction is confirmed, it should not be lost even if there's a power outage.
  - **Importance of ACID:**
    - **Reliability in Databases:** ACID properties are crucial for maintaining reliable databases, especially in high-stakes applications like finance, healthcare, and e-commerce.
    - **Example in Practice:** SQL databases like MySQL, PostgreSQL, and Oracle implement ACID properties to ensure data consistency and reliability.

---

4. Data Validation and Verification Methods (15 minutes)

- **Objective:** Understand the techniques and methods for validating and verifying data to ensure its quality.
- **Content:**
  - **Data Validation:**
    - **Definition:** The process of ensuring that data meets defined criteria and is suitable for analysis or processing.
    - **Types of Validation:**
      - **Format Validation:** Ensuring data is in the correct format (e.g., dates, email addresses).
      - **Range Validation:** Ensuring numerical data falls within a specified range (e.g., age should be between 0 and 120).
      - **Consistency Validation:** Ensuring data values are consistent across different fields (e.g., start date should be before end date).
    - **Example:**

```
# Example of data validation in Python
import pandas as pd
```

```
# Load dataset
df = pd.read_csv('data.csv')

# Validate email format
df['valid_email'] = df['email'].str.contains(r'^[\w\.-]+@[\w\.-]+\.\w+$')

# Validate range
df['valid_age'] = df['age'].between(0, 120)
```

-
    - **Data Verification:**
        - **Definition:** The process of ensuring that data is accurate and reflects real-world conditions.
        - **Techniques:**
            - **Cross-Verification:** Comparing data from different sources to ensure consistency.
            - **Manual Review:** Checking a sample of the data manually to ensure accuracy.
            - **Automated Verification:** Using scripts or tools to verify data against known benchmarks or constraints.
        - **Example:**
            - **Verification in Data Entry:** Cross-checking entered data with source documents (e.g., invoices, surveys).
            - **Verification in Data Migration:** Ensuring that data transferred from one system to another remains accurate and complete.
    - **Common Tools for Data Validation and Verification:**
        - **Pandas in Python:** For data validation and transformation.
        - **SQL Queries:** For verifying data in relational databases.
        - **Excel:** For basic data validation using data validation rules and conditional formatting.

---

5. Practical Examples and Case Studies (5 minutes)

- **Objective:** Apply the concepts through practical examples and case studies.
- **Content:**
    - **Case Study 1:** Implementing data validation in a healthcare database to ensure patient records are accurate and consistent.
    - **Case Study 2:** Using ACID properties to ensure reliable transaction processing in an e-commerce application.
    - **Hands-On Example:** Demonstrate data validation techniques in Python and SQL.

---

6. Q&A and Discussion (5 minutes)

- **Objective:** Address any questions and discuss real-world challenges related to maintaining data consistency, accuracy, and integrity.
- **Content:**
    - **Q&A Session:** Open the floor for questions from students.
    - **Discussion:** Explore common challenges in data maintenance and how to overcome them in different industries.

Key Takeaways

- **Data Maintenance:** Importance of keeping data accurate, consistent, and reliable.
- **ACID Properties:** Understanding how ACID ensures reliable transaction processing in databases.
- **Data Validation/Verification:** Techniques for ensuring data meets quality standards before analysis.

Resources:
Data Maintenance: https://www.validity.com/data-quality/data-maintenance-and-quality/
Data Maintenance vs. Data Cleansing: https://www.indeed.com/career-advice/career-development/data-maintenance-vs-data-cleansing
Implementing a Data Maintenance Strategy: https://help.hcl-software.com/commerce/8.0.0/admin/concepts/cdbdatamaintenance.html
Recommended Data Maintenance Activities: https://iawindows.zendesk.com/hc/en-us/articles/26770147652499-Recommended-Data-Maintenance-Activities
8 Best Practices: https://www.tierpoint.com/blog/data-center-maintenance/
9 Ways to Maintain Data Quality: https://www.actian.com/blog/data-management/data-management-quality/
ACID for DBMS: https://www.geeksforgeeks.org/acid-properties-in-dbms/
https://www.databricks.com/glossary/acid-transactions https://www.freecodecamp.org/news/acid-databases-explained/ https://www.bmc.com/blogs/acid-atomic-consistent-isolated-durable/
In MongoDB: https://www.mongodb.com/resources/basics/databases/acid-transactions

Lecture Outline: Data Exploration and Model Planning in the Data Analysis Lifecycle
Duration: 50 minutes

