Boolean Axioms

$(T \wedge T) = T, \quad (F \wedge F) = F, \quad (T \wedge F) = F, \quad (T \vee T) = T,$
$(T \vee F) = (F \vee T) = T, \quad (F \vee F) = F, \quad T = T, \quad F = F.$

Commutativity:

$(A \wedge B) = (B \wedge A), \quad (A \vee B) = (B \vee A).$

Identity:

$(A \wedge T) = A, \quad (A \vee T) = T, \quad (A \wedge F) = F, \quad (A \vee F) = A.$

1. Data Wrangling and Analysis Process

Data Collection: JSON, SQL, XML, CSV; sources like data lakes, files, databases.
Data Processing Pipeline: Acquisition, Storage, Cleaning, Transformation, Exploration.
Frameworks: OSEMN Framework, CRISP-DM, KDD Process.
Data Wrangling Steps:
- **Discovering:** Understand and structure data.
- **Transformation and Cleaning:** Normalize, clean errors, remove biases.
- **Enriching:** Add more data or features if necessary.
- **Validation and Publishing:** Confirm correctness, share data securely.

4. Data Models

Relational Model: Uses tables with rows (tuples) and columns (attributes).
Entity-Relationship (E-R) Model: Conceptual model with entities and relationships.
Semistructured Model: Flexible attribute sets, e.g., JSON, XML.
Object-Based Model: Uses objects to represent real-world entities.

5. Components of Data Models

Structure: Defines data format (tables, nodes, key-value pairs).
Constraints: Relationships and rules (e.g., primary key, foreign key).
Operations: Store, retrieve, add, update, delete data.

6. The Relational Model

Instance: A table with rows and columns.
Schema: Specifies table name and types.
Keys:
- **Superkey:** A set of attributes identifying a tuple.
- **Primary Key:** A candidate key selected for uniqueness.
- **Foreign Key:** An attribute referencing another table.

Constraints: Not null, primary key, unique, check conditions.

7. Drawbacks of File Systems for Data Storage

Data redundancy, difficulty accessing data, isolation, integrity issues.
Atomicity Issues: Inconsistent state due to partial updates.
Concurrency: Multiple users accessing data simultaneously.
Security Problems: Difficulty in providing restricted access.

1. Data Models and Types

Structured Data: Organized in tables (SQL).
Semi-structured Data: XML, JSON.
Unstructured Data: Text, multimedia.
Data Model Components: Data, structure, semantics, operations.

2. Relational Algebra Operators

Union ($\cup$): Combines rows from two relations.
Intersection ($\cap$): Returns rows common to both relations.
Difference ($-$): Subtracts rows from the first relation that are in the second.
Selection (σ): Returns rows that satisfy a specified condition.
Projection (π): Returns a relation with only specified attributes (columns).
Join ($\bowtie$): Combines rows from two relations based on a related column.
Rename (ρ): Renames the output of a relational expression.
Extended Operators:
Duplicate Elimination (δ): Removes duplicate rows.
Grouping and Aggregation (γ): Groups rows and applies aggregation functions (SUM, AVG, etc.).
Sorting (τ): Sorts the result based on specified attributes.

Relational Algebra Examples

Q: Find the titles of courses in the Comp. Sci. department that have 3 credits.

$$\pi_{title}\left(\sigma_{dept\_name=Comp.Sci. \wedge credits=3}(course)\right)$$

Q: Find the IDs of all students who were taught by an instructor named Einstein.

$$\pi_{ID}\left(takes \bowtie \pi_{course\_id}\left(teaches \bowtie \sigma_{name=Einstein}(instr\right.\right.$$

Q: Find the highest salary of any instructor.

$$\gamma_{max(salary)}(instructor)$$

Q: Find all instructors earning the highest salary (there may be more than one with the same salary).

$$\sigma_{salary=\gamma_{max(salary)}(instructor)}(instructor)$$

3. String Operations in SQL

LIKE: Matches patterns in strings.
%: Matches any sequence of characters.
_: Matches a single character.
ESCAPE: Defines escape characters for special patterns.

4. Set Operations

UNION: Combines results from two queries.
INTERSECT: Returns common rows from two queries.
EXCEPT: Returns rows from the first query that aren't in the second.