---

1. Introduction to Data Exploration (10 minutes)

- **Objective:** Understand the purpose and techniques of data exploration in the data analysis lifecycle.
- **Content:**
  - **Definition:**
    - **Data Exploration:** The process of analyzing and understanding the characteristics of a dataset before applying statistical or machine learning models.
  - **Purpose of Data Exploration:**
    - **Understanding Data Distribution:** Identifying patterns, trends, and anomalies.
    - **Formulating Hypotheses:** Generating hypotheses about relationships within the data.
    - **Preparing for Modeling:** Identifying preprocessing needs and feature selection.
  - **Key Steps in Data Exploration:**
    - **Summary Statistics:** Using measures like mean, median, mode, standard deviation.
    - **Data Visualization:** Creating plots to visualize data distributions and relationships.
    - **Correlation Analysis:** Checking relationships between variables.
  - **Example:**

```
import pandas as pd
import seaborn as sns
```

```
import matplotlib.pyplot as plt

# Load dataset
df = pd.read_csv('data.csv')

# Summary statistics
print(df.describe())

# Data visualization
sns.pairplot(df)
plt.show()

# Correlation analysis
print(df.corr())
```

Key Techniques in Data Exploration (15 minutes)

- **Objective:** Explore specific techniques and tools used during the data exploration phase.
- **Content:**
    - **Univariate Analysis:**
        - **Definition:** Analysis of individual variables to understand their distributions.
        - **Techniques:**
            - **Histograms:** Visualizing the distribution of a single variable.
            - **Box Plots:** Identifying outliers and understanding the spread.
        - **Example:**

```
# Histogram
df['variable'].hist()
plt.title('Histogram of Variable')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.show()

# Box Plot
sns.boxplot(x='variable', data=df)
plt.title('Box Plot of Variable')
plt.show()
```

Bivariate Analysis:

- **Definition:** Analyzing relationships between two variables.
- **Techniques:**
    - **Scatter Plots:** Examining the relationship between two continuous variables.
    - **Heatmaps:** Visualizing correlation matrices.
- **Example:**

```
# Scatter Plot
plt.scatter(df['variable1'], df['variable2'])
plt.title('Scatter Plot of Variable1 vs. Variable2')
plt.xlabel('Variable1')
plt.ylabel('Variable2')
```

```
plt.show()

# Heatmap
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```

Multivariate Analysis:

- **Definition:** Analyzing interactions between more than two variables.
- **Techniques:**
  - **Pair Plots:** Visualizing relationships between multiple variables.
  - **Principal Component Analysis (PCA):** Reducing dimensionality while preserving variance.
- **Example:**

```
# Pair Plot
sns.pairplot(df, hue='target_variable')
plt.show()
```

Model Planning (15 minutes)

- **Objective:** Learn how to plan and prepare for model development based on data exploration insights.
- **Content:**
  - **Defining Objectives:**
    - **Problem Statement:** Clearly define the problem to be solved (e.g., classification, regression).
    - **Success Metrics:** Determine how success will be measured (e.g., accuracy, F1-score).
  - **Feature Selection:**
    - **Definition:** Choosing which features to include in the model based on exploration.
    - **Techniques:**
      - **Correlation Analysis:** Selecting features with significant correlations to the target variable.
      - **Feature Importance:** Using models like Random Forest to evaluate feature importance.
    - **Example:**

```
from sklearn.ensemble import RandomForestClassifier

# Feature Importance
model = RandomForestClassifier()
model.fit(df.drop('target', axis=1), df['target'])
feature_importances = model.feature_importances_
features = df.columns[:-1]
importance_df = pd.DataFrame({'Feature': features, 'Importance': feature_importances})
importance_df.sort_values(by='Importance', ascending=False, inplace=True)
print(importance_df)
```

Data Preprocessing:

- **Definition:** Preparing data for modeling, including handling missing values and encoding categorical variables.
- **Techniques:**
    - **Handling Missing Data:** Imputation or removal of missing values.
    - **Encoding Categorical Variables:** One-hot encoding or label encoding.
- **Example:**

```
# Handling Missing Data
df.fillna(df.mean(), inplace=True)