5. Boolean Operators in SQL

AND ($\wedge$): Returns rows satisfying both conditions.
OR ($\vee$): Returns rows satisfying either condition.
NOT ($\neg$): Excludes rows that satisfy the condition.

6. Null Values

NULL: Represents unknown or missing values.
IS NULL: Checks for null values.
Three-valued Logic: SQL uses three logical values (true, false, unknown) when handling nulls.
AND, OR, NOT with null values result in "unknown" in some cases.

1. Integrity Constraints (ICs)

Definition: Enforces data consistency and accuracy in DBMS.
Types of ICs:
Value-based constraints: Define acceptable data types (e.g., CHAR, VARCHAR, INT).
Primary Key: Unique set of attributes per table. One primary key per table.
Foreign Key: Values correspond to another table.

2. Primary Key and Superkey

Primary Key: Uniquely identifies each row in a table.
Superkey: Set of attributes that uniquely identifies tuples.
Multiattribute Key: A key with more than one attribute.

3. Foreign Keys and Referential Integrity

Foreign Key: Refers to the primary key of another table.
Referential Integrity: Ensures references between tables remain consistent.
Actions on Violation: CASCADE, SET NULL, NO ACTION.

4. The CHECK Constraint

CHECK: Ensures that data satisfies a condition.
Example: CHECK (year >1990).

5. NOT NULL Constraint

Ensures a column cannot have NULL values.
Example: name VARCHAR(20) NOT NULL.

6. Database Design

ER Model: Represents data as entities and relationships.
Entities: Objects with attributes (e.g., Student, Instructor).
Relationships: Associations between entities (e.g., Advises, Teaches).
Relationship Cardinality: Defines how entities relate.
Cardinality Types: 1-to-1, 1-to-many, many-to-1, many-to-many.

7. Creating Relations in SQL

Primary Key Declaration: PRIMARY KEY (id).
Foreign Key Declaration: FOREIGN KEY (id) REFERENCES Students(id).

8. Views

A view is a virtual table from a SELECT query.
Example:
CREATE VIEW ActiveStudents AS
SELECT S.name, T.grade
FROM Students S, takes T
WHERE S.id = T.id AND S.tot_cred <31.

Cardinality in Relationships

One-to-One (1:1): An entity in one set relates to only one entity in another. **Example:** Each student has one advisor, and each advisor advises one student.
One-to-Many (1:N): An entity in one set relates to multiple entities in another, but the reverse is not true. **Example:** One department can have many employees, but each employee belongs to one department.
Many-to-One (N:1): Many entities in one set relate to a single entity in another. **Example:** Many students may enroll in one course.
Many-to-Many (M:N): Entities in both sets can relate to multiple entities in the other. **Example:** Students can enroll in many courses, and each course can have many students.

1. Functional Dependencies (FD)

Definition: X functionally determines Y (X $\rightarrow$ Y) if, whenever two tuples have the same X values, they must have the same Y values.
Example: name $\rightarrow$ dept_name in an instructor relation.

2. Use of Functional Dependencies

FDs are used to test if a relation satisfies dependencies, check if decomposition is lossless, and detect inconsistencies/redundancies.

3. Superkey and Candidate Key

Superkey: Set of attributes that uniquely identifies a tuple in a relation.
Candidate Key: A minimal superkey with no proper subset being a superkey.

4. Armstrong's Axioms

Reflexivity: If Y subset of X, then X $\rightarrow$ Y.
Augmentation: If X $\rightarrow$ Y, then XZ $\rightarrow$ YZ (for any Z).
Transitivity: If X $\rightarrow$ Y and Y $\rightarrow$ Z, then X $\rightarrow$ Z.

5. Normal Forms

1NF: All columns contain atomic values, with no repeating groups.
Goal: Eliminate multiple values in a field.
Example: Instead of multiple phone numbers in one field, create separate rows for each phone number.
2NF: In 1NF, all non-key attributes are fully dependent on the entire primary key (no partial dependencies).
Goal: Eliminate partial dependencies.
Example: If Course_ID determines Instructor_Name, but not the full key (e.g., Student_ID + Course_ID), it violates 2NF.
3NF: In 2NF, there are no transitive dependencies (non-key attributes cannot depend on other non-key attributes).
Goal: Eliminate transitive dependencies.
Example: If Student_ID determines Advisor_Name and Advisor_Name determines Department, this creates a transitive dependency.
BCNF: For every FD X $\rightarrow$ Y, X is a superkey.
Goal: Ensure all determinants are superkeys.
Example: If Instructor determines Department, but Instructor is not a superkey, it violates BCNF.

6. Example of FD Violations

If name $\rightarrow$ dept_name is violated (i.e., same name is associated with different departments), this violates the functional dependency.

7. Conditional Functional Dependencies (CFD)

These are FDs that hold under specific conditions.
Example: In the UK, postal code determines street: [country = 'UK', zip] $\rightarrow$ street.

Indexing

Indexes: Speed up selections on search key fields. An index contains data entries that allow efficient retrieval of records.
Search Key: Any subset of fields used to speed up queries (not the same as primary key).

Types of Indexes

B+ Tree Index: Efficient for range queries; leaf pages contain sorted data entries. Supports efficient search and updates.
Hash-based Index: Best for equality searches (e.g., id = 123); uses buckets to store data entries. Not suitable for range queries.

Clustered vs. Unclustered Index

Clustered Index: Data entries stored in the same order as the index, improving performance for range queries.
Unclustered Index: Index points to data stored elsewhere, less efficient for range queries but useful for lookups.

Composite Search Keys

Composite Index: Index on multiple attributes (e.g., age, salary), useful when queries involve multiple attributes (e.g., WHERE age = 30 AND salary = 4000).

Index-Only Plans

Some queries can be answered using only index entries (e.g., using a tree index for COUNT or MIN) without retrieving actual table data.

Choosing an Index

Choose based on the query workload.
Hash Index: For exact match queries.
Tree Index: For range queries.
Trade-off: Indexes improve query performance but slow down updates and require disk space.

Data Integration

Challenges: Different systems have different schemas, data formats, and purposes.
Uniform access needed for interoperation between systems (e.g., relational DBs, XML, web services).

Virtual Integration

Data remains in original sources, and data is fetched on-demand, ensuring it's up-to-date.
On-demand Integration: Data fetched only when needed.

Schema Mappings

Mappings define how local schemas relate to the global schema. These mappings help translate queries and combine results from different sources.

Query Processing in Integration

Global queries are broken down into sub-queries for individual sources, executed, and combined.
Challenges: Inconsistencies, different query languages, and merging results from different sources.

Heterogeneous Data Analysis

Entity Linkage: Different names or descriptions for the same entity (e.g., London, Londres).

String Similarity Methods

Edit Distance: Measures operations to convert one string into another.
Formula: $D(i,j) = \min\{D(i-1,j)+1, D(i,j-1)+1, D(i-1,j-1)+cost\}$,
where cost is 0 if characters are the same, otherwise 1.
Jaro Similarity: Based on common characters and transpositions.

$J_{\text{sim}}(S_1, S_2) = \frac{1}{3}\left(\frac{|C|}{|S_1|} + \frac{|C|}{|S_2|} + \frac{|C|-T}{|C|}\right)$.

Jaro-Winkler: Extension of Jaro, higher weight for common prefixes.
$JW(S_1, S_2) = J_{\text{sim}} + P \times L \times (1 - J_{\text{sim}})$.
Soundex: Four-character code for names. First letter retained, encode consonants as: B,F,P,V $\rightarrow$ 1; C,G,J,K,Q,S,X,Z $\rightarrow$ 2; D,T $\rightarrow$ 3; L $\rightarrow$ 4; M,N $\rightarrow$ 5; R $\rightarrow$ 6. Remove vowels (A,E,I,O,U), W, H. Example: "Smith" $\rightarrow$ S530.

Set-Based Similarity Methods

Jaccard Similarity: Measures similarity between two sets.
$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$.
Sørensen-Dice Coefficient: More weight to common elements.
$C_C(A,B) = \frac{2\times|A \cap B|}{|A|+|B|}$.
Tversky Index: Generalized by weighting differences.
$T(A,B) = \frac{|A \cap B|}{|A \cap B| + \alpha|A \setminus B| + \beta|B \setminus A|}$.
Overlap Coefficient: Focuses on overlap, normalized by smaller set size.
$OC(A,B) = \frac{|A \cap B|}{\min(|A|,|B|)}$.