# One-Hot Encoding
df_encoded = pd.get_dummies(df, columns=['categorical_feature'])
```

Model Selection:

- **Definition:** Choosing appropriate models based on the problem and data characteristics.
- **Types of Models:**
    - **Classification Models:** Logistic Regression, Decision Trees, etc.
    - **Regression Models:** Linear Regression, Ridge Regression, etc.
- **Example:**

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(df.drop('target', axis=1), df['target'], test_size=0.2)

# Model Selection
model = LogisticRegression()
model.fit(X_train, y_train)
print(f"Model Accuracy: {model.score(X_test, y_test)}")
```

Planning for Validation and Evaluation (5 minutes)

- **Objective:** Understand how to plan for model validation and evaluation.
- **Content:**
    - **Cross-Validation:**
        - **Definition:** Technique for assessing the performance of the model by dividing the data into multiple folds.
        - **Types:** K-Fold Cross-Validation.
    - **Evaluation Metrics:**
        - **Classification Metrics:** Accuracy, Precision, Recall, F1-Score.
        - **Regression Metrics:** Mean Absolute Error (MAE), Mean Squared Error (MSE), R-squared.
    - **Example:**

```
from sklearn.model_selection import cross_val_score

# Cross-Validation
```

```
scores = cross_val_score(model, df.drop('target', axis=1), df['target'], cv=5)
print(f"Cross-Validation Scores: {scores}")
```

Q&A and Discussion (5 minutes)

- **Objective:** Address any questions and discuss real-world challenges in data exploration and model planning.
- **Content:**
    - **Q&A Session:** Open the floor for student questions.
    - **Discussion:** Explore practical challenges and solutions in data exploration and planning for models.
-
    -

Key Takeaways

- **Data Exploration:** Techniques to understand and visualize data distributions and relationships.
- **Model Planning:** Steps for preparing data, selecting features, choosing models, and planning for validation.
- **Practical Skills:** Applying data exploration insights to plan effective models and evaluate their performance.

Resources:
Data Exploration: https://www.heavy.ai/learn/data-exploration
https://www.matillion.com/learn/blog/data-exploration https://www.geeksforgeeks.org/what-is-data-exploration-and-its-process/ https://www.analyticsvidhya.com/blog/2016/01/guide-data-exploration/
https://www.simplilearn.com/data-exploration-article
Theory and Techniques: https://www.keboola.com/blog/data-exploration-techniques
In Python: https://www.kaggle.com/code/pmarcelino/comprehensive-data-exploration-with-python
https://www.geeksforgeeks.org/exploratory-data-analysis-in-python/
https://www.analyticsvidhya.com/blog/2015/04/comprehensive-guide-data-exploration-sas-using-python-numpy-scipy-matplotlib-pandas/ https://www.omdena.com/blog/a-beginners-guide-to-exploratory-data-analysis-with-python https://hex.tech/blog/explore-data-with-python-and-pandas/

Lecture Outline: Imputations, Dealing with Missing Values, Writing Functions, and Simulations in Python
Duration: 50 minutes

---

1. Introduction to Missing Values and Imputations (10 minutes)

- **Objective:** Understand common techniques for handling missing values and performing imputations.
- **Content:**
    - **Definition of Missing Values:**
        - **Missing Values:** Data entries that are not recorded or are incomplete.
    - **Importance of Handling Missing Values:**
        - **Impact on Analysis:** Missing values can skew results and reduce the accuracy of models.
        - **Techniques for Imputation:** Strategies to fill in missing values.
    - **Example Dataset:**

```
import pandas as pd
```

```
import numpy as np

# Create example DataFrame with missing values
data = {'A': [1, 2, np.nan, 4],
        'B': [5, np.nan, 7, 8],
        'C': ['x', 'y', 'z', np.nan]}
df = pd.DataFrame(data)
print(df)
```

Techniques for Handling Missing Values (15 minutes)

- **Objective:** Learn and apply different methods for imputing missing values.
- **Content:**
    - **1. Removing Missing Values:**
        - **Definition:** Removing rows or columns with missing values.
        - **Example:**

```
# Drop rows with missing values
df_dropped_rows = df.dropna()

# Drop columns with missing values
df_dropped_cols = df.dropna(axis=1)

print("Dropped Rows:\n", df_dropped_rows)
print("Dropped Columns:\n", df_dropped_cols)
```

Mean/Median Imputation:

- **Definition:** Filling missing values with the mean or median of the column.
- **Example:**

```
# Mean imputation
df['A'].fillna(df['A'].mean(), inplace=True)

# Median imputation
df['B'].fillna(df['B'].median(), inplace=True)

print("Mean Imputation:\n", df)
```

3. Mode Imputation:

- **Definition:** Filling missing values with the mode (most frequent value) of the column.
- **Example:**

```
# Mode imputation for categorical data
mode_value = df['C'].mode()[0]
df['C'].fillna(mode_value, inplace=True)

print("Mode Imputation:\n", df)
```

4. Interpolation:

- **Definition:** Estimating missing values based on the values of surrounding data.