Document Similarity

Shingling: Converts documents into sets of k-length substrings.
Min-Hashing: Reduces large sets to short signatures while preserving similarity.
LSH: Hashes similar documents into the same bucket for comparison.

Hashing

Formula: Converts a key into an index in a hash table. For integers: $h(x) = x \mod m$, where $m = 7$.
Example: Hashing keys 10, 20, 35, 50, 21, 14:

$h(10) = 3,\ h(20) = 6,\ h(35) = 0,\ h(50) = 1,\ h(21) = 0,\ h(14) =$ (

For strings: $h(x) = \sum \text{ascii}(x[i]) \mod m$.

Data Extraction in Python

Pandas DataFrames: A DataFrame is a rectangular table with ordered columns, where columns can have different types (e.g., numeric, string). Columns are Series, and rows/columns are indexed.
Example:
```
import pandas as pd
data = [['Ohio', 2000, 1.5], ['Ohio', 2001, 1.7], ['Nevada', 2001, 2.4]]
cols = ['State', 'Year', 'Population']
df = pd.DataFrame(data, columns=cols)
```
Loading Data into DataFrames:
From CSV: `df = pd.read_csv('file.csv')`
From Excel: `wb = pd.ExcelFile('file.xlsx')`
`dict = {sheet: wb.parse(sheet) for sheet in wb.sheet_names}`
From MySQL:
```
cnx = connection.MySQLConnection(user='user', password='pwd',
host='127.0.0.1', db='db')
cursor.execute("SELECT * FROM table")
df = pd.DataFrame(cursor.fetchall(), columns=["ID", "Name"])
cursor.close(); cnx.close()
```
Extracting Data from DataFrames:
Rows: `row1 = df.iloc[0], rows_set = df.iloc[5:10]`
Columns: `df['column_name'], df[['column1', 'column2']]`
Filtering Data:
`df.loc[df['column'] == 'value']`
Example: `df.loc[df['content_rating'] == 'PG-13', ['actor_1_facebook_likes', 'budget']]`
Profiling DataFrames:
`print(len(df.columns)), print(len(df)), df.count()`
`df.dtypes, df['column_name'].unique()`
Aggregate Queries:
`df['column'].min(), df['column'].max(), df['column'].mean()`
Set Operations:
Concatenation: `union_df = pd.concat([df1, df2])`
Join/Merge: `df_merge = pd.merge(df1, df2, on='common_column')`

Data Preparation Overview

Data Quality Dimensions: **Accuracy:** Correctly recorded data. **Completeness:** All necessary data is present. **Uniqueness:** No duplicates. **Timeliness:** Data is up to date. **Consistency:** Data doesn't contradict itself.

Data Cleaning Techniques

Handling Missing Data: *Causes:* equipment malfunction, misunderstood data. *Solutions:* ignore missing tuples or fill values manually/automatically using constants or means.

Handling Noisy Data: *Noise:* random errors or variance in data. **Methods:** **Binning:** Smooth by bin means, medians, or boundaries. **Regression:** Fit data into regression functions. **Clustering:** Group data and remove outliers.

Handling Outliers

Types of Outliers: **Global:** Deviates from the entire dataset. **Contextual:** Deviates based on context (e.g., time, location). **Collective:** A group of points that collectively deviate.

Outlier Detection Methods: **Statistical:** Identify data points in low-probability regions of a model. **Distance-Based:** Judge based on distance from neighboring points. **Density-Based:** Compare local density to neighbors (e.g., Local Outlier Factor - LOF).

Data Smoothing Techniques

Binning Methods: **Equal-frequency Binning:** Divide into bins with equal frequencies. **Bin Mean:** Replace values with the bin's mean. **Bin Median:** Replace values with the median. **Bin Boundaries:** Replace values with closest boundary (min/max).

Regression: Use regression models (e.g., linear regression) to smooth data. **Clustering:** Group similar data and treat outliers as noise.

Outlier Detection Techniques

Global vs Local Approaches: Global compares a data point to the entire dataset, while local compares it to its neighborhood.

Scoring vs Labeling Approaches: **Scoring:** Assign an outlier score (continuous measure). **Labeling:** Classify points as "normal" or "outlier."

Statistical Approaches

Parametric: Assume known distribution (e.g., Normal distribution). Flag points in low-probability regions. **Non-parametric:** No distribution assumption (e.g., histograms, kernel density).

Distance-Based Approaches

A point is an outlier if its distance to its neighbors exceeds a threshold. **DB(ϵ, π)-Outliers:** A point is an outlier if fewer than π% of data points fall within radius ϵ.

Density-Based Approaches

Local Outlier Factor (LOF): Compares the local density of a point with its neighbors. A higher LOF score indicates the point is less dense compared to its surroundings.

Model-Based Approaches

Use regression or data-fitting models to detect general trends in the data. Points that don't conform to the model are flagged as potential outliers.

Data Transformation

Definition: Maps values of an attribute to new ones.

Smoothing: Remove noise from data (e.g., binning).

Attribute Construction: Create new attributes from existing ones.

Aggregation: Summarize data (e.g., using data cubes).

Normalization

Min-Max: Rescale data into a specific range.

Formula: $v' = \frac{v - \min}{\max - \min} \times (\text{new\_max} - \text{new\_min}) + \text{new\_min}$

Example: Normalize income of €73,600 from range [12,000, 98,000]:

$v' = \frac{73{,}600 - 12{,}000}{98{,}000 - 12{,}000} = 0.716$

Z-Score: Converts values to a scale with $\mu = 0$, $\sigma = 1$.

Formula: $v' = \frac{v - \mu}{\sigma}$

Example: €73,600 with $\mu = 54{,}000$, $\sigma = 16{,}000$:

$v' = \frac{73{,}600 - 54{,}000}{16{,}000} = 1.225$

Decimal Scaling: Normalize by moving the decimal.

Formula: $v' = \frac{v}{10^j}$

Example: If $j = 3$, $v = -986$, $v' = -0.986$.

Standard Deviation

Formula: $\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(v_i - \mu)^2}$

Example: Data points: 10, 12, 23, 23, 16. $\mu = 16.8$. Deviations: (10-16.8), (12-16.8)... Squared deviations: 46.24, 23.04... Sum = 146.8

$\sigma = \sqrt{\frac{146.8}{5}} = 5.42$

Data Integration

Definition: Combine data from multiple sources into a coherent dataset.

Schema Integration: Aligns schemas from different sources (e.g., Customer ID vs. Cust-No).

Entity Resolution: Identifies different representations of the same entity (e.g., Bill Clinton = William Clinton).

Redundancy: Detected via correlation, covariance analysis. Example: Resolve different customer names "John Smith" vs. "J. Smith."

Data Reduction

Dimensionality Reduction:

PCA: Projects data onto fewer dimensions to retain variance.

Attribute Subset Selection: Removes redundant attributes (e.g., Product Price vs. Sales Tax).

Instance-Based Reduction:

Sampling: Select a subset to represent the dataset.

Simple Random Sampling: Each point has equal chance of selection.

Stratified Sampling: Samples proportionally from each partition (e.g., 70% class A, 30% class B).

Histograms

Equal-width: All buckets have the same range.

Equal-frequency: Each bucket contains the same number of points.

Clustering: Group similar points; store only the cluster representation (e.g., centroids). Example: Cluster customer behaviors for segmentation.

Data Discretization

Definition: Divide continuous data into discrete intervals.

Equal-width Binning: Formula: $W = \frac{B - A}{N}$. Example: Values 1-100 with 5 bins $\rightarrow$ bin width = 20.

Equal-frequency Binning: Each bin has the same number of records.

Supervised vs. Unsupervised: Supervised uses class labels (e.g., decision trees), unsupervised doesn't (e.g., clustering).

Data Visualization Principles

Purpose: Visualization is used for exploration, communication, and entertainment.

Effective Graphics for Data Analysis

Common Problems: Misrepresentation due to omitting baselines or distorting scales.

Grammar of Graphics (Wickham's Version)

Elements:

Aesthetics: Mapping data to visual properties (position, color, size).

Geoms: Points, lines, bars, etc.