- **Example:**

```
# Interpolation
df_interpolated = df.interpolate()

print("Interpolated Data:\n", df_interpolated)
```

3. Writing Functions in Python (10 minutes)

- **Objective:** Learn how to create reusable functions for data handling and imputation tasks.
- **Content:**
    - **Function Definition:**
        - **Syntax:** How to define a function using def keyword.
        - **Example:**

```
def impute_missing_values(df):
   """Impute missing values with mean for numerical columns and mode for categorical columns."""
   for column in df.columns:
      if df[column].dtype == 'object':
         mode_value = df[column].mode()[0]
         df[column].fillna(mode_value, inplace=True)
      else:
         mean_value = df[column].mean()
         df[column].fillna(mean_value, inplace=True)
   return df

df_imputed = impute_missing_values(df.copy())
print("Data After Imputation:\n", df_imputed)
```

Writing Functions for Simulations:

- **Example Function to Simulate Random Data:**

```
def simulate_random_data(size, mean=0, std=1):
   """Simulate random data with specified mean and standard deviation."""
   return np.random.normal(loc=mean, scale=std, size=size)

simulated_data = simulate_random_data(100, mean=5, std=2)
print("Simulated Data:\n", simulated_data[:10])
```

4. Performing Simulations (10 minutes)

- **Objective:** Understand how to perform simulations to generate data or test hypotheses.
- **Content:**
    - **1. Simulating Data Distributions:**
        - **Example:**

```
import matplotlib.pyplot as plt

# Simulate normal distribution data
data_normal = np.random.normal(loc=0, scale=1, size=1000)
plt.hist(data_normal, bins=30, edgecolor='k')
plt.title('Normal Distribution')
```

```
plt.show()
```

2. Bootstrapping:

- **Definition:** A resampling technique to estimate the distribution of a statistic.
- **Example:**

```
from sklearn.utils import resample

# Original dataset
original_data = df['A'].dropna()

# Bootstrapping
bootstrap_sample = resample(original_data, n_samples=len(original_data), replace=True)
print("Bootstrap Sample:\n", bootstrap_sample)
```

3. Monte Carlo Simulations:

- **Definition:** Using randomness to solve problems or simulate processes.
- **Example:**

```
def monte_carlo_simulation(num_trials):
  results = []
  for _ in range(num_trials):
    result = np.random.normal(loc=0, scale=1)
    results.append(result)
  return np.mean(results)

mean_result = monte_carlo_simulation(1000)
print("Mean Result of Monte Carlo Simulation:", mean_result)
```

5. Practical Examples and Q&A (5 minutes)

- **Objective:** Apply the concepts through practical examples and answer any questions.
- **Content:**
    - **Hands-On Exercise:** Implement a function to handle missing values and perform simulations on a sample dataset.
    - **Q&A Session:** Address any questions and clarify concepts discussed during the lecture.

Key Takeaways

- **Imputations:** Techniques for handling missing values including mean, median, mode imputation, and interpolation.
- **Functions:** How to write reusable functions in Python for data handling and simulations.
- **Simulations:** Performing simulations to generate data and test hypotheses.

Resources:
Writing Functions in Python: https://www.w3schools.com/python/python_functions.asp https://www.geeksforgeeks.org/python-functions/ https://www.programiz.com/python-programming/function https://www.datacamp.com/tutorial/functions-python-tutorial
Simulations: https://discovery.cs.illinois.edu/learn/Simulation-and-Distributions/Simple-Simulations-in-Python/ https://realpython.com/simpy-simulating-with-python/

Bootstrapping in Python: https://www.digitalocean.com/community/tutorials/bootstrap-sampling-in-python https://carpentries-incubator.github.io/machine-learning-novice-python/07-bootstrapping/index.html
https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.bootstrap.html
Joining Dataframes in Pandas: https://www.datacamp.com/tutorial/joining-dataframes-pandas
Merging, Joining and Concatenating: https://www.geeksforgeeks.org/python-pandas-merging-joining-and-concatenating/