Scales: Continuous or discrete (e.g., Cartesian coordinates).

Facets: Split data into sub-plots (small multiples).

Statistical transformations: Summarize data (binning, median).

Coordinate system: Defines how data is displayed (Cartesian, polar).

Example (ggplot2 in R):

```
ggplot(data = mpg) +
geom_point(mapping = aes(x = displ, y = hwy, color = class))
```

Aesthetics: x: Engine size (displ), y: Fuel efficiency (hwy), *color*: Car type (class). Geom: Points. Scale: Continuous, Cartesian.

Gestalt Principles of Visual Perception

Proximity: Close objects are related.

Similarity: Similar-looking objects are related.

Connection: Connected objects seem related.

Continuity: Partially hidden objects are perceived as continuous.

Closure: Incomplete shapes are perceived as complete.

Figure/Ground: Elements perceived in foreground/background.

Common Fate: Objects moving together are perceived as a unit.

Principles from Tufte

Data-to-ink ratio: Maximize relevant data and minimize unnecessary decoration.

Present more data: Show different levels of detail without losing clarity.

Guide viewer's eye: Lead to the right comparisons.

Color in Visualizations

HSB Model: Hue, saturation, and brightness. Proper use helps guide attention and distinguish data.

Exploratory Data Analysis (EDA)

Definition: Explore data to generate insights/hypotheses.

Exploratory vs. Confirmatory:

Exploratory: Generate insights/hypotheses.

Confirmatory: Test pre-existing hypotheses.

Good Practices for EDA

Tukey's Approach:

1. Center/spread of data. **2.** Compare across features. **3.** Transform skewed data (e.g., logarithms). **4.** Use models to remove patterns.

Peng's EDA Checklist:

1. Formulate question. **2.** Read in data, check structure (e.g., `str()`). **3.** Check top/bottom for outliers, missing values. **4.** Validate with external sources. **5.** Start simple, challenge your results.

Visual EDA: Key Questions

Variation: Univariate: How is a single feature distributed?
Specific: Variation in a specific population.
General: Variation across all units of measurement.

Association: Do features covary (e.g., dep. delay vs. arr. delay)?

Common Graph Types in EDA

Bar Chart: Categorical data (**Geom:** Bars).

Histogram: Continuous data (**Geom:** Bars).

Density Plot: Smooths continuous data (**Geom:** Line).

Scatter Plot: Two continuous variables (**Geom:** Points).

Box Plot: Median, quartiles, outliers (**Geom:** Box/IQR, whiskers).

Facets: Multiple subplots for categories.

Solutions to Overplotting

Transparency (alpha): Reduce opacity to avoid clutter.

Binning: Use `geom_bin2d` for 2D scatter plots.

Additional Techniques in EDA

Straightening: Logarithmic transformations for skewed data.

Flattening: Use models to remove patterns and focus on residuals.

Machine Learning Overview

1. Definition (Samuel/Mitchell, 1959):
A computer program learns from experience (E) with respect to some task (T) and performance measure (P) if its performance improves with experience.

2. Types of Machine Learning:

- **Supervised Learning:** Learn from labeled data to make predictions (e.g., classification, regression).
- **Unsupervised Learning:** Find hidden patterns in data (e.g., clustering).
- **Reinforcement Learning:** Learn through interactions with the environment to maximize a reward.

Supervised Learning: Regression vs. Classification

1. **Regression:** Predicts a continuous outcome (e.g., house prices).
2. **Classification:** Predicts a discrete label (e.g., fraud detection: fraud vs. not fraud).

Regression Model: Key Concepts

1. **Linear Regression:**
Formula: $y_i = b_0 + b_1 x_i + \epsilon_i$ where:
- y_i is the outcome (dependent variable).
- x_i is the predictor (independent variable).
- b_0 is the intercept, and b_1 is the slope.
- ϵ_i represents the error term (irreducible error).

2. **Quadratic Regression:**
Adds polynomial terms to the model: $y_i = b_0 + b_1 x_i + b_2 x_i^2 + \epsilon_i$ allowing the model to capture non-linear relationships.

Goals of Regression

1. **Prediction:** Estimate the function $f(x)$ as closely as possible to predict future outcomes.
2. **Inference:** Understand the relationship between x and y (e.g., how strongly is x related to y? How precise are the estimates?).

Bias-Variance Tradeoff

1. **Bias:**
Measures how far off predictions are from true values on average.
- High bias: The model is too simple (underfitting).

2. **Variance:**
Measures how much predictions vary with different training datasets.
- High variance: The model is too complex (overfitting).

3. **Tradeoff:** Increasing model complexity reduces bias but increases variance. An ideal model minimizes mean squared error (MSE).

Train-Validation-Test Paradigm

1. **Training Set:** Used to fit the model.
2. **Validation Set:** Used to tune model parameters and select the best model.
3. **Test Set:** Used to estimate the model's generalization performance.

Model Evaluation Metrics

1. **Mean Squared Error (MSE):**
Measures the average squared difference between predicted and actual values.
Formula: $MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$ where y_i is the true value, and $\hat{y}_i$ is the predicted value.

2. **Cross-Validation:**
K-fold Cross-Validation: Split data into K subsets, train the model K times, each time using a different subset as the validation set. Average the results for better generalization.

K-Nearest Neighbors (kNN) for Regression

1. **Concept:** Predict the outcome by averaging the outcomes of the k-nearest neighbors.
Works well with low-dimensional data but suffers from the curse of dimensionality with more predictors.

SQL Queries and Additional Concepts

Unique Index:

```
CREATE UNIQUE INDEX index_name ON table_name(column_name);
```

Ensures uniqueness for a column.

Subquery:

```
SELECT * FROM table_name WHERE column_name = (SELECT
MAX(column_name) FROM another_table);
```

Uses a result from a subquery in the main query.

CASE Statement:

```
SELECT column_name, CASE WHEN condition1 THEN result1 ELSE result2
END FROM table_name;
```

Adds conditional logic in `SELECT`.

Window Function (RANK):

```
SELECT column_name, RANK() OVER (PARTITION BY category_column ORDER
BY value_column DESC) AS rank FROM table_name;
```

Ranks rows within partitions.

Recursive Query:

```
WITH RECURSIVE subordinates AS (SELECT employee_id, manager_id FROM
employees WHERE employee_id = ?  UNION ALL SELECT e.employee_id,
e.manager_id FROM employees e INNER JOIN subordinates s ON
s.employee_id = e.manager_id) SELECT * FROM subordinates;
```

Handles hierarchical data like employees reporting to managers.

Transaction Handling:

```
BEGIN TRANSACTION;
-- SQL queries here
COMMIT;
```

Executes multiple queries as a single unit.

Python (Pandas DataFrame Operations)

Handling Missing Data:

```
df.fillna(value=0, inplace=True)
df.dropna(inplace=True)
```

Fills or drops missing data.

Pivot Table:

```
pivot_table = df.pivot_table(values='column_name',
index='index_column', columns='column_to_pivot', aggfunc=np.sum)
```

Creates pivot table to aggregate data.

Apply Function:

```
df['new_column'] = df['existing_column'].apply(lambda x:  x * 2)
```

Applies a function element-wise to a column.

Datetime Operations:

```
df['date_column'] = pd.to_datetime(df['date_column'])
df['year'] = df['date_column'].dt.year
```

Converts strings to datetime and extracts parts like year.

GroupBy with Aggregations:

```
df.groupby('group_column').agg({'column1':  'sum', 'column2':
'mean'})
```

Aggregates data by groups.

R Code

mutate():

```
df <- df %>% mutate(new_column = column1 * 2)
```

Creates new column by applying an operation.

Summary by Group:

```
df %>% group_by(group_column) %>% summarize(mean_value =
mean(numeric_column))
```

Calculates summary statistics per group.

Join DataFrames:

```
df <- left_join(df1, df2, by = 'common_column')
```

Joins two data frames on a common column.

Visualization

Python Subplots (Matplotlib):

```
fig, axes = plt.subplots(nrows=2, ncols=2)
df.plot(ax=axes[0, 0])
plt.show()
```

Creates multiple plots in a grid.

R Boxplot (ggplot2):

```
ggplot(data = df) + geom_boxplot(mapping = aes(x = factor_column, y
= numeric_column))
```

Draws a boxplot for categorical vs numeric data.